Trabajo Final Máster Software Libre

Título:

Análisis, desarrollo e implantación de plataforma de traceo y monitorización en redes de acceso de telecomunicación basadas en protocolos Ethernet/IP mediante uso de herramientas libres en entornos GNU/Linux

Documento:

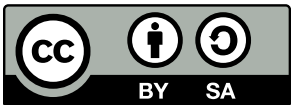
PAC5. Memoria Final TFM

Especialidad:

Administración de redes y sistemas operativos.

Autor: **Antonio Gandía Ortiz**
Consultor: **Miguel Martín Mateo**

Enero 2012



Resumen

Con el presente trabajo se trata de dar respuesta a la necesidad planteada dentro de un operador de telecomunicaciones para poder hacer traceos y análisis para troubleshooting de bajo nivel en redes UTRAN con transporte Ethernet/IP con software única y exclusivamente de licencia libre.

Tras una etapa de planificación (metodología a seguir, plazos, estudio de necesidades,...) se concluyó que era viable un prototipo que demostrase que podría satisfacer todos los requisitos mínimos y buena parte de los deseables planteados. Estos requisitos los resumimos en tres grupos:

- Hacer el conexionado de sondas para interceptar y analizar comunicaciones. Elegimos el modo *Remote Port Mirroring* porque es una funcionalidad soportada por la red Backhaul del operador. Pero se dejaba como extensión a futuro la posibilidad del *Port Mirroring* desplazando físicamente al punto de monitorización, un ordenador de los técnicos de campo.
- Qué sistema base montar. Para satisfacer ambos escenarios de captura, se optó por una solución de virtualización que funcionara en cualquier equipo (Linux o Windows) basada el *Virtualbox*. El sistema operativo se concluyó que fuera la distribución ligera *Lubuntu*. Usando sus paquetes y utilidades se habilitaron accesos remotos al escritorio de usuario, servicio de transferencia de ficheros y se abrieron canales para administración remota.
- Qué software de análisis y traceo usar. Tras hacer un repaso de los principales paquetes de monitorización de red con licencia libre, concluimos que *Wireshark* satisfacía todos los requisitos mínimos y casi todos los máximos.

Se estudió la manera de cumplir otros requisitos accesorios, como *coste próximo a cero*, s*eguridad de acceso* al sistema, *privacidad/separación de comunicaciones* entre la red corporativa y la red de telecomunicaciones comercial,... Se concluyó que la plataforma pensada los satisfacía.

Por deseo de la empresa, en este proyecto se iba a diseñar e implementar un prototipo en Remote Port Mirroring a pequeña escala, para su estudiar su ampliación futura. Para éste se contó con material antiguo reusado con el que se montó físicamente la plataforma de monitorización. Se diseñaron las configuraciones, algunas de ellas, muy específicas (*habilitar captura en Wireshark sin ser root, arranque automático de Virtualbox como Servicio (servidor headless), diseño del hardware virtual y elección de versión de software adecuado para evitar el VLAN stripping*,...).

El sistema se montó y validó según el plan de aceptación diseñado. Se estudiaron y realizaron casos de usos, validándose igualmente el correcto funcionamiento de la plataforma. También se realizaron una sesiones formativas a usuarios, así como un manual de uso de la misma.

Por último, se hizo un análisis crítico en profundidad de las posibilidades de la plataforma para analizar las comunicaciones de un IuB de nodo B. Se vio que *el suministrador de Backhaul, todavía no tiene implementado el sentido egress de la funcionalidad Remote Port Mirroring* y se pidió para su futura release. Se hicieron estudios de *cómo analizar tráficos de gestión, sincronismo, señalización y de usuario* mediante Wireshark. También se vio que corrigiendo desde Wireshark el payload type de SCTP (está mal etiquetado por el suministrador), *se puede decodificar toda la señalización entre nodo B y RNC* del protocolo de aplicación NBAP, pudiendo seguir cualquier proceso de la red UMTS como establecimiento de celdas, llamadas, handovers,...

Con ello, se concluye que el prototipo (salvando el fallo del fabricante para el egress del Remote Port Mirroring) puede realizar con éxito las funciones que se buscaban, pudiendo proceder a un estudio para una implantación a mayor escala.

Índice de contenido

1. Tipo, Metodología y Ciclo de Vida del Proyecto

Comenzaremos la presente memoria, especificando el marco general del proyecto y la manera de trabajar que hemos empleado durante el mismo.

En primer lugar, clasificaremos el tipo de proyecto realizado. Esta clasificación se puede hacer según diferentes criterios, pero según se vio en la asignatura *"Implantación de proyectos de software libre"[ImpSL]*, generalmente se puede analizar el **tipo de proyecto**:

- Según el ámbito: Claramente es un **proyecto interno**, ya que es desarrollado dentro de la misma organización y el uso para el que se piensa es a priori también interno.

- Según su objetivo: Principalmente podemos clasificarlo como un **proyecto de software**, ya que lo que se persigue es la implantación de una aplicación o grupo de aplicaciones que respondan a las necesidades del cliente. No obstante, también tiene una vertiente relativamente importante de **proyecto de infraestructura**, ya que tendremos que diseñar e implementar una plataforma con sondas y máquinas que no forman parte de los equipos en producción antes del comienzo del mismo.

En cuanto a la metodología a seguir escogemos seguir una **metodología clásica en cascada**, donde las fases se van sucediendo hasta llegar al producto. Los motivos por los que escogemos esta metodología son:

- Tiempo de entrega predefinido e inflexible. Como el proyecto debe estar finalizado con la finalización del semestre académico, con una planificación temporal exacta de las fases nos aseguramos que llegaremos a obtener un resultado (mejor o peor).

- Experiencia. En mi carrera profesional he podido participar en varios proyectos en los que se han seguido este tipo de metodologías y cómo aplicarlas.

Durante la asignatura *"Ingeniería de software en entornos de software libre"[IngSL]*, tuvimos la ocasión de estudiar una metodología en cascada como es *Métrica v3* (muy utilizada en desarrollos en la administración y grandes corporaciones tradicionales). Ésta lleva un proceso muy férreo y estructurado de fases con definiciones de entradas/salidas, tipo y estructura de documentación, criterios para la gestión del proyecto (costes, tiempos, recursos,...) de seguridad y aseguramiento de calidad final, que por falta de recursos queda muy lejos de nuestras intenciones. Sin embargo, sí que asumimos que la filosofía e ideas clave de la misma, iban a ser la inspiración de la metodología que se intentaría aplicar en nuestro proyecto.

En la literatura existen otras metodologías realmente atrayentes, como son la *incremental* o *evolutivas/en espiral* (por ejemplo la metodología *eXtreme Programming* o *XP*) las descartamos porque:

- Exigen de una mayor implicación de clientes que vayan aportando ideas, priorizando requisitos y evaluando resultados en un ciclo continuo junto al desarrollo. Este proyecto no se puede considerar como clave para la organización (ni para el departamento) y no se podría involucrar a tanta gente en el mismo.

- No hemos participado nunca en proyectos desarrollados con este tipo de metodologías y seguro que esta inexperiencia nos hubiera llevado a cierta desorganización, pérdidas de tiempo y equivocaciones que pusieran en riesgo el objetivo de entregar el proyecto en forma y tiempo.

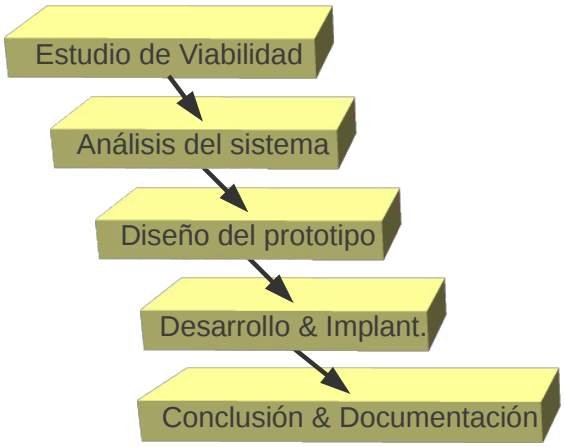
- Esta misma desorganización probablemente no habría sido buena para dar una clara y ordenada visibilidad del trabajo realizado en la memoria que será la base de la evaluación del proyecto para la Universidad.

- Aunque estas metodologías suelen llevar a un resultado más satisfactorio para el cliente y más cercano a sus necesidades, dado que es un proyecto interno y que seremos nosotros mismos el cliente, consideramos que no nos alejamos tanto de éstas. No obstante sí que existía la posibilidad de que durante la fase de implantación nos diéramos cuenta que había otras herramientas disponibles o diseños mejores o simples. Pero finalmente, siendo conservadores en cuanto a plazos de entrega, no nos quedó alternativa más que conocer y asumir estos riesgos.

Para terminar, hay señalar que si bien utilizaremos una metodología clásica en este proyecto, si pensamos en el ciclo de vida de la aplicación que es objeto del mismo, es nuestro deseo que este proyecto sólo sea una *primera iteración de un ciclo de vida iterativo de largo recorrido* con mejoras sucesivas (a sabiendas de los comentados riesgos que asumimos de errores/omisiones en el diseño) fuera ya del ámbito meramente académico. De hecho, **el sistema que implantado es más bien un prototipo funcional que demuestra a pequeña escala, la utilidad para resolver las necesidades**.

2. Fases del Proyecto

Como hemos dicho, en el proyecto aplicaremos una metodología clásica en cascada, pero adaptándola a cómo pensábamos que se iba a desarrollar nuestro proyecto. Concretamente *agrupamos las fases de Desarrollo e Implantación*, porque pensamos poner en marcha un prototipo (dejando una implantación a gran escala tras la finalización de este proyecto y previa evaluación de resultados por la empresa). También añadimos una fase final para sacar conclusiones y documentar convenientemente en memoria y presentaciones hacia la Universidad.



A continuación especificaremos las tareas involucradas en cada una según la idea de nuestro proyecto en concreto. El resto de esta memoria detallará exhaustivamente todo el trabajo desarrollado en cada fase.

2.1. Estudio de Viabilidad

El estudio de viabilidad tiene como objetivo, fijar en un documento acordado por el cliente y el desarrollador del proyecto las necesidades y cómo se pretenden satisfacer (solución de alto nivel) diciendo si es viable y con qué costes.

Dentro del estudio de viabilidad realizamos las siguientes tareas:

- **Análisis del estado actual.** Veremos qué tipo de red queremos analizar. También describiremos someramente el entorno de la red corporativa en que integraremos el sistema.
- **Identificación de requisitos/necesidades.** Tras reuniones con personal implicado de la empresa, fijaremos unos requisitos mínimos que se deben cumplir y unos requisitos máximos que se intentarán conseguir con el proyecto.
- **Estudio y valoración de alternativas de solución.** Para cada requisito haremos un estudio de soluciones a alto nivel que permiten satisfacerlo, considerando que pensamos en un prototipo funcional. Las valoraremos y emitiremos unos cuadro comparativos/DAFO que nos ayuden a la toma de decisiones.
- **Selección de la solución.** Elegiremos una de las alternativas de solución para cada requisito, valorando si en términos generales es viable su implantación a priori. Este primer paso supone un primer compromiso en cuanto al trabajo involucrado en el proyecto.

2.2. Análisis del Sistema

Con la solución acordada en el apartado anterior, deberemos proceder a un análisis en profundidad de las posibilidades y limitaciones que nos podamos encontrar, cómo se usará e integrará en las actividades de la empresa y un plan de pruebas que deberá cumplir el sistema para verificar que funciona correctamente una vez esté en marcha.

Tendremos estas tareas:

- **Definición exacta del sistema y requisitos adicionales.** En la fase anterior, se seleccionó una solución, pero ahora deberemos detallarla y estructurarla (será la base para el diseño). Este punto será clave para analizar si deberemos hacer un diseño de integración de productos ya desarrollados o deberemos de hacer también desarrollos de software adicionales. También con este diseño detallado deberemos precisar cómo solventar algunos requisitos más concretos y prácticos (legales, de seguridad, de acceso, de integración en la empresa,...).
- **Definición de casos de uso.** Se especificará el procedimiento que un usuario deberá seguir para usar la plataforma que diseñaremos.
- **Definición interfaz de usuario.** Como continuación de la tarea anterior, especificaremos quién y cómo accederá al sistema y qué verá éste.
- **Plan pruebas de aceptación.** Documento en el que se especificará previamente al diseño e implantación del sistema, qué pruebas se deberán hacer para comprobar la validez y buen funcionamiento del sistema, previo paso a la aceptación por parte del cliente.

En todas estas tareas, aunque estemos centrados en la idea de un prototipo funcional, es muy conveniente que ***en la fase de Análisis, tengamos en cuenta una futura extensión de este prototipo a un uso masivo en toda la compañía***, ya que así nos asegurará una fácil extensión y facilitará el éxito para que el proyecto tenga continuidad en el futuro.

2.3. Diseño del Prototipo

Con el análisis hecho, procederemos a diseñar a bajo nivel el prototipo del que es objeto este proyecto. Las tareas a realizar serán:

- **Definición de la arquitectura.** Especificaremos los componentes internos del sistema y cómo se interrelacionarán entre ellos.
- **Funcionamiento interno para los casos de uso.** Para los casos de uso vistos en la fase de análisis, veremos qué implicaciones y funcionamiento interno tienen para cada componente.
- **Especificación del desarrollo e implantación.** Especificaremos cómo y con qué herramientas haremos el desarrollo del sistema y qué requisitos debemos seguir para garantizar el éxito de la implantación.

Es importante que en esta fase sí que debemos centrarnos en el prototipo concreto que deseamos implantar a pequeña escala.

También si se hubiera requerido desarrollo adicional de software, tendríamos que realizar un diseño específico del mismo (estructura de clases, métodos/propiedades, funcionamiento, interfaces,...).

2.4. Desarrollo e Implantación

Una vez diseñado el prototipo debemos hacerlo realidad. En la literatura, se suele subdividir en dos fases diferenciadas el desarrollo y la implantación. Dado que lo que pretendemos es una implantación de un prototipo básico funcional, no tiene mucho sentido separarlas en dos (sobre todo la parte de implantación).

Las tareas a hacer serán:

- **Integración del sistema.** Montaje físico, instalación de paquetes de software y configuración de todo el sistema. Será conveniente ir haciendo pruebas de funcionamiento de cada componente/paquete conforme se vaya integrando.
- **Realización del plan de pruebas de aceptación.** Ejecución de las pruebas especificadas en la fase de análisis.
- **Manuales de usuario.** Guías básicas y funcionales del uso del sistema.
- **Realización y análisis crítico de ejemplos de casos de uso**.
- **Formación y soporte a los usuarios.** El prototipo quedará disponible durante el periodo de evaluación para un grupo reducido de técnicos de operación (de la territorial de Levante y Baleares) a los que se les dará unas charlas formativas de cómo manejarlo.

2.5. Conclusión y Documentación

Aunque en la literatura, esto no es una fase propiamente dicha, dado el carácter académico/pedagógico del proyecto (desde el punto de vista del alumno/universidad) y de evaluación para un uso futuro masivo (para la empresa), al finalizar la implantación del mismo, deberemos someter todo el proyecto a un análisis crítico y confeccionar una memoria documental de todo el proyecto.

Así pues, vemos las siguientes tareas adicionales necesarias:

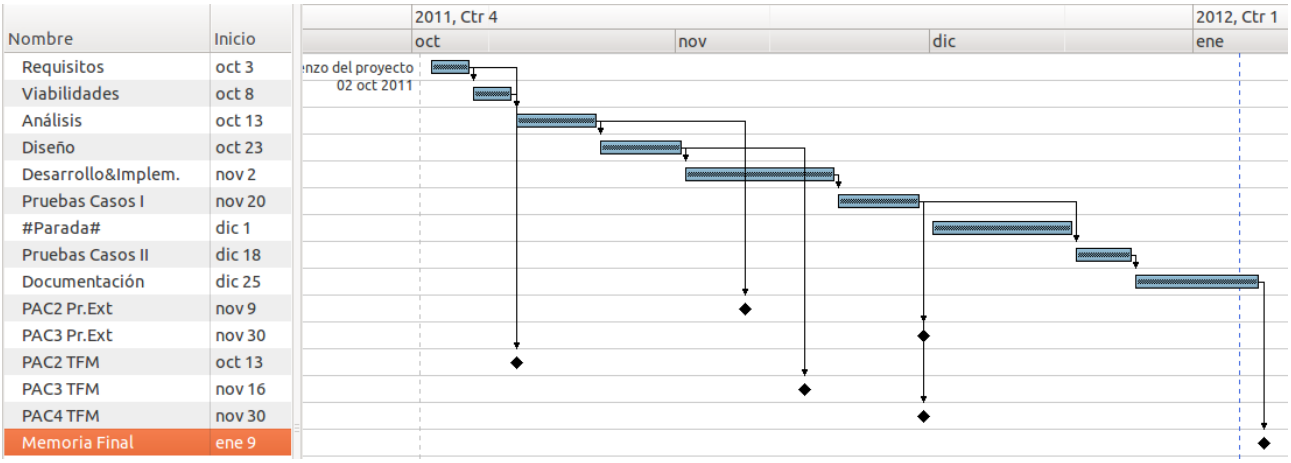
- **Memoria del proyecto.** Integración en un único documento autocontenido de todos los documentos intermedios realizados durante el proyecto.
- **Evaluar el cumplimiento de los requisitos/necesidades** del cliente detectadas en la primera fase.
- **Evaluar la extensión del prototipo** a un sistema implantado a nivel de toda la compañía.
- **Conclusiones** extraídas de todo el trabajo dentro de la asignatura Proyecto Final de Máster.
- **Preparación de presentaciones, vídeos y defensa del proyecto académico**.

2.6. Calendario

Las tareas descritas son las acordadas en el plan de trabajo de las prácticas externas que acordamos con la empresa (nuestro cliente).

Dado el carácter académico del proyecto, se estudió y justificó un calendario para cada una de esas tareas de acuerdo a los hitos intermedios de entrega de documentación que la universidad nos ha exigido tanto en la asignatura *Trabajo Final de Máster* y *Prácticas Externas en Empresa*.

Aunque hemos tenido que ir adaptándola a los diferentes cambios y problemáticas surgidas (por ejemplo, el proyecto tuvo que ser parado por motivos personales a principios de diciembre) la calendarización final seguida se parece bastante a la planteada inicialmente y es la que se muestra en el siguiente diagrama de Gantt:



3. Estudio de viabilidad

3.1. Análisis del Estado Actual

a) Situación actual de las Redes de Telecomunicaciones

En la literatura, las grandes redes de telecomunicaciones se suelen estructurar en dos grandes bloques:

- **Red de Acceso:** aquella parte de la red encargada de gestionar los recursos físicos para el acceso múltiple simultáneo de los usuarios al servicio.
- **Núcleo de Red o Red Core:** aquella formada por los equipos capaces de dar el servicio de conmutación final (de circuitos -p.e. Voz- y/o de paquetes -p.e. Datos-) y todos los servicios suplementarios que permite poner en comunicación dos o más clientes.

Las *redes core* suelen estar compuestas por unos *pocos elementos pero de gran capacidad* de conmutación. Las *redes de acceso* suelen ser *muchos elementos de una gran capilaridad y estructura en árbol* descendente hasta llegar al enlace final con el cliente.

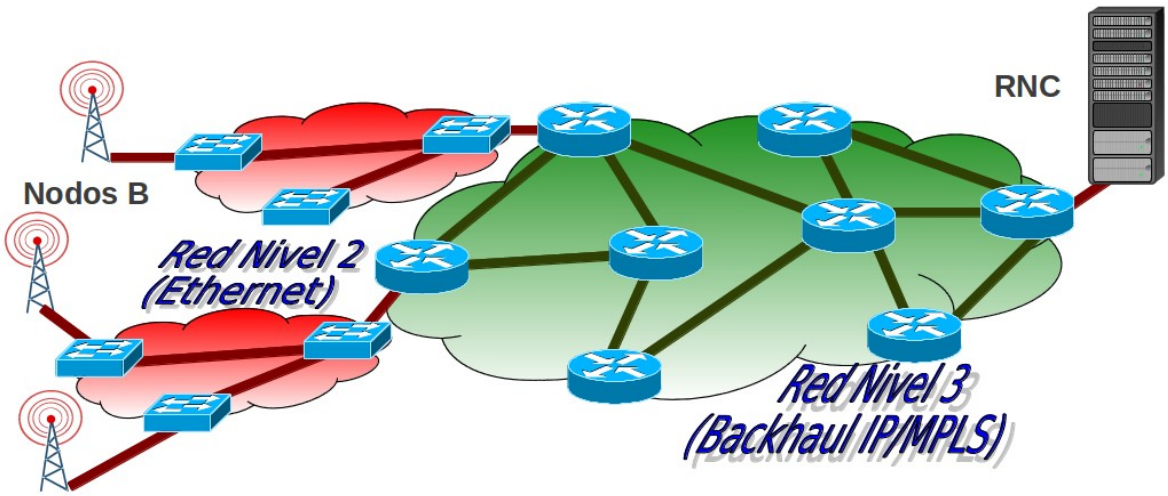
Para ambos tipos de redes, en la actualidad estamos asistiendo a la *sustitución de los tradicionales estándares de transporte* en redes de telecomunicaciones transnacionales tanto fijas como móviles[1] *por la adopción del protocolo IP*[2] *sobre redes de paquetes de nivel 1/2*[3] . Para que el protocolo IP (que fue pensado para transmisión best-effort de paquetes en redes de ordenadores) se comporte adecuadamente en este entorno se le hicieron algunas modificaciones para permitir:

- Clasificación de tipo de tráfico (*Traffic Shaping*).
- Priorización de cada clase de tráfico en las colas de los conmutadores según la calidad demandada para dicha clase (*QoS*).
- Mejora de eficiencia de conmutación *orientándolo a conexión* (p.e. el estándar *MPLS*, que basa la conmutación no de datagramas sino de túneles por los que transitan los paquetes).

En el presente proyecto, nos centraremos en las redes de acceso y más concretamente en las encargadas de servicios móviles 3G (UTRAN, UMTS Terrestrial Radio Access Network).

En la siguiente figura vemos un ejemplo clásico de red UTRAN según la evolución a IP ya prevista en Release 4 por el 3GPP[4] [3GoIP]. Básicamente lo que vemos es que todo el transporte se soporta sobre una capa única de de nivel 3 IP (comúnmente llamada **Backhaul**) formada por routers. Dada la capilaridad de la red hasta los últimos puntos de acceso (nodos B), no sería rentable llegar con conmutación de nivel 3 hasta ellos, sino que los últimos tramos de la red suelen ser segmentos de redes de nivel 2 interconectados por equipos capaces de hacer switching ethernet.

1 Estándares basados en PDH, SDH, X.25, ATM, MTP3, ...
2 Y toda la familia de protocolos de nivel 3 asociados: ICMP, ARP, OSPF, RIP,...
3 Generalmente alguna versión de Ethernet o enlaces punto a punto con PPP/HDLC
4 3rd Generation Partnership Project, que es el comité para el acuerdo de colaboración de los diferentes estamentos de normalización de cada país o zona.



En cuanto a los enlaces físicos punto a punto entre elementos (líneas rojas y verdes) pueden ser de muy diferentes tecnologías (cableados eléctricos Ethernet, fibras ópticas, radioenlaces de microondas, enlaces IP sobre estándares de circuitos SDH/PDH,...).

Esta diferente tipología de elementos de interconexión física, nos dificultará la tarea de monitorizar lo que pasa por dichos enlaces, aunque el protocolo de nivel 2 sea siempre Ethernet y el de nivel 3 IP.

Por último, es necesario destacar la gran versatilidad del Backhaul IP en cuanto a diferentes servicios que se soportan sobre ella:

- **VLL** (Virtual Leased Line) o **E-Line**: Enlace punto a punto desde un puerto Ethernet de un router frontera hasta otro. A efectos prácticos se comporta como un cable Ethernet.
- **VPLS** (Virtual Private LAN Service) o **E-LAN**: La red se comporta como un gran switch cuyas bocas ethernet pueden ser los puertos que deseemos de cualquier router frontera.
- **L3-VPN** (Layer 3 Virtual Private Network): La red se comporta como un gran router capaz de conmutar los paquetes IP que entran por los puertos que configuremos de los router frontera.
- **Ethernet Port Mirroring**: Se puede enviar a cualquier puerto Ethernet de la red una réplica de todas las tramas Ethernet que pasan por cualquier otro puerto que esté funcionando con alguno de los servicios mencionados anteriormente.
- **CES** (Circuit Emulation Service): La red emula circuitos TDM punto a punto (PDH/SDH).
- **ATM** (Asynchronous Transfer Mode Service): La red emula transporte de celdas ATM.

Gracias a esta versatilidad, se podrían proponer varios esquemas de monitorización y traceo de forma remota con una sola sonda pinchada en algún punto del Backhaul.

b) Sistema de Gestión de la Red

La gestión de toda red de comunicaciones requiere que pueda operarse y mantenerse. En estos conceptos se engloban las siguiente tareas:

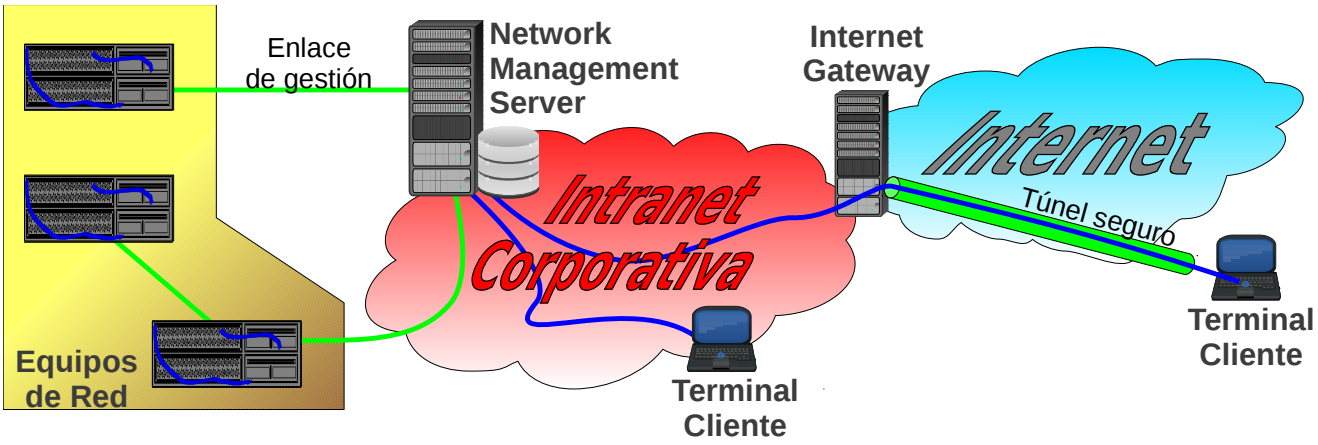
- **Configurar** cualquier elemento de la red.
- Captura y tratamiento de **eventos y alarmas** que se producen en los elementos, pues suelen ser indicativos de problemas o riesgos.
- Extracción y almacenamiento de **estadísticas** de uso del servicio (p.e tráfico), de calidad del mismo (p.e. Disponibilidad) y del performance de los elementos (p.e. carga cpu, memoria).
- Realización de **tests y traceos** para comprobar el correcto funcionamiento de los procesos y protocolos y ayudar a la eliminación de problemas (*troubleshooting*).
- Generación de **copias de seguridad** (*backups*) periódicos para ser usados en caso de pérdidas catastróficas de datos de configuración.

Todas estas tareas por motivos de eficiencia, se tienden a hacer de forma remota desde los *OMC*s (*Operation and Maintenance Center*) por parte de los *técnicos de operaciones* sin necesidad de acceder físicamente a los equipos (tarea de *técnicos u operarios de campo*).

Para resolver en mayor o menor medida las tareas rutinarias de operación y mantenimiento de forma remota, prácticamente todos los fabricantes de equipos proponen una arquitectura cliente-servidor con estas características:

- Uno o varios servidores que actúan de BBDD de configuraciones, eventos, estadísticas,...
- Los servidores tienen comunicación con todos los elementos de la red (a través de enlaces dedicados o mediante túneles a través de la misma intranet corporativa del operador).
- Los operarios desde sus ordenadores acceden a los servidores mediante clientes donde se pueden ordenar al servidor la ejecución de todas las tareas descritas. Este acceso suele hacerse a través de la intranet corporativa.

No es el objeto de este trabajo (por motivos de confidencialidad) describir la topología y funcionalidades de la red corporativa. Pero para lo que nos atañe al proyecto, podemos resumirla en la siguiente figura general, que se corresponde bastante bien con el aspecto de cualquier red de corporativa:



En lo que respecta a la intranet, cabe comentar que es una *red del tipo VPN* que llega a todos los centros de trabajo de la compañía, viéndose todos los *equipos interconectados a nivel IP dentro de una subred privada de clase A*. Por políticas de seguridad, no todas las comunicaciones entre todas las direcciones y puertos TCP/UDP están permitidos, marcándose unas *reglas de firewall* internas.

El *acceso a internet* desde los terminales cliente se realiza a través de los servidores que ejercen tareas de *gateway* (*proxy*). También es posible acceder con restricciones a la intranet desde internet a través de *túneles securizados* (*clientes de VPN*).

En cuanto a la arquitectura interna de las máquinas involucradas, los *servidores* son sistemas propietario diseñados a medida por cada fabricante de equipos de telecomunicación. Por su parte los *terminales de cliente*, suelen ser ordenadores personales (fijos/portátiles) con sistema operativo del tipo *Windows*, con paquetes de software privativo base preinstalados (p.e. Ofimáticos) y sobre los que se instalan aplicaciones cliente específicas para dialogar con los servidores.

c) Problemática Actual

Aunque prácticamente todos los fabricantes de equipamiento para redes de paquetes IP, ofrecen plataformas de gestión orientadas a ayudar al *troubleshooting*, éstas en la mayoría de los casos adolecen de varios problemas:

- Pocas (nulas) posibilidades de traceo. Hay veces que con una simple traza grabada de lo que envía/recibe un nodo de la red, se puede detectar la causa de los problemas (ver si las tramas son mal formadas, si algún tipo de trama no se envía, si se fragmenta la trama a partir de cierto tamaño, si se pierden o llegan con errores,...), si hay violación del protocolo estándar,... Esto prácticamente ninguna plataforma de gestión lo ofrece.
- Estadísticas poco segmentadas. Con suerte, las herramientas de fabricante suelen dar algún tipo de estadística de paquetes envíados/recibidos globales a un interfaz, pero sin apenas segmentación por tipo, calidades, VLANs o direcciones origen/destino.
- Muy heterogéneas y de diferentes enfoques de unos suministradores a otros.
- Si se va a buscar al mercado una plataforma paralela (ajena a los equipos) de monitorización comercial, éstas suelen tener un coste económico muy elevado.
- Por la extensa capilaridad de la red de acceso (llegar hasta el último elemento de una red que da servicio a millones de usuarios allá donde estén) el acceso a estos elementos suele ser complejo y no realizado por personal humano de alta cualificación técnica. Si hay que pinchar sondas de traceo en estos puntos, la tarea puede ser muy costosa.
- Poco flexibles para estudiar problemas de este otros puntos de vista (p.e. capturar solo algún tipo de tráfico, de algún cliente/elemento de la red, desde algún punto intermedio o remoto).

3.2. Identificación de Requisitos y Necesidades del Cliente

Como se desprende del análisis de la situación actual y su problemática, la principal necesidad de la empresa es **poner al alcance del técnico de operaciones una nueva herramienta** (accesible remotamente desde los habituales terminales de cliente de operación) **que le ayude a diagnosticar la causa de un problema en la red IP de acceso, basándose en capturas y análisis de trazas y estadísticas detalladas de los interfaces**.

Esta herramienta, por deseo del cliente, será en primera instancia un **prototipo que demuestre las posibilidades y capacidades del sistema**, y que tras esta etapa, se pueda evaluar la posibilidad de extenderla o implantar en un sistema mayor.

De acuerdo con el cliente, establecemos una serie de requisitos mínimos y otros deseables que la herramienta debe cumplir.

Requisitos Mínimos:

- **Grabar y Analizar Trazas.** Poder grabar trazas de las tramas que pasan por un interfaz bajo observación de la sonda y pueda ser **analizado desde una herramienta** para tal fin.
- **Analizar cualquier interfaz de cualquier router frontera del Backhaul IP.** Con estos interfaces que son la primera puerta de nivel 3, ya se podrán diagnosticar un gran número de problemas (sólo echaremos en falta, las tramas que se puedan perder por el camino en las últimas redes de nivel 2).
- **Operación remota.** La operación sobre la sonda de captura y la recuperación de datos de la misma, se debe poder hacer de forma remota desde cualquier terminal de cliente conectado a la intranet corporativa.
- **Estadísticas.** Posibilidad de hacer algún tipo de estudio estadístico de los paquetes que pasan por el interfaz durante el periodo de observación.
- **Coste económico próximo a cero.** Para el proyecto no se contempla habilitar ninguna partida de presupuesto de inversión para la compra de grandes equipos. Habrá que utilizar los equipos y materiales actualmente disponibles dentro del departamento (ordenadores, cableados y equipos de red corporativa).
- **Separación de tráfico de telecomunicaciones y tráfico de la red corporativa.** En ningún momento debe viajar por el mismo interfaz (salvo debidamente tunelizado) los dos tipos de tráfico mezclados.
- **Seguridad.** Aplicar las máximas garantías de seguridad para que no se permitan accesos de usuarios no autorizados o no autenticados al sistema.

Requisitos Deseables Adicionales:

- Poder pinchar **cualquier interfaz ethernet** con la sonda (no sólo el Backhaul).
- **Detalle de desglose/segmentación de estadísticas**:
  - Volumen de tramas/paquetes por: MAC origen/detino, IP origen/destino, DSCP, CoS, VLAN, Tamaño trama/paquete, Tipo (Uni/Multi/Broadcast).
  - Tramas erróneas o mal formadas.
  - Ancho de banda utilizado (kbps).
- Preconfiguración de **filtros de tráfico** para que la traza sólo sea del tráfico deseado.
- **Inyección de tráfico.** La sonda no es solo un elemento pasivo, sino que permite generar algún tipo de tráfico:
  - ICMP: Pings, traceroute
  - Generación de tráfico para evaluar caudal máximo de transferencia extremo a extremo.
  - Inyectar tramas pregrabadas/sintetizadas hacia la red.
- **Sin desplazamientos.** Sería deseable que todo el sistema no requiriera de la presencia física de personas (técnicos de campo) que se desplacen a emplazamientos remotos.
- **Operación remota encriptada.** Las operaciones remotas sobre la plataforma, aunque viajen sobre la intranet corporativa (o sobre internet por túnel seguro), no deberían ser interceptables por nadie conectado a la red.

3.3. Identificación y Valoración de Alternativas

Haciendo un preanálisis de los requisitos del cliente podemos ver 3 aspectos bien diferenciados que el cliente nos está pidiendo evaluar y proponer soluciones alternativas:

- **Herramientas** de captura de datos y análisis de interfaces ethernet.
- **Sistema base** donde se ejecutará dicha herramienta, y que tenga características de fiabilidad, seguridad y acceso remoto.
- Propuestas de **esquemas para pinchar sondas** en la red de telecomunicaciones.

A continuación vamos a hacer propuestas de solución para cada apartado y analizarlas someramente.

a) Herramientas de Captura y Análisis de Red

En este apartado vamos a hacer una relación de las herramientas de software libre disponibles en sistemas GNU/Linux, describiendo sus características principales. El análisis exhaustivo de cómo lo hace, se realizará en la siguiente fase del proyecto (Análisis).

Wireshark

Wireshark es una herramienta cuyo principal cometido es el analizar redes de paquetes [WireUS]. Captura los paquetes que pasan por interfaces y trata de mostrarlos con el mayor nivel de detalle posible. Entre sus características principales tenemos:

- Disponible tanto en sistemas tipo UNIX (GNU/Linux, Solaris, BSD, OS X) como Windows.
- Bajo licencia libre GNU GPL v2.
- Captura paquetes en tiempo real de cualquier interfaz de red (Ethernet 802.11, enlaces punto a punto PPP/HDLC, ATM, Bluetooth, USB, Frame Relay, FDDI, Token Ring,...).
- Analiza los paquetes y los muestra con todo detalle y correlándolos entre ellos según el protocolo que está utilizando. El listado de protocolos que es capaz de identificar y analizar es innumerable (sobre 100.000 filtros de protocolos [WireFil]) abarcando prácticamente todos los niveles OSI y todas las aplicaciones (incluida telefonía y redes de telecomunicaciones).
- Puede abrir y guardar trazas para análisis en tiempo diferido. De igual forma puede importar y exportar los datos para otros programas de análisis o formatos estándar (XML, CSV,...).
- Puede filtrar y buscar paquetes tanto en la captura en tiempo real, como en el proceso de análisis en tiempo diferido, en base a muchísimos parámetros y condiciones complejas.
- Interfaz gráfica y presentación de trazas intuitiva y que facilita el análisis. Dispone de una larga lista de herramientas para hacer seguimiento de hilos de conversación, secuencias de protocolos, volumen de información intercambiada y desencriptado de paquetes en protocolos seguros.
- Crea estadísticas en base a la captura, según una gran cantidad de parámetros.

- Posibilidad de crear scripts mediante el lenguaje de programación ligero Lua [WireLua].
- También dispone de una versión para terminales no gráficas, *tshark* y de una completa suite de utilidades para línea de comandos:
  - dumpcap: permite hacer volcados a fichero de capturas de paquetes en formato libpcap.
  - *capinfos*: printa información de una captura pregrabada.
  - *rawshark*: filtra un tubería de entrada -p.e. fichero con captura- volcando el resultado en la salida estándar.
  - *editcap*: edita y elimina paquetes de una captura.
  - *mergecap*: une varias capturas en un solo fichero.
  - *text2pcap*: convierte un fichero en formato hexadecimal ASCII a un formato de captura.
- Amplia documentación libre tanto oficial como de la comunidad.

Es sin duda el mejor analizador de protocolos del mundo del software libre (y posiblemente también del privativo). Pensamos que encajará perfectamente con casi todos los requerimientos de herramientas de este proyecto.

Entre las cosas que Wireshark declara no hacer está que no es sonda activa, es decir que no es capaz de enviar paquetes a la red. Tampoco genera alertas (p.e. No es un detector de intrusiones al sistema).

tcpdump/libpcap

tcpdump (desarrollada en 1987) fue la primera utilidad de software libre para realizar capturar paquetes de interfaces de red [tcpdump]. Actualmente sus características son:

- Se ejecuta desde la línea de comandos.
- Licencia libre BSD.
- Captura los paquetes y los muestra por la salida estándar formateados de muy diferente maneras (sólo hora/origen/destino, seleccionar campos extra a visualizar, volcado completo hexadecimal o ASCII del contenido del paquete,...).
- Permite hacer filtrado de sólo subconjuntos de paquetes que nos interesa capturar haciendo uso del eficiente filtro BPF (BSD Packet Filter) y con ello ayudando a aliviar la carga del sistema para procesar interfaces de red muy cargados [BPF].

Como base a tcpdump, fue desarrollada libpcap por los mismos desarrolladores. libpcap es una librería escrita en lenguaje C que facilita la captura de paquetes de interfaces de red, ofreciendo una API sencilla a los programas de análisis de red (tcpdump). La librería tiene también licencia BSD.

Prácticamente todos los programas importantes de análisis de red se basan en esta librería (incluído el propio Wireshark).

Por último, comentar que tcpdump/libpcap fueron desarrollados para entornos de tipo UNIX. Posteriormente fueron portados a entornos Windows, para los que cambiaron los nombres de las herramientas a WinDump/WinPCap.

Otros Analizadores y Monitorizadores

Existen muchas otras herramientas de análisis de red pero más enfocadas a *aspectos de seguridad* como alertas de intrusión en sistemas, detección de puertas abiertas y vulnerabilidades, descubrimiento y análisis remoto (no cooperativo) de equipos conectados a la red, etc.

En este proyecto, los equipos monitorizables no son equipos informáticos clásicos (servidores, máquinas cliente,...) sino equipos de telecomunicaciones con protocolos de transporte/aplicación muy específicos. También hemos de pensar que partimos de la premisa de que nuestros equipos se encuentran en una red privada teóricamente segura, no pública. Por tanto, parte de los objetivos de estas herramientas no tienen mucho sentido, porque están pensados para entornos distintos.

También existen otras herramientas más orientadas al campo de la *monitorización de rendimiento y buen estado de los sistemas*, que incluyen testeos y monitorizado del estado de los interfaces y servicios de red.

A continuación hacemos repaso de algunas de las herramientas más importantes en estos campos por si en posteriores fases pudiéramos encontrarles utilidad:

- ***Nmap***: Herramienta de escaneo que explora una máquina remota, detectando sus puertos abiertos, servicios de red disponibles, tipo de máquina y sistema operativo que lleva instalado. Licencia GNU GPLv2. Es una herramienta de línea de comandos pero también hay interfaces gráficos para ella.
- ***Snort***: Es un sistema de detección y prevención de intrusos a una máquina (*IDS/IPS*). También tiene licencia GNU GPLv2. Básicamente se comporta como un *sniffer* que monitoriza los interfaces de red y sobre él se definen unas reglas que detectan situaciones que pueden *disparar acciones automáticamente*. Existe toda una biblioteca de reglas complejas que el usuario puede aplicar, que intentan detectar los ataques e intentos de intrusión por red más habituales y ante los cuales la máquina puede tomar acciones programables (enviar avisos al administrador, ejecutar bloqueos automáticos,...).
- ***Netcat***: Herramienta de línea de comandos que puede leer/escribir en puertos TCP/UDP. Tiene licencia libre permisiva sin restricciones (similar a la BSD).
- ***Kismet***: Sniffer capaz de detectar redes y ataques de intrusión, especializado en redes inalámbricas Wifi. Licencia GNU GPL.
- ***Ettercap***: Herramienta interceptadora/capturadora de paquetes que tiene como objetivo registrar información sensible como parejas usuario/contraseña. Tiene métodos de interceptación muy complejos, capaces de inyectar tráfico (técnicas “man-in-the-middle”) y pudiendo bajo ciertas condiciones acceder a protocolos “seguros” (SSH/HTTPS). También tiene licencia GNU GPL.
- ***Munin***: Es una completa aplicación para monitorizar el funcionamiento de una máquina. Básicamente monta un servidor Web, al que se puede acceder para comprobar el estado y estadísticas del sistema (carga de CPU, memoria, procesos, estado de servicios como MySQL,...). Tiene un apartado dedicado a monitorizar el estado de la red que muestra estadísticas de uso bastante completas. También puede generar alertas que se pueden enviar por correo o a otros sistemas/máquinas. Se distribuye bajo licencia GNU GPL.

- ***MRTG***: Herramienta capaz de recopilar información periódicamente y generar informes en formato HTML con gráficas de uso. Está pensada para recoger información de equipos remotos a través del protocolo SNMP (por ejemplo, volumen de tráfico cursado por routers remotos). Tiene licencia GNU GPL.
- ***Nagios***: Es una de las más importantes plataformas de monitorización de toda la infraestructura de IT de una organización. Se monitoriza desde funcionamiento de todas las máquinas y equipos (carga, condiciones ambientales, alarmas externas) hasta los servicios de red (HTTP, SMTP, POP3, SNMP, FTP, SSH,...). Básicamente habrá un servidor y en el resto de máquinas remotas se instala el Nagios Remote Plugin Executor. Se pueden consultar el estado de todas las máquinas, almacenar históricos, crear informes y generar alarmas que se envíen por e-mail o SMSs. También su licencia es GNU GPL.

b) Esquemas de Conexionado de Sondas

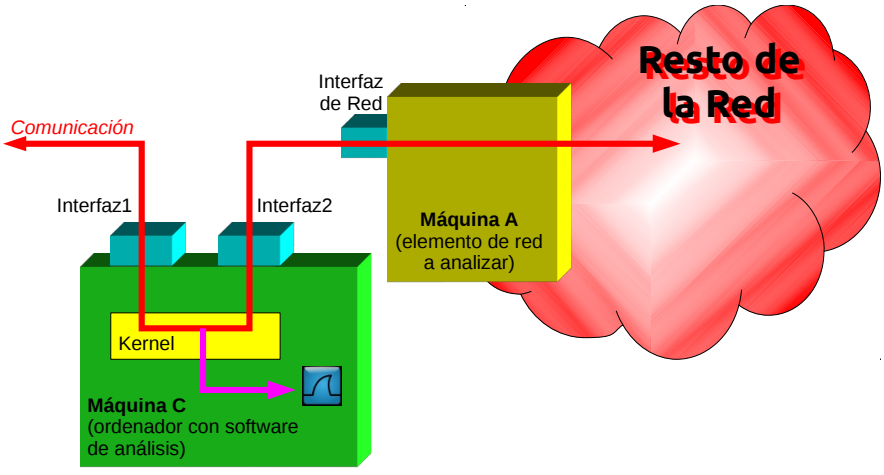
Las herramientas descritas se encargan de analizar lo que pasa por los interfaces de red de la máquina donde están instaladas. Pero en nuestro caso, lo que se requiere es analizar lo que entra o sale de una tercera máquina (llamaremos *máquina A*).

La *máquina A* podría ser cualquiera de los elementos de la red de telecomunicaciones que describimos en el apartado de “Situación Actual”. En la *máquina A* no podremos instalar o alterar software, puesto que no son equipos informáticos. Por lo tanto tendremos que hacer llevar el interfaz de esa *máquina A* a una tercera máquina que actuará de sonda (llamaremos *máquina C*).

A priori podemos plantear diferentes alternativas para el conexionado de la sonda:

Paso a Través

Este modelo lo podemos representar en el siguiente esquema:



Es decir, abrimos el lazo de la comunicación para insertar en medio nuestra máquina mediante dos interfaces de red. El kernel del sistema operativo debería permitirnos configurar en modo switch repetidor las comunicaciones entre ambos interfaces, para que no se rompan y para la máquina A, todo sea transparente.

Este esquema tiene varios problemas:

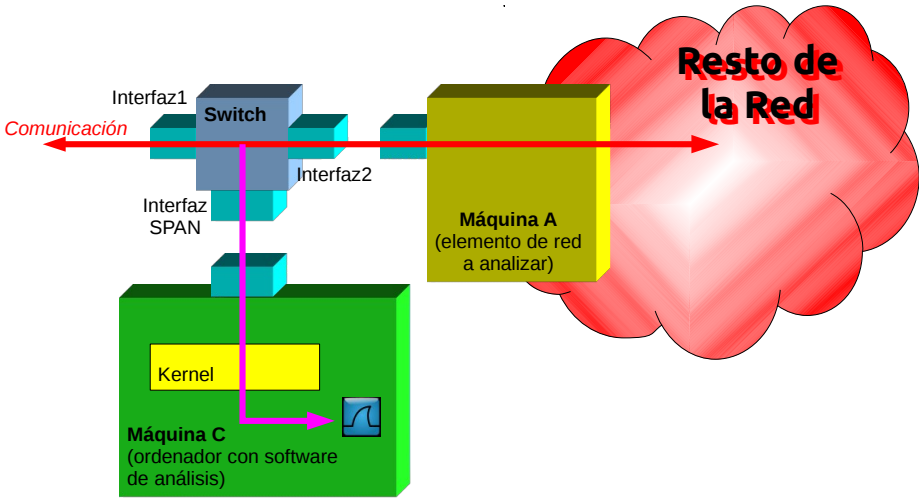
- Es *disruptivo*, puesto que para insertar la sonda hemos de cortar la comunicación temporalmente.
- Requiere de *dos interfaces de red dedicados a la monitorización* (si además queremos que la máquina C tenga salida hacia la red corporativa, necesitaríamos un tercero!!).
- *Fiabilidad*. Si a la máquina C le ocurre algo, afectará al servicio (la comunicación se corta hacia la máquina A).
- Requiere de un *desplazamiento* a conectar la máquina C a un punto remoto.

Port Mirroring

Para intentar solventar algunos de los problemas del esquema anterior, podemos en vez de utilizar el kernel del sistema y varios interfaces, intercalar otro equipo que permita redirigir las comunicaciones hacia la *máquina C*, de forma transparente a la máquina A.

Este equipo podría ser un *hub* (retransmite todas las tramas a todos los puertos), pero hoy en día es un equipo difícil de encontrar comercialmente. Es más sencillo encontrar equipos *switch* con la funcionalidad *Port Mirroring* (poder desviar todo hacia un puerto de forma configurable) o que tuviera preconfigurado un puerto para tal fin (suele llamarse *puerto SPAN* en terminología Cisco -Switched Port Analyzer-).

El esquema sería:



Con esto solucionamos el que *la máquina C no requiere 2 interfaces de red*, sino uno sólo.

También evitamos que el tráfico tenga que depender del buen funcionamiento de la *máquina C* (los *switches* a priori parecen más estables por ser equipos hardware pensados para este cometido).

Pero seguimos teniendo el problema del *cambio disruptivo* (insertar el switch) y el que alguien se tenga que *desplazar* a pinchar la máquina.

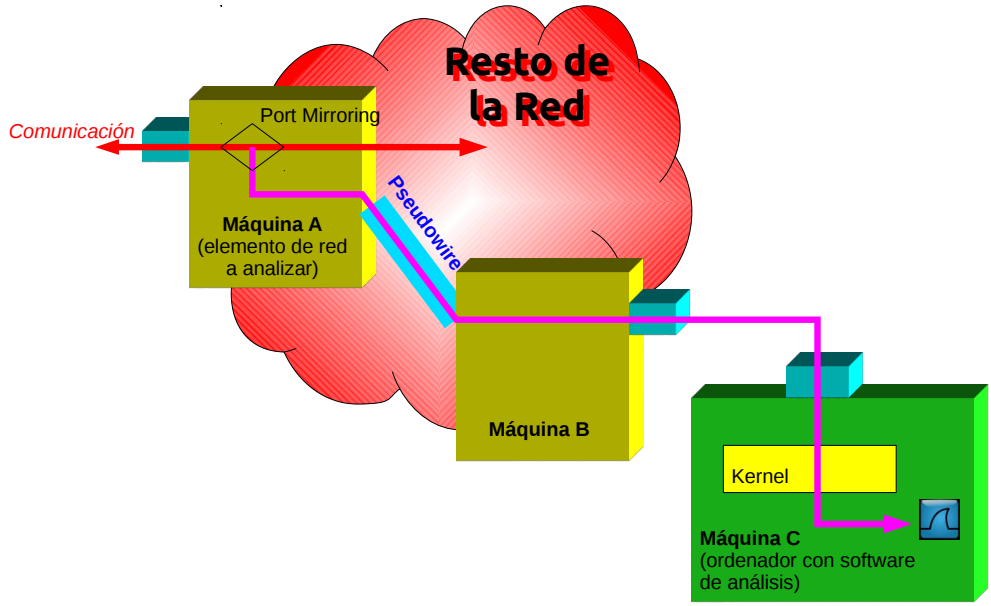
En muchas situaciones no tendremos más opciones, y no descartamos este esquema de conexionado.

Remote Port Mirroring

Por último, existe una tercera posibilidad y es aprovechar que la máquina A se encuentra conectada a la red, si existiera la posibilidad de redireccionar el port mirroring hacia otro punto de esa misma red, solventaríamos todos los problemas.

Esta funcionalidad está disponible en los equipos integrantes del *Backhaul IP*.

El esquema sería el siguiente:



Se solucionarían los problemas del cambio *disruptivo* (sólo sería un cambio de configuración software en las *máquinas A y B*, que no afectarían a la comunicación).

También se solucionarían los problemas de los *desplazamientos* a conectar físicamente la *Máquina C*, puesto que ésta puede estar en un centro de trabajo (p.e. Centro de Conmutación). Esta ubicación tiene las ventajas de que podríamos tener una máquina más *estable*, las *24 horas* funcionando y con *conexión directa a la red corporativa* (en las otras configuraciones, si queremos operar remotamente la *máquina C* requeríamos una conexión securizada a través de internet).

Sin embargo aparecen algunas desventajas/riesgos:

- *Sólo se podrían monitorizar interfaces de la red de Backhaul*. Esto casa con los requisitos mínimos del cliente, puesto que esta red ya tiene bastante capilaridad.
- *Se duplicaría el caudal de tráfico de salida de la máquina A hacia la red*, puesto que a la comunicación en sí, habría que añadir el tráfico que pasa por el pseudowire, que es exactamente lo mismo. En general, esto no es problema porque muchas de estas salidas suelen ser fibras con 1/10 Gbps. Sin embargo, en algunos casos puede que si la salida sea un radioenlace de baja capacidad, habrá que vigilar que la configuración del Mirroring no reduzca el ancho de banda disponible para las comunicaciones de los clientes.
- El port mirroring, aunque declarado por el fabricante, *todavía no se ha utilizado/probado masivamente en los equipos de red* y puede llevarnos alguna sorpresa (requiere de unas pruebas y análisis en profundidad).

c) Sistema Base

Sistema Operativo

Las herramientas que describimos están pensadas para la ejecución en sistemas informáticos (la llamada *máquina C*). También será parte del proyecto la elección de la plataforma base informática.

Ya hemos comentado que en la actualidad prácticamente la totalidad de los equipos dentro de la compañía son del tipo ***Windows***. Aunque muchas de las herramientas tienen su versión para *Windows*, pensamos que no es un entorno adecuado porque:

- *No tiene la facilidad y flexibilidad de gestión de tráfico de red* que posee GNU/Linux, donde ésta ya se hace desde el núcleo del sistema operativo (*iptables*).
- *No tiene la ligereza de carga* que tiene el núcleo de GNU/Linux que puede ser ejecutado en máquinas antiguas o de pocos recursos sin muchos problemas.
- Por experiencia, *no tiene ni la estabilidad ni nivel de seguridad de acceso* (local o remota) deseables.

Por ello, atendiendo a los requisitos de acceso remoto, seguridad y coste económico bajo, pensaremos en una plataforma basada en GNU/Linux.

El siguiente paso será *elegir la distribución* que consideremos mejor. Este es un aspecto bastante delicado, puesto que aunque en esencia todas llevan lo mismo, sí que van a *diferir en cuanto modo de configuración, aplicaciones disponibles, ligereza del sistema de partida, entorno gráfico etc etc*.

Lo primero sería elegir una distribución que disponga del *mayor repositorio de aplicaciones* directamente instalable y con una *documentación gratuita más extensa*. Así pues nos centraremos en las 2 familias de distribuciones más implantadas, que son las distribuciones basadas en **Fedora** y en **Debian**.

Es difícil elegir una familia. Quizás Fedora esté más orientado a funcionalidades de seguridad para entornos profesionales y Debian más a máquinas cliente con una configuración algo más sencilla. Sin embargo, dado que en las que *tenemos mayor experiencia son las basadas en Debian* nos centraremos en dicha familia.

Podríamos escoger directamente la distribución Debian. Sin embargo, pensamos que **Ubuntu** al ser una distribución basada en Debian (incorpora prácticamente todos sus paquetes y su filosofía de configuración) le da el punto a favor adicional de que *incluye muchos más paquetes para configurar hardware específico* (incluso con controladores propietarios).

Por último, siguiendo en el razonamiento, aunque Ubuntu es una distribución muy buena (de hecho es la más extendida con diferencia), para nuestros propósitos quizás adolece de un problema y es el *exceso de carga y recursos que necesita para mover tanto el entorno Gnome como su versión KDE* (Kubuntu). Sin querer abandonar las virtudes que tiene Ubuntu, pero buscando un sistema de base más ligero, nos decidimos por la versión ligera de Ubuntu: **Lubuntu**.

Lubuntu hasta su versión 11.10 era una versión no oficial del conglomerado bajo responsabilidad de Canonical (gestor de Ubuntu). Pero ésta a partir de octubre del 2011, ya entra dentro de la familia de distribuciones xxxUbuntu con las mismas garantías. No existen grandes diferencias en su manera de funcionar, ya que tiene el *mismo repositorio de programas que Ubuntu*, pero de partida elige

unos *componentes más ligeros* que la hacen necesitar muchos menos recursos:

- *LXDE* en lugar de Gnome
- *Abiword/Gnumeric/Powerdot* en lugar de la suite *LibreOffice*
- *Chromium* en lugar de *Firefox*
- *Mplayer* en lugar de *Totem*.

Lógicamente, son programas de aspecto visual algo inferior y con menos características, pero totalmente operativas para un uso normal. Aún así, si queremos las versiones pesadas o cualquier otro programa de Ubuntu, podremos instalarlo desde el habitual *Synaptic*.

En nuestro proyecto no requeriremos de estas características visuales aunque sí que queremos que estén disponibles e instalables herramientas como *Wireshark*. Por ello **elegiremos Lubuntu**.

Aquí tenemos un resumen comparativo de las alternativas analizadas:

Sistema	Ventajas	Desventajas
Windows	-Ya instalado en todas las máquinas	-Poca flexibilidad y facilidad para manejar tráfico de red -Poca estabilidad y seguridad -No ligero para mover programas, incluso en una instalación reciente.
Fedora	-Excelente distribución (repositorio de programas y distribución) -Orientada a entornos profesionales y seguros con muchas funcionalidades avanzadas	- Tenemos menos experiencia con ella
Debian	-Excelente distribución (repositorio de programas y distribución) -Sencilla configuración	- Menor disponibilidad de controladores de Hardware
Ubuntu	-Excelente distribución (repositorio de programas y distribución) basada en Debian -Excelente disponibilidad de controladores HW -Muy fácil de configurar	- Alta necesidad de recursos para la distribución base (Gnome)
Lubuntu	-Excelente distribución (repositorio de programas y distribución) basada en Ubuntu/Debian -Excelente disponibilidad de controladores HW -Muy fácil de configurar -Muy ligera y requiere muy pocos recursos	-Aplicaciones base ligeras pero con menores características visuales y de funcionalidad

Plataforma Hardware

Ahora debemos decidir sobre qué máquinas vamos a instalar el sistema operativo con las herramientas indicadas.

Atendiendo al *tipo de conexionado de sonda*s, parece que la solución que adoptaremos en el prototipo objeto de este proyecto es el *Remote Port Mirroring*. Sin embargo, como aún es una funcionalidad del *Backhaul* no probada y además no nos sirve para las redes de nivel 2 más allá del *Backhaul*, no podemos descartar la opción del *Port Mirroring* in situ.

Intentaremos pues, *dar una solución para ambas configuraciones* y que la misma máquina y configuración nos pueda servir para los dos casos.

Para el *Remote Port Mirroring* tendremos una *máquina dedicada* para tal fin, permanentemente conectada a un puerto de un equipo del Backhaul. Pero para el *Port Mirroring*, exigirá el desplazamiento de una máquina portátil al lugar donde queremos monitorizar. La solución más inmediata es utilizar la *máquina portátil de los técnicos de campo* que se desplazan a los emplazamientos.

Estas máquinas, como hemos dicho tienen preinstalado *Windows*. Por lo tanto habría que instalar de alguna manera Lubuntu+Herramientas. Podemos:

- **Redimensionar la partición Windows**, crear una nueva partición para GNU/Linux y con un *dual boot* el operario decide qué arrancar.
- Crear una **máquina virtual** dentro de Windows.

El *dual boot*, obviamente nos daría un resultado mucho más estable y sin bajada de performance. Pero nos ocasiona varios inconvenientes:

- *No tendríamos las herramientas de Windows* habituales.
- *No tendríamos ni la aplicación ni la configuración para el acceso a la red corporativa* a través de internet mediante túneles (VPN). Esta aplicación creemos que es difícil de replicar en un sistema GNU/Linux sin la colaboración del departamento de IT.
- Como cada portátil tiene un hardware diferente, *para cada máquina habría que hacer un proceso de instalación de todo el sistema operativo, herramientas y configurarlo*.

En una solución sobre *máquina virtual*:

- Replicando el fichero de imagen de disco duro virtual, podemos tener nuestra máquina operativa en cualquier ordenador. *Sólo instalaremos y configuraremos una vez*.
- *Se reducirá el rendimiento y habrá que hacer pruebas de perfomance*. Pero dado que ya estamos eligiendo una distribución ligera (Lubuntu), no esperamos tener grandes problemas.
- Para que pueda haber conexión remota desde la red corporativa o realizar otras tareas, el operario de campo tendrá *disponible todo el entorno habitual Windows*.

Para probar esta configuración y que pueda ser extensible a futuro rápidamente, *aunque no sea necesario en un prototipo de configuración de conexionado en modo Remote Port Mirroring,* también vamos a realizar una virtualización. En este caso, como será una máquina fija dedicada y conectada a la red corporativa mediante LAN no será necesario *Windows* y como sistema anfitrión usaremos otra distribución GNU/Linux (por performance y no diversificar elegimos **Lubuntu** nuevamente).

En cuanto al software de virtualización, existen muchas soluciones libres (*Qemu*, *KVM*, *Xen*,...) pero nos decantamos por **VirtualBox OSE** por:

- Sencillez de configuración.
- Permite configurar gran cantidad de dispositivos, y modos de red virtual.
- Tenemos software tanto para GNU/Linux como para Windows.
- Eficiencia de virtualización (especialmente con la aceleración hardware).
- Permite la operación remota de la máquina virtual a través de protocolos VNC/VRDP.
- Licencia GNU GPL v2.

Acceso Remoto

Sólo nos queda un último requisito del que no hemos hablado, y es el acceso remoto a la máquina donde están las herramientas de análisis, desde los terminales de usuario de los técnicos de operación.

Al margen de los accesos remotos a consolas de texto clásicas (por ejemplo usando el protocolo SSH, en su versión *OpenSSH* para Linux), estamos pensando inicialmente en *herramientas que nos permitan manipular gráficamente la máquina C desde un punto remoto*.

Existen una gran cantidad de soluciones de los llamados "escritorios remotos" disponibles en las distribuciones GNU/Linux. Según el protocolo que utiliza tenemos:

- **VNC:** Es un protocolo de compartición de escritorio que se basa en el envío de un streamming de vídeo al cliente desde el servidor y de envío de eventos de teclado/ratón en sentido inverso. Existen varios productos con licencia GNU GPL disponible, como son *TightVNC*, *UltraVNC*, *RealVNC* o *FreeNX*.
- **RDP:** Es un protocolo similar al VNC pero algo más elaborado porque es semántico, teniendo en cuenta el conocimiento de controles, textos, fuentes, y demás elementos gráficos. Esto hace que no haya que transmitir tanta información de vídeo y sea algo más eficiente. Existen también algunas soluciones con licencia GNU GPL como *rdesktop* o *xrdp*.
- **X11 redirigido a red:** El sistema de ventanas X de GNU/Linux tiene una arquitectura cliente-servidor (máquina donde se ejecutan las aplicaciones frente a máquina con el interfaz físico de interacción humana). En el funcionamiento habitual, ambos procesos están en la misma máquina local, pero también dicha comunicación puede ser redirigida a un puerto de la red (*X11 forwarding*), permitiendo acceder a máquinas remotas fácilmente. Suele tener un peor performance en ciertas circunstancias, pero es sencillo y simple.

De los 3 tipos de protocolos tenemos *clientes para plataformas Windows* (para ejecutar desde los terminales de los técnicos).

También de los 3 tipos de tenemos *soluciones de autenticación para evitar accesos no autorizados y de encriptado* (p.e. a través de túneles SSH).

A priori, no vemos una clara solución mejor, puesto que son similares. Deberemos de concretarlo en posteriores fases de análisis, comparando el performance y las opciones de seguridad que nos brindan en el marco de nuestra implementación particular.

3.4. Selección de la Solución

Aunque aún queda la F*ase de Análisis,* donde podremos vernos en la necesidad de cambiar algunas decisiones, para cumplir con los requisitos del cliente y teniendo en cuenta que buscamos un prototipo (que en futuras etapas pueda ser extensible a otros ámbitos) proponemos esta solución:

- Físicamente, necesitaremos una **máquina fija conectada a un puerto de un elemento concreto del Backhaul-IP** (preferentemente en el Centro de Conmutación) mediante un interfaz *Fast Ethernet.*
- Esta máquina tendrá otro interfaz *Fast Ethernet* **conectado a la red local corporativa**.
- Se configurará **el *Backhaul,* en modo Remote Port Mirroring** para poder monitorizar cualquier interfaz del *Backhaul*. Sólo se podrá monitorizar un puerto a la vez y requerirá de que una reconfiguración (*redireccionado*) por parte del técnico de operaciones.
- Dicha máquina será una **máquina dedicada a este proyecto** (el protipo) con una **distribución GNU/Linux ligera (Lubuntu)**. No será necesario comprar una nueva máquina y se podrá reutilizar alguna antigua disponible en el departamento que esté en desuso.
- En ella se instalará y configurará **VirtualBox OSE** para crear una máquina virtual. Esta **máquina virtual también será Lubuntu**. La máquina virtual tendrá acceso al interfaz de red que mira a la red de telecomunicaciones.
- Bien en la máquina host o guest o ambas, **se instalará una solución de acceso remoto (VNC, RDP o X11 forwarding)** a decidir según performance. Este acceso será *autenticado y encriptado a través de la intranet corporativa* y disponible desde los terminales de los técnicos de operaciones.
- En la máquina virtual **se instalará y configurará Wireshark como herramienta de análisis**. Si tras las pruebas no se cumplieran los requisitos del cliente, *se completaría con otros paquetes de software descritos (o desarrollos propios)*.
- Esta **máquina virtual totalmente configurada, debe ser exportable** (copia del disco duro virtual) para que se *pueda ejecutar en cualquier terminal Windows* de los técnicos de campo y pueda funcionar igualmente desde ella con un conexionado de *Port Mirroring* in situ.

Con esto, pensamos que también cumplimos con el requisito de *"Coste económico próximo a cero"* puesto que no se comprará ningún material (el hardware será reutilizado o virtualizado) o licencia (software libre).

4. Análisis del Sistema

En el sistema propuesto como solución quedaban algunos puntos por detallar. Esta fase del proyecto es fundamental para completar el documento de referencia sobre el que se basará el diseño concreto de nuestro prototipo. Como comentamos en la descripción del proyecto, *en esta fase de Análisis debemos tener en consideración la futura extensión de este prototipo a un uso masivo en toda la compañía (no sólo el prototipo)*.

4.1. Definición exacta del sistema y requisitos adicionales

A continuación haremos un repaso de algunos detalles que quedaron fuera del estudio de viabilidad y que son necesarios para tomar decisiones finales de diseño:

a) Aspectos de seguridad

En la medida de lo posible vamos a limitar los accesos a cada una de las máquinas, maximizaremos la seguridad de las conexiones y separaremos tráfico de red corporativa del de telecomunicaciones.

Esto en nuestro sistema propuesto (máquina anfitriona-huésped mediante *Virtualbox*) lo podemos traducir haciendo uso de las funcionalidades disponibles:

Máquina Virtual Huésped

Aquí es donde correrá la aplicación de análisis de red (Wireshark). Lógicamente esta máquina deberá poder acceder de forma transparente a la red de telecomunicaciones (modo bridge y promiscuo de la configuración en Virtualbox).

No es necesario que tenga acceso a la red corporativa ya que gracias a Virtualbox podemos solventar las necesidades de:

- Control remoto: funcionalidades de escritorio remoto de Virtualbox (sólo la máquina anfitriona ha de exponerse a la red corporativa).
- Intercambio de ficheros (p.e. trazas): funcionalidades de carpeta compartida de Virtualbox. Anfitrión y huésped podrán intercambiar ficheros a través de esa carpeta. El acceso externo (desde red) a esos ficheros se podrá hacer mediante algún FTP/NFS servidor en el anfitrión.

En cuanto a accesos, habrá que crear un único usuario/contraseña que deberán conocer y compartir todos los usuarios. Como tendremos copias de seguridad de la máquina (fichero con disco duro virtual) y el acceso a red es límitado, no preveemos más necesidades de usuarios distintos que puedan ser traceables. Si el prototipo se extendiera en un futuro, esta política cambiaría.

Por último, como no hay acceso a internet, desactivaremos las actualizaciones de software automáticas.

Máquina Anfitriona

En el caso de que ésta sea un terminal Windows corporativo, los aspectos de seguridad (usuarios/políticas de acceso, firewall, antivirus,...) son manejados por el departamento de TI de la compañía y sobre los que no actuaremos.

En el caso de una máquina dedicada (el prototipo) donde instalaremos un sistema GNU/Linux base, a ésta aplicaremos unas políticas muy restrictivas de acceso para que nadie (a excepción del

administrador) pueda controlarla ni reconfigurarla:

- Acceso remoto únicamente mediante protocolo seguro SSH. No habrá cuentas de usuario, sólo una de administración. A pesar de estar en entorno seguro (red corporativa) se encriptarán las comunicaciones para no ser interceptada la password del administrador.
- Un puerto abierto de escritorio remoto, pero que visualizará el escritorio de la máquina huésped. Este acceso será protegido por contraseña. Como estamos en un entorno de red corporativa segura, como requisito mínimo de cliente no será necesario (a priori) encriptar esta comunicación. En requisitos máximos sí se especificaba, así pues durante el diseño se estudiará su conveniencia, y se podría tunelizar esta comunicación.
- Acceso por FTP/NFS únicamente a una carpeta compartida con el húesped. También protegido con contraseña (aunque por el mismo motivo, a priori no encriptaremos).
- Se desactivarán todas las opciones de actualización automática.

b) Análisis comparativo de escritorios remotos

Ya en el apartado anterior nos hemos puesto un condicionante a los escritorios remotos: ***la máquina virtual no tendrá el software de escritorio remoto corriendo porque no verá la red corporativa***.

Esto aparte de tener unas ventajas de aislamiento de tráfico y de seguridad, tiene otra ventaja y es el performance. Si instaláramos algún software de acceso remoto en el huésped, éste tendría que correr virtualizado. Aprovechándonos de las facilidades de escritorios remotos de máquinas virtualizadas en Virtualbox, este software correrá en la máquina anfitriona físicamente real (más rápida).

Sin embargo, esto tiene la desventaja que nos ata a lo que nos ofrezca Virtualbox. Concretamente tenemos dos tipos de acceso:

- **VRDP** (Virtualbox Remote Desktop Protocol). Es un sistema del tipo RDP (ver PAC anterior) que funciona muy bien, con buenas características de seguridad de acceso controlado y performance. Pero tiene un problema y es que no viene con la versión de *Virtualbox OSE* (*Open Source Edition*, con licencia GNU/GPL). Para poder usarla habría que instalar un *"extension pack"* con muchas funcionalidades pero bajo licencia no libre (licencia *PUEL*, Personal Use and Evaluation License [PUEL]).
- **VNC**. En la versión OSE, se encuentra la funcionalidad de acceso VNC mantenida por la comunidad. Esta opción sólo está disponible para máquinas virtuales arrancadas desde línea de comandos con *VBoxHeadless*. VNC es un protocolo algo peor en cuanto a performance y en el módulo de Vbox realmente dan muy pocas opciones de configuración (password y puerto). Además en las pruebas que hemos hecho, detectamos que hay un *bug que hace que a veces se pierda alguna pulsación de teclado* (especialmente al inicio de sesión). Sin embargo, es nuestra única opción libre, así que será prácticamente una obligación.

En cuanto a la máquina anfitriona (en el caso de ser la GNU/Linux dedicada), ya hemos dicho que sólo tendremos acceso vía terminal SSH. Es más, posiblemente esta máquina si el performance lo requiere no tenga ni interfaz gráfico. Aún así, por si fuera necesario para facilitar tareas de administración, podríamos pensar en instalar algún escritorio remoto tunelizado a través del SSH. Dado que optamos por la distribución *Lubuntu* (sin GNOME ni KDE) nos parece una buena idea que en caso de necesidad se pueda hacer *X11 forwarding* o mejor aún, instalar el paquete *x11vnc* y ejecutarlo puntualmente bajo demanda del administrador).

c) Cumplimiento de requisitos funcionales de Wireshark

Aunque durante la etapa de viabilidad claramente nos decantamos por Wireshark como herramienta de análisis de red, nos quedaba hacer un análisis de si cumple todos los requisitos del cliente y cómo lo hace, para detectar carencias (previamente a la etapa de diseño) y plantearnos hacer desarrollos de software (o scripts) a medida.

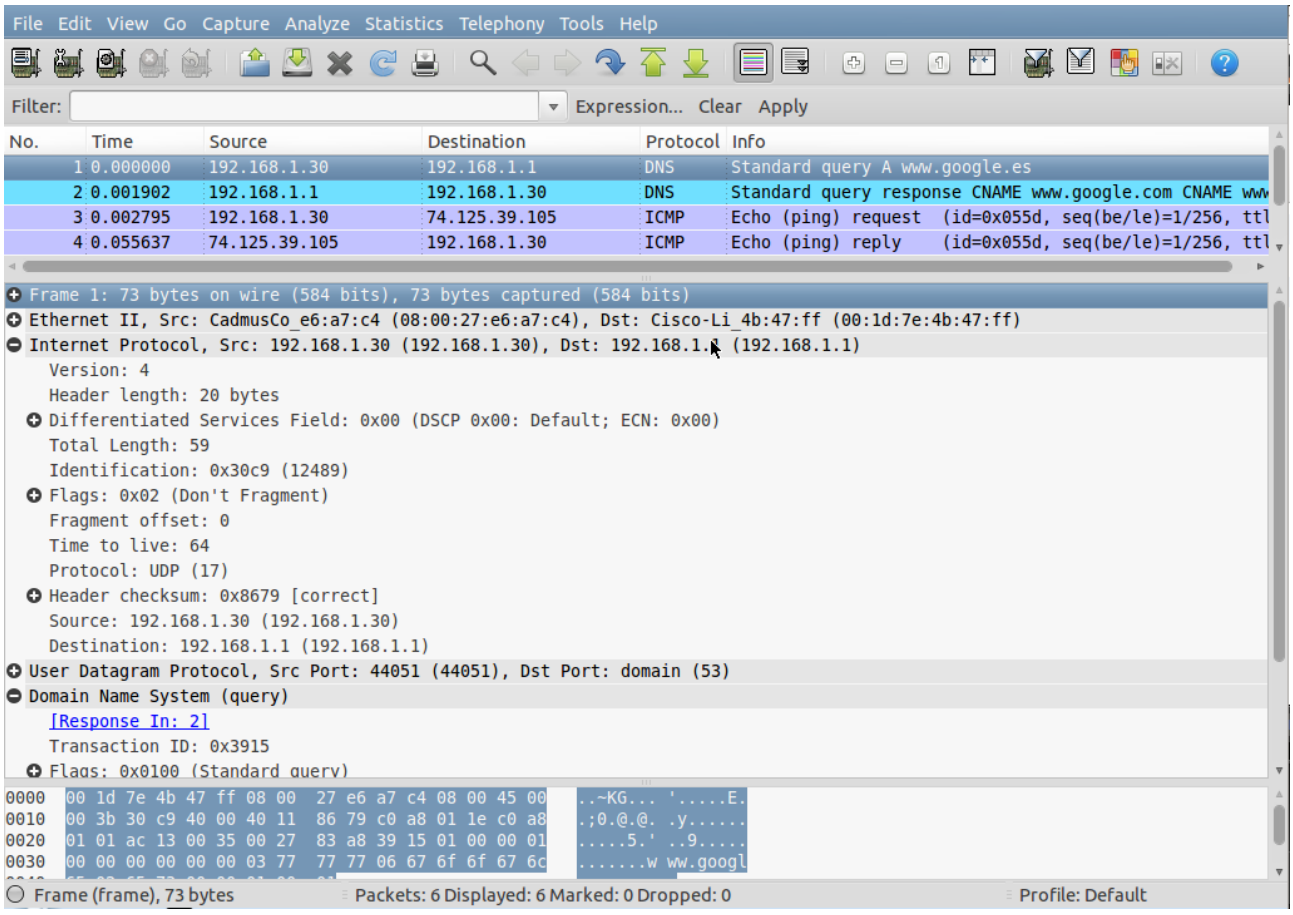
Repasemos los requisitos del cliente que aplican a la herramienta:

- **Grabar y Analizar Trazas** (requisito mínimo)

Wireshark fue creado para tal fin. Permite la captura de cualquier interfaz de red (y USB). Para ello, por tema de permisos tendremos que habilitar de alguna forma la ejecución de la captura como "root" del sistema (veremos en etapa de diseño). Una vez arrancada la aplicación, en "Capture-Interfaces" seleccionamos el interfaz de red a capturar.

Es importante que en opciones del interfaz habilitemos la opción "*Capture packets in promiscuous mode*" para que capture todos los paquetes y no sólo los que son originantes/destinatarios al interfaz de red de nuestra máquina. Existe una amplia lista de opciones adicionales a la captura (filtrado, resolución de direcciones, actualización en pantalla en tiempo real,...)

Para el análisis, tendremos la siguiente pantalla principal (ejemplo de ping a *www.google.es*):



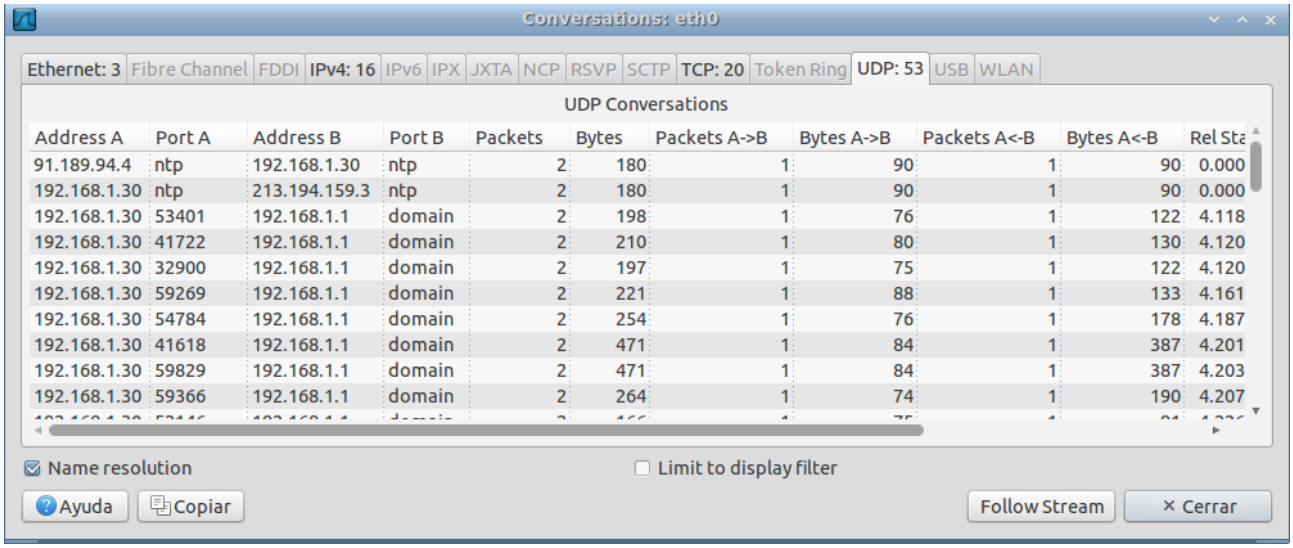
Vemos 3 áreas bien diferenciadas:

1. **Lista de paquetes capturados**, especificando IP origen, destino, tipo de protocolo (el mayor nivel OSI) y la información principal asociada.
2. Para el paquete seleccionado, se hace un **desglose de protocolo nivel a nivel OSI** (trama física, trama Ethernet nivel enlace de datos, paquete IP de nivel de red, paquete TCP/UDP de nivel de transporte,...). En cada desglose se pueden ver las cabeceras decodificadas en un formato amigable. La estructura es en árbol, para poder ir al desglose que nos interese en cada momento.
3. Abajo del todo tenemos el **paquete en bruto** seleccionado en modo hexadecimal y ASCII.

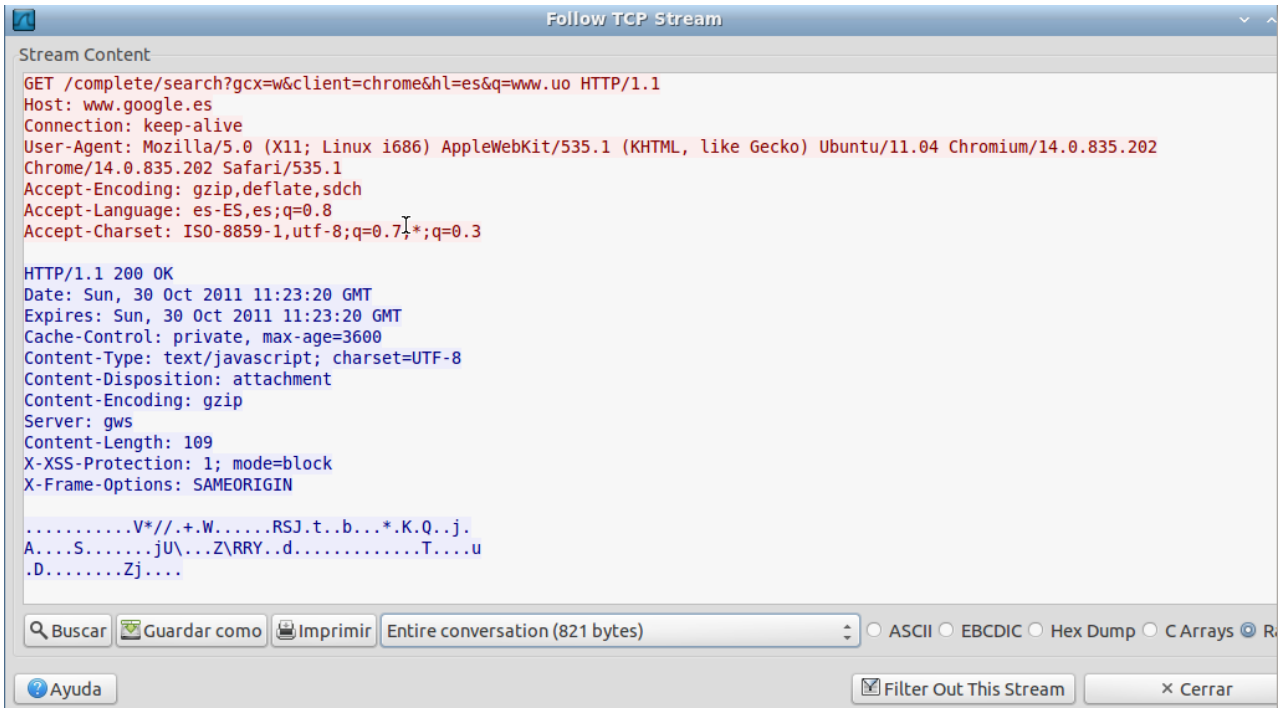
Sobre esta pantalla se pueden hacer filtrados de paquetes por múltiples opciones para poder seleccionar únicamente la información de la comunicación que nos interesa. Con ello, gran parte de los problemas de *troubleshooting* podrán ser analizados fácilmente.

También existen numerosas herramientas de análisis más complejo de las que destacamos por su utilidad para nuestras necesidades:

Análisis de conversaciones capturadas. Analiza las diferentes tríada origen/destino/protocolo capturadas:

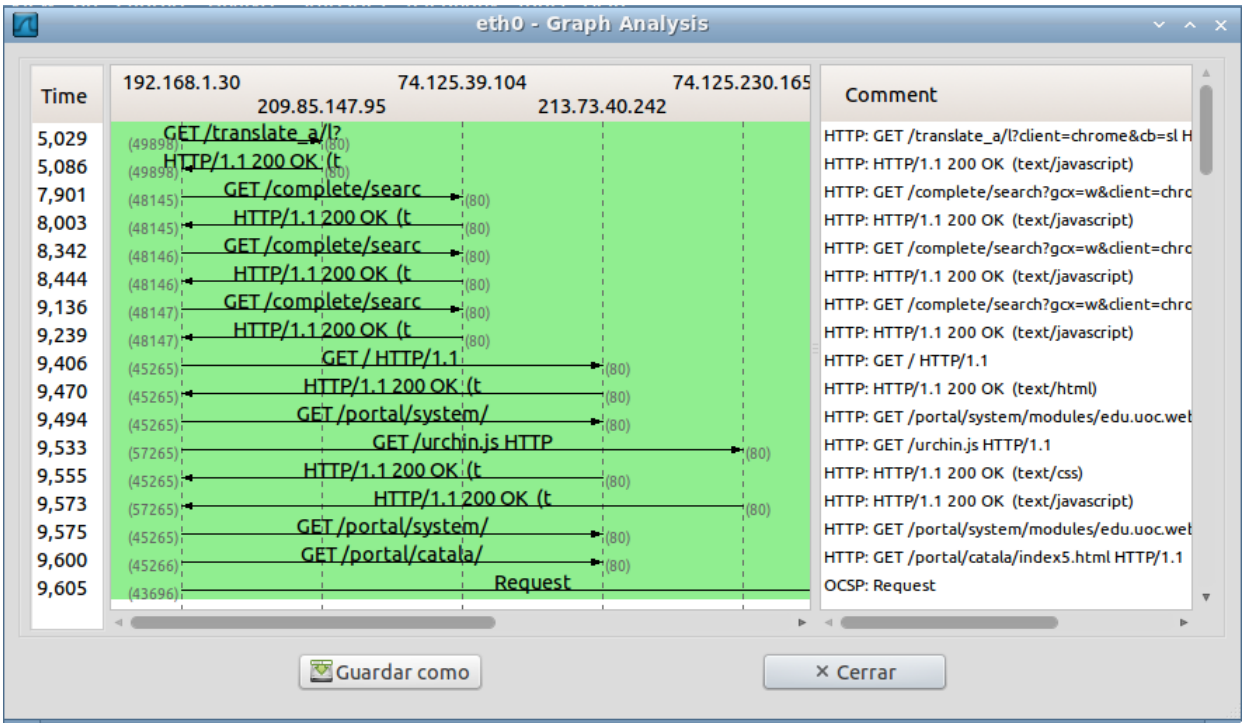


Análisis de Streams. Para cada conversación (p.e. TCP), Wireshark puede automáticamente hacer un filtrado de paquetes y centrar el análisis en únicamente dicha conversación. En la pantalla anterior, si seleccionamos una conversación http y pinchamos en el botón “Follow Streams” se filtran los siguiente paquetes:



Análisis gráfico de intercambio de paquetes y secuencia de protocolo. Si lo anterior no nos aporta la información muy claramente, podemos filtrar lo que deseamos y hacer una representación gráfica secuencial del protocolo seguido. Desde una simple resolución de ARP:

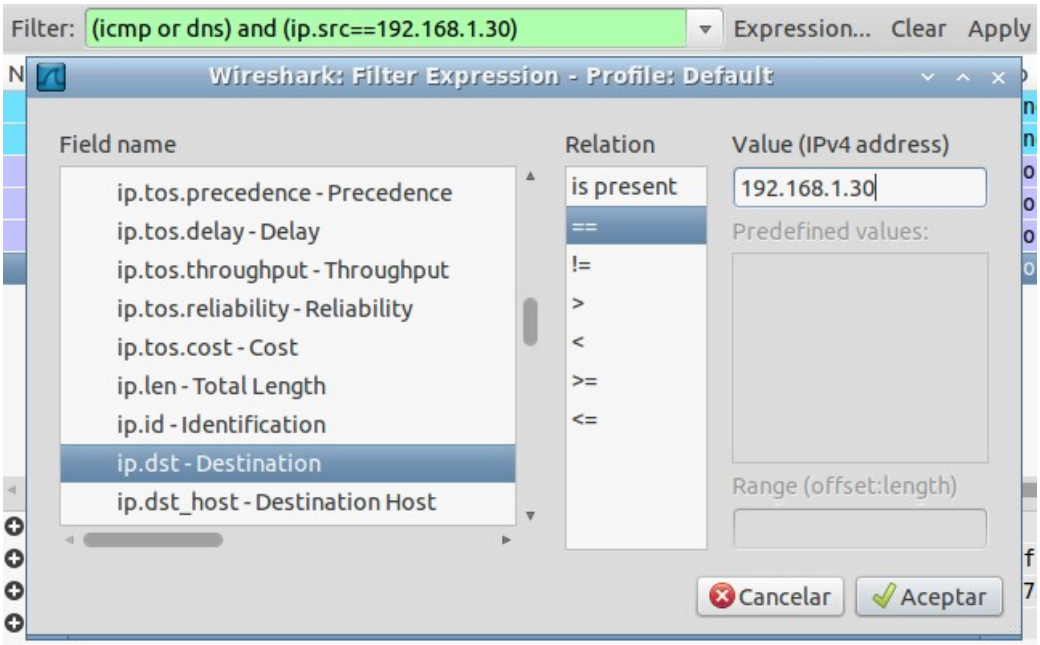
Hasta una compleja descarga de página web (ejemplo web de www.uoc.edu):



Existen adicionalmente multitud de herramientas para aplicaciones específicas (para servidores HTTP, SMB,...). De entre ellas vemos un menú para aplicaciones de telefonía que incluye análisis de protocolos específicos como de SCTP, ISUP, MTP3, CAMEL,... Cuando empecemos a capturar trazas reales de red, estudiaremos la utilidad de estas herramientas.

- **Filtros de tráfico** (requisito deseable)

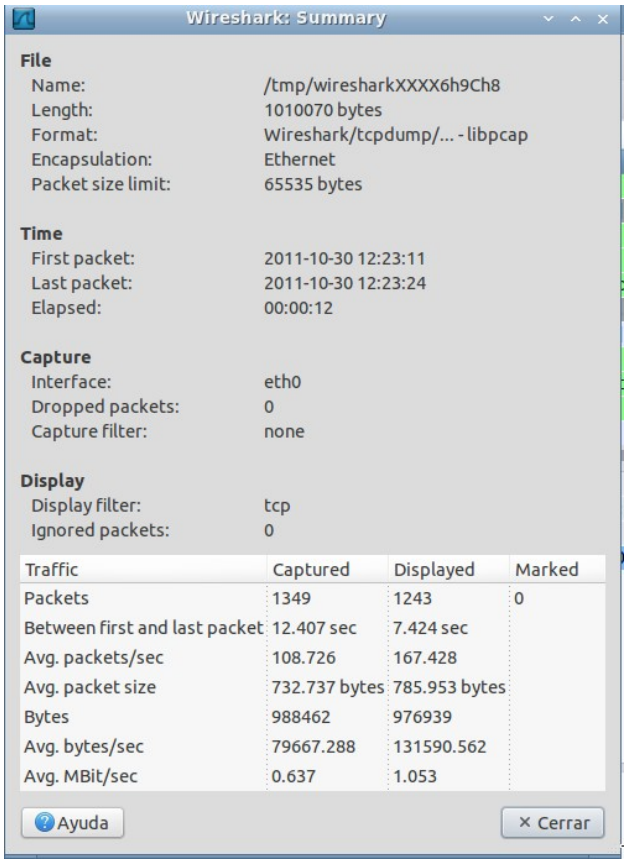
Como hemos dicho anteriormente, tanto en la etapa de captura, como en la etapa de análisis se puede hacer un filtrado de tráfico según una serie de condiciones realmente muy complejas (con infinidad de campos, operadores aritméticos y lógicos):



- **Estadísticas básicas** (requisito mínimo)**.**

Posibilidad de hacer algún tipo de estudios estadístico de los paquetes que pasan por el interfaz durante el periodo de observación.

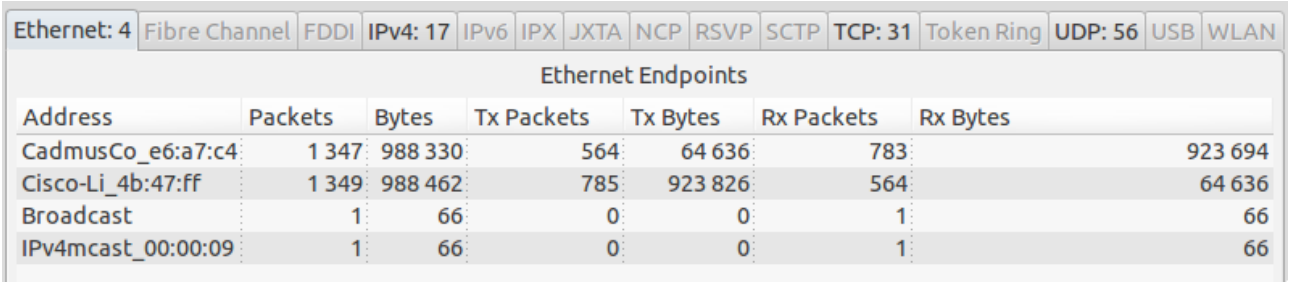
La pantalla principal sería la que encontramos en Statistics-Summary, que nos da la información de la captura total (y filtrada) sin desglose:



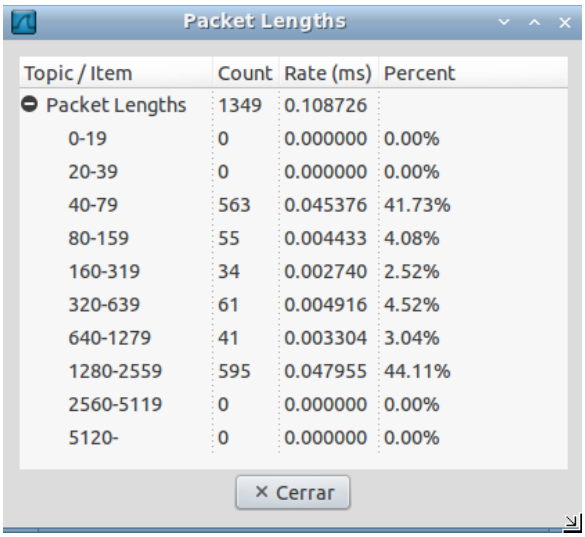
- **Detalle de desglose/segmentación de estadísticas** (requisito deseable):

Pero si queremos hacer desgloses, vemos que Wireshark nos abre un abanico enorme de posibilidades, ya que combinamos los filtros de paquetes/protocolos de la pantalla principal, con las presentaciones en tablas/gráficas del menú Statistics.

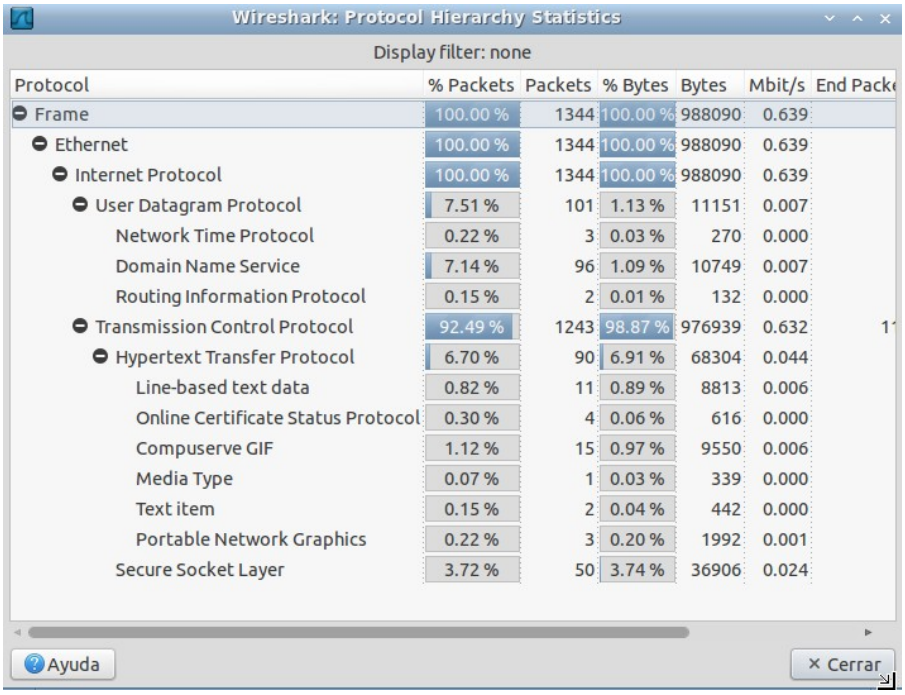
Por ejemplo, si queremos ver qué tramas ethernet se han transmitido/recibido por cada MAC:



Si queremos un desglose por tamaño de los paquetes:

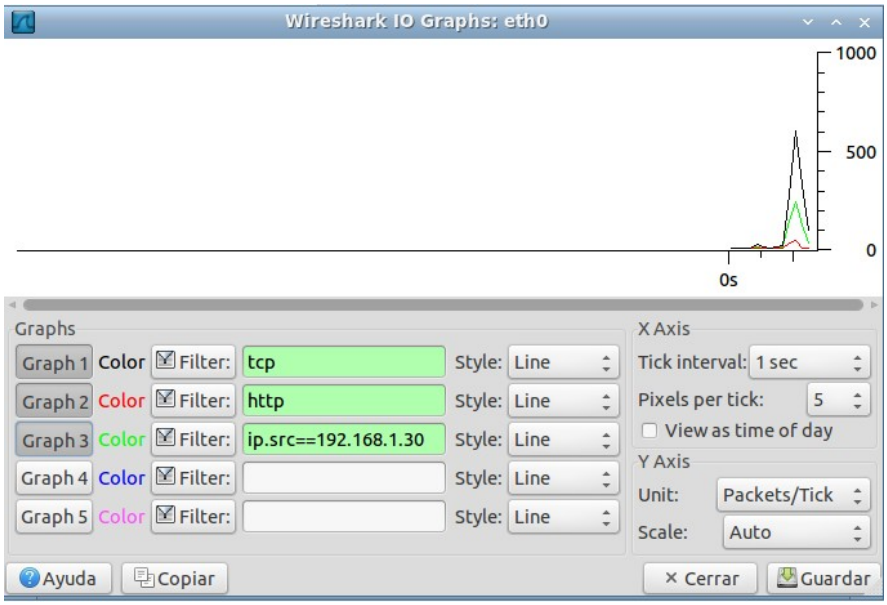


Si queremos un desglose por protocolo:



Si queremos sacar cualquiera de las pantallas anteriores pero para un subconjuto de paquetes sólo tendremos que aplicar filtros sobre la pantalla principal. Por ejemplo ***"ip.dsfield.dscp==46"*** nos filtrará los paquetes de calidad IP EF y podremos hacer estadísticas para ese tipo de tráfico.

Si queremos un análisis gráfico de ancho de banda utilizado, podemos hacer filtros en varias series y representarlas:



Por último, hay un requisito deseable que es sacar estadísticas de paquetes mal formados o con errores. Si el error es a bajo nivel (Ethernet), según la documentación de Wireshark es probable que no se pueda extraer, ya que o bien el driver o bien el sistema operativo o bien el libpcap puede estar descartando paquetes ethernet que directamente ve que están mal. Durante la fase de pruebas intentaremos estudiar estos casos y configurar el sistema adecuadamente para sacar la mayor información posible.

- **Inyección de tráfico** (requisito deseable)**.**

Wireshark no es capaz de inyectar tráfico a la red, así pues deberemos buscar alternativas si queremos implementar alguna funcionalidad de inyección de tráfico. Como es un requisito deseable no principal del proyecto, salvo que en la etapa de pruebas veamos la necesidad real de estas funcionalidades, de momento las dejamos aparcadas.

No obstante, si así surgiera, tampoco vemos la necesidad de ningún desarrollo adicional porque podemos usar herramientas disponibles en cualquier distribución GNU/Linux, como ping, traceroute,... Para evaluaciones de ancho de banda end to end, podemos usar alguna de las herramientas netcps, iperf o ttcp.

Por todo esto, concluimos que a priori **no será necesario desarrollar ningún nuevo software adicional**, ya que con lo disponible podemos cumplir prácticamente todos los requisitos. Por lo tanto, las siguientes fases del proyecto se dedicarán a elegir los componentes de software que instalaremos y crear una configuración válida, funcional y coherente a lo que queremos hacer.

4.2. Definición de casos de uso

A priori es complicado definir con detalle todos los casos de uso de esta herramienta, puesto que son tan amplios como los problemas que se pueden encontrar en la operación y mantenimiento de una red de telecomunicaciones. En función del problema y del interfaz pinchado por el analizador la herramienta será usada de una manera u otra.

No obstante sí vamos a ver algunos casos comunes generales y necesidades del usuario y cómo se debería enfrentar el técnico de operaciones que será el usuario del sistema.

a) Captura del interfaz IuB de un solo nodo B. Análisis de posibles problemas

Este es el caso más sencillo. Un nodo B sobre el que tenemos algún problema (p.e. Que no tengamos gestión remota de él, o que no levante el interfaz radio porque alguno de los mensajes que debería intercambiar con la RNC se pierde por el camino,…). De este nodo somos capaces de **pinchar un interfaz ethernet por el que sólo pase comunicaciones destinadas a este nodo**.

Los pasos a seguir por el usuario serán:

- Estudiar el modo de pinchar la sonda (bien en remoto o bien haciendo desplazar un técnico de campo al emplazamiento para que pinche un ordenador). Si es con remote port mirroring, deberá reconfigurar la red Backhaul para hacer llegar el interfaz remoto hasta el ordenador que esté en el centro de operaciones con la plataforma de análisis. Si es con un técnico desplazado, dicho técnico deberá preparar y ejecutar la virtualización así como el cableado y conexiones de red para que el técnico de operaciones acceda de forma remota a dicho ordenador.
- Una vez la sonda está pinchada, en funcionamiento y accesible, el técnico de operaciones lanzará un escritorio remoto contra ella desde su terminal de usuario (normalmente Windows). Aunque el escritorio remoto será de la máquina virtualizada, el acceso será a través de un puerto de la máquina huésped.
- En el escritorio habrá un acceso directo para abrir Wireshark.
- El técnico realizará una captura de paquetes que pasa por el interfaz. Con unos pocos segundos de captura podremos detectar la mayor parte de situaciones problemáticas.
- Bien en tiempo real o bien sobre la captura previamente grabada, usará las herramientas de Wireshark para analizar, diagnosticar y corregir el problema.

b) Captura del interfaz IuB en un punto con varios nodos agregados

Es básicamente el mismo caso que el anterior, pero dado que la estructura de la red de acceso que estamos analizando tiene forma de árbol, normalmente a los interfaces del Backhaul que podemos acceder mediante Remote Port Mirroring no tiene porqué pasar un único nodo B. Normalmente llegan hasta el primer router frontera varios nodos agregados mediante alguna red de nivel 2, como vimos en el análisis de situación del estudio de viabilidad.

Por ello, en la mayor parte de los casos deberemos aplicar un filtrado previo para quedarnos sólo con los paquetes del nodo B que deseemos.

Este filtrado se realizará en Wireshark (a ser posible en las opciones de captura, para que el fichero de captura no sea muy grande). Las condiciones del filtro serán del tipo:

(ip.src==x.x.x.x or ip.dst==x.x.x.x) or (ip.src=y.y.y.y or ip.dst==y.y.y.y)

donde x.x.x.x e y.y.y.y son las direcciones del nodo B.

La existencia de dos direcciones IP del nodo B se debe a que para el modelo de nodos B que estamos estudiando, aunque éstos sólo tienen un interfaz ethernet físico, sobre él se montan dos VLANs para separar el tráfico de gestión de remota del nodo B (que va a la plataforma de gestión de red) del tráfico de control y usuario de IuB (que va a la RNC). Ambas direcciones pertenecen a redes de clases diferentes (la de gestión a la clase A privada (10.x.x.x) y la de IuB a una clase B privada (172.19.x.x)).

Para filtrar un solo nodo, no filtramos el tráfico por VLAN, porque a la entrada del Backhaul, la VLAN es compartida por todos los nodos B (se usa para separar gestión de tráfico IuB).

c) Intercambio de ficheros con el resto de la red corporativa

En un momento dado, el usuario puede tener la necesidad de subir/bajar ficheros a/desde el sistema donde está Wireshark. Para ello únicamente tendrá que lanzar un cliente FTP contra la misma dirección que el VNC y accederemos con permisos de escritura/lectura a una carpeta que esté mapeada en el árbol de directorios del sistema donde se ejecute Wireshark.

d) Administración remota del sistema

Si el administrador del sistema requiere acceder de forma remota al anfitrión, tendrá la posibilidad lanzando un ssh contra la misma dirección IP del VNC (del huésped) o el FTP. Con el usuario de administración, accederá a una terminal de comandos del sistema anfitrión donde podrá realizar cualquier tarea de cambio del sistema.

También incluye la posibilidad de lanzar de manera puntual un servidor de sesión VNC sobre el escritorio del anfitrión, pero sobre un puerto distinto.

4.3. Definición de interfaz de usuario

La interfaz del usuario de la herramienta será bastante sencilla y no vamos a entrar a detallar menús, botones, etc, porque serán los que brinden las aplicaciones que usaremos. Únicamente enumeramos qué conceptos estarán disponibles por el usuario y cómo accederá a ellos:

- Aplicación cliente VNC para acceder a la plataforma (Windows):
  - En el caso de que la sonda esté en una máquina fija (con remote port mirroring) se lanzará el VNC contra la IP (estática) de la red corporativa de dicha máquina, que será previamente conocida (y preconfigurada en el cliente VNC).
  - En el caso de una máquina portátil que ha llevado al emplazamiento un técnico de campo, el técnico de operaciones deberá indicar al técnico de campo cómo averiguar y decirle qué IP tiene dicha máquina (dependerá en cada momento según la conexión que éste haga a través de internet y la pasarela securizada a la red corporativa).
- En ambos casos, habrá un password para abrir el VNC a la plataforma, que conocerá el técnico de operaciones.
- Una vez establecida la conexión VNC, habrá que loggearse a la máquina virtual. Si la máquina era fija, posiblemente no será necesario hacer login porque alguien ya se habrá loggeado previamente (dicha máquina estará conectada las 24 horas), pero por contrapartida seguramente habrá saltado el salvapantallas que también estará protegido por la misma contraseña.
- Con el escritorio remoto visualizado, tendremos disponible un acceso directo para lanzar Wireshark (evitaremos que el usuario tenga que configurar o preocuparse por permisos adicionales).
- Una vez tenemos abierto Wireshark, usaremos su interfaz gráfico para hacer la captura y análisis. Supondremos que el usuario sabe utilizar dicho programa. Si no es así se deberá dar un mínimo de formación.
- Si requerimos traernos/subir ficheros a/desde el terminal Windows de usuario, podremos lanzar un FTP (contra la misma IP de antes, que será la de la máquina anfitrión) que nos abrirá una carpeta que esté mapeada en el ordenador donde se ejecuta Wireshark.
- El usuario desde el escritorio remoto tendrá las aplicaciones básicas de Lubuntu por si le fueran necesarias (terminal de comandos, navegador web, aplicaciones ofimáticas,...).

4.4. Plan de pruebas de aceptación

Una vez esté montada la plataforma, deberemos asegurarnos que funcione correctamente, sometiéndola a un plan de pruebas de aceptación. Las pruebas que realizaremos serán:

- Captura de trazas básicas.

Es la primera prueba. Pincharemos el interfaz de red que nos vaya a hacer las veces de sonda en un punto de la red corporativa (aún no probaremos la red de telecomunicaciones). Capturaremos y verificaremos que son de acuerdo a protocolo algunos procesos como:

  - Resolución de MAC mediante ARP contra otra máquina en la red corporativa.
  - Ping ICMP contra otra máquina de la red corporativa.
  - Apertura de la web de la intranet (HTTP).

- Test de velocidad máxima de captura.

Dado que la máquina sobre la que se va a ejecutar el software de captura posiblemente sea una máquina antigua sin muchos recursos y además virtualizada (que hace bajar su performance drásticamente), haremos una prueba para analizar si puede capturar una traza de paquetes de muy alta velocidad.

Con la sonda pinchada sobre la red corporativa, lanzaremos una descarga FTP contra un servidor. Haremos un análisis estadístico de una traza en Wireshark de esta descarga. El objetivo de este análisis es ver si ha conseguido capturar todos los paquetes y el ancho de banda que mide (debe coincidir con el que vemos en la aplicación FTP).

- Captura de traza de 802.1Q (taggeado VLAN).

Con lo anterior, nos aseguramos que Wireshark y el interfaz de red Ethernet básicamente funciona bien y puede capturar paquetes a velocidades razonables.

Pero en la red de telecomunicaciones, en general, no funciona según la norma Ethernet básica, sino que la mayor parte del tráfico se encuentra bajo la norma 802.1Q que separa el tráfico etiquetando (taggeando) VLANs y calidades de servicio. Esto requerirá de alguna configuración extra (y que el hardware pueda reconocer y trabajar con tráfico taggeado).

Así pues, una vez chequeado el tráfico sobre un ethernet normal de la red corporativa, la siguiente prueba será pinchar algún equipo de la red de telecomunicaciones (sin afectar a la red real). Existe un nodo B de maqueta en el centro de conmutación. Configuraremos dicho nodo B para que genere tráfico taggeado y pincharemos su interfaz Ethernet.

Aunque poco nos importará la coherencia del protocolo (el nodo B contra el ordenador no va a conseguir establecer ninguna comunicación), sí que podremos capturar las tramas Ethernet 802.1Q y veremos si la plataforma es capaz de decodificarlos correctamente.

- Captura de interfaz IuB de un nodo B remoto.

Hasta aquí tendremos probado que la plataforma funciona correctamente como sonda. Pero necesitamos ponernos en un caso real para certificar que todo funciona como esperamos (incluyendo las técnicas de remote port mirroring).

Así pues, la siguiente prueba será pinchar la máquina al puerto del Backhaul IP y configurar el

remote port mirroring para traernos de forma remota el interfaz de un nodo distante. En Wireshark capturaremos dicho interfaz y comprobaremos la coherencia de los resultados (p.e. haremos un análisis estadístico y/o intentaremos capturar algún proceso básico de IuB de UMTS como el Cell Setup).

- Acceso remoto.

Todas estas pruebas se podrían realizar físicamente sobre el ordenador. Pero lo que nos queda por probar es que desde los terminales de usuario Windows podemos:

  - acceder a la máquina virtual con wireshark (VNC).
  - intercambiar ficheros con la plataforma (FTP).
  - acceder por línea de comandos al sistema anfitrión (SSH) en el caso del prototipo fijo.

- Arranque de todos los servicios automáticamente.

En el sistema montado en el prototipo, como será desatendido y funcionando 24x365, se deberán iniciar automáticamente todos los servicios con el arranque del sistema. El apagado debe ser controlado para que al siguiente arranque no haya ningún problema. En general, esto no debe ser problemático a excepción de las máquinas virtuales, para la que deberemos hacer alguna configuración.

5. Diseño del Prototipo

Con el análisis hecho, procederemos a diseñar a bajo nivel el prototipo del que es objeto este proyecto. Es importante que en esta fase sí que debemos centrarnos en el prototipo concreto que deseamos implantar a pequeña escala.

Como hemos concluido en la fase anterior, no requeriremos de ningún desarrollo de software adicional ya que con los componentes y paquetes disponibles con licencia de software libre tendremos suficiente para cubrir nuestros objetivos.

Dejamos fuera del alcance de la memoria, detalles de diseño y configuración de la red de telecomunicaciones para hacer llegar los datos del remote port mirroring al interfaz ethernet del prototipo.

5.1. Definición de la arquitectura de la máquina anfitriona

a) Hardware

Uno de los objetivos del proyecto es que el prototipo tuviera un coste próximo a cero. Para poder llegar a ese objetivo, deberemos reutilizar hardware antiguo en desuso (o incluso obsoleto para aplicaciones Windows).

Además, pensando en el caso de que quisiéramos llevar un ordenador portátil (con una máquina huésped virtualizada dentro) a un lugar remoto no debemos imponer muchos requisitos. Por lo tanto, vamos a trabajar con el siguiente hardware:

- CPU: El ordenador en el que montaremos el prototipo es un simple Pentium 4 a 1,8 GHz. Aunque para que funcionara correctamente la máquina virtual sería deseable que la CPU implementara la extensión de instrucciones VT-x o AMD-V, no la vamos a poner como requisito (de hecho la máquina nuestra NO la usará).
- RAM: Para poder mover los 2 sistemas operativos (Linux) con garantías, debería tener un mínimo de 1GB. Nuestro prototipo así lo tendrá.
- Disco Duro: Para poder instalar dos sistemas operativos de una distribución ligera no debería ser necesario mucho. Con unos 5-10 GB por máquina sería suficiente (en total 10 o 20GB). En nuestro caso, el prototipo llevará un disco antiguo de 38GB por interfaz IDE.
- Tarjetas de Red: Debe implementar al menos dos interfaces de red: uno que se direccione dentro de la red corporativa y otro que se conecte a la red de telecomunicaciones que queremos monitorizar. En el prototipo, como estará en un lugar fijo, deberemos tener dos interfaces ethernet independientes. Uno normalmente va integrado en la misma placa base (y así es en nuestro caso). Para el otro utilizaremos una tarjeta Ethernet suplementaria conectada al bus PCI. Por si hubiera problemas de controladores, haremos acopio de diferentes tarjetas de red de entre todos los ordenadores en desuso que teníamos.
- Resto de periféricos: No requerimos de mucho más, salvo una tarjeta gráfica simple (integrada en la placa base). En el momento de la instalación (hasta que podamos acceder a la máquina remotamente y se convierta en un servidor headless) requeriremos de monitor+teclado+ratón+unidad CDROM externa. Todo esto también será reutilizado de equipos en desuso.

b) Sistema Operativo Base

El sistema operativo que instalaremos, como justificamos en el estudio de viabilidad será Lubuntu.

En principio íbamos a instalar la más moderna 11.10 (con kernel de la familia 3.0), pero tras un primer intento llegamos a la conclusión que la versión de Virtualbox incluída (4.1.2) no permitía capturar los paquetes por la tarjeta de red virtual si éstos llevaban etiquetado 802.1Q (hacía "*VLAN tag stripping*"). Por ello, hemos tenido que buscar una versión anterior de Virtualbox (3.2.8). Pero como Virtualbox requiere de la compilación de algunos módulos dinámicos (mediante dkms) y los header del kernel necesarios eran de la versión 2.6, tuvimos que hacer un downgrade del sistema operativo.

En resumen, optamos por Lubuntu 10.10.

En cuanto a los detalles de instalación:

- Partición completa del disco duro excepto 1GB. Se formateará con el sistema ext4 y se montará en el raíz "/".
- Partición de 1GB para swapping de memoria virtual.
- Para la red usaremos una IP fija para el interfaz de red integrado en la placa base (el que esté conectado al a red corporativa). Evitaremos la asignación dinámica de IPs por DHCP (procedimiento interno estándar) para tener localizada la máquina de forma remota. Para ello será necesario petición de puerto e IP fija a los administradores de la red corporativa.
- Habrá un único usuario (administrator) que será a su vez administrador del sistema (será un "sudoer").
- Una vez instalado y actualizado el sistema (configurando el proxy interno de la red corporativa), se deshabilitarán todas las opciones de chequeos de paquetes de software nuevos y/o actualización automática.
- Se configurará la resolución de la pantalla del escritorio (por si deseamos acceder de forma remota por un VNC puntual lanzado por el administrador) a 1024x768. Esta configuración se puede hacer así:

```
sudo /etc/init.d/lxdm stop
sudo Xorg -configure
sudo mv xorg.conf.nes /etc/X11/xorg.conf
sudo nano /etc/X11/xorg.conf
```

añadimos en la sección Screen 0

```
Default Depth 24
```

añadimos en la subsección Display correspondiente al Depth 24

```
Modes "1024x768"
```

```
sudo /etc/init.d/lxdm start
```

- Por motivos de seguridad, se activa el protector de pantalla para que se active a los 5 minutos y sólo se pueda desactivar mediante reintroducción del password.
- Por último, para prevenir que se apague accidentalmente la máquina anfitriona, procedemos

a cambiar la política de apagado según se explica en un foro de Ubuntu [Apag]. En el fichero `/usr/share/polkit-1/actions/org.freedesktop.consolekit.policy` cambianos a "no" la entrada allow_interactive siguiente:

```
<action id="org.freedesktop.consolekit.system.stop">
    <description>Stop the system</description>
    <message>System policy prevents stopping the system</message>
    <defaults>
      <allow_inactive>no</allow_inactive>
      <allow_active>no</allow_active>
    </defaults>
  </action>
```

c) Paquetes de software adicionales

Virtualbox

Instalaremos la aplicación Virtualbox que montará la máquina virtual con todo nuestro sistema. Para ello haremos:

```
sudo apt-get install virtualbox-ose
```

La configuración de la máquina virtual, será vista en el apartado de definición de hardware de la máquina virtual huésped.

Como nuestra intención es hacer un servidor desatendido remoto headless cuyas virtualizaciones sean accesibles vía VNC, deberemos añadir una configuración que nos haga cómodo (de forma manual o automática durante el arranque/apagado del anfitrión) el lanzamiento, consulta de estado y parada (controlada, salvando el estado) de la máquina virtual. Para ello vamos a crear un script alojado en /etc/init.d que nos permita trabajar como un demonio/servicio más:

- Modificamos la configuración por defecto de Virtualbox para que el apagado de las máquinas virtuales sea salvando el estado (en el siguiente arranque, seguirá con ella). Modificaremos el fichero `/etc/default/virtualbox-ose`, incluyendo las líneas:

  ```
  SHUTDOWN_USERS="administrator"
  SHUTDOWN=savestate
  ```

- Creamos el script `/etc/init.d/virtualbox-lubuntu` (nos basamos en uno similar en [VBSrv]). Podemos ver dicho script en el **Anexo 1**.
- Metemos el script dentro del listado dentro de los directorios de scripts de arranque (runlevel de System V) pero para que sea uno de los últimos (número 99) y no entre en conflicto con el arranque de módulos de virtualbox (p.e. el S20virtualbox-ose) mediante:

  ```
  sudo update-rc.d virtualbox-lubuntu defaults 99
  ```

Notar que en el script incluimos la opción de autoarranque del servidor VNC para esta máquina virtual en el puerto 5910 con el password que deseemos:

```
sudo -H -u $VM_OWNER VBoxHeadless -s "$VM_NAME" -n -m 5910 -o password
```

FTP

Para el intercambio de ficheros con el exterior montaremos un servidor ftp. Usaremos el paquete *vsftpd*, que instalaremos mediante:

```
sudo apt-get install vsftpd
```

Deberemos habilitar el intercambio con un usuario/password conocido por los usuarios y diferente al del administrador. Por los requisitos de seguridad, además, sólo podrá acceder a una carpeta específica (enjaularlo a /home/ftp) y que será la que compartirá Virtualbox con la máquina huésped. Para ello:

- Crearemos la carpeta compartida y pública:

  ```
  mkdir /home/ftp
  chmod 777 /home/ftp
  ```

- Creamos el usuario userftp que será el que conocerán los usuarios (pero sin acceso a shell):

  ```
  sudo addgroup userftp
  sudo useradd userftp -g userftp -d /home/ftp -s /bin/false
  sudo passwd userftp
  ```

- Creamos una lista de usuarios no permitidos en `/etc/vsftpd.deny_list` que contenga todos los usuarios excepto `userftp`.

- Rehacemos la configuración del servicio en el fichero `/etc/vsftpd.conf`:

```
#configuracion general
write_enable=YES
dirmessage_enable=YES
ftpd_banner="Bienvenido a FTP Lubuntu"
xferlog_enable=YES
vsftpd_log_file=/var/log/vsftpd.log
log_ftp_protocol=YES
connect_from_port_20=YES
idle_session_timeout=600
data_connection_timeout=120
pasv_enable=YES
pam_service_name=vsftp
listen=YES
max_clients=5
max_per_ip=3
pasv_min_port=40000
pasv_max_port=40020

#acceso a solo los directorios home locales
local_enable=YES
local_umask=0000
chroot_local_user=YES

#acceso anonimo
anonymous_enable=NO

# Listas de bloqueo
userlist_enable=YES
userlist_file=/etc/vsftpd.deny_list
```

SSH

SSH nos valdrá tal cual sale de la instalación:

```
sudo apt-get install ssh
```

Como sólo tenemos un usuario con shell válido (administrator, ya que userftp apuntaba a /bin/false), sólo podremos hacer ssh contra él. Recordamos que hemos bloqueado el servicio vsftpd para este usuario. Pero gracias a que ssh proporciona el sftp/scp podremos igualmente intercambiar ficheros mediante este usuario de forma segura.

Otros paquetes opcionales

Por si nos fuera necesario en algún momento para el administrador del sistema, instalamos los paquetes servidor y cliente de VNC y wireshark:

```
sudo apt-get install wireshark x11vnc xtightvncviewer
```

5.2. Definición de la arquitectura de la máquina huésped

El objetivo central del proyecto es tener esta máquina virtual huésped que sea instalable en cualquier máquina anfitrión que pueda ejecutar Virtualbox. Vamos a especificar qué hardware virtualizado verá, qué sistema operativo instalaremos (y con qué opciones) y qué paquetes de software adicionales instalaremos y configuraremos.

a) Hardware virtualizado

El hardware virtualizado (máquina que llamaremos LubuntuVirtual) es el que configuraremos en la aplicación Virtualbox que se ejecute en la máquina anfitriona. A continuación detallamos la configuración que consideramos óptima:

- CPU: 1 Única CPU. No habilitado la extensión PAE/NX ni VT-x/AMD-V.
- RAM: 512 Mbytes.
- Placa Base: Deshabilitado IO APIC, EFI y dispositivo apuntador absoluto.
- Tarjeta Gráfica: 12 Mbytes de memoria de video. Sin habilitar aceleración 3D.
- Disco duro: Emulación de SATA, tipo AHCI. Está mapeado sobre un fichero (LubuntuVirtual.vdi) de tamaño 8Gbytes dinámico.
- Controlador IDE: definido pero sin CDROMs cargados por defecto.
- Audio: Deshabilitado.
- Interfaces de Red: Un único adaptador, de tipo “adaptador puente” o bridge (modo promiscuo), sobre el interfaz físico que mira a la red de telecomunicaciones. Simula un controlador físico del tipo Pcnet-FAST III (Am79C973) que tiene buen soporte para decodificar VLAN tags.
- Puerto Serie: Deshabilitado.

b) Sistema Operativo Base

Para la máquina huésped, como no va a ser anfitriona de otro huésped anidado, no va a ejecutar VirtualBox, y por lo tanto, no tendremos la restricción de kernel que hemos comentado anteriormente. Así pues, escogemos instalar la última versión de Lubuntu que es la 11.10.

Como la máquina virtual sólo verá uno de los interfaces de red, es importante que durante el proceso de instalación este interfaz tenga salida a internet para poder descargar y actualizar paquetes.

El proceso de instalación es el estándar con las siguientes opciones elegidas:

- Usamos todo el disco duro virtual para una partición ext4, sin swap. Esta máquina ya estará bastante justa de recursos de CPU y disco como para añadir swapping. Como Lubuntu es un sistema ligero y sólo pensamos ejecutar Wireshark, pensamos que es una buena opción. Si más adelante lo requiriéramos, podríamos añadirle el swap a posteriori.
- Creamos el usuario “user” que será administrador local de la máquina.

- Tras el primer arranque, Lubuntu 11.10 detecta que se ejecuta sobre una máquina virtual y pide instalar los controladores “x86 virtualization solution” y “guest-addition source for dkms”. Los instalamos y habilitamos.

Al igual que en el anfitrión, una vez instalado y actualizado el sistema se deshabilitarán todas las opciones de chequeos de paquetes nuevos y/o actualización automática.

También activamos el salvapantallas con protección de contraseña, para que a los 5 minutos de desuso sólo un usuario con el password pueda reactivar el sistema.

A continuación, configuraremos la carpeta compartida. En el virtualbox anfitrión, en las opciones de “Carpetas compartidas” se añadirá la carpeta “/home/ftp” (ver el diseño del FTP anfitrión), que para el huésped se llamará “ftp”. En el huésped añadiremos al fichero `/etc/fstab` la línea siguiente para que se automonte dicha carpeta en /home/user/ftp:

```
ftp   /home/user/ftp vboxsf rw,exec,uid=1000,gid=1000,dev  0 0
```

c) Paquetes de software adicionales

Wireshark

La instalación de wireshark se realizará mediante la instrucción:

```
sudo apt-get install wireshark
```

Ahora bien, por seguridad si arrancamos wireshark tal cual sale de la instalación, veremos que éste no es capaz de encontrar ningún interfaz de red para capturar y debemos agregar permisos. Esto se puede hacer de forma permanente dando los privilegios adecuados al programa de captura dumpcap:

```
sudo setcap 'CAP_NET_RAW+eip CAP_NET_ADMIN+eip' /usr/bin/dumpcap
```

Haremos también algunos cambios de configuración para facilitar al usuario las capturas:

- Ocultaremos los interfaces de red no deseados (p.e. “lo” o “any”). Para ello vamos a Edit – Preferences – Capture – Interfaces Edit. Y le marcamos el tick “hide” en todos menos en eth0.
- Por motivos de performance, vamos a deshabilitar el actualizado interactivo de paquetes en pantalla durante la captura. Para ello vamos a Edit – Preferences – Capture y desmarcamos la opción “Update list of packets in real time” [5]. Con esto, durante la captura, la ventana se queda en blanco mientras que en la barra de estado vemos cómo se actualiza el número de paquetes capturados.

5 En las pruebas sobre la virtualización observamos que con esta opción activada, la carga de CPU se iba al 100%, mientras que con ella desactivada, no superaba el 50%.

5.3. Funcionamiento interno de casos de uso

Veamos la correlación entre los casos de uso vistos en la fase de análisis y los componentes de nuestra arquitectura:

a) Captura del interfaz IuB de un solo nodo B. Análisis de posibles problemas

En este caso de uso intervendrán:

- Reconfiguración de la red Backhaul para que funcione en remote port mirroring (esto queda fuera del objeto del diseño).
- Virtualbox del anfitrión. De dos maneras:
  - Proporciona la plataforma hardware virtual donde correr el huésped.
  - Proporciona el servidor VNC (puerto 5910 y password según configuración) al huésped.
- Lubuntu huésped 11.10. Proporciona el sistema base desde donde se ejecutarán la suite Wireshark con los programas de captura (dumpcap) y herramientas de visualizado/análisis. Es importante que el interfaz de red deje pasar todo el paquete sin recortar cabeceras (p.e. VLAN tags). El usuario de este sistema será “user”
- Wireshark. Es la base de todo el caso de uso y sobre el que interactuará el usuario principalmente, viendo todas sus opciones disponibles.

b) Captura del interfaz IuB en un punto con varios nodos agregados

Desde el punto de vista de funcionamiento interno, es el mismo caso de uso que el anterior. Sólo aplicarán filtros de Wireshark.

c) Intercambio de ficheros con el resto de la red corporativa

Intervendrán:

- Servicio FTP del anfitrión. Es un servicio FTP de puertos estándar, sólo accesible al usuario “userftp” y “enjaulado” a la carpeta /home/ftp.
- Virtualbox. Mediante las guest-additions, el sistema huésped podrá montar la carpeta /home/ftp del anfitrión en su árbol de directorios.

d) Administración remota del sistema

En este caso de uso intervendrá el servidor SSH instalado en el anfitrión. Sólo podrá accederse mediante el usuario administrator. El puerto será el estándar TCP 22. Una vez abierta la sesión, se podrá incluso lanzar un servicio puntual de VNC sobre el anfitrión usando el componente x11vnc. Por ejemplo el siguiente comando nos abrirá el puerto 5901 para aceptar peticiones desde clientes VNC para acceder al el escritorio anfitrión mediante contraseña:

```
x11vnc -passwd contrasenya -display :0 -rfbport 5901 -forever
```

5.4. Especificación del desarrollo e implantación

Del diseño planteado, vemos que para el desarrollo e implantación deberemos realizar diferentes tareas de las que especificamos cómo y con qué herramientas deberíamos atacar su desarrollo:

- Adquirir y preparar la máquina física con el hardware mínimo especificado. Usaremos materiales proporcionados por Vodafone.
- Adecuar el entorno físico: junto con Vodafone se decidirá la ubicación del equipo en el Centro de Conmutación, así como la manera de realizar el montaje de cableado de red tanto para la red corporativa como para la de telecomunicaciones.
- Configuración de la red de telecomunicaciones para funcionar en Remote Port Mirroring. Usaremos las herramientas del plataforma de gestión de red del fabricante de equipos para Backhaul (en este caso Huawei [U2000]).
- Instalación del software en la máquina anfitriona. Se hará directamente sobre la máquina, conectada a la red corporativa. A través de un proxy de red, se podrá acceder a Internet y mediante ésta a los repositorios oficiales de Lubuntu para actualizaciones y descargas.
- Instalación de la máquina virtual. Dado que la máquina anfitriona tendrá poco performance de CPU para mover el sistema virtual y como la instalación puede durar varias horas en este entorno, crearemos una máquina virtual de similares características en en otro ordenador personal (no destinado a este proyecto) con procesador de instrucciones AMD-V. En éste, dejaremos preparado todo el sistema y se traspasará al prototipo copiando el fichero que contiene la imagen del disco duro virtual (LubutuVirtual.vdi).

6. Desarrollo e implantación

Esta fase del proyecto debería comprender la integración del sistema, realización de plan de pruebas de aceptación, documentación y formación de usuarios. Aunque entra dentro de la implantación, por su importancia en esta memoria dejamos en otro apartado la realización y análisis crítico de ejemplos de casos de uso con sus resultados.

6.1. Integración del sistema

Repasemos las tareas principales de integración del sistema, viendo algunos detalles:

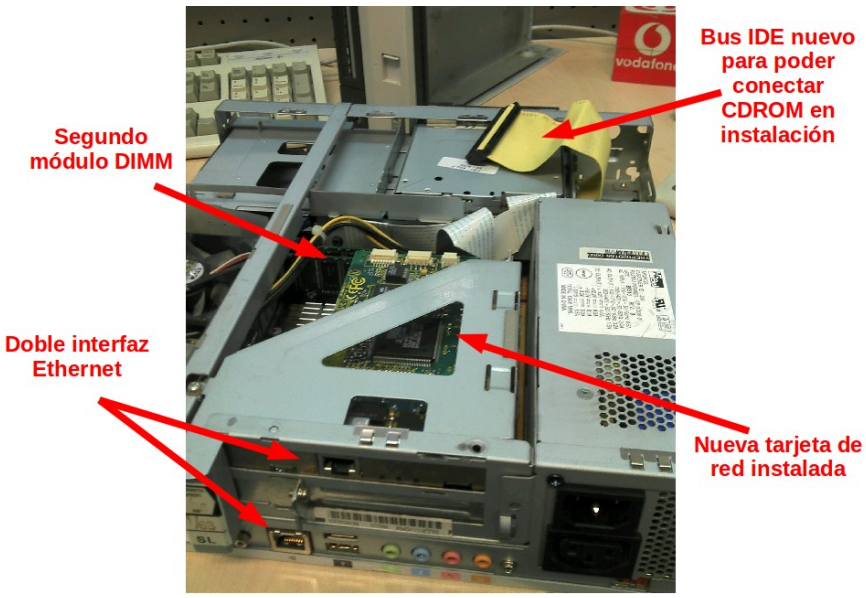
a) Montaje físico del prototipo

Como uno de los requisitos del proyecto era que tuviera coste cercano a cero, se optó por reutilizar hardware obsoleto o en desuso. Para ello, de todas las máquinas que estaban físicamente en el centro de trabajo y que estuvieran aparcadas, se eligió aquella que, teniendo mejores prestaciones, permitiera el montaje físico de más memoria y tarjetas de red.

Se optó por un pequeño equipo Pentium 4, 512 Mbytes de RAM (con un slot DIMM libre) y 38 Gbytes de disco duro IDE. Físicamente disponía de 2 slots libres para insertar tarjetas PCI de tamaño completo. No disponía de CDROM ni podía arrancar desde pendrives USB.

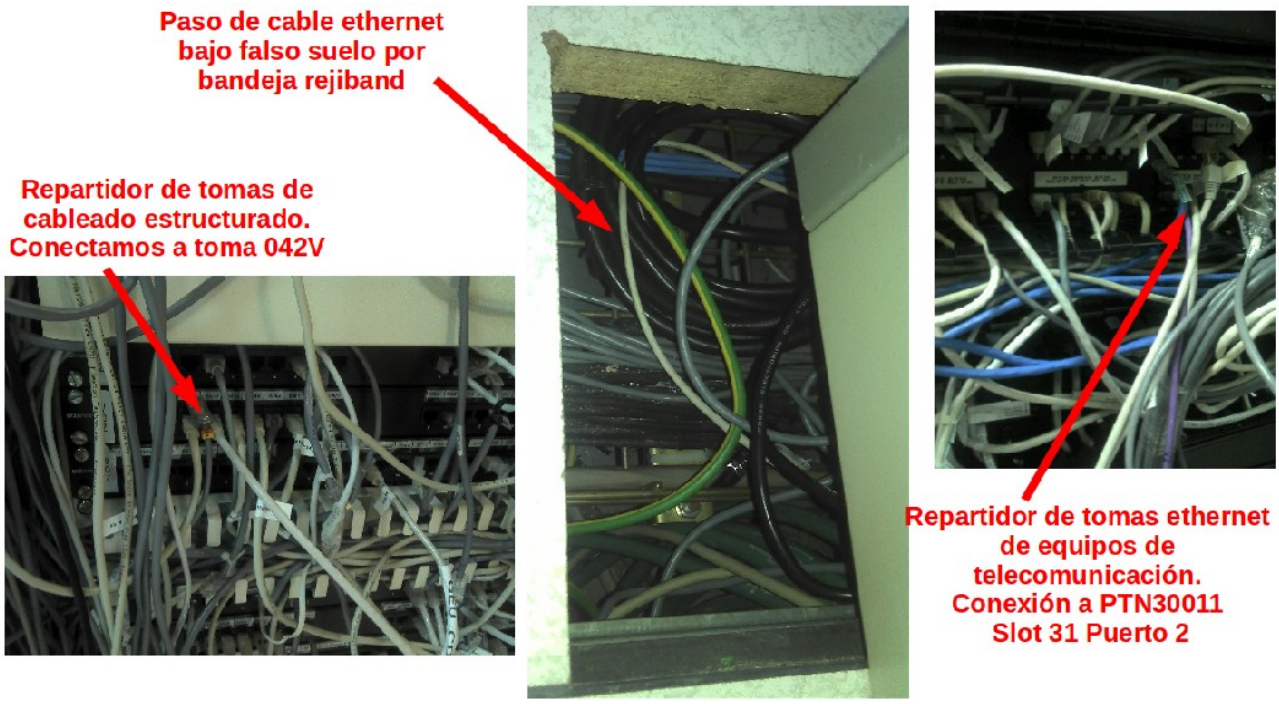
Se equipó con:

- Otra tarjeta de red (PCI), tomada de otro equipo y que resultó tener buen soporte en Linux.
- Otro módulo de memoria de un ordenador exactamente igual, para completar 1Gbyte.
- Se reemplazó el cable IDE de un solo puerto, por un cable para dos dispositivos IDE.
- Se reconfiguró el disco duro para que fuera máster del bus IDE.
- Se equipó de forma provisional (fuera del chásis) una unidad CDROM IDE, para poder hacer la instalación del sistema operativo. Luego se retiró, una vez instalado.



Se eligió una ubicación en la sala de operadores, donde había además de toma de energía eléctrica conectada al SAI del centro, había disponibles 3 tomas Ethernet con cableado estructurado del edificio hasta la sala de equipos.

En la sala de equipos, las tomas del cableado llegaban a un repartidor. En esta misma sala se encuentran switches de la red corporativa y en otro extremo repartidores de puertos Ethernet de las centrales y equipos de telecomunicaciones. Una de las tomas Ethernet se conectó al puerto de la red corporativa (configurado por IT). La otra se tuvo que tirar un cable Ethernet (categoría 6 debido a la distancia) hasta el repartidor donde estaban los puertos de los equipos del Backhaul.



b) Configuración de la red de telecomunicaciones

Se realizó la petición y reserva formales de un puerto en la red de telecomunicaciones que no fuera a ser utilizada por ningún otro departamento para otro fin. Este puerto iba a ser el punto fijo de observación y que se conectó a la máquina anfitriona. Con este puerto fijado (llamemos puerto C) y sabiendo el puerto remoto que queremos monitorizar (llamemos puerto A) la configuración en la red de telecomunicaciones se basa en lo siguiente:

- En el equipo donde está el puerto C, se configura sobre él un Port Mirror del tipo “Remote Mirror Observation Service”. Para ello elegimos el puerto y creamos un pseudowire (con un id) del tipo dinámico y que viaja por un túnel MPLS hacia el equipo donde está el puerto A. El puerto C será fijo siempre y se ha configurado de forma permanente a 100 Mbps Full Dúplex y para aceptar cualquier tipo de trama ethernet (802.1, 802.1Q o QinQ -doble taggeado-).
- En el equipo donde está el puerto A, se configura también el Port Mirror pero esta vez del tipo “Remote Mirror Service”. Se configura igualmente el puerto y un pseudowire (con el mismo id que el anterior) pero viajando por un túnel en el sentido opuesto.

Como no hay configuración en cuanto a QoS específica, la priorización con la que viajan los paquetes replicados por el mirroring a través de la red Backhaul, será la misma que la del paquete original. Es decir, si un paquete que entró al Backhaul se propaga con un Per Hop Behaviour de EF o BE, su réplica del mirroring viajará duplicada también con PHB igual EF o BE, con esto los paquetes más críticos para el funcionaimiento del nodo B serán también los más prioritarios para ser capturados por la plataforma en caso de congestión.

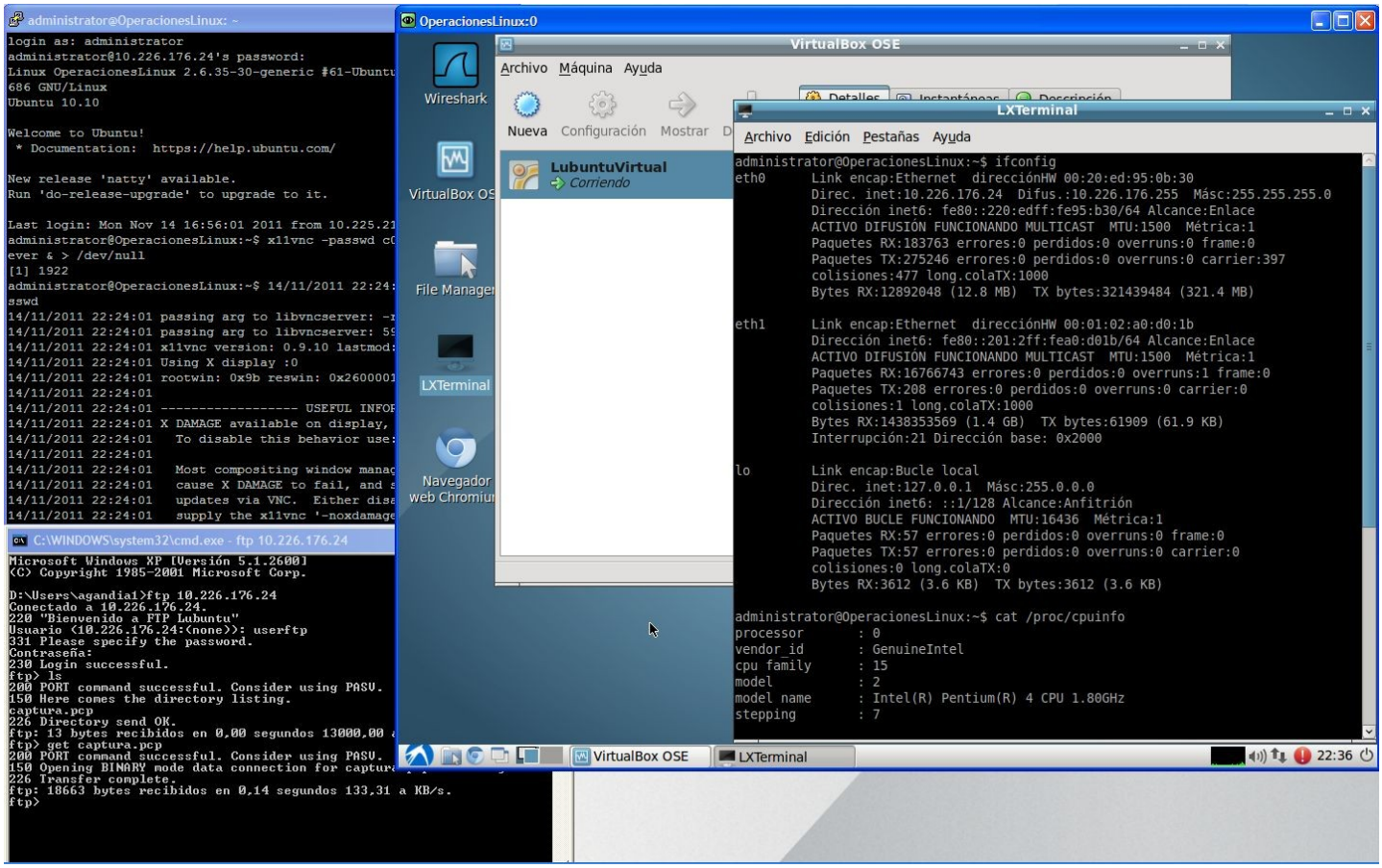
c) Preparación de la máquina anfitriona

Como comentamos, para la máquina anfitriona tuvimos que localizar e instalar una versión más antigua de Lubuntu (la 10.10) para que evitar el VLAN tag stripping que hacía VirtualBox.

Seguimos las instrucciones acorde al diseño del apartado anterior, sin incidencias.

Se fueron probando en cada paso todos los servicios (SSH, FTP, Virtualbox como demonio y el VNC).

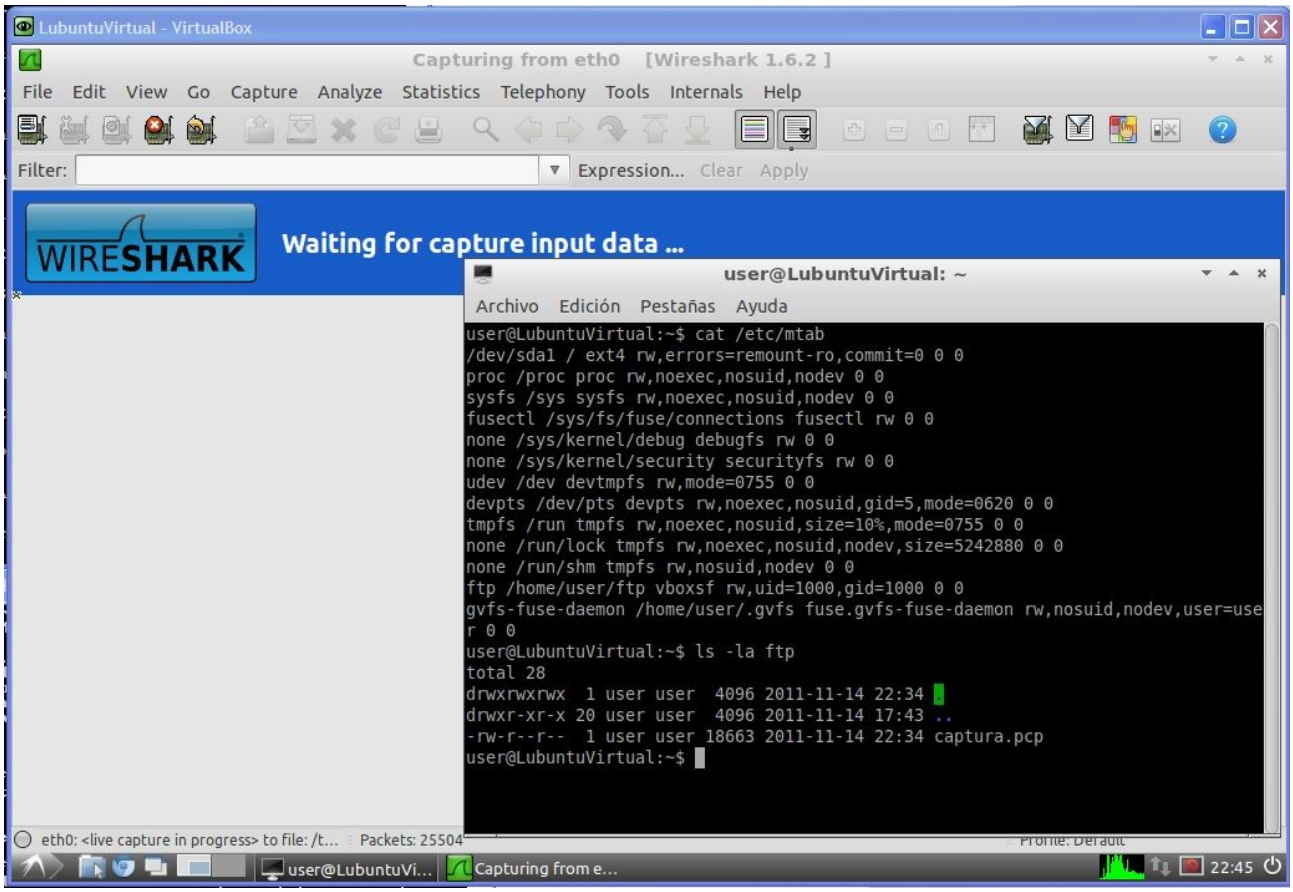
Aquí tenemos una captura de pantalla del aspecto de la máquina anfitriona desde un terminal de cliente Windows. Tenemos un cliente SSH Putty -donde se abrió temporalmente el servidor VNC al puerto 5901 para ver su escritorio-, un cliente VNC UltraVNC y un cliente FTP (comando Windows) desde una estación de trabajo de usuario Windows:



d) Preparación de la máquina huésped

La máquina huésped Lubuntu 11.10 fue preparada en un ordenador más potente y transportada mediante su fichero de disco duro virtual `LubuntuVirtual.vdi` al prototipo (a través del FTP). Se crearon accesos directos y retocó el escritorio (p.e. se incluyó una gráfica de análisis de carga de CPU en tiempo real).

Aquí tenemos una captura del aspecto del escritorio del huésped (visto a través de un cliente VNC al puerto 5910) con el Wireshark abierto en proceso de captura (ver la carga de CPU moderada) y un shell donde podemos ver que está montada la unidad ftp del tipo vboxsf:

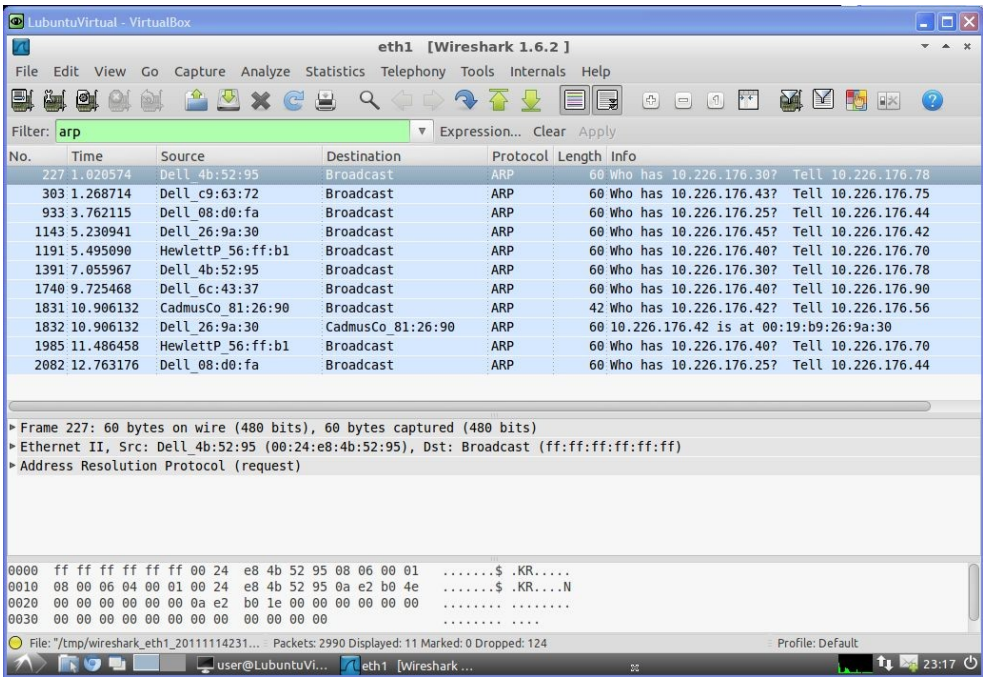


6.2. Realización del plan de pruebas de aceptación

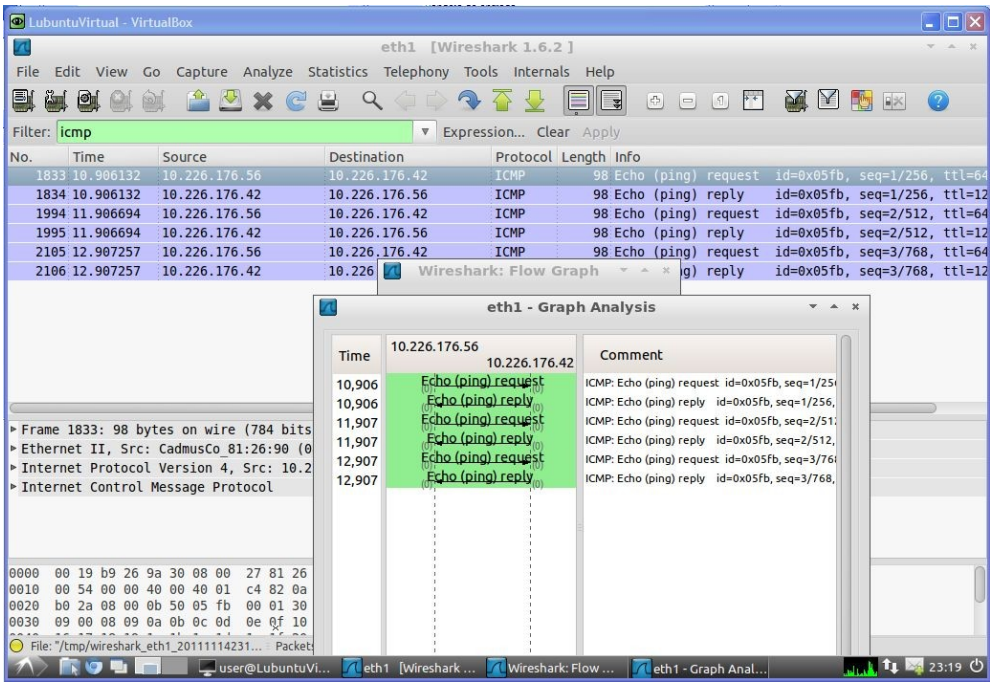
a) Captura de trazas básicas

En la máquina virtual, habilitamos temporalmente el interfaz de red que mira a la red corporativa y podemos capturar estos procesos:

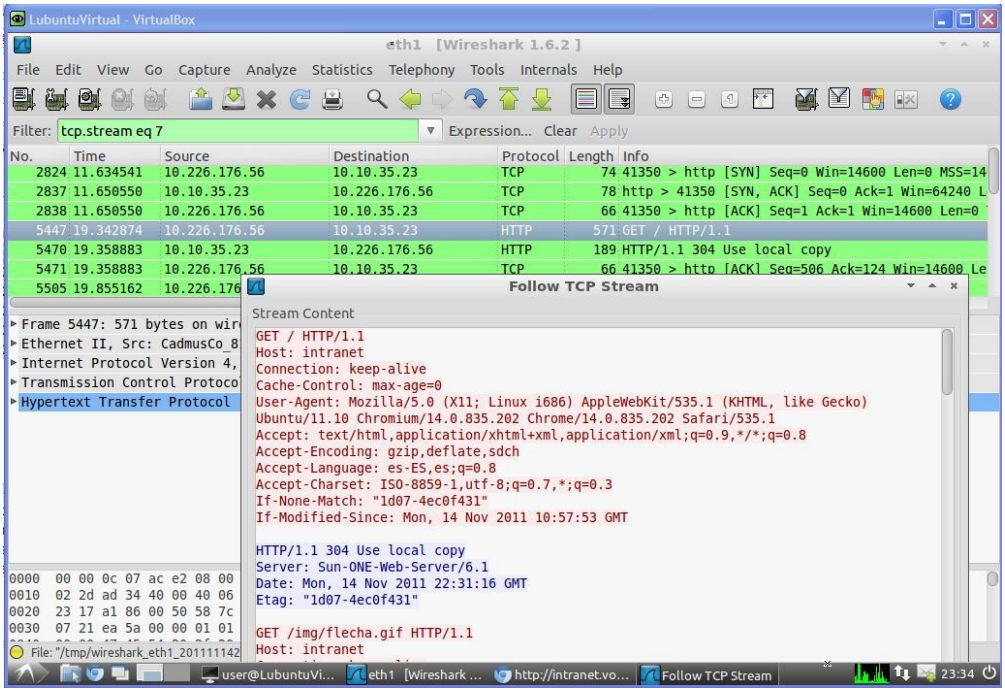
- Resolución de MAC mediante ARP contra otra máquina en la red corporativa.



- Ping ICMP contra otra máquina de la red corporativa.



- Apertura de la web de la intranet (HTTP).



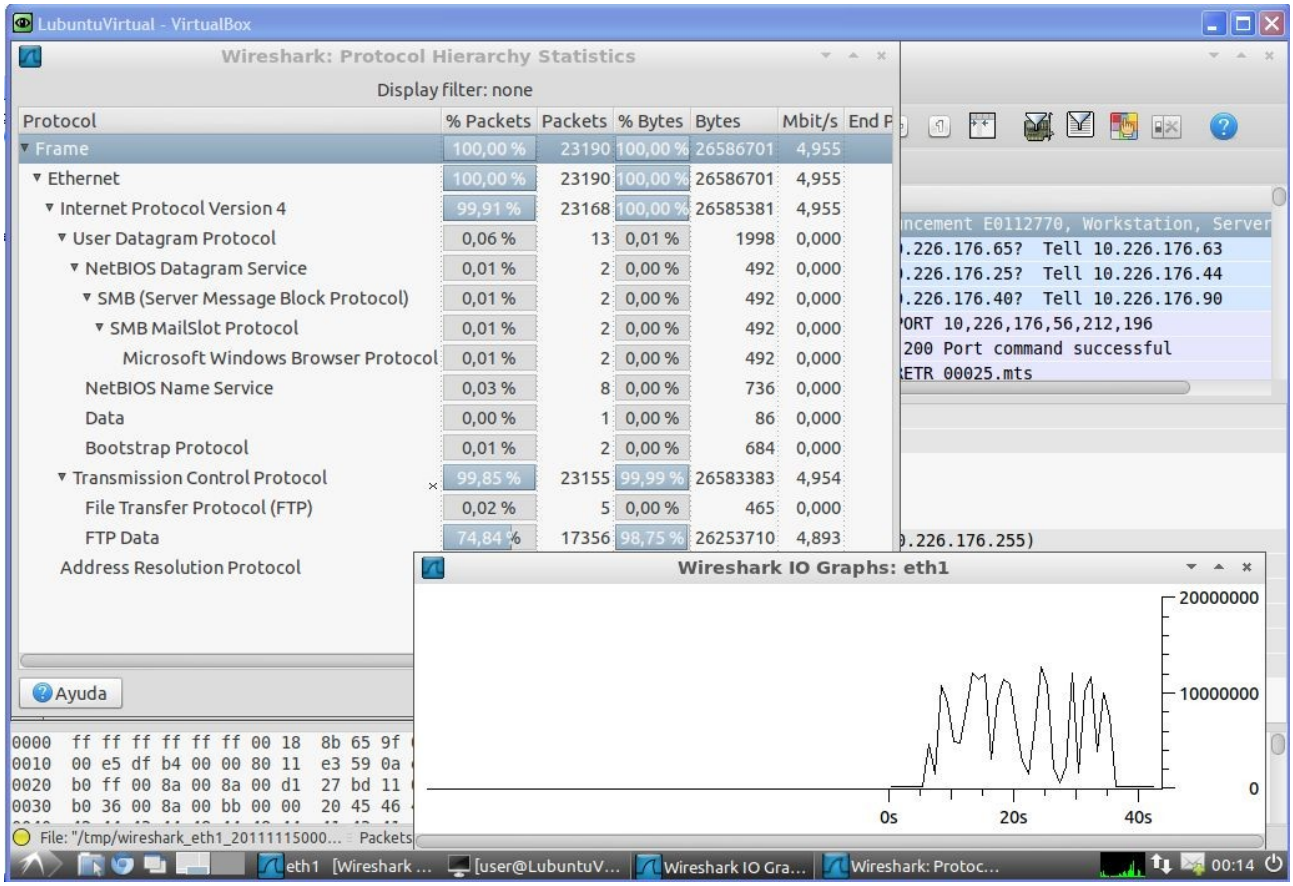
b) Test de velocidad captura

Para probar el buen rendimiento en cuanto a captura del prototipo, vamos a lanzar un FTP contra otra máquina de la red y hacemos un análisis de paquetes transmitidos y velocidad.

Según los resultados arrojados por el cliente FTP, se recibió un fichero de unos 26 Mbytes a una velocidad media de unos 7,4 Mbps durante la transferencia que duró 29 segundos.

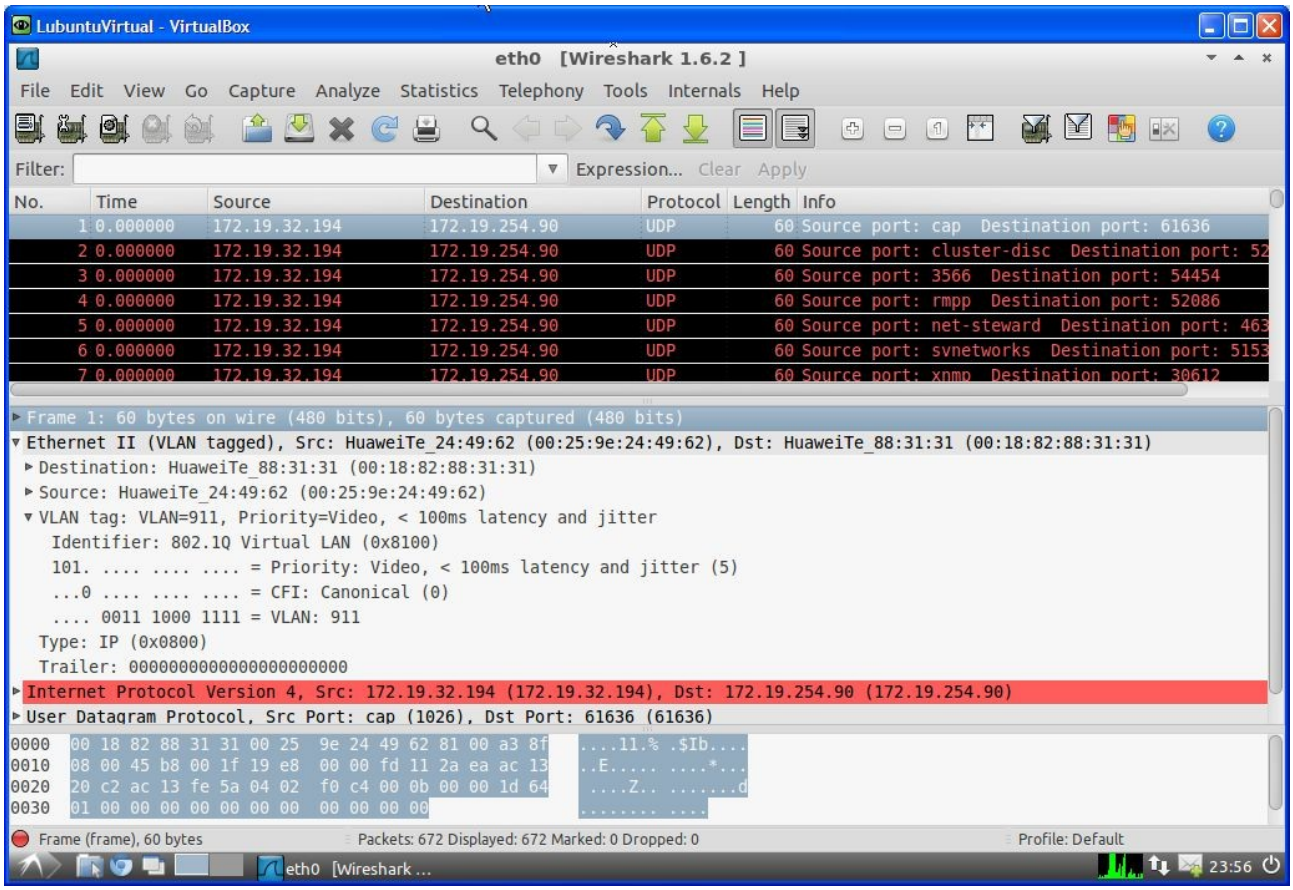
En el siguiente pantallazo se muestra lo que capturó el prototipo y que se ajusta bastante bien a lo esperado. Se capturaron unos 26Mbytes de protocolo FTP. La captura fue de 43 segundos, y la velocidad media es 4,9Mbps. Si asumimos que fuera de los 29 segundos la captura es prácticamente 0, podemos decir que la velocidad de transferencia media fue 4,9*43/29 = 7,3 Mbps.

Además, en la gráfica de IO se aprecian unos 30 segundos con un rizado bastante importante con picos de por encima los 10 Mbps:



c) Captura de traza de 802.1Q (taggeado VLAN) de interfaz IuB de un nodo B remoto

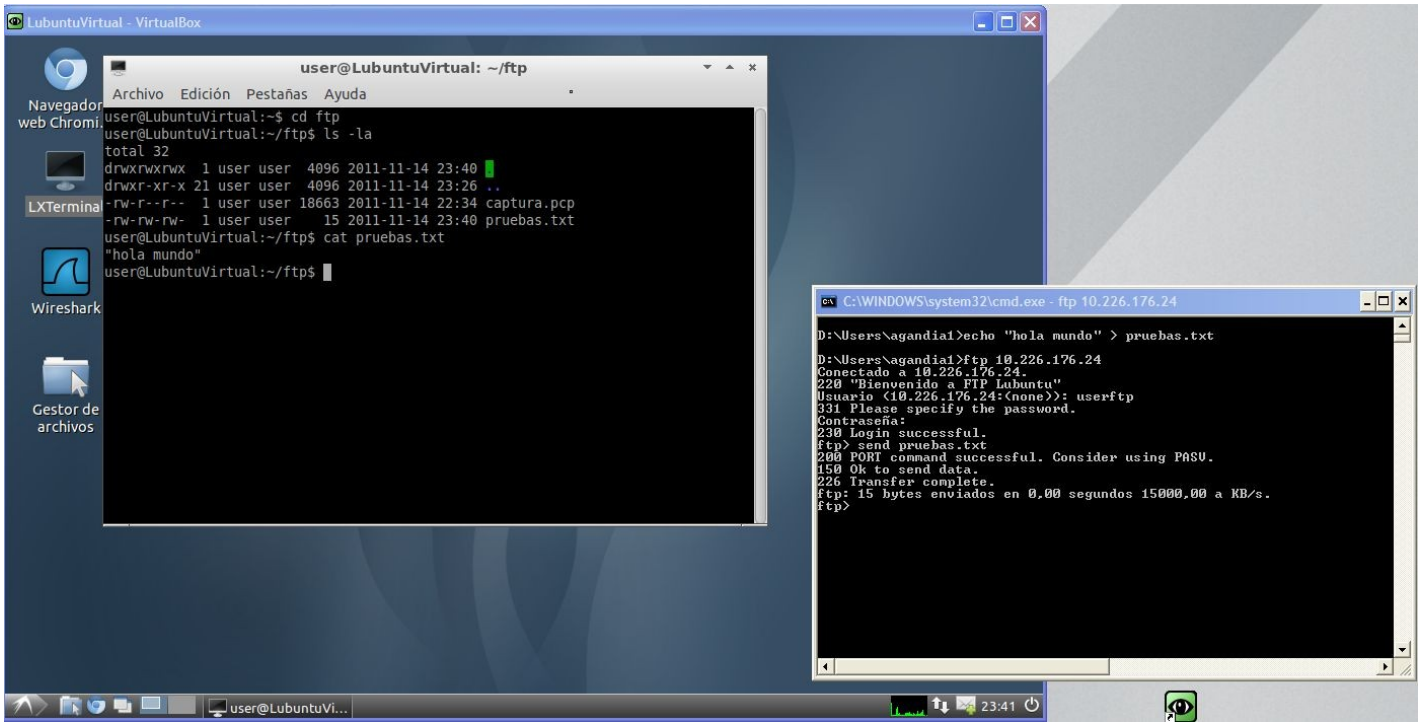
En un solo paso hacemos las dos pruebas. Vemos si somos capaces de ver los paquetes que envía un nodo B remoto y como éste manda paquetes taggeados 802.1Q comprobamos que se ven bien las etiquetas. Reconfiguramos la red de telecomunicaciones para que actúe como Remote Port Mirroring y obtenemos una captura correcta de paquetes del nodo B donde se muestra uno de ellos que fue por la VLAN=911 y Priority=5:



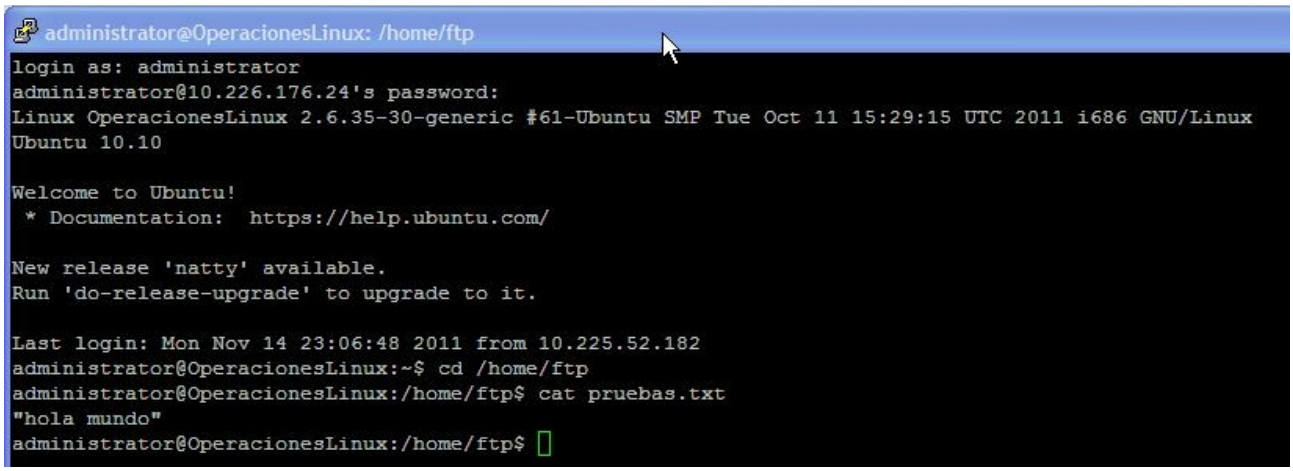
d) Acceso remoto

Debemos probar que podemos:

- Acceder a la máquina virtual con wireshark (VNC). Las anteriores capturas son de VNCs de la máquina virtual.
- Intercambiar ficheros con la plataforma (FTP).

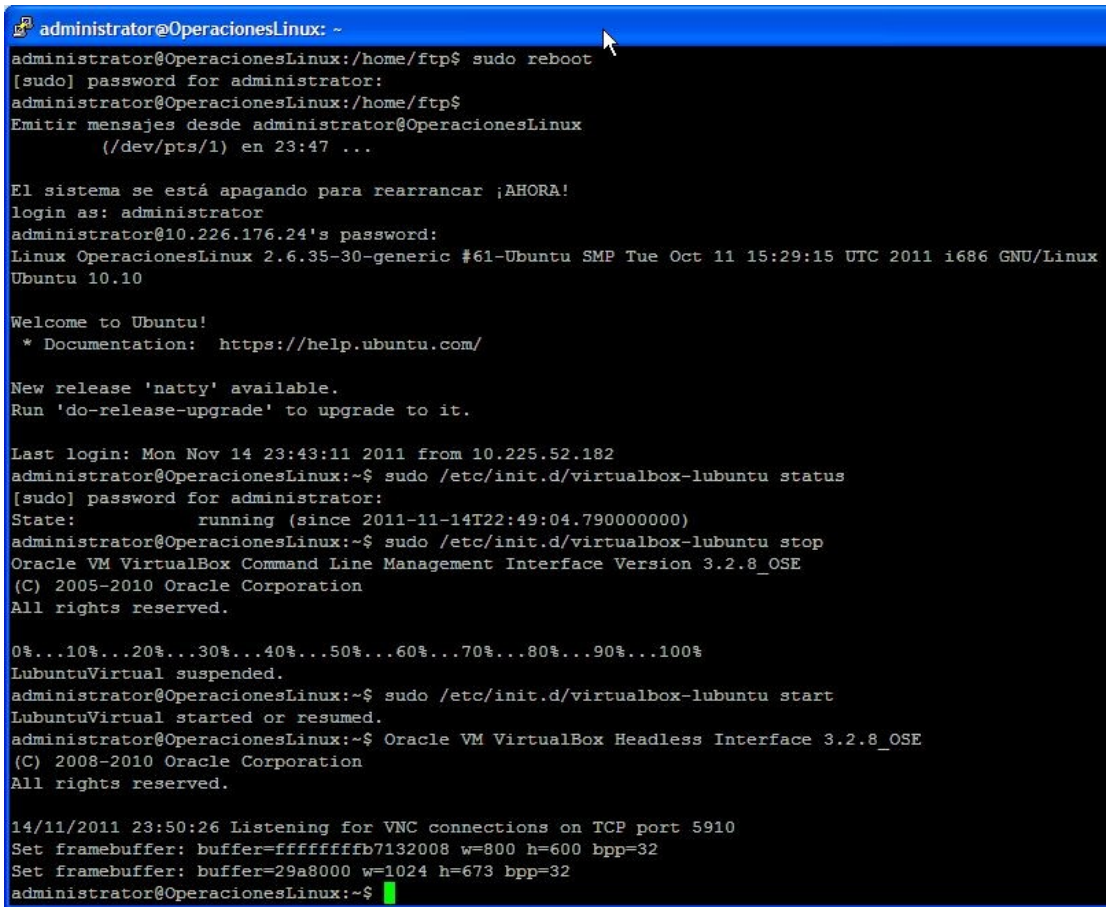


- Acceder por línea de comandos al sistema anfitrión (SSH) en el caso del prototipo fijo.



e) Arranque de todos los servicios automáticamente

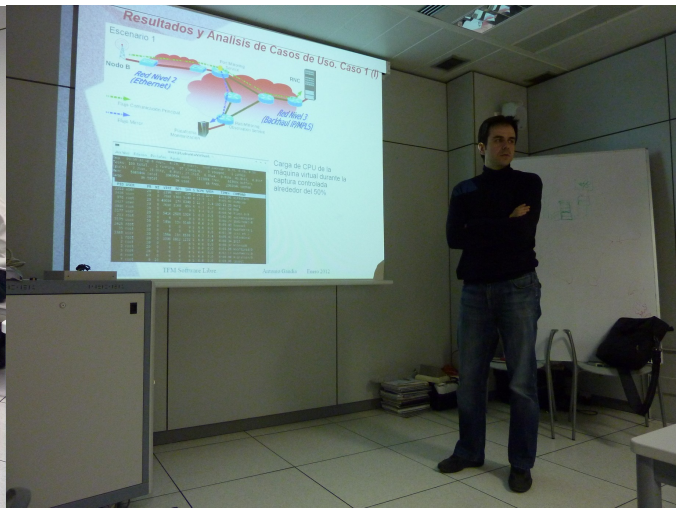
Los servicios SSH, FTP arrancan automáticamente tal cual salen de la instalación (out-of-the-box). Vamos a comprobar que funciona también el servicio creado para la máquina virtual virtualbox. Reiniciamos la máquina y entramos en modo consola, comprobamos el estado (running), lo paramos (hace un salvado del estado de la máquina virtual) y lo rearrancamos.



6.3. Documentación y formación de usuarios

Aunque la plataforma implantada ha sido pensada como un prototipo para evaluar el grado de utilidad de la misma, sí que ha sido intención desde el principio, que esté disponible para un grupo reducido de técnicos que la prueben y ayuden a mejorarla y recomendar su extensión (o no). Este grupo de usuarios es el departamento de operaciones de la región de Levante e islas Baleares.

A ellos se les ha dedicado una sesión de charla formativa (a nivel usuario, no administrador). Además, como el autor de este trabajo es parte de dicho departamento estaré dando soporte al resto de usuarios, además de realizar las tareas de administración de la plataforma.



Por último, se ha realizado un pequeño manual de usuario que sirve como guía para poder comenzar a usar la plataforma. El contenido de dicha guía son estos apartados.

1. Introducción
2. Configuración del Remote Port Mirroring en Backhaul IP
3. Acceso a la plataforma de traceo
4. Captura de una traza
5. Opciones de análisis de una traza
6. Intercambio de ficheros con la plataforma

Con la presente memoria, se anexa otro fichero con una versión de la misma, de la que se han eliminado algunos capítulos de información crítica por su confidencialidad.

7. Resultados y Análisis de Ejemplos de Casos de Uso

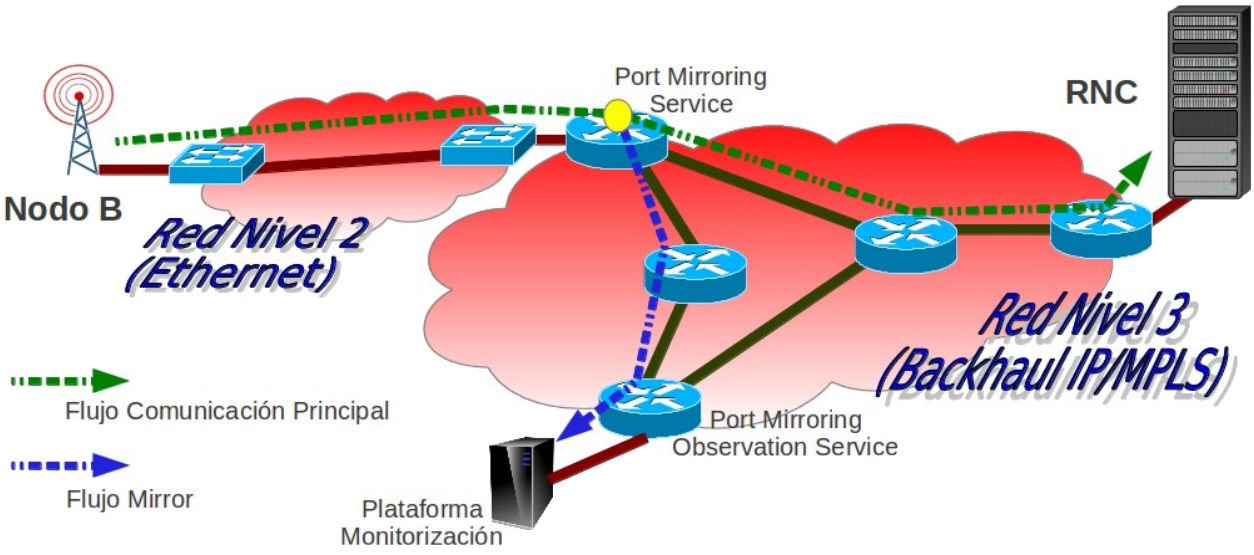
Para finalizar con el trabajo del proyecto final de máster, vamos a someter a la plataforma construida a unas pruebas donde se verifique su utilidad en cuanto a capacidad de análisis de redes de telecomunicaciones basadas en protocolos Ethernet de forma remota.

Ya en el capítulo de implementación, se realizaron las pruebas de aceptación que verifican un correcto funcionamiento de la plataforma de acuerdo a su diseño y requerimientos. En este capítulo final, nos centraremos en hacer pruebas en escenarios reales y analizar con nuestra plataforma los tipos de tráfico que viajan por el interfaz IuB de la red UTRAN.

7.1. Escenario A. Monitorización de un nodo B en servicio real a través de un remote port mirroring efectuado por la red Backhaul

a) Planteamiento de escenario y captura

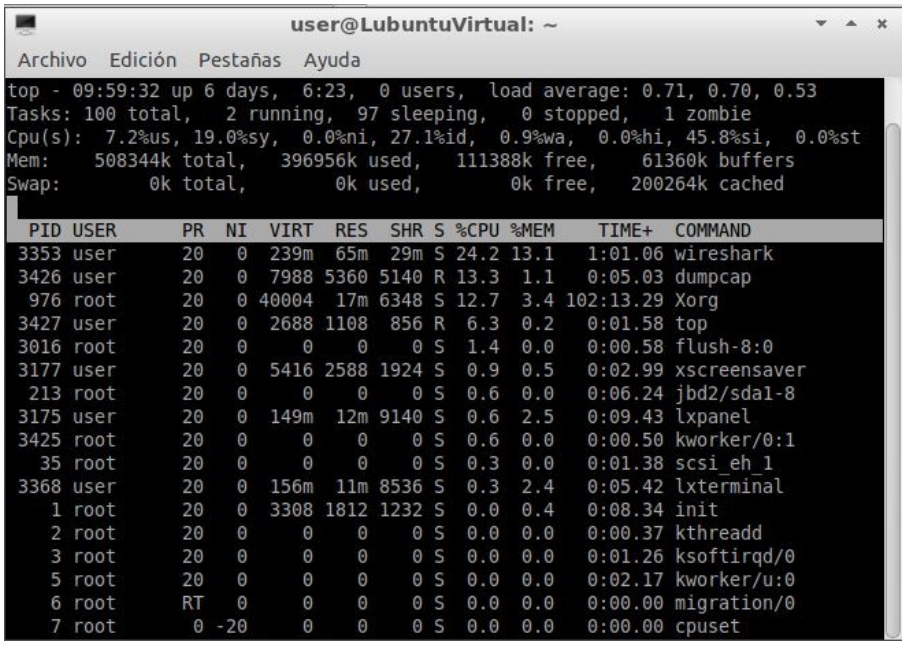
En la siguiente figura vemos el montaje para este escenario.



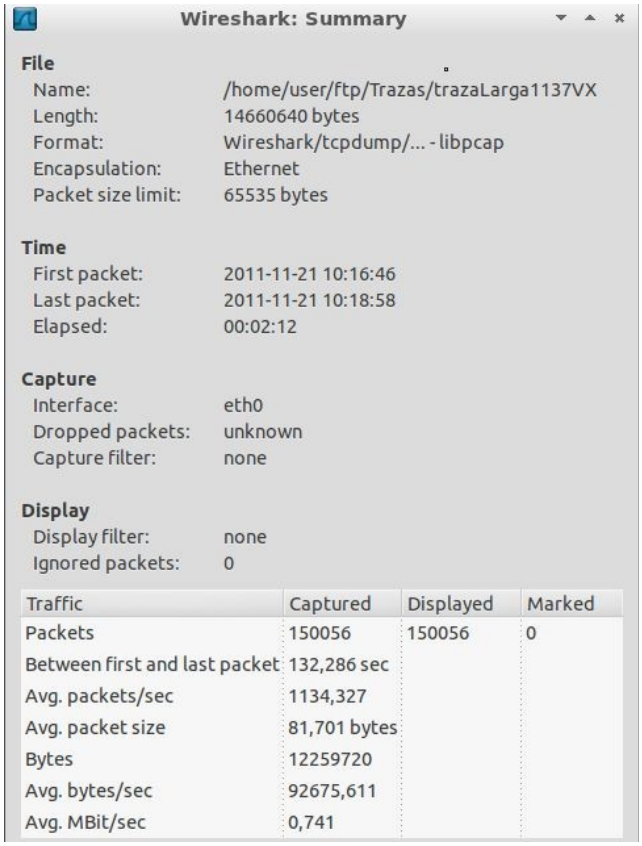
Recordar que las comunicaciones del nodo B con el resto de la red están segmentadas en 2 VLANs:

- Una VLAN reservada para transmisión de señalización, sincronismo y tráfico de usuario con la RNC. Existen diferentes tipos de tráfico priorizados a nivel 3 por DSCP y a nivel 2 por Prio/CoS. El nodo B, RNC (y equipos de sincronización temporal) y puertos de Backhaul tienen IPs de una de las clases B reservadas como privadas en el estándar (172.x.x.x).
- Otra VLAN reservada para transmisión de información de gestión remota del nodo B (intérprete de comandos, ficheros de configuración, descarga de software,...). En principio tiene priorización DSCP Best Effort y la más baja CoS (0), aunque se reserva un canal CoS=3 para comunicaciones de alta prioridad. En esta VLAN, tanto el nodo B (que tiene dos direcciones, una física y un loopback interno), como la máquina de entrada al sistema de gestión como los puertos de Backhaul tienen IPs de la clase A privada estándar 10.x.x.x.

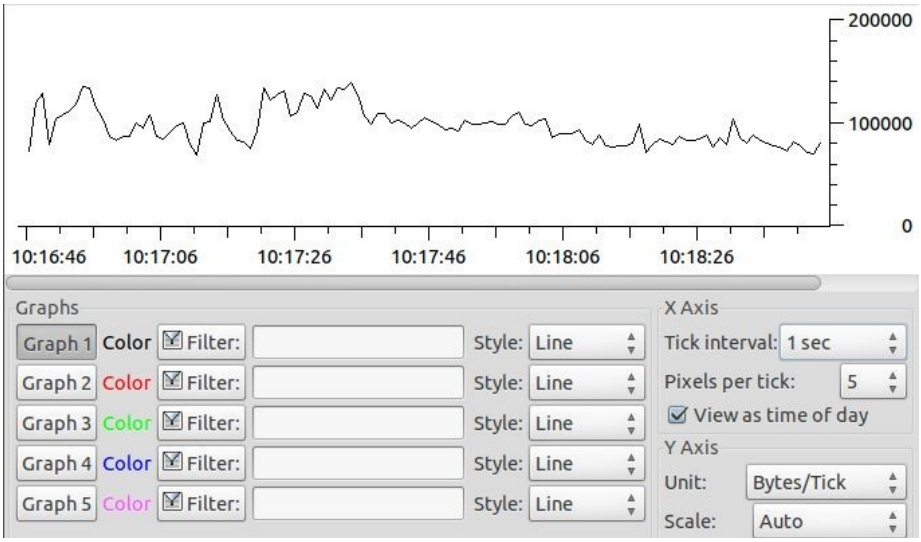
Elegimos un nodo B de la red real en servicio y configuramos el escenario para capturar sus comunicaciones en nuestra plataforma. En primer lugar comprobamos que el sistema no se encuentra saturado y vemos que la carga de la CPU virtual es bastante estable alrededor del 50% (se la lleva principalmente entre los procesos wireshark, dumpcap y Xorg):



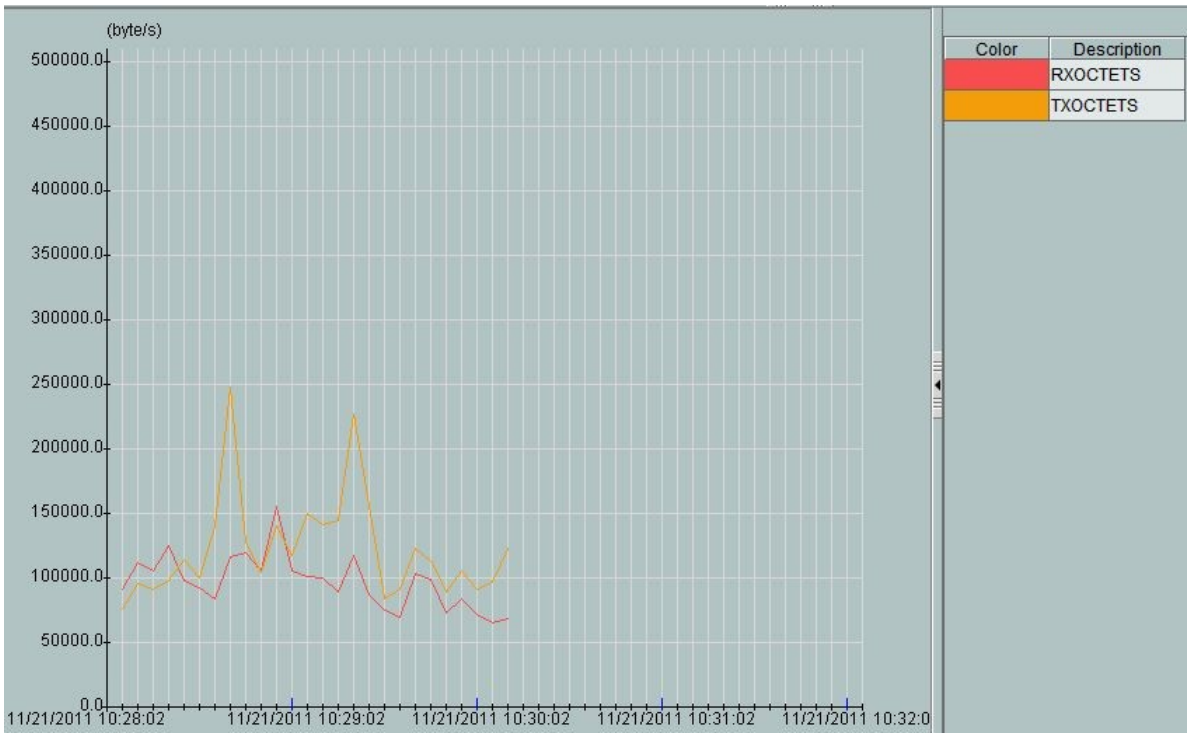
Para un periodo de muestra de unos 2 minutos, capturamos unos 150.000 paquetes (12 Mbytes), a una velocidad media de 0,74 Mbps:



La gráfica de entrada/salida (bytes/sec) de Wireshark para esta captura es:



Comparando con lo que podemos obtener del propio sistema de gestión de los equipos del Backhaul, para el puerto en observación para el mismo periodo de captura es (bytes/sec):



En ella vemos en rojo, los paquetes que recibe el Backhaul (enviados por el nodo B o uplink) y en amarillo los que envía el Backhaul (que manda la red hacia el nodo B/usuario, o downlink). Mientras que el tráfico de voz y señalización es bastante simétrico, no es así para el tráfico de datos, puesto que suele haber más descargas por parte de los usuarios y por ello mayor downlink.

En el interfaz capturado por Wireshark debería verse la suma de ambos tráficos... pero no es así, solo vemos uno de ellos (por el volumen y ausencia de picos, deducimos que el rojo). ¿Por qué?

Usando el analizador de conversaciones, podemos encontrar la razón. A nivel Ethernet, vemos que sólo tenemos dos tipos de conversaciones, una entre las MACs xxx:88:31:31 --- xxx:24:49:62 y que tiene todo el peso del tráfico y otra residual desde la MAC xxx:8A:AA:88.

Ethernet: 2 | Fibre Channel | FDDI | IPv4: 4 | IPv6 | IPX | JXTA | NCP | RSVP | SCTP: 2 | TCP: 1 | Token Ring | UDP: 639 | USB | WL

Ethernet Conversations

Address A	Address B	Packets	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B
00:18:82:88:31:31	00:25:9e:24:49:62	149 926	12 251 920	0	0	149 926	12 251 920
00:18:82:8a:aa:88	01:80:c2:00:00:02	130	7 800	130	7 800	0	0

La segunda conversación se corresponde a los paquetes que manda el Backhaul en el puerto donde hemos conectado la máquina, para buscar otros equipos de la red Backhaul (y no los encuentra porque está conectado nuestro PC). No hemos de prestarle atención y es un efecto colateral.

La primera conversación, es la que va entre el equipo Backhaul remoto donde se monta el punto de observación y el nodo B. Pero... vemos que en dirección A->B no hay paquetes!!!

Si nos vamos a nivel IP, se ve muchísimo más claro:

Ethernet: 2 | Fibre Channel | FDDI | IPv4: 4 | IPv6 | IPX | JXTA | NCP | RSVP | SCTP: 2 | TCP: 1 | Token Ring | UDP: 639

IPv4 Conversations

Address A	Address B	Packets	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B
172.19.32.194	172.19.254.90	149 943	12 252 209	149 906	12 249 471	37	2 738
10.18.11.19	10.115.88.84	16	2 129	0	0	16	2 129
172.19.32.194	172.19.251.6	2	160	2	160	0	0
172.19.32.194	172.19.251.2	2	160	2	160	0	0

El nodo B tiene la IP 172.19.32.194. Se ven 3 conversaciones, una de gran volumen (con la RNC) y un par de menor volumen (con equipos de sincronismo). Para el otro rango de direcciones, el nodo B tiene la ip física 10.115.88.84 y se ve una única conversación con el sistema de gestión. Los paquetes capturados son únicamente desde las IPs del nodo B hacia el interior del Backhaul.

Haciendo averiguaciones sobre la funcionalidad Remote Port Mirroring del Backhaul con el propio suministrador de los equipos de Backhaul, llegamos a la conclusión que *en la release actual de software, esta funcionalidad sólo captura el sentido "ingress", es decir, los paquetes entrantes al Backhaul*. Se ha hecho una solicitud para que en las próximas releases (en unas semanas) también podamos tener el sentido "egress", no obstante estos resultados quedarán fuera del proyecto actual por plazos de tiempos.

Así pues, continuamos el análisis de este caso, sabiendo que para él, sólo tendremos uno de los sentidos de la comunicación.

Para comprobar la validez del sistema montado cuando tengamos la funcionalidad de Remote Port Mirroring ingress/egress, en el siguiente apartado haremos capturas bidireccionales mediante un switch específico.

b) Análisis de Señalización

La señalización en interfaces IuB de UMTS, se establece a través de un par de puertos reservados para señalización SCTP (protocolo de nivel de aplicación llamado NBAP que tiene dos subprocesos un por puerto, el NCP y el CCP).

Viendo las conversaciones, Wireshark ha detectado 2 conversaciones SCTP, que se corresponden a los protocolos NCP y CCP montados sobre los puertos indicados por Wireshark:

SCTP Conversations

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B
172.19.254.90	58080	172.19.32.194	1025	6 067	562 894	0	0	6 067	562 894
172.19.254.90	58080	172.19.32.194	1024	2 884	306 636	0	0	2 884	306 636

Utilizando las herramientas de Telefonía de Wireshark, podríamos hacer un análisis más exhaustivo de las asociaciones, pero al no tener la comunicación bidireccional, no tiene sentido y la herramienta no se comporta bien. Haremos un análisis más exhaustivo del tráfico de señalización en el escenario siguiente donde sí dispongamos de comunicaciones bidireccionales.

c) Análisis de Tráfico de Gestión (O&M)

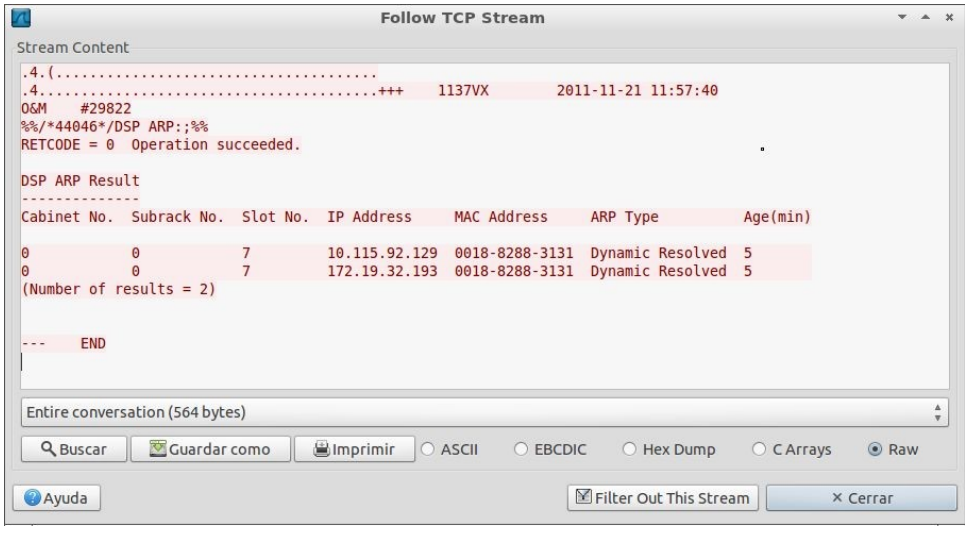
Volviendo a las conversaciones, vemos una única conversación mantenida a través de TCP:

TCP Conversations

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B
10.18.11.19	41388	10.115.88.84	6007	7	1 139	0	0	7	1 139

Y es la que lleva toda la comunicación de gestión. Si hiciéramos un análisis del stream de esta conversación obtendríamos que está basada en en un intercambio de paquetes de protocolos TURN (Transversal Using Relays around NAT) y STUN (Session Transversal Utilities for NAT), que son protocolos usados para comunicación entre elementos detrás de NAT o Firewall.

Podemos incluso decodificar su contenido y vemos en texto plano los comandos que se envían al nodo B. Por ejemplo, si enviamos al nodo B el comando DSP ARP:; (que hace un printado de la tabla de ARP aprendida) tenemos (como sólo vemos el sentido entrante del backhaul, sólo tenemos la respuesta del nodo y no vemos cómo lo envió el sistema del gestión):



d) Análisis de Tráfico de Sincronismo

Toda red de telecomunicaciones debe permanecer sincronizada entre todos los elementos que la componen para evitar problemas por deslizamiento (adelantos o retrasos) de paquetes en protocolos de tiempo real (p.e. Transmisión de voz, especialmente en handovers entre dos estaciones base).

Existen muchos métodos de mantener sincronizados los elementos de la red. Uno de ellos, es mediante el protocolo PTP (Precision Time Protocol) definido en el estándar IEEE 1588. En él, se establece una comunicación bidireccional entre dos elementos, enviando el nodo central (o IPClock) correciones en nanosegundos al elemento remoto que queremos sincronizar.

En nuestra captura, vimos que habían conversaciones entre nodo B y nodos centrales de sincronismo (redundantes). Si vemos el desglose de estas conversaciones, aunque nos falta un sentido de la comunicación, vemos que Wireshark decodifica correctamente los paquetes del protocolo PTP que viajan como protocolo de aplicación sobre UDP puerto 320:

```
 43301 36.732651   172.19.32.194   172.19.251.6    PTPv2          Unknown PTP Message (4)
119060 99.575980   172.19.32.194   172.19.251.6    PTPv2          Unknown PTP Message (4)

⊞ Frame 43301: 80 bytes on wire (640 bits), 80 bytes captured (640 bits)
⊞ Ethernet II, Src: HuaweiTe_24:49:62 (00:25:9e:24:49:62), Dst: HuaweiTe_88:31:31 (00:18:82:88:31:31)
⊞ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 911
⊞ Internet Protocol, Src: 172.19.32.194 (172.19.32.194), Dst: 172.19.251.6 (172.19.251.6)
⊞ User Datagram Protocol, Src Port: ptp-general (320), Dst Port: ptp-general (320)
⊟ Precision Time Protocol (IEEE1588)
  ⊞ 0000 .... = transportSpecific: 0x00
    .... 0100 = messageId: Unknown (0x04)
    .... 0010 = versionPTP: 2
    messageLength: 34
    subdomainNumber: 0
  ⊟ flags: 0x0000
      0... .... .... .... = PTP_SECURITY: False
      .0.. .... .... .... = PTP profile Specific 2: False
      ..0. .... .... .... = PTP profile Specific 1: False
      .... .0.. .... .... = PTP_UNICAST: False
      .... ..0. .... .... = PTP_TWO_STEP: False
      .... ...0 .... .... = PTP_ALTERNATE_MASTER: False
      .... .... ..0. .... = FREQUENCY_TRACEABLE: False
      .... .... ...0 .... = TIME_TRACEABLE: False
      .... .... .... 0... = PTP_TIMESCALE: False
      .... .... .... .0.. = PTP_UTC_REASONABLE: False
      .... .... .... ..0. = PTP_LI_59: False
      .... .... .... ...0 = PTP_LI_61: False
  ⊟ correction: 0,000000 nanoseconds
      correction: Ns: 0 nanoseconds
      SubNs: 0,000000 nanoseconds
    ClockIdentity: 0x00259e00ac1320c2
    SourcePortID: 1
    sequenceId: 0
    control: Other Message (5)
    logMessagePeriod: 6
```

e) Análisis de Tráfico de Usuario

En cuanto al tráfico de usuario, en un interfaz IuB se pueden establecer diferentes caminos para cada tipo de tráfico, siguiendo cada uno de ellos una parametrización de QoS distinta. Esto se implementa con distintos etiquetados de DSCP a nivel IP y CoS/Prio a nivel Ethernet.

En nuestra traza, Wireshark no es capaz de distinguir ningún protocolo específico para el plano de usuario, y en las utilidades de análisis se ve como un bloque de muchas conversaciones UDP desde distintos puertos. Lo que sí que podemos sacar es un análisis estadístico a nivel de DSCP para distinguir entre los diferentes tipos de tráfico y ver sus características/volumen.

Wireshark originalmente no lleva esta funcionalidad, y si quisiéramos automatizarla podríamos hacer uso de herramientas (con licencias libres) como C5 Sigma, que a partir de un fichero de captura producen una base de datos relacional sobre la que poder lanzar consultas SQL (cargándolas en un MySQL o PostgreSQL).

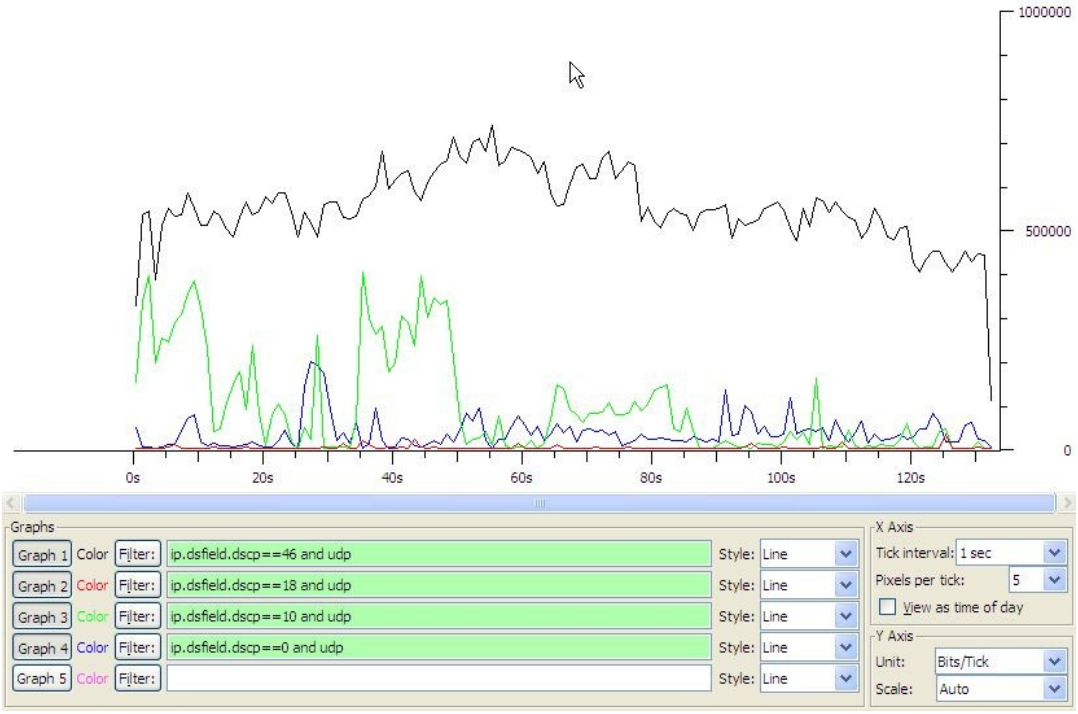
Para el caso que nos ocupa, no parece compensar el esfuerzo para las pruebas que podemos hacer. Así pues lo más rápido y operativo, será hacer un filtro por DSCP (p.e. `"ip.dsfield.dscp==46 and udp"`) y anotar las estadísticas. Así tenemos la siguiente tabla para unos cuantos tipos de tráfico:

		DSCP=46 (EF, Expedited Forward)	DSCP=18 (AF21, Assured Forward 21)	DSCP=10 (AF11, Assured Forward 11)	DSCP=0 (BE,Best Effort)
Convesaciones UDP		509	8	97	61
Rango puertos RNC		29122-64896	29758-57818	30310-64856	29714-64946
Rango puertos NodoB		1024-5822	2174-4590	1086-5794	1130-5773
Paquetes		123053	287	12381	5156
%Paquetes		*82,0%*	*0,2%*	*8,3%*	*3,4%*
Bytes		9170000	30704	1568966	603691
%Bytes		*74,8%*	*0,3%*	*12,8%*	*4,9%*
Histograma Longitud Paquetes	**0-19**	*0,0%*	*0,0%*	*0,0%*	*0,0%*
	20-39	*0,0%*	*0,0%*	*0,0%*	*0,0%*
	40-79	*49,9%*	*5,9%*	*0,6%*	*1,2%*
	80-159	*50,1%*	*90,2%*	*82,0%*	*88,0%*
	160-319	*0,0%*	*3,8%*	*16,3%*	*10,2%*
	320-639	*0,0%*	*0,0%*	*1,1%*	*0,6%*
	640-1279	*0,0%*	*0,0%*	*0,0%*	*0,0%*
	1280-2559	*0,0%*	*0,0%*	*0,0%*	*0,0%*
	2560-5119	*0,0%*	*0,0%*	*0,0%*	*0,0%*
	>5120	*0,0%*	*0,0%*	*0,0%*	*0,0%*

Comprobamos que en ninguna de las conversaciones hemos observado texto en plano legible por Wireshark (p.e. código de http/ftp de aplicaciones de clientes). Esto es así porque, por seguridad, las conversaciones privadas de usuario son encriptadas según los protocolos para UTRAN.

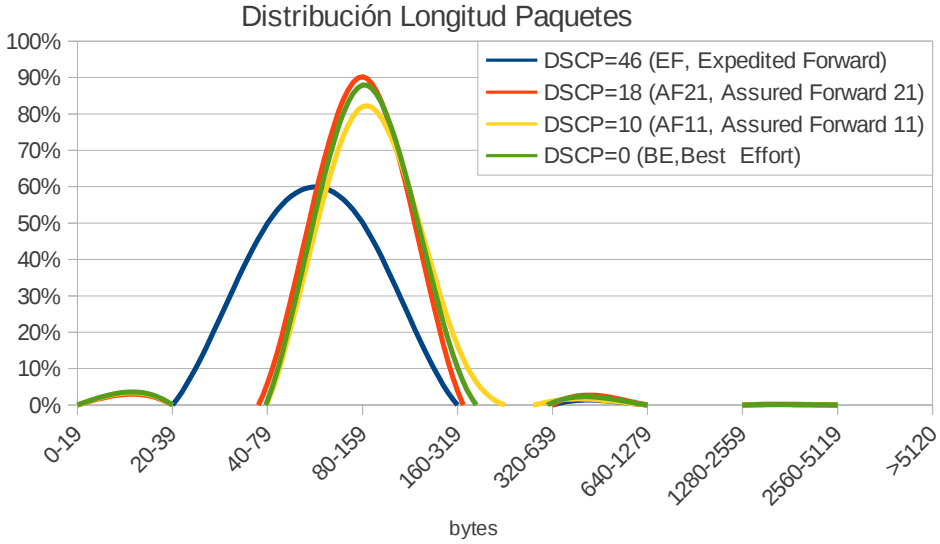
Es el operador quien juega con qué tipo de tráfico va por cada uno de los DSCPs, pero normalmente, los de mayor prioridad (DSCP más alto) suelen ser tráfico con requisitos de tiempo real (p.e. Voz o señalización del interfaz radio) y los de menor prioridad (DSCP más bajo) suelen corresponderse a tráficos de paquetes de baja prioridad (p.e. descargas de datos).

Vemos que la mayor parte del tráfico se corresponde al DSCP de mayor prioridad (que suele ser el tráfico de voz). Si miramos la distribución en tiempo de bits/seg, vemos que el tráfico de voz es bastante constante en el tiempo, mientras que el de datos, va a ráfagas:



Respecto a los puertos utilizados, ninguno de ellos corresponde a un protocolo reconocible. Pero sí que vemos que mientras la RNC asigna puertos en al banda alta (29.000-65.000), el nodo B los va asignado en una banda inferior (1.024-6.000).

En cuanto al histograma de longitudes de paquetes, vemos claramente que los paquetes de alta prioridad (voz) suelen ser más pequeños alrededor de 80 bytes y sin apenas paquetes largos. Para los de prioridad más baja, la media es más alta y además aparecen paquetes de longitudes largas.

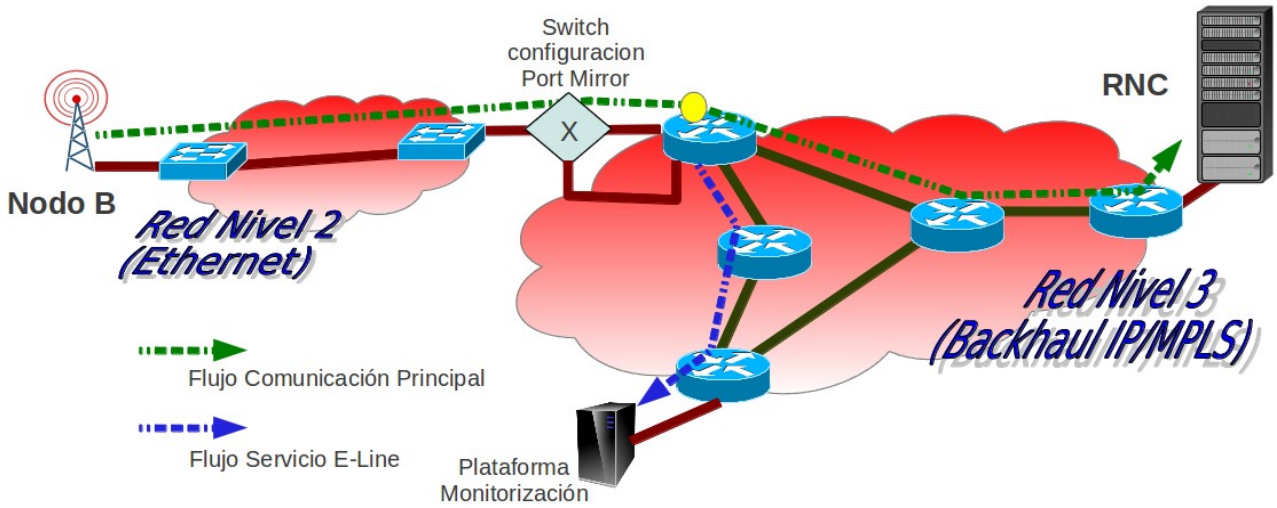


7.2. Escenario B. Monitorización de un nodo B de pruebas a través de un port mirroring efectuado por switch

a) Planteamiento de escenario y captura

Dado que la funcionalidad para poder tener comunicaciones bidireccionales a través de un remote port mirroring tardará en implementarse más allá de la entrega de la presente memoria, hemos decidido hacer pruebas del comportamiento de la plataforma sobre otro tipo de mirroring. Para ello, adquirimos un switch[6] programable con buenas características para no afectar al servicio del nodo B[7], capaz de desviar a un puerto las comunicaciones destinadas a los otros y a un precio bajo.

Se montó el siguiente escenario, en horas de bajo tráfico (de noche), puesto que el proceso de sacar un cable y pinchar el switch iba a suponer un corte de unos segundos que afectaría al servicio real de voz y datos en la zona de cobertura del nodo B.



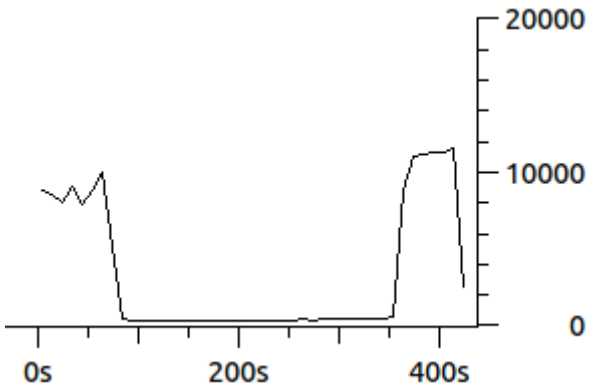
Desde el primer router del backhaul, aparte de la comunicación principal, se lanzó hasta el puerto donde está conectada la plataforma de monitorización un servicio E-Line o VPL, que consiste en emular un cable ethernet desde un puerto de entrada a otro entre diferentes equipos de la red.

Nuestro principal objetivo para este escenario es capturar procedimientos de UTRAN estándar fácilmente identificables para comprobar la bondad del sistema. Pensamos en capturar el proceso de configuración y activación de las celdas radio de un nodo B. Para ello, el proceso que hicimos fue:

- Iniciar la captura tal cual estaba el nodo en funcionamiento.
- Tras unos segundos, resetear el nodo B para reinicializar todos los procesos.
- Esperar a que se restableciera el servicio en la zona (levantado de las celdas UMTS a nivel radio).
- Capturar unos primeros segundos y detener la captura.

6 Modelo Netgear GS105D
7 Ancho de banda de procesado de 10Gbps, tratamiento de VLAN 802.1Q y QoS 802.1p (con 4 colas de priorización sobre el campo Prio/CoS)

Como era de esperar, obtuvimos 3 zonas diferenciadas de paquetes capturados (estado normal, tiempo de reseteo -unos 5 minutos- y recuperación del servicio):



Ahora sí que obtuvimos conversaciones bidireccionales para todas las IPs que intervienen en un interfaz IuB:

IPv4 Conversations: 8

Address A	Address B	Packets	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B
172.19.32.194	172.19.254.90	129 176	10 386 325	44 540	3 578 592	84 636	6 807 733
172.19.32.194	172.19.251.6	6 765	608 952	5	420	6 760	608 532
10.18.11.19	10.115.88.84	344	71 377	153	12 869	191	58 508
172.19.32.194	172.19.251.2	2 274	204 804	6	520	2 268	204 284
192.168.0.238	192.168.0.255	30	3 504	30	3 504	0	0
10.18.11.19	10.115.92.130	6	564	3	282	3	282
0.0.0.0	255.255.255.255	9	3 264	9	3 264	0	0
172.19.32.194	192.168.0.239	399	25 536	0	0	399	25 536

b) Análisis de Señalización

Como dijimos anteriormente, la señalización en el IuB de UTRAN viaja sobre el protocolo de nivel de transporte SCTP, estableciendo dos puertos para llevar la capa de aplicación con el protocolo NBAP (canal NCP y canal CCP). Ahora con la comunicación bidireccional, sí que podemos hacer un análisis exhaustivo.

En nuestro escenario, nos fijaremos en el momento en que el nodo B tras el reset, comienza el restablecimiento del servicio (alrededor del segundo 200 de la captura).

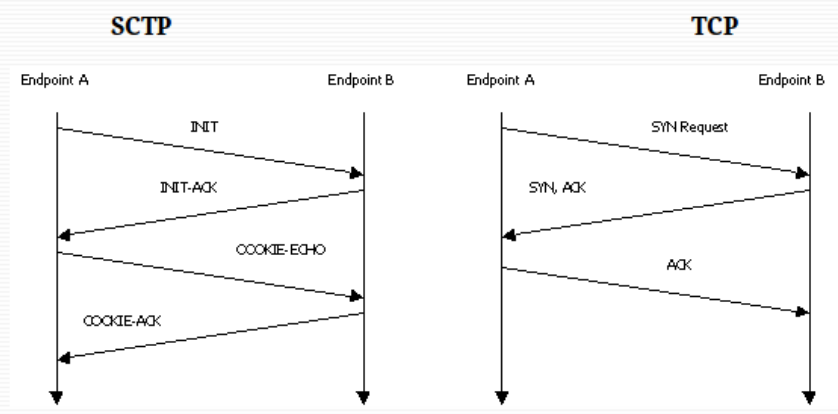
A nivel SCTP vemos, tras un periodo de tiempo en el que la RNC trata de buscar inútilmente al Nodo B, establece los dos canales SCTP por dos puertos diferentes:

Filter: sctp — Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.19.254.90	172.19.32.194	SCTP	142	HEARTBEAT
146	8.000056	172.19.254.90	172.19.32.194	SCTP	142	HEARTBEAT
291	16.000284	172.19.254.90	172.19.32.194	SCTP	142	HEARTBEAT
344	19.000273	172.19.254.90	172.19.32.194	SCTP	60	ABORT
772	42.782643	172.19.32.194	172.19.254.90	SCTP	82	INIT
773	42.784422	172.19.254.90	172.19.32.194	SCTP	798	INIT_ACK
774	42.786716	172.19.32.194	172.19.254.90	SCTP	766	COOKIE_ECHO
775	42.788758	172.19.254.90	172.19.32.194	SCTP	60	COOKIE_ACK
776	42.790398	172.19.254.90	172.19.32.194	DUA	82	UNKNOWN [Malformed Packet]
777	42.792521	172.19.32.194	172.19.254.90	SCTP	66	SACK
842	45.830618	172.19.32.194	172.19.254.90	SCTP	82	INIT
843	45.832738	172.19.254.90	172.19.32.194	SCTP	798	INIT_ACK
844	45.834730	172.19.32.194	172.19.254.90	SCTP	766	COOKIE_ECHO
845	45.836262	172.19.254.90	172.19.32.194	SCTP	60	COOKIE_ACK
846	45.837088	172.19.32.194	172.19.254.90	DUA	90	UNKNOWN [Malformed Packet]
847	45.850616	172.19.32.194	172.19.254.90	SCTP	142	HEARTBEAT
848	45.852180	172.19.254.90	172.19.32.194	SCTP	142	HEARTBEAT_ACK
853	46.070654	172.19.32.194	172.19.254.90	SCTP	66	SACK
901	48.824955	172.19.254.90	172.19.32.194	DUA	82	UNKNOWN [Malformed Packet]
902	48.824967	172.19.254.90	172.19.32.194	DUA	90	UNKNOWN [Malformed Packet]

▸ Frame 842: 82 bytes on wire (656 bits), 82 bytes captured (656 bits)
▸ Ethernet II (VLAN tagged), Src: HuaweiTe_24:49:62 (00:25:9e:24:49:62), Dst: HuaweiTe_88:31:31 (00:18:82:88:31:31)
▸ Internet Protocol Version 4, Src: 172.19.32.194 (172.19.32.194), Dst: 172.19.254.90 (172.19.254.90)
▸ Stream Control Transmission Protocol, Src Port: 1025 (1025), Dst Port: 58080 (58080)

Vemos el protocolo de establecimiento de SCTP que es un protocolo orientado a conexión con establecimiento a 4 bandas (diferente al TCP que es a tres), y que tiene los paquetes INIT – INIT_ACK - COOKIE_ECHO - COOKIE_ACK.



Una vez establecida la comunicación, el canal es mantenido con reconocimientos de paquetes HEARTBEAT-HEARTBEAT_ACK.

El paquete de SCTP tiene una estructura bastante similar al TCP, pero la información que contiene se encuentra estructurada en subbloques llamados chunks. Cada chunk puede llevar información de un mensaje de protocolo o información de usuario según el campo type [SCTP]:

Bits	Bits 0 - 7	8 - 15	16 - 23	24 - 31
+0	Source port		Destination port	
32	Verification tag			
64	Checksum			
96	Chunk 1 type	Chunk 1 flags	Chunk 1 length	
128	Chunk 1 data			
...	...			
...	Chunk N type	Chunk N flags	Chunk N length	
...	Chunk N data			

Value	Abbreviation	Description
0	DATA	Payload data
1	INIT	Initiation
2	INIT ACK	initiation acknowledgement
3	SACK	Selective acknowledgement
4	HEARTBEAT	Heartbeat request
5	HEARTBEAT ACK	Heartbeat acknowledgement
6	ABORT	Abort
7	SHUTDOWN	Shutdown
8	SHUTDOWN ACK	Shutdown acknowledgement
9	ERROR	Operation error
10	COOKIE ECHO	State cookie
11	COOKIE ACK	Cookie acknowledgement
12	ECNE	Explicit congestion notification echo (reserved)
13	CWR	Congestion window reduced (reserved)
14	SHUTDOWN COMPLETE	Shutdown complete

Lo que ocurre es que, si vemos los siguientes paquetes con datos de usuario de SCTP (chunk type=0), debería decodificar la señalización UTRAN (entre las características de Wireshark, estaba decodificar NBAP), pero sólo encuentra paquetes mal formados del protocolo DUA (acrónimo de DPNSS/DASS2- User Application). El protocolo DUA es un protocolo que no tiene sentido en este contexto. ¿Por qué ocurre esto? Pues bien, si analizamos el chunk de payload data, vemos que el fabricante está etiquetando (fuera de norma) los paquetes con el tipo de protocolo 10 (payload protocol id):

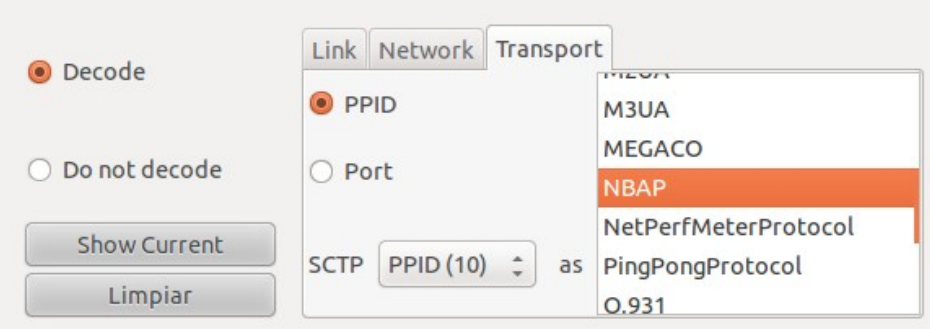
```
▼ Stream Control Transmission Protocol, Src Port: 58080 (58080), Dst Port: 1024 (1024)
    Source port: 58080
    Destination port: 1024
    Verification tag: 0x63682d43
    Checksum: 0xd7370f0d (not verified)
  ▼ DATA chunk(ordered, complete segment, TSN: 447948207, SID: 0, SSN: 0, PPID: 10, payload length: 14 bytes)
    ▶ Chunk type: DATA (0)
    ▶ Chunk flags: 0x03
      Chunk length: 30
      TSN: 447948207
      Stream Identifier: 0x0000
      Stream sequence number: 0
      Payload protocol identifier: DUA (10)
      Chunk padding: 0000
▶ DPNSS/DASS2-User Adaptation Layer
▶ [Malformed Packet: DUA]
```

Cuando según la especificación de SCTP, para NBAP debería ser 25 [SCTP-P]:

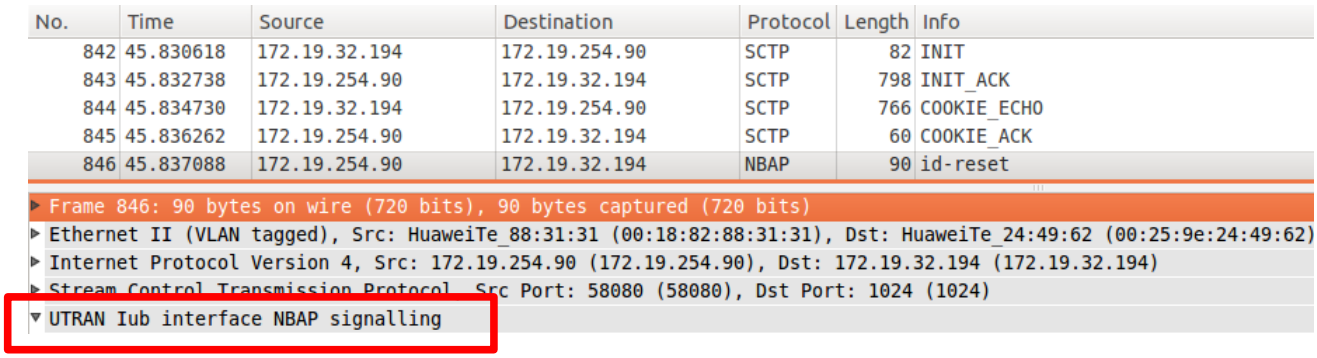
Payload Protocol Identifiers.

Identifier	
0	
1	IUA, ISDN Q.921-User Adaptation.
2	M2UA, MTP2-User Adaptation.
3	M3UA, MTP3-User Adaptation.
4	SUA, Signalling Connection Control Part User Adaptation Layer.
5	M2PA, MTP2 User Peer-to-peer Adaptation Layer.
6	V5UA, V5.2-User Adaptation.
7	H.248.
8	BICC / Q.2150.3.
9	TALI, Transport Adapter Layer Interface.
10	DUA.
11	ASAP, Aggregate Server Access Protocol.
12	ENRP, Enpoint Name Resolution Protocol.
13	H.323.
14	Q.IPC / Q.2150.3.
15	SIMCO.
16	DDP Segment Chunk.
17	DDP Stream Session Control.
18	S1AP, S1 Application Protocol.
19	RUA.
20	HNBAP.
21	ForCES-HP.
22	ForCES-MP.
23	ForCES-LP.
24	SBc-AP.
25	NBAP.

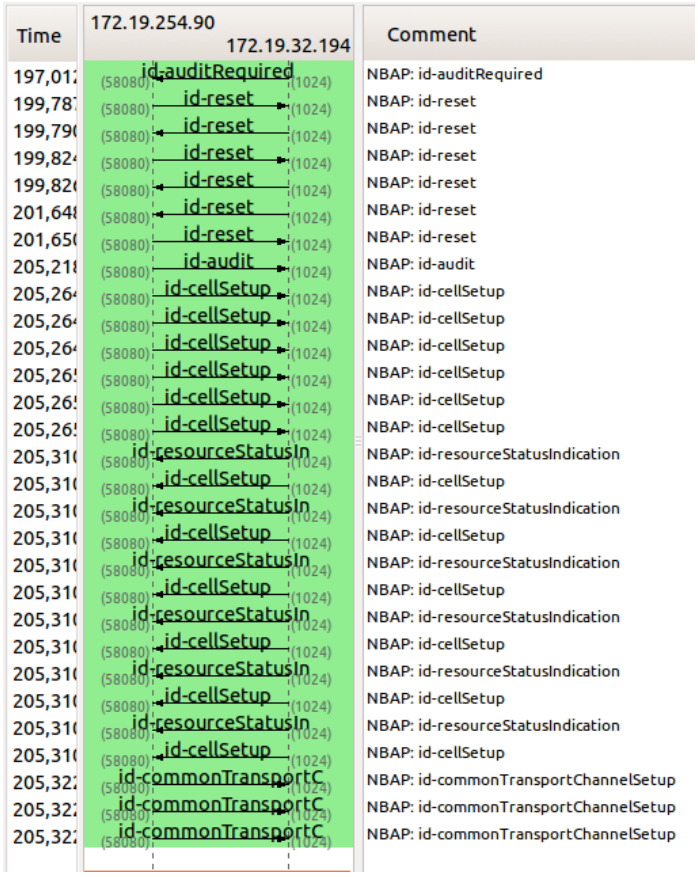
Afortunadamente, en Wireshark están previstas estas contingencias, y podemos decidir con qué protocolo se interpreta el payload. Simplemente debemos ir a “Analyze-Decode As” y configurar el sistema para el nivel OSI 4 (Transport), para el protocolo SCTP, el payload del identificador de protocolo (PPID) 10 sea interpretado como protocolo “NBAP”:



Con esto, nos desaparecen los DUAs mal formados y decodifica perfectamente toda la señalización NBAP que viaja por el Iub:



Así, podemos seguir todo el proceso de establecimiento de las celdas del nodo B (reset-cellSetup-resourceStatus-... varias veces porque el nodo radía en las n celdas diferentes que tiene configuradas):



Y si necesitamos, Wireshark decodifica perfectamente todos los campos de un mensaje. Por ejemplo, en el siguiente mensaje capturado -y exportado a texto-, se puede ver[8] cómo la RNC envía al nodo B la identificación de celda (LocalCellId y CellId), las radiofrecuencias a utilizar en uplink y downlink (UARFCN), la potencia con la que va a radiar, Scrambling Code configurado, etc.:

```
UTRAN Iub interface NBAP signalling
    NBAP-PDU: initiatingMessage (0)
        initiatingMessage
            procedureID
                procedureCode: id-cellSetup (5)
                ddMode: fdd (1)
            criticality: reject (0)
            messageDiscriminator: common (0)
            transactionID: longTransActionId (1)
                longTransActionId: 1600
            value
                CellSetupRequestFDD
                    protocolIEs: 16 items
                        Item 0: id-Local-Cell-ID
                            ProtocolIE-Field
                                id: id-Local-Cell-ID (124)
                                criticality: reject (0)
                                value
                                    Local-Cell-ID: XXXXXXXXXXX
                        Item 1: id-C-ID
                            ProtocolIE-Field
```

8 Por motivos de confidencialidad, se ocultan los valores exactos de configuración en la red radio.

```
        id: id-C-ID (25)
        criticality: reject (0)
        value
            C-ID: XXXXXXXXXXX
Item 2: id-ConfigurationGenerationID
    ProtocolIE-Field
        id: id-ConfigurationGenerationID (43)
        criticality: reject (0)
        value
            ConfigurationGenerationID: XXXXXXXXXXX
Item 3: id-T-Cell
    ProtocolIE-Field
        id: id-T-Cell (276)
        criticality: reject (0)
        value
            T-Cell: XXXXXXXXXXX
Item 4: id-UARFCNforNu
    ProtocolIE-Field
        id: id-UARFCNforNu (282)
        criticality: reject (0)
        value
            UARFCN: XXXXXXXXXXX
Item 5: id-UARFCNforNd
    ProtocolIE-Field
        id: id-UARFCNforNd (281)
        criticality: reject (0)
        value
            UARFCN: XXXXXXXXXXX
Item 6: id-MaximumTransmissionPower
    ProtocolIE-Field
        id: id-MaximumTransmissionPower (131)
        criticality: XXXXXXXXXXX
        value
            MaximumTransmissionPower: XXXXXXXXXXX
Item 7: id-Closed-Loop-Timing-Adjustment-Mode
    ProtocolIE-Field
        id: id-Closed-Loop-Timing-Adjustment-Mode (333)
        criticality: reject (0)
        value
            Closedlooptimingadjustmentmode: XXXXXXXXXXX
Item 8: id-PrimaryScramblingCode
    ProtocolIE-Field
        id: id-PrimaryScramblingCode (181)
        criticality: reject (0)
        value
            PrimaryScramblingCode: XXXXXXXXXXX
```

Podemos seguir este proceso para cualquier tipo de comunicación de señalización que queramos analizar en el IuB (establecimiento de llamada, señalización de posición de terminales, reconfiguración de parametrización de las celdas, medidas periódicas de potencia recibida en todos los terminales de la zona,...), que nos permita detectar y analizar problemas o realizar optimizaciones sobre la red.

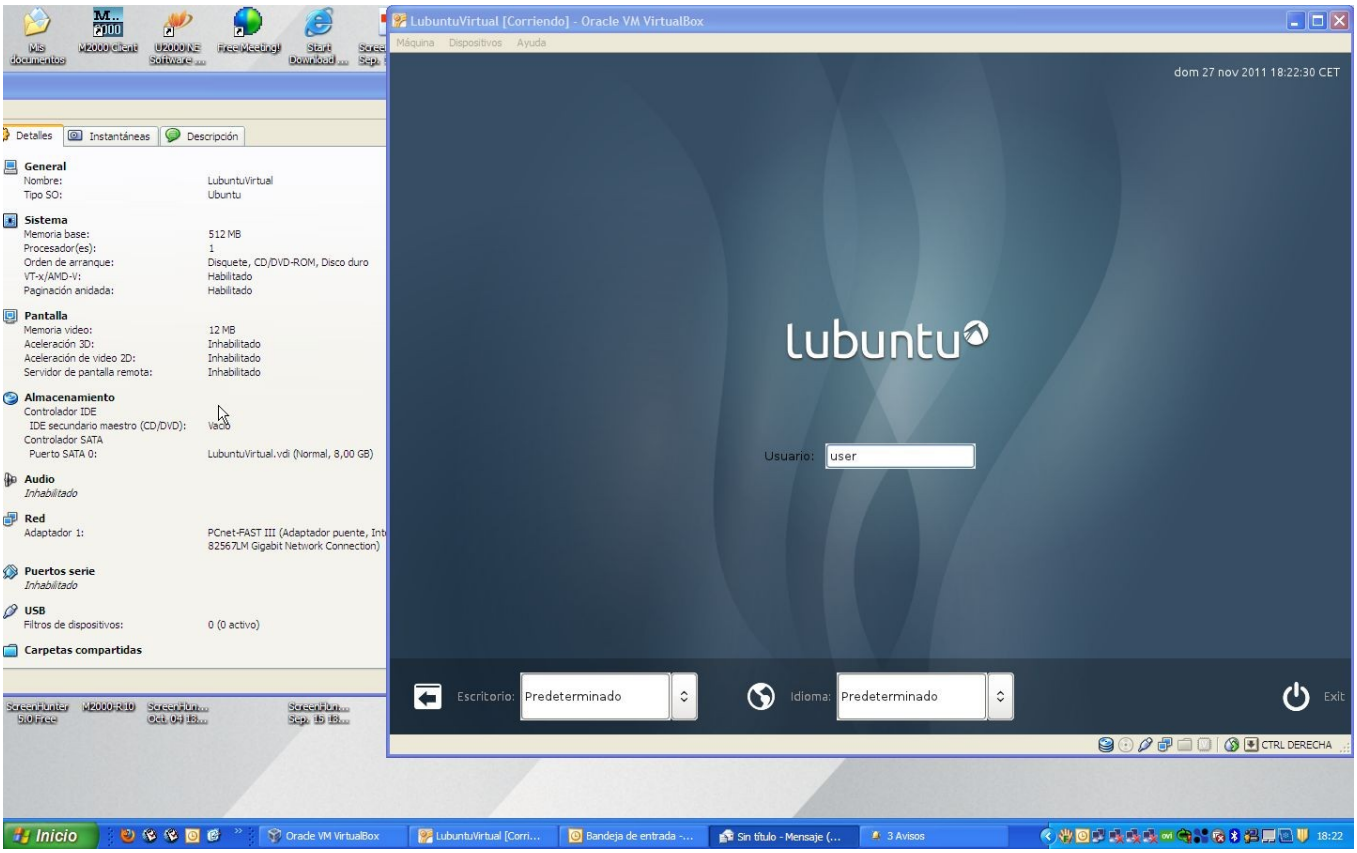
7.3. Extensibilidad de la máquina virtual a un sistema Windows remoto

Concluimos esta ronda de resultados haciendo una prueba de ejecución de la máquina virtual en un host con el sistema operativo corporativo oficial (Windows).

Para ello, simplemente lo que debemos hacer es instalar una versión de Virtualbox para Windows en la máquina cliente, que será un ordenador portátil intel Centrino 2.

Elegimos una versión de Virtualbox de la misma familia que la utilizada en el prototipo fijo (3.2), y la instalamos. Configuramos exactamente igual la máquina virtual y copiamos el fichero `LubuntuVirtual.vdi` tal como hicimos en la versión con host Linux.

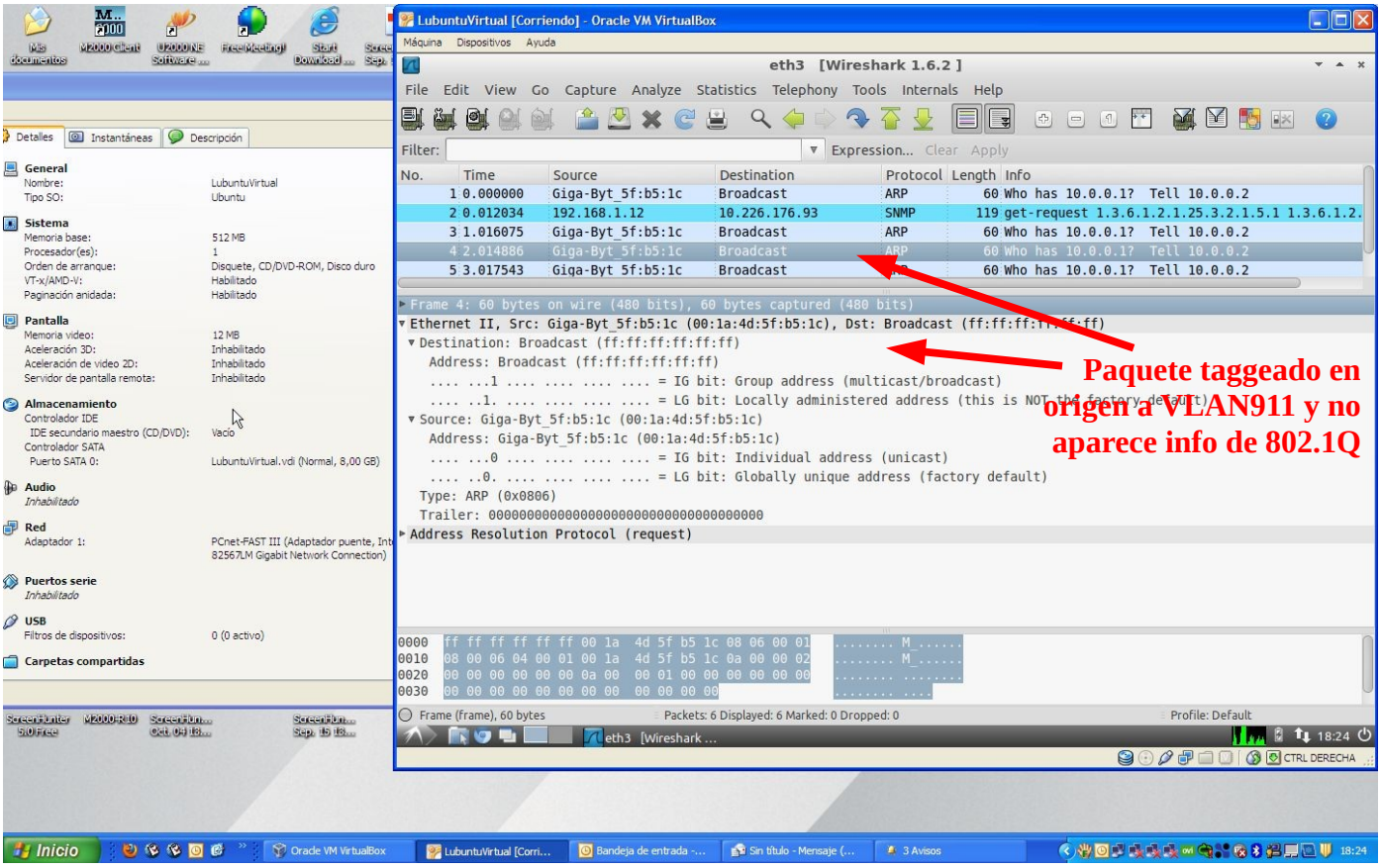
El sistema virtual arrancó sin problemas y todo funcionaba igual que en el prototipo de la plataforma fija (salvo las velocidades de ejecución porque son CPUs distintas).



Los problemas volvieron a surgir cuando intentamos decodificar paquetes con VLAN 802.1Q. Nos encontramos con el mismo problema de VLAN tag stripping que en la máquina virtual, ya que el equipo portátil tenía como dispositivo de red un "Intel 82567LM Gigabit", conocido como uno de los más problemáticos para VLANs.

Según la web oficial de Intel [IntelVLAN], en algún modelo de controlador este problema se podía solventar cambiando la parametrización del controlador desde el editor de registro de Windows (regedit). Localizamos la clave, pero no aparecía exactamente la misma referencia ("Monitor Mode"). Actualizamos el controlador e hicimos alguna prueba más con otras claves ("Tagging Mode", "VLAN Filtering",...) pero con resultados nulos. Por lo tanto, concluimos que el problema de drivers de interfaces Intel es el mismo en Windows y en Linux. En este caso, como es el

dispositivo físico real que tiene la máquina host, es insalvable y hemos de conformarnos con capturar paquetes de red sin la información de la etiqueta 802.1Q como vemos en la siguiente captura.



8. Conclusiones

8.1. Valoración

Vamos a hacer un repaso de los **requisitos iniciales planteados** y valoraremos el grado de cumplimiento de los mismos.

Mínimos:

- *Grabar y Analizar Trazas.* Cumplido.
- *Analizar cualquier interfaz de cualquier router frontera del Backhaul IP.* Cumplido en el sentido ingress en Remote Port Mirroring (cuando esté disponible el egress, el cumplimiento será del 100%).
- *Operación remota.* Cumplido.
- *Estadísticas Básicas.* Cumplido.
- *Coste económico próximo a cero.* Cumplido.
- *Separación de tráfico de telecomunicaciones y tráfico de la red corporativa.* Cumplido.
- *Seguridad.* Cumplido

Deseables:

- Poder *pinchar cualquier interfaz ethernet con la sonda* (no sólo el Backhaul). Probado pero habría que estudiar el acceso remoto a una máquina no conectada a la intranet.
- Detalle de *desglose/segmentación de estadísticas:* Cumplido pero mejorable (se hace filtrado manual y no hay interfaz de BBDD relacional, no hay estadísticas de errores).
- *Filtros de tráfico* para que la traza sólo sea del tráfico deseado. Cumplido.
- *Inyección de tráfico.* No considerado, pero posible en futuros trabajos.
- *Sin desplazamientos.* Cumplido para interfaces de Backhaul.
- *Operación remota encriptada.* Cumplido para la administración. Posible para el escritorio remoto de usuario normal, aunque no considerado en el prototipo.

Aparte del cumplimiento de los requisitos, para el presente proyecto se marcaron otros **objetivos adicionales** y que se han ido cumpliendo todos como se ha demostrado en la presente memoria:

- *Profundización de conocimientos:* herramientas de monitorización de red, técnicas de captura, puesta en producción de sistemas de virtualización, acceso remoto, administración de máquinas GNU/Linux, análisis de bajo nivel de protocolos UTRAN.
- *Aplicación de una metodología de desarrollo de proyectos* (clásica en cascada) tanto desde el punto de vista teórico, como en el sentido práctico de abordar una problemática compleja.
- *Conocimiento, difusión y uso dentro de la empresa de herramientas de Software Libre.*

Así pues, podemos concluir que **la valoración global del presente proyecto ha sido de un grado muy satisfactorio**, tanto desde el punto de vista práctico de la empresa como del crecimiento personal en un sentido académico.

8.2. Líneas futuras de trabajo

El objetivo principal de este trabajo era construir un prototipo totalmente funcional, pero a pequeña escala que ofreciera una herramienta de troubleshooting a los técnicos de operación de red de acceso de una región. Sin embargo, algunos aspectos podrían ser trabajados en mayor profundidad o ampliados en el futuro:

- Se deberá hacer un seguimiento de la implementación por parte del fabricante de equipos de Backhaul, de la funcionalidad Remote Port Mirroring en modo de captura "Egress". Este punto es básico para el éxito del sistema.
- Extensión de la posibilidad de monitorizar más allá de los puertos frontera del Backhaul. Esto pasaría por la instalación de la virtualización en ordenadores de técnicos de campo. Aunque la virtualización actual está probada y es operativa para sus ordenadores, para que sea utilizable realmente se deberá estudiar:
    - cómo realizar el conexionado local para capturar (lo más fácil probablemente sea añadir un switch con capacidades de Mirroring en el equipamiento básico personal de un técnico de campo).
    - cómo configurar el anfitrión (Windows) para el acceso remoto seguro al escritorio huésped desde la intranet corporativa (p.e. mediante securización de túneles a través de una conexión de internet móvil).
- Extensión de la plataforma a una máquina de mayores prestaciones (mayor capacidad de proceso, instrucciones de virtualización AMD-V/VT-x, más interfaces de red pinchados).
- Mayor difusión del conocimiento de la plataforma y uso a otros departamentos y regiones de la compañía.
- Aunque en su estado actual el mantenimiento de la plataforma es sencillo y casi desatendido, si se ampliara el uso del mismo y el número de usuarios, se podría convertir en un sistema crítico que requiera de protocolos de actuación específicos para mantenimiento y responsables de los mismos.
- Ampliación de funcionalidades: sonda activa (p.e. pruebas de medición de ancho de banda end-to-end mediante inyección de tráfico), procesado de estadísticas de capturas mediante base de datos relacionales (p.e. usando C5 Sigma), programado temporal y automático de capturas de forma desatendida (p.e. administrado desde un servidor web),...

Anexo 1. Script de configuración de Virtualbox como servicio

Este es el script que añadimos en /etc/init.d/virtualbox-lubuntu (basado en [VBSrv]):

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          virtualbox-lubuntu
# Required-Start:    $local_fs $remote_fs vboxdrv vboxnet
# Required-Stop:     $local_fs $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: LubuntuVirtual virtual machine
# Description:       LubuntuVirtual virtual machine hosted by VirtualBox
### END INIT INFO

# Based on Brendan Kidwell's script in
# http://www.glump.net/howto/virtualbox_as_a_service

# PATH should only include /usr/* if it runs after the mountnfs.sh script
PATH=/usr/sbin:/usr/bin:/sbin:/bin
DESC="lubuntu"
NAME=virtualbox-lubuntu
SCRIPTNAME=/etc/init.d/$NAME

MANAGE_CMD=VBoxManage
VM_OWNER=administrator
VM_NAME="LubuntuVirtual" #This has to be the name exactly as it appears in your
VirtualBox GUI control panel.

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Load the VERBOSE setting and other rcS variables
[ -f /etc/default/rcS ] && . /etc/default/rcS

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.0-6) to ensure that this file is present.
. /lib/lsb/init-functions

#
# Function that starts the daemon/service
#
do_start()
{
        # Return
        #   0 if daemon has been started
        #   1 if daemon was already running
        #   2 if daemon could not be started

        sudo -H -u $VM_OWNER $MANAGE_CMD showvminfo "$VM_NAME"|grep
"^State:\s*running" >/dev/null && {
            echo "$VM_NAME" is already running.
            return 1
        }
```

```
        sudo -H -u $VM_OWNER VBoxHeadless -s "$VM_NAME" -n -m 5910 -o password &
>/dev/null  || {
            echo Failed to start "$VM_NAME".
            return 2
        }

        echo "$VM_NAME" started or resumed.
        return 0
}

#
# Function that stops the daemon/service
#
do_stop()
{
        # Return
        #   0 if daemon has been stopped
        #   1 if daemon was already stopped
        #   2 if daemon could not be stopped
        #   other if a failure occurred

        sudo -H -u $VM_OWNER $MANAGE_CMD showvminfo "$VM_NAME"|grep
"^State:\s*running" >/dev/null || {
            echo "$VM_NAME" is already stopped.
            return 1
        }

        sudo -H -u $VM_OWNER $MANAGE_CMD controlvm "$VM_NAME" savestate || {
            echo Failed to stop "$VM_NAME".
            return 2
        }

        echo "$VM_NAME" suspended.
        return 0
}

#
# Display "State" field from showinfo action
#
do_status()
{
        sudo -H -u $VM_OWNER $MANAGE_CMD showvminfo "$VM_NAME"|grep
"^State:\s*.*$"
}

case "$1" in
  start)
        [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
        do_start
        case "$?" in
                0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
                2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
        esac
        ;;
  stop)
```

```
        [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
        do_stop
        case "$?" in
                0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
                2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
        esac
        ;;
  restart|force-reload)
        #
        # If the "reload" option is implemented then remove the
        # 'force-reload' alias
        #
        log_daemon_msg "Restarting $DESC" "$NAME"
        do_stop
        case "$?" in
          0|1)
                do_start
                case "$?" in
                        0) log_end_msg 0 ;;
                        1) log_end_msg 1 ;; # Old process is still running
                        *) log_end_msg 1 ;; # Failed to start
                esac
                ;;
          *)
                # Failed to stop
                log_end_msg 1
                ;;
        esac
        ;;
  status)
        do_status
        ;;
  *)
        #echo "Usage: $SCRIPTNAME {start|stop|restart|reload|force-reload}" >&2
        echo "Usage: $SCRIPTNAME {start|stop|restart|force-reload|status}" >&2
        exit 3
```

Bibliografía

[ImpSL]: "Implantación de Sistemas de Software Libre", Marcelo D'Elia Branco, Mónica León, Alejandro Novo, Alberto Otero , http://cv.uoc.es/cdocent/9690KDU35DP1RF8OK9B0.pdf

[IngSL]: "Ingeniería del Software en Entornos del Software Libre", David Aycart, Marc Gibert, Martín Hernández, Jordi Mas , http://cv.uoc.es/cdocent/APK5H8_0MRUMW3K7YJKJ.pdf

[3GoIP]: "IP Transport in UTRAN Work Task Technical Report", 3GPP , http://www.3gpp.org/ftp/tsg_ran/WG3_Iu/TSGR3_15/TRs/25933_v020_IpTransport.doc

[WireUS]: "Wireshark's User Guide", Ulf Lamping, Richard Sharpe, Ed Warnicke , http://www.wireshark.org/docs/wsug_html/#ChIntroWhatIs

[WireFil]: "Wireshark Display Filter Reference Index", Wireshark , http://www.wireshark.org/docs/dfref/

[WireLua]: "Lua in Wireshark", Wireshark , http://wiki.wireshark.org/Lua

[tcpdump]: "TCPDUMP Man Page", Van Jacobson, Craig Leres, Steven McCanne , http://www.tcpdump.org/tcpdump_man.html

[BPF]: "The BSD Packet Filter", Steven McCanne, Van Jacobson , http://www.tcpdump.org/papers/bpf-usenix93.pdf

[PUEL]: "Virtualbox Personal Use and Evaluation License", Virtualbox , https://www.virtualbox.org/wiki/VirtualBox_PUEL

[Apag]: "Disable user ability to shut down (10.10) ", , http://ubuntuforums.org/showthread.php?t=1670897&page=2

[VBSrv]: "How to Setup VirtualBox as a Service in Linux", Brendan Kidwell , http://www.glump.net/howto/virtualbox_as_a_service

[U2000]: "Huawei U2000 — Unified Network Management Solution", Huawei Technologies , http://www.huawei.com/enterprise/u2000-unified-network-management-solution.do

[SCTP]: "SCTP packet structure", , http://en.wikipedia.org/wiki/SCTP_packet_structure

[SCTP-P]: "Stream Control Transmission Protocol (SCTP) Parameters", Internet Assigned Numbers Authority (IANA) , http://www.iana.org/assignments/sctp-parameters

[IntelVLAN]: "My sniffer is not seeing VLAN, 802.1q, or QoS tagged frames", Intel , http://www.intel.com/support/network/sb/CS-005897.htm