

**Universidad Nacional Autónoma de Nicaragua, UNAN-León
Y
Universidad de Alcalá**



**Programa de Maestría en Computación
Énfasis en redes y comunicaciones**

Desarrollo de una interfaz de configuración para la herramienta TC e implementación del protocolo RADIUS en la red inalámbrica de la UNAN-RUCFA

Autor: José Manuel Aguilar González

Tutor: Javier de Pedro Carracedo

**León, Nicaragua
Noviembre 2012**

Índice de contenido

1. Introducción.....	5
1.1 Planteamiento del problema.....	5
1.2 Antecedentes.....	7
1.3 Justificación.....	8
1.4 Objetivos.....	9
1.4.1 General.....	9
1.4.2 Específicos.....	9
2 MARCO CONCEPTUAL.....	10
2.1 Disciplinas de colas.....	10
2.1.1 Introducción.....	10
2.1.2 Disciplinas de colas sin clases.....	11
2.1.2.1 FIFO.....	11
2.1.2.2 TBF.....	11
2.1.2.3 SFQ.....	13
2.1.2.4 Otras.....	16
2.1.3 Disciplinas de colas con clases - Classfull.....	17
2.1.3.1 CBQ.....	18
2.1.3.2 HTB.....	23
2.1.3.3 Otras.....	27
2.2 AUTENTICACIÓN DE USUARIOS.....	28
2.2.1 AAA (Autenticación, Autorización y Contabilización).....	28
2.2.1.1 Autenticación (Authentication).....	29
2.2.1.2 Autorización (Authorization).....	30
2.2.1.3 Contabilidad (Accounting).....	30
2.2.2 RADIUS.....	30
2.2.2.1 Métodos de autenticación.....	30
2.2.2.2 Autenticación EAP.....	31
2.2.2.2.1 EAP-MD5.....	32
2.2.2.2.2 EAP-TLS	32
2.2.2.2.3 EAP-TTLS.....	32
2.2.2.2.4 EAP-PEAP.....	33
2.2.2.2.5 Tabla comparativa de tipos de EAP.....	34
2.2.2.3 Shared secret.....	35
2.2.3 Atributos AVP.....	35
2.2.4 Secuencia de autenticación de RADIUS.....	35
2.2.5 El estándar 802.1X.....	37
2.2.6 Estructura de las comunicaciones EAP.....	38
2.2.6.1 Secuencias de autenticación EAP.....	38
3 CONFIGURACIÓN DE PAQUETES.....	42
3.1 Configuraciones para soporte IMQ.....	42
3.1.1 Instalación del Sistema Operativo Debian 6.....	42
3.1.2 Instalación y configuración de herramientas.....	46
3.2 Desarrollo de la aplicación para administración de ancho de banda.....	48
3.2.1 Descripción general.....	48
3.2.2 Diagrama de clases.....	48

Índice de Contenidos

3.2.3	Uso de la aplicación.....	50
3.2.4	Esquema de administración de ancho de banda.....	52
3.3	Instalación de sistema operativo para FreeRADIUS.....	55
3.4	Instalación de FreeRADIUS.....	56
3.5	Guía de configuración de OpenSSL.....	57
3.6	Configuración de los archivos de freeRADIUS.....	64
3.6.1	radiusd.conf.....	64
3.6.2	eap.conf.....	65
3.6.3	sites-available/default.....	66
3.6.4	sql.conf.....	67
3.6.5	proxy.conf.....	67
3.7	Servidor Web - Apache.....	68
3.8	daloRADIUS y MySQL.....	71
3.9	Test de funcionamiento del servicio.....	72
4.	CONCLUSIONES Y RECOMENDACIONES.....	77
4.1	Conclusiones.....	77
4.2	Recomendaciones.....	77
5.	BIBLIOGRAFÍA.....	78

Índice de Ilustraciones

Ilustración 1: Conexión de red WAN-LAN.....	10
Ilustración 2: FIFO.....	11
Ilustración 3: TBF (Token Bucket Filter).....	12
Ilustración 4: SFQ (Stochastic Fairness Queueing).....	14
Ilustración 5: Árbol de disciplinas de colas con clases.....	17
Ilustración 6: Árbol clases ejemplo para CBQ.....	21
Ilustración 7: Árbol HTB simplificado.....	25
Ilustración 8: Secuencia de comunicación AAA.....	36
Ilustración 9: Estructura de comunicación resumida de EAP.....	38
Ilustración 10: Secuencias de autenticación EAP-PEAP/MS-CHAPv2.....	39
Ilustración 11: Inicio de instalación Debian.....	42
Ilustración 12: Instalación Debian, selección de tarjeta de red primaria.....	42
Ilustración 13: Instalación Debian, fallo de configuración IP automática.....	43
Ilustración 14: Instalación Debian, configurar red manualmente.....	43
Ilustración 15: Instalación Debian, configuración de particiones.....	44
Ilustración 16: Instalación Debian, ejemplo simple de particiones.....	45
Ilustración 17: Instalación Debian, selección de programas por defecto.....	45
Ilustración 18: Diagrama de Clases.....	49
Ilustración 19: Interfaz de configuración tc-htb.....	50
Ilustración 20: Interfaz de ejecución del servicio tc-htb.....	51
Ilustración 21: Interfaz de monitoreo de ejecución de tc-htb.....	51
Ilustración 22: Paquetes por defecto para el servidor freeRADIUS.....	55
Ilustración 23: Configuración de RADIUS para cliente NAS.....	72
Ilustración 24: Detección de SSID.....	73
Ilustración 25: Parámetros de configuración para red inalámbrica.....	73
Ilustración 26: Propiedades EAP-PEAP.....	74
Ilustración 27: Propiedades EAP-MSCHAPv2.....	75
Ilustración 28: Configuración 802.1X.....	76
Ilustración 29: Solicitud de credenciales de usuario.....	76

Índice de tablas

Tabla 1: Comparativa de tipos de EAP.....	34
Tabla 2: Esquema de prioridades para la administración del ancho de banda de UNAN-RUCFA.....	53

1. Introducción

1.1 Planteamiento del problema

La Facultad de Ciencias Económicas, comúnmente conocida como RUCFA (Recinto Universitario Carlos Fonseca Amador), de la Universidad Nacional Autónoma de Nicaragua actualmente posee contratado un ancho de banda de 3.3Mbps el cual es distribuido entre 233 usuarios de los cuales se mantienen activos 40 usuarios en la mayor parte del día teniendo picos de entre 50-60 usuarios en algún determinado momento del día. Cabe afirmar que el mayor consumo de ancho de banda es generado por los usuarios de los laboratorios y la red inalámbrica. Esto se deduce por los datos históricos de navegación registrados en el servidor que permite el acceso a Internet en la facultad, ya que la navegación de todos los usuarios se realiza mediante un servidor proxy que distribuye la conexión que se tiene a Internet.

La velocidad de conexión a Internet para los usuarios se reduce en gran medida al tener un uso normal de 40 usuarios activos en gran parte del día, así, calculando $3.3\text{Mbps}/40\text{usuarios}$ resulta una conexión de 82.5Kbps para cada usuario. Este cálculo indica que cuando un usuario desee navegar por algún sitio de Internet verá reducida drásticamente su velocidad de conexión lo que conlleva a una queja con el administrador de red. La velocidad de conexión se reduce ya que los sitios de Internet cada día demandan más ancho de banda y la visualización de una página web toma más tiempo debido al peso del contenido (imágenes, vídeo, audio) de la misma. En determinados momentos del día cuando la concurrencia de usuarios es baja, la navegación se realiza con mejores prestaciones.

Teniendo en cuenta que en los últimos 3 años la cantidad de usuarios ha aumentado de 150 a 233 y afirmando que en 2 ó 3 años este número va a crecer dado que se construirá un nuevo edificio y además se ampliará la población estudiantil por la modalidad sabatina, se aprecia que la velocidad de conexión a Internet se verá más reducida porque habrá más competencia por el recurso.

Un punto interesante a recalcar es que muchos de los usuarios que acceden a Internet en la facultad son estudiantes ante lo cual la dirección se planteó la posibilidad de establecer algunas políticas de restricciones de tráfico lo que llevó a hacer restricciones de contenido pero aún quedó la limitante de que todos los usuarios compiten por una misma conexión.

El punto anterior indica que se debe establecer un mecanismo de administración del ancho de banda para la facultad de forma que se establezcan prioridades donde las unidades más críticas (consejo facultativo, profesores) de la facultad puedan disponer de una velocidad de conexión mayor que las unidades no muy críticas (secretarías, intendencia, archivo, etc.).

Sumado a lo planteado anteriormente, se tiene que en la facultad se brinda servicio de Internet Inalámbrica tanto para estudiantes como para docentes y el único mecanismo de restricción que se usa es el de la clave en el punto de acceso así como la habilitación en cada equipo por un encargado de la red inalámbrica que lleva un control semestral de los usuarios a quienes se les configura su portátil para

el acceso al servicio.

Ante las situaciones comentadas anteriormente, se plantean las siguientes interrogantes que sirven como directrices para realizar la presente investigación,

1. ¿Qué factores llevan a la necesidad de utilizar una aplicación administradora de ancho de banda?
2. ¿Cuál es el esquema de administración del ancho de banda que se va a implementar en la UNAN-RUCFA?
3. ¿Cuáles son las políticas de seguridad a implementar para el buen uso del ancho de banda?
4. ¿Cuáles son los métodos utilizados para la manipulación de paquetes de red?
5. ¿Cuáles son las herramientas software necesarias para la implementación de un esquema de administración de ancho de banda?
6. ¿Qué software se puede utilizar para gestionar a los usuarios que se conecten a la red inalámbrica?

1.2 Antecedentes

La Universidad Nacional Autónoma de Nicaragua (UNAN-Managua) se compone de 6 recintos universitarios ubicados en Carazo, Chontales, Estelí, Matagalpa y 2 en Managua que son la Facultad de Ciencias Económicas (UNAN-RUCFA) y el Recinto Universitario Rubén Darío que es la sede central.

De los 6 recintos solamente uno de ellos cuenta con un sistema de administración de ancho de banda. Este recinto es la sede central de la universidad quienes adquirieron un hardware con su respectivo software para las tareas de administración del ancho de banda.

En la Facultad de Ciencias Económicas (UNAN-RUCFA) no se cuenta con un sistema de administración de ancho de banda de la red informática de modo que todos los usuarios se dividen el recurso que posee la facultad.

Debido a que todos los usuarios comparten el ancho de banda, un docente investigador tiene limitado su ancho de banda disponible por algún otro usuario llámese, estudiante o secretaria, que busca contenido no productivo. Por ello se busca la forma de implementar, mediante opensource, un mecanismo de control sobre la administración del ancho de banda. Un punto que pone en evidencia aún más la problemática es que la población estudiantil ha aumentado en los últimos años en aproximadamente 750 estudiantes debido a que se han aperturado carreras con modalidad sabatino y el número de estudiantes va a crecer aún más cuando se complete el ciclo de 5 años en la modalidad sabatino.

En herramientas comerciales encontramos por ejemplo NetEnforcer de Allot(<http://www.allot.com>) que es un equipo hardware de altas prestaciones para la administración del ancho de banda. En herramientas de libre distribución tenemos por ejemplo la plataforma Untangle(<http://www.untangle.com>) que es un derivado de Debian e incluye, limitadamente, un mecanismo de calidad de servicio. Otras distribuciones Linux que aportan características de calidad de servicio son Zeroshell(<http://www.zeroshell.net>) y BrazilFW(<http://www.brazilfw.com.br>).

1.3 Justificación

A medida que la Facultad de Ciencias Económicas de la Universidad Nacional Autónoma de Nicaragua UNAN-Managua crece en población estudiantil también lo hace la demanda de ancho de banda tanto para los usuarios existentes como para los nuevos. Aquí se presenta un problema de competencia por el recurso dado que actualmente la facultad no cuenta con un mecanismo para regular la cantidad de ancho de banda que tiene asignada cada área (Administración, Escuelas, Estudiantes, Red Inalámbrica).

Dado que todos los usuarios compiten por un ancho de banda global para toda la facultad, esto conlleva a que la navegación sea lenta en los momentos de mayor utilización de equipos que se conectan a Internet por el tipo de contenido al cual acceden los usuarios y es que mientras un estudiante puede estar navegando a través de contenido multimedia (Video, Audio, imágenes) un docente investigador puede estar tratando de descargar un documento de gran tamaño para sus investigaciones.

Lo anterior hace suponer que se necesita, en primera instancia, incrementar el ancho de banda que tiene contratado la facultad, el que se tiene fijado en 3.3Mbps. Con este ancho de banda y tomando como una media de 40 usuarios en línea en la mayor parte del día, nos indica que cada usuario tiene asignado 82.5Kbps lo que a simple vista hace suponer que es muy poco ancho de banda para navegar a través de sitios web que tienen una carga de contenido muy alta. Además, en segunda instancia, sumado a necesitar más ancho de banda también se debe llevar a cabo la implementación de una herramienta que nos permita administrar el ancho de banda de modo que se establezcan prioridades entre las distintas áreas de la facultad para que los usuarios con labores más especializadas puedan disponer de más ancho de banda que aquellos usuarios que no tienen tareas tan determinantes.

Con la implementación de una herramienta administradora del ancho de banda en la facultad, como es el caso del programa TC del kernel de Linux y agregando autenticación de usuarios a la red inalámbrica, se contará con lo siguiente:

- Limitar el ancho de banda que dispondrá cada área en la facultad en base a las necesidades de las mismas.
- Utilizar un mecanismo de control de acceso para los usuarios de red inalámbrica

También, las unidades de la administración central de la facultad agilizarán sus procesos dado que dispondrán de mayor ancho de banda para sus propósitos. Así por ejemplo:

- Un docente investigador, al disponer de más ancho de banda podrá reducir el tiempo de descarga de los documentos de interés facilitándose de esta manera su trabajo.
- El área administrativa agilizará la recepción de información desde el recinto central y empresas proveedoras.

- Las áreas menos sensibles de la facultad, al disponer de menos ancho de banda, buscarán como aprovechar de mejor forma el recurso por lo que se verán obligados a sensibilizarse en un cambio de conducta con el tipo de contenido al cual acceden desde Internet.

Para llevar a cabo el diseño e implementación de un esquema para administrar el ancho de banda de la facultad, se hará en primera instancia un estudio acerca de la forma en la cual manejan los paquetes de red los distintos métodos (HTB, CBQ, etc.) por medio de los cuales se puede controlar el ancho de banda. Esto tendrá como fin, escoger el método de mejores prestaciones para la red en particular en la cual se va a ejecutar esta herramienta. El entorno de configuración del método escogido así como la ejecución del mismo será bajo una distribución GNU/Linux.

1.4 Objetivos

1.4.1 General

- Desarrollar una interfaz de configuración para la herramienta TC e implementar el uso de freeRADIUS en la red inalámbrica de la UNAN-RUCFA

1.4.2 Específicos

1. Identificar los factores que conllevan a la implementación de un esquema de administración del ancho de banda en la UNAN-RUCFA
2. Definir el esquema de administración del ancho de banda a implementar en la UNAN-RUCFA
3. Analizar algunos de los métodos utilizados para la manipulación de paquetes de red.
4. Desarrollar una interfaz de configuración para la herramienta TC que provee GNU/Linux.
5. Implementar las características que provee el software freeRADIUS para la autenticación de usuarios.

2 MARCO CONCEPTUAL

2.1 Disciplinas de colas

2.1.1 Introducción

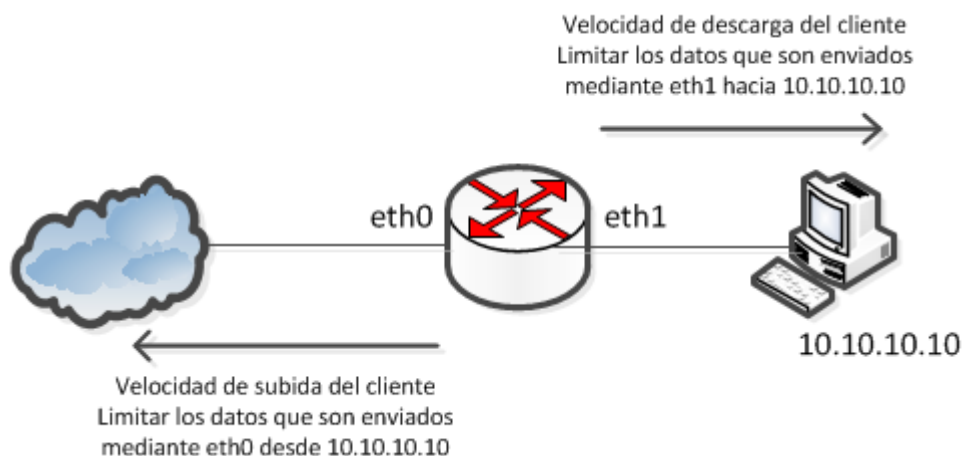


Ilustración 1: Conexión de red WAN-LAN

En el mundo de las redes de computadoras, uno de los esquemas más habituales es tener un router con dos interfaces de red que se usan una para la red local y otra para la conexión a Internet a como puede apreciarse en la Ilustración 1. El tráfico que se genera desde y hacia la red local puede ser manejado estableciendo prioridades de entrega del mismo. Esto se consigue con las disciplinas de colas que son mecanismos que se utilizan para ralentizar y ajustar el tráfico que circula por una red de computadoras donde se necesita gestionar un recurso, como es el caso del ancho de banda a Internet.

En el presente trabajo se pretende dar una de las tantas posibles soluciones al manejo o administración del ancho de banda mediante algoritmos de disciplinas de colas.

Dado que se va a trabajar sobre Linux con la herramienta TC hay que tener presente, para evitar confusión, la interpretación de las tasas de transferencia que usan los diferentes mecanismos de disciplinas de colas:

mbit: Megabits por segundo

kbit: Kilobits por segundo.

2.1.2 Disciplinas de colas sin clases

2.1.2.1 FIFO

First In First Out/primer en entrar, primero en salir. Es una qdisc sin clases según la cual se atiende primero al cliente que antes haya llegado (ver Ilustración2). El algoritmo FIFO es la base de la qdisc por defecto en todas las interfaces de red de Linux (*pfifo_fast*). Se lleva a cabo sin la formación o reorganización de los paquetes. Simplemente transmiten los paquetes tan pronto como pueda después de recibirlos y encolarlos.

First In First Out

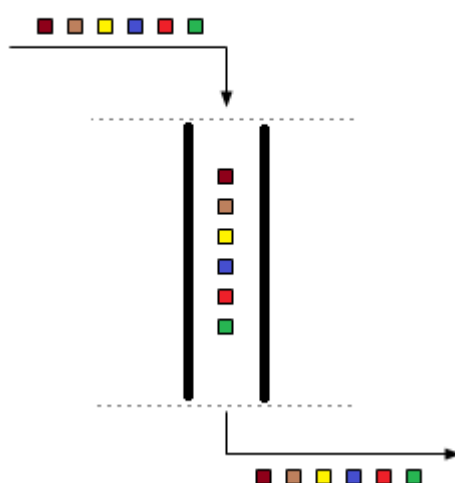


Ilustración 2: FIFO

Una disciplina de cola FIFO debe tener un tamaño de buffer límite para evitar que se desborde en el caso de que no sea capaz de quitar de la cola los paquetes tan pronto como los reciba.

El sistema Linux emplea dos qdiscs FIFO básicas. Una es basada en bytes y otra es basada en paquetes. Independientemente del tipo de FIFO a usar, el tamaño de la cola es definido por el parámetro *limit*. Para una *pfifo* la unidad se entiende que es paquetes mientras que para una *bfifo* la unidad será en bytes.

La qdisc más utilizada es la *pfifo* ya que es la que viene por defecto en Linux.

2.1.2.2 TBF

Token Bucket Filter, es un algoritmo de disciplina de encolado sin clases que proporciona una forma sencilla de establecer límites administrativos a la velocidad de conexión de una interfaz con el fin de acelerar el tráfico que pasa por ésta. Una aplicación muy usada de este algoritmo son los Delay Pools de Squid.

En el arranque, TBF está lleno de tokens los cuales corresponden a la cantidad de tráfico que se puede procesar en ráfaga. Los tokens llegan a un ritmo constante hasta que el bucket esté lleno. Ver Ilustración 3

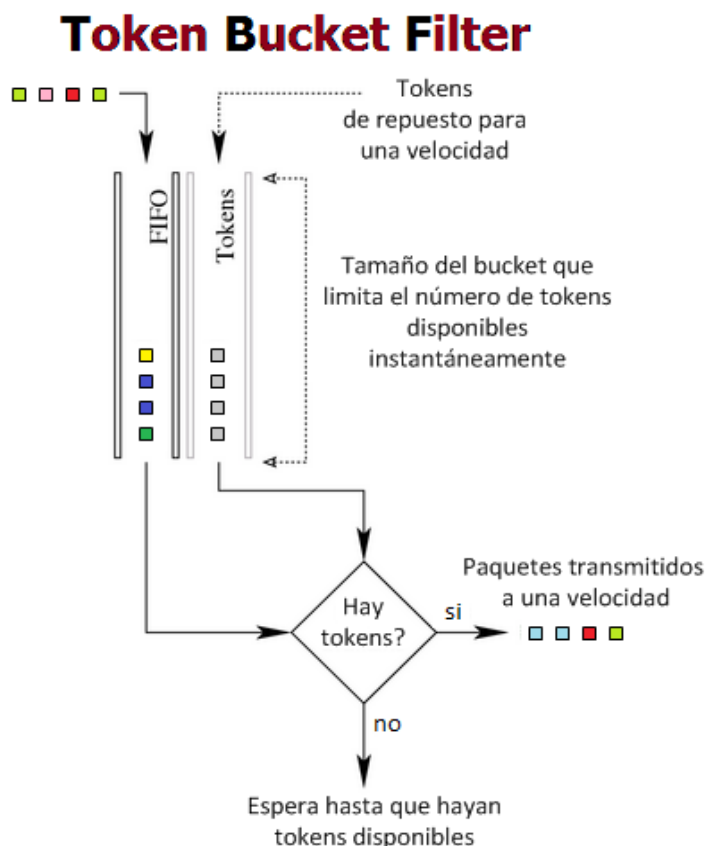


Ilustración 3: TBF (Token Bucket Filter)

Si no hay tokens disponibles para los paquetes, éstos se ponen en la cola hasta un límite configurado. El TBF calcula el déficit de tokens y acelera el proceso hasta el primer paquete en la cola que puede ser enviado. La velocidad con la cual se vacía el bucket se puede implementar con otro TBF más pequeño a una velocidad pico o tasa máxima.

Una utilidad de ejemplo puede ser que TBF se centre en que la cola será manejada en nuestro Linux y no en el dispositivo router de salida a Internet como por ejemplo un módem DSL o un cable módem. Esto se debe a que enviar datos al módem llena su cola, que debe ser muy grande, por lo que manejar la cola en nuestro Linux hará que podamos establecer un tamaño de cola no muy grande para no perder interactividad y que se puedan realizar otras tareas mientras se envían datos.

Los parámetros que se deben tener en cuenta en TBF son:

```
tc qdisc ... tbf rate rate burst bytes/cell ( latency ms | limit bytes ) [ mpu bytes [ peakrate rate mtu bytes/cell ] ]
```

limit o latency: Limit es el número de bytes que pueden ser encolados a la espera de que haya tokens disponibles. También puede especificar esto estableciendo el parámetro latency, que indica el período máximo de tiempo que puede pasar un paquete en el TBF. Este último cálculo tiene en cuenta el tamaño del bucket, la tasa y posiblemente el peakrate (la tasa de picos, si es que la ha configurado).

burst/buffer/maxburst: Tamaño del bucket, en bytes. Esta es la máxima cantidad de bytes para los que pueden haber tokens disponibles instantáneamente. En general, grandes tasas precisan grandes buffers. Si el buffer es demasiado pequeño, se descartarán paquetes debido a que llegan más tokens por tick del temporizador de los que caben en el bucket

mpu: Un paquete de tamaño cero no usa un ancho de banda cero. En ethernet, ningún paquete usa menos de 64 bytes. La Minimum Packet Unit determina el uso mínimo de tokens (especificado en bytes) por paquete.

rate: El ajuste de la velocidad.

Si se desea usar una velocidad pico o tasa máxima, los parámetros a tener en cuenta son:

peakrate: Tasa máxima de agotamiento del bucket. Se usa para especificar que tan rápido se le permite al bucket vaciarse.

mtu/minburst: Especifica el tamaño del bucket del peakrate.

Ejemplo de uso de TBF:

Configurar TBF para una velocidad máxima sostenida de 0.5mbit/s, con una tasa máxima de vaciado de 1mbit/s, un buffer de 5kilobytes, con un límite de cola calculado por TBF que a lo sumo tenga 70ms de latencia.

```
# tc qdisc add dev eth0 root tbf rate 0.5mbit burst 5kb latency 70ms peakrate 1mbit  
minburst 1540
```

2.1.2.3 SFQ

Stochastic Fairness Queueing (SFQ) es una disciplina de colas sin clases que intenta distribuir equitativamente la oportunidad de transmitir datos a la red entre un número arbitrario de flujos. Esto se logra mediante el uso de una función hash para separar el tráfico de distintas (mantenidas internamente) FIFOs que se desencolan mediante round robin, dando a cada sesión la oportunidad de enviar datos mediante turnos.

Debido a que existe la posibilidad de injusticia en la elección de la función hash, esta función debe ser alterada periódicamente, generalmente cada 10 segundos pero puede variar. La perturbación (con el parámetro *perturb*) establece esta periodicidad para garantizar que un flujo no ahogue a los demás. Este hecho puede tener algún efecto en la mitigación de un intento de ataque de Denegación de Servicio.

El comportamiento de SFQ es bastante equitativo y se llama estocástica por que no crea una cola para cada sesión, sino que tiene un algoritmo que divide el tráfico en un número limitado de colas utilizando un hash. La Ilustración 4 nos muestra un poco el diseño.

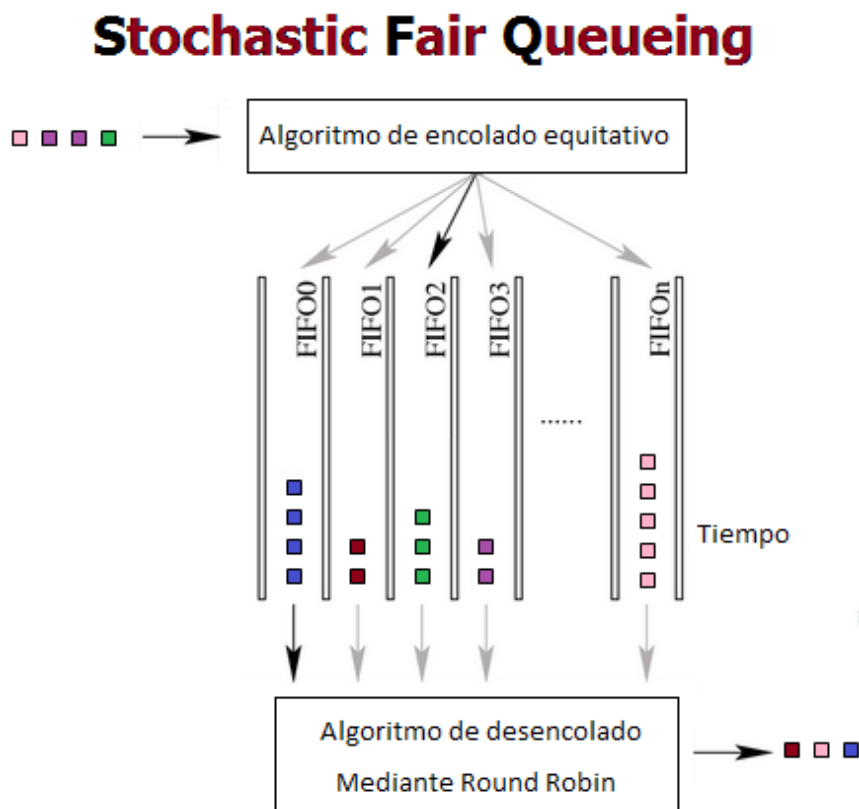


Ilustración 4: SFQ (Stochastic Fairness Queueing)

SFQ es una implementación sencilla de la familia de algoritmos de colas justas(fair queuing) como Completely Fair Scheduler (CFS) o Weighted Fair Queuing (WFQ). Tiene menos precisión pero lo compensa realizando menos cálculos mientras trata de ser lo más justo posible.

El algoritmo SFQ

Al momento del encolado, asigna cada paquete a un bucket, esto basado en: Dirección IP Origen, Dirección IP Destino, Puerto Origen, Puerto Destino y un Número Aleatorio. El algoritmo tiene conocimientos de los protocolos IPv4, IPv6, UDP, TCP y ESP. Los paquetes con otros protocolos son ordenados sobre la base de una representación de 32bits de su destino y el socket al cual pertenece. Generalmente un flujo corresponde a una conexión TCP/IP.

Cada uno de los buckets de SFQ deben representar un flujo único pero debido a que pueden ocurrir injusticias en la ordenación del hash donde varios flujos pueden quedar en el mismo bucket, se necesita perturbar el algoritmo de la función hash cada cierto tiempo para hacer una reordenación de paquetes y que la colisión de los flujos dure el menor tiempo posible. Luego, cuando se da el desencolado, cada bucket con datos es consultado mediante round robin para la transmisión de datos.

La principal ventaja de este método es que ningún paquete tendrá la prioridad sobre otro. Ninguna conexión puede sobreponerse a otra y todas las conexiones pueden transmitir. El reparto de ancho de banda trata de ser justo pero puede haber alguna limitación como por ejemplo que el algoritmo de hash podría producir el mismo resultado para varios paquetes provocando que se envíen a un mismo bucket. Así, un bucket se llenará más rápido que otro rompiendo la equidad. Para resolver esto, se tiene al parámetro *perturb* para que se haga un reordenamiento cada cierto tiempo y en caso de desbordamiento, se ejecuta el descarte al final (tail-drop) en el bucket que está más lleno manteniendo así la equidad.

Parámetros en SFQ

perturb

Intervalo en segundos para la perturbación del algoritmo de encolado o lo que es la reconfiguración del hash cada # segundos. El valor predeterminado es 0, lo que significa que no se producen perturbaciones y no es recomendable. El valor que se aconseja es 10.

quantum

Es la cantidad en bytes que se permite en un flujo para desencolarlo durante una ronda del proceso round robin. Por defecto toma el valor de la MTU de la interfaz el cual también es el valor aconsejado y el valor mínimo.

limit

Es el número de paquetes a encolarse una vez que se ha iniciado el descarte. Opcional.

Ejemplo de la implementación de SFQ

```
# tc qdisc add dev imq0 root sfq perturb 10
```

Para ver estadísticas de la qdisc

```
# tc -s -d qdisc ls
qdisc sfq 8001: dev imq0 limit 127p quantum 1500b flows 127/1024 perturb 10sec
Sent 0bytes 0pkts (dropped 0, overlimits 0 requeues 0)
```

Este resultado nos indica que usará el identificador 8001 para la qdisc que fue asignado automáticamente. El dispositivo sobre el cual funcionará SFQ es el dispositivo de encolado intermedio imq0 limitando la cantidad de paquetes que pueden esperar en esta cola a 127. El quantum queda en 1500bytes que es el valor por defecto de las redes ethernet. Se tienen disponibles 1024 flujos de los cuales se pueden tener activos 127 y la reconfiguración hash se hará cada 10 segundos.

Se debe tener presente que SFQ es útil solamente sobre su propia cola. Es el caso de cuando la velocidad del enlace es igual al ancho de banda actual. Esto es válido para módems telefónicos, conexiones ISDN y enlaces ethernet directos no conmutados.

Generalmente el uso más común de SFQ se da cuando se utiliza en conjunto con alguna disciplina de colas con clases (como HTB o CBQ entre otras) de manera que la qdisc con clases contenga en sus interior varias clases hijas SFQ. Así se puede tener la certeza de que el algoritmo SFQ funcionará correctamente sobre una cola asignada en un ancho de banda específico.

2.1.2.4 Otras

RED

Random Early Detect (detección aleatoria temprana), pero también es conocida como Random Early Drop (descarte aleatorio temprano), ya que es así como funciona. Es una qdisc sin clases que busca como evitar la congestión.

El comportamiento normal de las colas de router en Internet se denomina "tail-drop". El tail-drop (descarte de últimos) funciona encolando hasta una cierta cantidad, y descartando después todo el tráfico que rebese esa cantidad. Esto es bastante poco justo, y lleva a la sincronización de retransmisión. Cuando sucede esta sincronización, la súbita ráfaga de descartes del router que ha llegado a su tope causa una ráfaga retardada de retransmisiones, que vuelve a llenar en exceso el router congestionado.

Para copar con la congestión transitoria de los enlaces, los router de backbone a menudo implementan grandes colas. Desafortunadamente, mientras estas colas están bien para la transferencia, pueden incrementar sustancialmente la latencia y causar que las conexiones TCP se comporten a ráfagas durante la congestión. RED es una buena disciplina de cola para backbones pero no es un remedio milagroso para todo lo que se presente. Las aplicaciones que no implementen esperas tras colisión (backoff) exponencial seguirán llenando una porción demasiado grande del ancho de banda, pero con RED no causan tanto daño a la transferencia y la latencia de otras conexiones.

RED descarta paquetes de los flujos estadísticamente antes de que lleguen a un límite absoluto (hard). Esto hace que un enlace congestionado en un backbone reduzca la marcha de una manera más elegante, y evita la sincronización de retransmisiones. También ayuda a TCP a encontrar su velocidad "correcta" más rápido permitiendo que algunos paquetes sean eliminados, manteniendo bajo el tamaño de las colas y la latencia bajo control. La probabilidad de que se descarte un paquete de una conexión particular es proporcional a su uso de ancho de banda en lugar de al número de paquetes que transmite.

Para usar RED, se deben tomar decisiones sobre tres parámetros: **Min**, **Max**, y **burst**. **Min** establece el tamaño mínimo de la cola en bytes antes de que empiecen los descartes, **Max** es un máximo preventivo (soft) bajo el que intenta mantenerse el algoritmo, y **burst** especifica el número máximo de paquetes que pueden "pasar en ráfaga".

GREED

Generic Random Early Detection, es una qdisc sin clases que soporta múltiples prioridades de descarte tanto como sean requeridas para asegurar el reenvío. Tanto RED como GREED son adecuados para encolado de datos en backbone.

Solo por mencionar, debido a su poca popularidad, se tienen también las siguientes disciplinas de colas sin clases: TEQL(True Link Equalizer) y NETEM(Network Emulator).

2.1.3 Disciplinas de colas con clases - Classfull

Las disciplinas de colas con clases pueden ser subdivididas en varias clases y generalmente se utilizan para dividir los diferentes tipos de tráfico que circulan por la red. Así, cada tipo de tráfico puede tener una clase asignada por donde saldrán sus paquetes.

Cuando entra tráfico dentro de una qdisc con clases, hay que enviarlo a alguna de las clases que contiene (se necesita «clasificarlo»). Para determinar qué hay que hacer con un paquete, se consulta a los «filtros». Es importante saber que los filtros se llaman desde dentro de una qdisc, y no al revés. Los filtros asociados a esa qdisc devuelven entonces una decisión, y la qdisc la usa para encolar el paquete en una de las clases. Cada subclase puede probar otros filtros para ver si se imparten más instrucciones. En caso contrario, la clase encola el paquete en la qdisc que contiene.[1]

Aparte de contener otras qdisc, la mayoría de las qdisc con clases también realizan «shaping». Esto es útil tanto para reordenar paquetes (con SFQ, por ejemplo) como para controlar tasas. [1]

En vista que el presente trabajo se basa en la implementación de una qdisc con clases, veremos cual es el procedimiento que sigue esta implementación.

Primero, cada interfaz en nuestro equipo tiene una disciplina de cola raíz (root qdisc) que es quien se encarga de establecer la comunicación con el kernel del sistema operativo. Luego, se tiene una clase hija que está unida o es descendiente de la *root qdisc*, y de esta clase parten otras clases hijas que contienen alguna qdisc (que se encargará de manejar los datos) y las clases terminales las cuales también son clases hijas de las clases hijas.

Para comprender mejor el funcionamiento de la qdisc con clases veamos la Ilustración 5

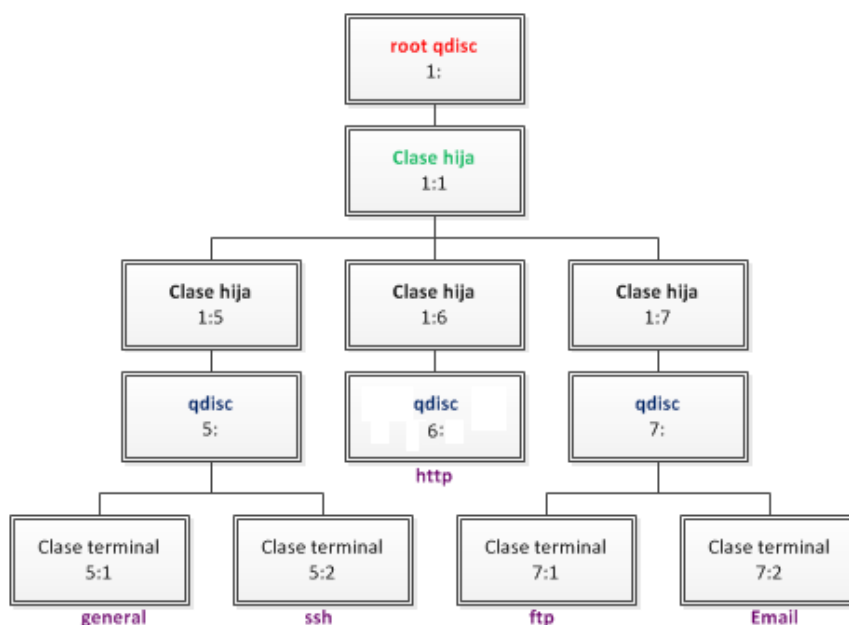


Ilustración 5: Árbol de disciplinas de colas con clases

Las disciplinas de colas con clases más utilizadas son CBQ (Class Based Queueing) y HTB (Hierarchical Token Bucket)

2.1.3.1 CBQ

Si bien es cierto CBQ no será la elección del presente trabajo se dan algunos detalles para tener en cuenta en vista del amplio uso que tiene.

Class Based Queueing (CBQ) es la disciplina de encolado con clases más famosa por su recorrido pero también la más compleja dado que su configuración no es nada fácil de llevar a cabo para que funcione correctamente. La principal razón[1] de la complejidad de CBQ es que el algoritmo no es muy preciso a la hora de trabajar en conjunto con Linux y los cálculos de tiempo ocioso.

El proyecto de investigación del algoritmo CBQ fue creado en la Network Research Group at Lawrence Berkeley National Laboratory por Floyd, S., and Jacobson, V. en 1995 como una alternativa a la tecnología tradicional basada en router.

En la página del manual (man tc-cbq) del kernel de Linux, definen CBQ como una disciplina de colas con clases que implementa una jerarquía de clases en un enlace compartido. Este contiene manejo de elementos así como capacidades de priorización de tráfico. El manejo de tráfico se realiza mediante cálculos de tiempo sobre la inactividad del enlace basándose en el calendario de eventos de desencolado y el ancho de banda del enlace adyacente.

CBQ opera dentro de la capa de red IP y ofrece los mismos beneficios a través de la capa 2 y es igualmente efectiva con el protocolo IP trabajando con TCP y UDP. Funciona con la pila de protocolos TCP/IP ya sea por parte del Cliente o Servidor puesto que se aprovecha de los mecanismos de control de flujo de TCP/IP para controlar el tráfico extremo a extremo.

Dentro de una instancia CBQ pueden existir muchas clases. Cada una de estas clases contiene otra disciplina de cola que en su configuración por defecto es *pfifo*.

En [1] se indica que, *además de ser de clases, CBQ también es ajustadora (shaper) y es en este aspecto en el que no funciona del todo bien. Debería funcionar de esta manera. Si intenta ajustar a 1mbit/s una conexión de 10mbit/s, el enlace debería estar ocioso el 90% del tiempo. Si no lo está, necesitamos acelerar de manera que realmente ESTÉ ocioso el 90% del tiempo. Esto es bastante difícil de medir, de manera que en su lugar CBQ deriva el tiempo ocioso del número de microsegundos que se tarda entre cada petición de más datos por parte de la capa de hardware. Combinados, se pueden usar para aproximar cómo de lleno o vacío está el enlace.*

La gente que ha hecho mediciones ha descubierto que CBQ no es siempre muy preciso, y a veces se pierde del todo. Pero en la mayoría de las circunstancias funciona bien.

Parámetros de CBQ

qdisc

La raíz de un árbol de clases de CBQ tiene los siguientes parámetros para su funcionamiento encontrados en la página del manual de CBQ:

```
tc qdisc ... dev dev ( parent classid | root) [ handle mayor: ] cbq [ allot bytes ] avpkt bytes
bandwidth rate [ cell bytes ] [ ewma log ] [ mpu bytes ]
```

parent mayor:menor | root

Este parámetro mandatorio determina el lugar de la instancia CBQ, ya sea en la raíz de una interfaz o dentro de una clase existente.

handle

Cómo todas las disciplinas de colas anteriores, CBQ puede tener asignada un manejador (handle). Debe consistir solamente de un número mayor seguido por dos puntos. Es opcional.

allot

Valor opcional que por defecto tiene un valor razonable relacionado a *avpkt*.

Cuando se le pide a la CBQ externa un paquete para enviar por la interfaz, buscará por turnos en todas sus qdisc internas (en las clases), en el orden del parámetro de «prioridad». Cada vez que le toca el turno a una clase, sólo puede enviar una cantidad limitada de datos. «allot» es la unidad básica de esta cantidad.

avpkt

Tamaño medio de un paquete, medido en bytes. Se necesita para calcular *maxidle*, que se deriva de *maxburst*, que va especificado en paquetes. Este valor debe ser conocido para los cálculos

bandwidth

CBQ necesita conocer el ancho de banda físico de su dispositivo o qdisc padre, ya que es necesario para los cálculos de tiempo ocioso.

cell

El tiempo que tarda un paquete en ser transmitido sobre un dispositivo está escalonado, y se basa en el tamaño del paquete. Un paquete de tamaño 800 y uno de 806 pueden tardar lo mismo en ser enviados, por ejemplo (esto establece la granularidad de los cálculos de tiempo en la transmisión del paquete). A menudo está a "8". Debe ser una potencia entera de dos.

mpu

Por defecto este valor está a cero. Pero, debido a que en ethernet se necesitan al menos 64 bytes para una transmisión, si *mpu* está en 0 entonces se rellena el paquete hasta que tenga al menos 64 bytes y esa transmisión lleva un determinado tiempo por lo que CBQ necesita conocer el valor de este parámetro para hacer los cálculos adecuados del tiempo ocioso.

ewma

Cuando CBQ necesita calcular el tiempo ocioso promedio, lo hace usando una media móvil exponencial ponderada que suaviza las mediciones en una media móvil. El EWMA determina la cantidad de suavizado que se produce. Por defecto tiene el valor de 5 y debe estar entre 0 y 31. Un valor muy bajo implica mayor sensibilidad.

Clases

Una qdisc CBQ no hace ajuste de tráfico ya que sólo necesita conocer algunos parámetros acerca del enlace adyacente de ancho de banda y el ajuste real se lleva a cabo en las clases.

Los parámetros de las clases son los siguientes:

```
tc class ... dev dev parent major:[minor] [ classid major:minor ] cbq allot bytes [ bandwidth rate ]
[ rate rate ] prio priority [ weight weight ] [ minburst packets ] [ maxburst packets ] [ ewma log ]
[ cell bytes ] avpkt bytes [ mpu bytes ] [ bounded isolated ] [ split handle & defmap defmap ]
[ estimator interval timeconstant ]
```

parent mayor:menor

Nos brinda la ubicación de esta clase en la jerarquía. Si se llama directamente a una qdisc y no a otra clase, el valor *menor* puede ser omitido. Valor obligatorio.

classid mayor:menor

Como las qdiscs, las clases también pueden ser nombradas. El número mayor debe ser igual al número mayor de la qdisc a la cual pertenece. Es un valor opcional pero se necesita en caso de que la clase vaya a tener hijos.

weight

Cuando se desencola a la interfaz, las clases envían tráfico mediante round robin, por lo que es razonable que si una clase tiene asignado un ancho de banda mucho menor que las demás se le puede permitir enviar más datos en la ronda que le corresponde.

prio

En el proceso round robin, las clases con el campo de prioridad más bajo tiene mejor trato para el envío de paquetes

minburst

Como se mencionó previamente, CBQ necesita acelerar en caso de sobrelímite. La solución ideal es hacerlo exactamente durante el tiempo ocioso calculado, y pasar un paquete. Para los núcleos de Unix, sin embargo, normalmente es complicado organizar eventos menores a 10ms, de manera que es mejor acelerar durante un periodo algo mayor, y hacer pasar entonces "minburst" paquetes de una sólo tanda, para después dormir "minburst" veces más.

maxburst

Es el número de paquetes que se usa para calcular *maxidle* de manera que cuando *avgidle* esté a *maxidle*, se pueda enviar una ráfaga de esta cantidad de paquetes medios antes de que *avgidle* se ponga a 0. Para que sea tolerante a ráfagas se puede poner un valor alto. No se puede establecer *maxidle* directamente, sino sólo mediante este parámetro.

bounded

Significa que esta clase no va a pedir prestado ancho de banda a sus clases hermanas.

isolated

Significa que esta clase no va a prestar ancho de banda a sus clases hermanas.

split&defmap

Si los filtros vinculados a una clase no dan una respuesta, CBQ también puede clasificar en base a la prioridad en los paquetes. Hay disponibles 16 prioridades numeradas de la 0 a la 15.

El *defmap* especifica cual de las prioridades de esta clase quiere recibir, especificado como un mapa de bits. El bit menos significativo corresponde a cero prioridad. El parámetro *split* le dice a CBQ cual clase debe tomar la decisión, que debe ser un (gran) padre de la clase que está agregando.

estimator

CBQ puede medir la cantidad de ancho de banda que cada clase está usando. Con el fin de determinar el ancho de banda que utiliza, se tiene un estimador muy simple que mide una vez cada *interval* de microsegundos la cantidad de tráfico que ha pasado. De nuevo, esto es un EWMA, para lo cual se puede especificar la constante tiempo, también en microsegundos. El *time constant* corresponde a la lentitud de la medida o, por el contrario, a la sensibilidad de la media a ráfagas cortas. Los valores más altos significan una menor sensibilidad.

En la Ilustración 6 se muestra un ejemplo para CBQ [1]

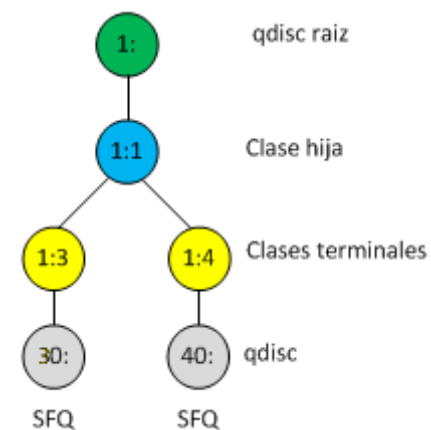


Ilustración 6: Árbol clases ejemplo para CBQ

Esta configuración limita el tráfico del servidor web a 5mbit y el SMTP a 3mbit. Juntos, no pueden alcanzar más de 6mbit. Tenemos una NIC de 100mbit y las clases pueden tomar ancho de banda prestado unas de otras.

```
# tc qdisc add dev eth0 root handle 1:0 cbq bandwidth 100Mbit avpkt 1000 cell 8
# tc class add dev eth0 parent 1:0 classid 1:1 cbq bandwidth 100Mbit \
    rate 6Mbit weight 0.6Mbit prio 8 allot 1514 cell 8 maxburst 20 avpkt 1000 bounded
```

Esta parte instala la raíz y la clase convenida 1:1. La clase 1:1 está limitada, de manera que su ancho de banda no puede pasar de 6mbit.

```
# tc class add dev eth0 parent 1:1 classid 1:3 cbq bandwidth 100Mbit \
    rate 5Mbit weight 0.5Mbit prio 5 allot 1514 cell 8 maxburst 20 avpkt 1000
# tc class add dev eth0 parent 1:1 classid 1:4 cbq bandwidth 100Mbit \
    rate 3Mbit weight 0.3Mbit prio 5 allot 1514 cell 8 maxburst 20 avpkt 1000
```

Estas son nuestras dos clases terminales. Fíjese en cómo se escala el peso con la tasa configurada. Ninguna clase está limitada, pero están conectadas a la clase 1:1, que sí lo está. De manera que la suma de anchos de banda de las dos clases pasará nunca de 6mbit. Los identificadores de clase deben estar dentro del mismo número mayor que su qdisc madre.

```
# tc qdisc add dev eth0 parent 1:3 handle 30: sfq
# tc qdisc add dev eth0 parent 1:4 handle 40: sfq
```

Ambas clases llevan una qdisc FIFO por defecto. Pero las hemos reemplazado con una cola SFQ de manera que cada flujo de datos sea tratado de igual forma.

```
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 match ip sport 80 0xffff flowid 1:3
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 match ip sport 25 0xffff flowid 1:4
```

Estas órdenes, asociadas directamente a la raíz, envían el tráfico a las qdisc correctas.

Fíjese que se usa «tc class add» para CREAR clases dentro de una qdisc, pero que se usa «tc qdisc add» para añadir qdiscs a estas clases.

2.1.3.2 HTB[2]

Introducción

El hecho de que CBQ se presenta como bastante complejo para configurar así como tener en ocasiones fallas para su correcto funcionamiento fueron los hechos más importantes para que Martin Devera aka devik se motivara a crear Hierarchy Token Bucket (HTB) el cual lo pensó como un reemplazo más comprensible, intuitivo y rápido para CBQ.

Al igual que CBQ, HTB ayuda en las tareas de control de ancho de banda sobre un determinado enlace. Utiliza una misma conexión física para simular varios enlaces más lentos para la transmisión de distintos tipos de tráfico (http, ftp, ssh, etc.) o bien, limitar el ancho de banda en general para uno o más usuarios. Se tiene que especificar cómo dividir el vínculo físico en enlaces simulados y cómo decidir qué enlace simulado se va a utilizar para enviar un determinado paquete.

HTB ajusta el tráfico basado en el algoritmo Token Bucket Filter (TBF) el cual no depende de las características de la interfaz y no necesita saber el ancho de banda base de la interfaz de salida a como si lo necesita saber CBQ para realizar sus cálculos.

Clasificación

Dentro de una instancia HTB pueden existir muchas clases o nodos. Cada una de estas clases contiene otra qdisc que por defecto es pfifo.

Cuando HTB encola un paquete, se inicia en la raíz y utiliza instrucciones internas para determinar qué clase debe recibir los datos.

Debido a las pocas opciones de configuración del algoritmo, el proceso es bastante fácil. En cada nodo, se busca una instrucción y luego se dirige a la clase a la cual apunta esa instrucción. Si la clase encontrada es una clase terminal o clase hoja o nodo hoja(sin hijos), se debe encolar ahí el paquete. Pero si aún no es un nodo hoja, entonces se realiza de nuevo el proceso partiendo desde el nodo donde se encuentra.

Las siguientes acciones se llevan a cabo en cada nodo que se visita, hasta que se realice un traslado a otro nodo o termine el proceso:

- Consultar los filtros asociados a la clase. Si devuelve una instrucción indicando que debe trasladarse a otro nodo, se ejecuta el traslado.
- Si lo anterior no devuelve instrucción alguna que indique trasladarse a otro nodo, entonces se procede a encolar los paquetes en este nodo.

Este algoritmo se asegura de que un paquete siempre termine en algún lugar. Los paquetes siempre se encolan hacia abajo y se desencolan hacia arriba para pasarlos a la interfaz física. Nunca pasan directamente desde las clases terminales hacia la interfaz.

Parámetros a usar en HTB

Los detalles de los parámetros de HTB se encuentran disponibles en la página del manual de Linux para HTB (man tc-htb).

La raíz de un árbol de clases HTB tiene los siguientes parámetros:

```
tc qdisc ... dev dev ( parent classid | root) [ handle major: ] htb [ default minor-id ]
```

parent major:minor | root

Este parámetro obligatorio determina la ubicación de la instancia HTB, ya sea en la raíz de la interfaz o dentro de una clase existente.

handle major:

Como las demás disciplinas de colas, HTB puede tener asignado un manejador (handle). Consiste solo de un número (*major*) seguido por dos puntos.

default minor-id

El tráfico que no está clasificado se envía a esta clase identificada con *minor-id*.

Las clases tienen una serie de parámetros a configurarse para su operación:

```
tc class ... dev dev parent major:[minor] [ classid major:minor ] htb rate rate [ ceil rate ] burst bytes [ cburst bytes ] [ prio priority ]
```

parent major:minor

Representa la ubicación de esta clase en la jerarquía. Si está asociada directamente a una qdisc y no a otra clase, el valor *minor* puede ser omitido.

classid major:minor

Al igual que las qdiscs, las clases pueden ser nombradas. El número *major* debe ser igual al número mayor de la qdisc a la cual pertenece la clase. Es un valor necesario si la clase tiene hijos pero opcional si es terminal.

prio

En el proceso round robin, las clases con el campo prioridad más bajo son beneficiadas para enviar paquetes de primero.

rate

Es la velocidad máxima que tiene garantizada esta clase y todos sus hijos.

ceil

Es la velocidad máxima a la cual una clase puede enviar datos, aunque su padre tenga ancho de banda de sobra. El valor predeterminado es la velocidad (*rate*) configurada, lo que implica que no puede haber préstamo o petición de más ancho de banda que esté sin utilizar.

burst

Es la cantidad de bytes que se pueden enviar en ráfagas a una velocidad (ceil) techo en exceso de la velocidad (rate) configurada. Debería ser al menos tan alta como la máxima ráfaga de todos sus hijos.

cburst

Es la cantidad de bytes en ráfagas en una velocidad “infinita”, en otras palabras, lo más rápido que la interfaz pueda transmitir. En un entorno perfecto, debería ser igual a un máximo de un paquete promedio. Debe ser al menos tan alta como la más alta *cburst* de todos los hijos.

Ejemplificación del funcionamiento de HTB.

Problema: Se tienen 2 clientes, A y B, ambos conectados a Internet a través de eth0. Se desea destinar 60 kbps a B y 40 kbps para A. Se quiere también subdividir el ancho de banda de A asignando 30 Kbps para la WWW y 10 Kbps para todo lo demás. Cualquier ancho de banda no utilizado puede ser utilizado por cualquier clase que lo necesite (en proporción de su cuota asignada).

HTB asegura que la cantidad de servicio prestado a cada clase es por lo menos el mínimo de la cantidad que ésta solicita y la cantidad máxima asignada para ella. Cuando una clase solicita menos de la cantidad asignada, el restante (el exceso) de ancho de banda se distribuye a otras clases que soliciten el servicio.

Los diferentes tipos de tráfico están representados por las clases de HTB. El enfoque más simple se muestra en la Ilustración 7.

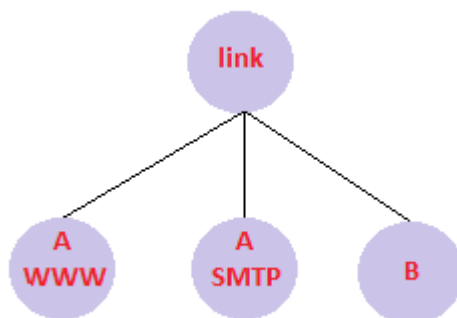


Ilustración 7: Árbol HTB simplificado

Vamos a ver lo que manda a utilizar:

```
tc qdisc add dev eth0 root handle 1: htb default 12
```

Este comando permite enlazar la disciplina de cola HTB a la interfaz eth0 y le asigna el manejador ("handle") **1:**. Esto es solo un nombre o identificador con el cual se referirán a ella más adelante. El valor por defecto **12 (default 12)** significa que cualquier tráfico que no está clasificado se le asignará a la clase 1:12.

Nota: En general (no sólo para HTB, sino para todas las qdiscs y clases en TC), los manejadores (handles) son escritos de la forma $x:y$ donde “ x ” es un entero que indentifica a una qdisc e “ y ” es un entero que identifica una clase perteneciente a la qdisc. El manejador (handle) de una qdisc debe tener cero para valor “ y ”, y el identificador de una clase debe tener un valor distinto de cero para su valor “ y ”. El “1:” es tratado como “1:0”.

```
tc class add dev eth0 parent 1: classid 1:1 htb rate 100kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:10 htb rate 30kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:11 htb rate 10kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:12 htb rate 60kbps ceil 100kbps
```

La primera línea crea una clase raíz (root) 1:1 bajo la clase 1:. La definición de una clase raíz es única con la qdisc HTB como su padre. Una clase raíz, al igual que otras clases bajo una qdisc HTB permite a sus hijas prestarse ancho de banda entre si, pero una clase raíz no puede pedir prestado de otra. Se podría haber hecho la creación de las 3 clases directamente bajo la qdisc HTB, pero entonces el exceso de ancho de banda de una clase no estaría disponible para las demás. En este caso, se quiere permitir el "endeudamiento", por lo que se tiene que crear una clase extra para que sirva como la raíz y dejar a las subclases que se lleven los datos reales bajo ella. Estos son definidos por las siguientes 3 líneas.

Nota: Por que se tiene que repetir "dev eth0" cuando ya han utilizado el "handle" o "parent". La razón es que los "handles" son locales a una interfaz, por ejemplo, eth0 y eth1 podrían tener cada una clases con "handle 1:1".

También se tiene que describir cuales paquetes pertenecen a cual clase. Lo anterior no está relacionado con la qdisc HTB y son los filtros a como se muestra a continuación:

```
tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 \
  match ip src 1.2.3.4 match ip dport 80 0xffff flowid 1:10
tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 \
  match ip src 1.2.3.4 flowid 1:11
```

(Se identifica a "A" por su dirección IP la cual debería ser 1.2.3.4).

Se puede ver que no se ha creado un filtro para la clase 1:12. Con lo cual se muestra el uso de la forma predeterminada. Cualquier paquete que no sea clasificado por las dos reglas anteriores (con cualquier paquete que tenga como IP origen la 1.2.3.4) se deberá poner en la clase 1:12.

Opcionalmente, se pueden adjuntar las disciplinas de colas a las clases terminales. Si no se especifica, el valor predeterminado es pfifo.

```
tc qdisc add dev eth0 parent 1:10 handle 20: pfifo limit 5
tc qdisc add dev eth0 parent 1:11 handle 30: pfifo limit 5
tc qdisc add dev eth0 parent 1:12 handle 40: sfq perturb 10
```

2.1.3.3 Otras

HFSC - (Hierarchical Fair Service Curve). Este algoritmo permite la distribución proporcional del ancho de banda así como el control y la asignación de las latencias. Esto da un mejor y más eficiente uso de una conexión para situaciones en las que tanto los servicios de ancho de banda de datos y servicios interactivos comparten un enlace de red.

PRIO (Multi Band Priority Queueing). La qdisc PRIO es una disciplina de encolado con clases sencilla que contiene un número arbitrario de clases con distintas prioridades. Las clases son desencoladas por orden de prioridad numérico de forma descendente. PRIO es un programador de paquetes que divide el tráfico basado en la configuración de los filtros. No es un ajustador de paquetes pero puede tener qdiscs en sus clases que si lo sean.

Un bug de PRIO es que con grandes cantidades de tráfico en las bandas bajas puede ocasionar la inanición de las bandas altas. Esto puede ser prevenido colocando una qdisc ajustadora (por ejemplo, TBF) a esas bandas para asegurarse que no puedan dominar el enlace.

Por mencionar, tenemos otras disciplinas de colas con clases como lo son: ATM (Asynchronous Transfer Mode), DSMARK (Differentiated Services Marker) e INGRESS (Ingress Qdisc).

2.2 AUTENTICACIÓN DE USUARIOS

Los servicios en una red, generalmente, están destinados a un grupo específico de usuarios ya sean de la misma organización/entidad/empresa o usuarios externos temporales tal es el caso de un hotel, comercio, entidad gubernamental, etc., que permite el uso gratis de su servicio a Internet. Sin embargo, en el amplio espectro de usuarios existen aquellos que quieren usar los servicios a los cuales no se les está permitido hacerlo y para ello emplean técnicas maliciosas en busca de debilidades en la red que puedan explotar para poder tener los permisos deseados.

Un aspecto importante a tener en cuenta en los servicios que presta una red es la identificación de los usuarios en la misma y para ello se pueden implementar los protocolos AAA - Autenticación, Autorización y Contabilización (Authentication, Authorization and Accounting en inglés). Entre estos protocolos encontramos RADIUS y TACACS como los más conocidos.

2.2.1 AAA (Autenticación, Autorización y Contabilización)

Es una colección de protocolos que se utilizan para el diseño de un sistema basado en la autenticación. AAA no es un sistema de autenticación sino un conjunto de normas sobre las cuales se creará el sistema.

La Autenticación permite controlar todos los aspectos relativos a la identificación de un usuario. Una vez autenticado permite la gestión de recursos o servicios para su uso mediante la Autorización. Una vez que un usuario está autenticado y autorizado se puede hacer una Contabilización para la gestión de reportes y estadísticas de uso de los recursos.

"En un esquema clásico cliente-servidor, el cliente es el equipo o usuario que solicitaría la autenticación o la entrada al sistema, y el servidor es el recurso que nos provee servicios. En el caso AAA, no se debe utilizar esta definición de cliente y servidor, ya que esto nos podría llevar a equívocos, puesto que estos conceptos "Cliente/Servidor" son más ambiguos. En esta cadena de participantes existen diferentes componentes que la hacen un poco más compleja.

- El equipo o usuario que solicita autenticación o entrada se llama **suplicante**, y no es obligatoriamente quien inicia la secuencia de autenticación.
- El equipo de red que hace de puerta de entrada física a la red, llamado **NAS** (Network Access Server) o servidor de acceso a la red, es el que permite la entrada física a la red y tramita nuestra autenticación. Puede ser una pila de modems, un switch de red, un punto de acceso (AP), un router, etc. Este equipo suele ser quien inicia la secuencia de autenticación al detectar una conexión activa en una de sus puertas, por ello se le denomina **autenticador**. En AAA también se le denomina "equipo de servicio" o Service Equipment. El NAS es en la realidad el centro de todo este sistema AAA, ya que es el responsable de abrir o limitar las características reales de funcionamiento de la infraestructura.

- El **Servidor de Autenticación** que puede ser RADIUS, TACACS, entre otros; es el que dirige todo el proceso de autenticación, autorización y arqueo de los equipos y usuarios que solicitan acceso. Por supuesto en AAA se le conoce como "Servidor AAA".
- El servidor de Autenticación o servidor AAA puede hacer el papel de **Proxy**, elevando las consultas a otros servidores AAA. De esta manera un Servidor AAA que hace de Proxy AAA se convierte en cliente de otro servidor.
- El **servidor de directorio** o **servidor de base de datos** de usuarios y credenciales, al cual el servidor de autenticación va a solicitar los datos de autenticación de los solicitantes de acceso. Ésta pudiera ser la misma máquina que el servidor de autenticación, aunque en instalaciones reales no suele serlo. Puede ser un servidor de AD (Active Directory), de LDAP (Lightweight Directory Access Protocol), una base de datos SQL (MySQL, Microsoft SQL Server, Oracle, etc.) o un servidor Unix con credenciales de usuario.
- El **servidor de recursos y servicios** que necesita el usuario para realizar su cometido en la red. Puede ser un servidor de almacenamiento de datos, un servidor Web, etc.
- El **Proveedor de Servicios** (Service Provider) en el modelo AAA es el propietario de la infraestructura de acceso a la que se conecta el usuario y por lo tanto es el propietario del servidor AAA y del equipo de servicio o NAS. Puede haber varios proveedores de servicio que colaboren entre sí. Si esto ocurre, y existen varios proveedores en colaboración para prestar un servicio, el servidor contractual del usuario se denomina **UHO** (User Home Organization)

Es por esta secuencia que se hace difícil definir de forma general en el estándar AAA quién es el cliente y quién es el servidor. En la terminología que utiliza AAA, el cliente es aquel que envía paquetes con estructura AAA a un servidor AAA. Pero eso lo puede hacer el suplicante, así como el equipo NAS o el propio servidor, por lo que dependería del contexto el poder utilizar el término cliente. Para evitar confusiones hablaremos de: suplicante, NAS o autenticador y Servidor de Autenticación. En líneas generales, en el proceso de configuración de RADIUS, se habla de clientes para definir a los NAS.", Fernández, Ramos & García-Morán(2009).

2.2.1.1 Autenticación (Authentication)

La Autenticación es el proceso más importante en AAA debido a que una entidad o usuario deberá probar su identidad ante otra entidad. Ante esto, la autenticación debe responder a la pregunta ¿Quién o qué entidad pretende acceder a lo servicios que presto?

En el proceso de autenticación que realiza una entidad o usuario para acceder a una red, no es la entidad quien se comunica con el servidor AAA sino que la entidad se comunica con el NAS y éste es quien se encarga de conversar con el servidor de autenticación enviando un mensaje de Solicitud de Acceso (Access – Request).

El suplicante envía el nombre de usuario y contraseña hacia el cliente NAS. El NAS envía un mensaje de solicitud de acceso al servidor de autenticación con los datos que le ha pasado el suplicante.

2.2.1.2 Autorización (Authorization)

En la Autorización se debe contestar la pregunta ¿A qué servicios voy a permitir acceder al solicitante, una vez autenticado?

La autorización se encarga de dar los privilegios necesarios a un suplicante en base a su identidad que previamente ha sido autenticada. Las autorizaciones pueden también estar basadas en restricciones, tales como restricciones horarias, sobre la localización de la entidad solicitante, la prohibición de realizar logins múltiples simultáneos del mismo usuario, etc. La mayor parte de las veces el privilegio concedido consiste en el uso de un determinado tipo de servicio como puede ser asignación de ancho de banda, cifrado, asignación de direcciones, entre otros.

2.2.1.3 Contabilidad (Accounting)

La Contabilidad es una fase de estadísticas y recolección de datos sobre la conexión que son almacenados generalmente en base de datos SQL. Son datos que permiten tomar decisiones para una administración, planificación o facturación de acuerdo al uso de los recursos de la red por parte de los usuarios.

2.2.2 RADIUS

La palabra RADIUS viene de Remote Authentication Dial-Up Server que significa Servidor de Autenticación Remota para sistemas de Mercado Telefónico a Redes. El nombre se deriva del hecho de que el único uso de RADIUS era el acceso a redes a través de MODEM.

Fue desarrollado por Livingston Enterprises en 1991. Un año más tarde se formó un grupo de trabajo en la IETF llamado NASREQ para gestionar los requerimientos de los equipos NAS de manera que existiese un estándar para ello. En 1994, Livingston liberó el código fuente para el grupo de trabajo dejando una puerta abierta a los fabricantes de Hardware y Software para que incorporaran o mejoraran dicho código.

El protocolo RADIUS se ejecuta en múltiples plataformas (Unix, GNU/Linux, Windows, Solaris, etc.) utilizando el protocolo UDP mediante los puertos 1812 para la autenticación y 1813 para la contabilidad. Además, algunos servidores como freeRADIUS utilizan el puerto UDP 1814 para la escucha de respuestas Proxy RADIUS de otros servidores.

2.2.2.1 Métodos de autenticación

"Los métodos de autenticación son paquetes de software o módulos de software sobre los que se basa el proceso de autenticación de usuario de una plataforma RADIUS. Estos módulos son complejas cajas matemáticas encargadas de realizar el cifrado, descifrado y empaquetado de todos los procesos complejos de autenticación. Desde el método nativo de RADIUS que es PAP hasta los más actuales como algunos tipos nuevos de EAP, la evolución en cuanto a seguridad ha sido notable. Cuando RADIUS recibe una solicitud de acceso, va pasándola por cada uno de los módulos de autenticación

que tenga activados en su configuración, hasta que alguno de esos módulos reconozca sus algoritmos o las credenciales del usuario y se encargue de validar la autenticación." Fernández, Ramos & García-Morán(2009)

- **PAP** (Password Authentication Protocol o protocolo de autenticación mediante contraseña). El más sencillo y vulnerable método de autenticación que hay actualmente ya que transmite el nombre de usuario y contraseña en texto plano.
- **CHAP** (Challenge Handshake Authentication Protocol o protocolo de desafío mutuo). Se creó como una mejora para PAP. Tanto el suplicante como el servidor tienen el conocimiento de una contraseña para cada uno. Cuando el suplicante trata de autenticarse, el servidor le envía un desafío y el suplicante la devuelve junto con su contraseña mediante MD5. El servidor, dado que ya conoce el valor calculado, compara el resultado obtenido y si todo está bien, permite la autenticación del suplicante.
- **MS-CHAPv1**. Se basa en desafíos para sistemas Microsoft. Lo bueno de MS-CHAP es que tanto cliente como servidor procesan el desafío utilizando el valor hash de la contraseña y no la contraseña en si por lo que no hay almacenamiento de contraseña en texto plano
- **MS-CHAPv2**. Es una mejora de la v1 soportando cambios de contraseña y mensajes de respuesta con estados. Tiene soporte en las versiones de Windows 2000 en adelante.
- **UNIX**. Se puede utilizar el sistema de almacenamiento de usuarios y contraseñas nativo en Unix y Linux.
- **HTTP Digest**. Autenticación por desafío para clientes de servidores web con autenticación RADIUS. A la contraseña se le aplica una función hash antes de enviarla sobre la red. Usa MD5 pero también puede manejar otros algoritmos como SHA-1.

Métodos EAP

Entre la lista de métodos EAP nos podemos encontrar con: **EAP-MD5**, **EAP-OTP** (One Time Password), **EAP-GTC** (Generic Token Card), **EAP-MS-CHAP**, **EAP-MS-CHAPv2**, **EAP-SIM**, **EAP-AKA**, basados en Cisco (**EAP-LEAP** y **EAP-FAST**), basados en PKI (**EAP-TLS**, **EAP-TTLS** y **EAP-PEAP**), Biometría, entre otros.

Hay otros métodos de autenticación, solamente por mencionarlos son: autenticación simple y autenticación mutua, autenticación contra archivos de usuarios, autenticación contra el sistema operativo, autenticación contra base de datos y autenticación contra servicios de directorio.

2.2.2.2 Autenticación EAP

Extensible Authentication Protocol es un protocolo que se encarga del transporte, encapsulado y de la seguridad en la autenticación; por lo que no se debe confundir a EAP como un protocolo de autenticación. En el interior de EAP se encuentran los métodos de autenticación que se van a utilizar como PAP, CHAP, TLS, TTLS, PEAP, SIM, AKA, Kerberos, entre otros.

EAP trabaja de forma independiente en la capa de enlace y por ser un protocolo de transporte dispone de sus propios sistemas de control de entrega, retransmisión y de integridad del paquete.

2.2.2.2.1 EAP-MD5

“EAP-MD5 es la primera versión, la más simple y, por lo tanto, la más insegura de EAP. El método utilizado desarrollado por RSA es análogo a CHAP. Como su nombre indica, utiliza el algoritmo MD5 para obtener un hash de la contraseña de usuario, lo que lo hace muy vulnerable a ataques de fuerza bruta o a ataques mediante Rainbow Tables. Es un método de autenticación de una sola dirección (el servidor autentica al cliente pero no viceversa). Se considera el más inseguro de los tipos de EAP, ya que no incorpora ningún sistema de encriptación de los paquetes, que circulan en texto plano. No incorpora la característica de generar claves de sesión para el cifrado de protocolos como WEP o WPA como lo hagan TLS, TTLS o PEAP. Solo se debe utilizar en canales de autenticación difíciles de interceptar como 802.1X para redes cableadas y con mucha precaución.”, Fernández, Ramos & García-Morán(2009)

2.2.2.2.2 EAP-TLS

EAP-TLS (Transport Layer Security) utiliza certificados X.509 en una infraestructura de clave pública (PKI). Cuando un suplicante envía una solicitud de acceso al servidor mediante su nombre de usuario, el servidor responde con un certificado que el suplicante deberá verificar. Luego de verificado el certificado del servidor, el suplicante envía su propio certificado al servidor para que éste también lo verifique. Si todo va bien, se establece un canal seguro mediante TLS para el intercambio de credenciales u otros métodos de autenticación. Finalizado este proceso, el servidor entrega al suplicante y al NAS una clave única para establecer una sesión segura en la comunicación.

EAP-TLS es un método de autenticación mutua ya que tanto el suplicante debe autenticarse contra el servidor como el servidor deberá hacerlo contra el suplicante. Esto conlleva a la necesidad de dos certificados X.509, uno para el servidor de autenticación y otro para el suplicante. La gran importancia de esto es que se evita el conocido ataque del hombre en medio donde un suplicante puede entregar sus credenciales a un servidor falso.

Una debilidad que tiene EAP-TLS es que al inicio del intercambio de datos desde el cliente, antes del intercambio de certificados, éste envía el nombre de usuario en texto plano por lo que si alguien intercepta ese tráfico podrá tener el nombre de usuario. Aunque el nombre de usuario no basta, ya es un dato más que se puede usar en la fase de enumeración en un ataque a la red.

2.2.2.2.3 EAP-TTLS

Es una extensión de EAP-TLS y no se basa en la autenticación mutua de dos certificados puesto que solo el servidor debe tener un certificado X.509. Pero el cliente tiene la opción de usar un certificado X.509 si así lo desea.

TTLS(Tunneled Transport Layer Security) crea dos túneles de seguridad. El primer túnel se crea para el intercambio de credenciales y el segundo para el envío de la contraseña cifrada para la sesión. Toda la información pasa cifrada incluyendo los mensajes EAP de autenticación exitosa o fallida.

Cuando el cliente comprueba el certificado del servidor, se crea un túnel para el traspaso de las credenciales del cliente. En este canal se pueden usar otros métodos de autenticación como PAP, CHAP, MS-CHAP sin suponer un riesgo de seguridad ya que la información va en un canal cifrado.

El hecho de tener un solo certificado para el servidor, quita carga al no tener que estar generando certificados para cada cliente que se desee conectar a la red.

Entre las ventajas de TTLS está el echo de usar un sistema de atributos parecidos al de RADIUS en la forma de AVP (Par Atributo/Valor) intercambiando estos datos en un canal cifrado. Además, en cierta forma resuelve el problema de seguridad de EAP-TLS referente a la captura del nombre de usuario puesto que al inicio de la autenticación lo que se envía es un usuario anónimo.

Para sistemas de Microsoft no hay soporte nativo para EAP-TLS pero si se puede implementar con suplicantes ajenos como SecureW2, WPA_Suppllicant, entre otros.

2.2.2.2.4 EAP-PEAP

Protected Extensible Authentication Protocol(Protocolo de Autenticación Extensible Protegido) o simplemente PEAP, es un protocolo que encapsula a EAP dentro de un tunel(TLS) encriptado y autenticado. PEAP fue desarrollado conjuntamente por Cisco Systems, Microsoft y RSA Security.

“PEAP es similar en diseño a EAP-TTLS, requiriendo un certificado del lado del servidor para crear un tunel TLS seguro para proteger la autenticación del usuario. PEAP crea un túnel cifrado TLS entre el cliente y el servidor de autenticación. En la mayoría de las configuraciones, las claves de este cifrado se transportan utilizando la clave pública del servidor. El subsiguiente intercambio de información de autenticación en el interior del túnel para autenticar al cliente se cifra, así las credenciales del usuario están seguras.”, Wikipedia

“EAP-PEAP soporta como métodos de autenticación a MS-CHAPv2 y GTC(Generic Token Card). PEAP también es un protocolo muy recomendado para la securización de redes inalámbricas como WiFi y está extensamente soportado por la mayoría de los fabricantes de equipos compatibles con 802.1X y 802.11i. EAP-PEAP dispone también de la función fase reconnect para el restablecimiento de sesiones TLS. Ambos protocolos permiten el intercambio dinámico de claves como las que se precisan en WiFi o VPN.”, Fernández, Ramos & García-Morán(2009)

2.2.2.2.5 Tabla comparativa de tipos de EAP

Dado que hay muchas implementaciones de EAP, en la Tabla 1 se muestran algunas muy comunes entre las que se puede elegir. Los datos fueron tomados de: Fernández, Ramos & García-Morán(2009).

	EAP-MD5	LEAP	EAP-TLS	EAP-TTLS	EAP-PEAP	EAP-FAST
Certificado de Servidor	No	Challenge	Si	Si	Si	No PKI Shared Secret
Certificado de Cliente	No (Nombre de usuario y contraseña mediante challenge)	No (Nombre de usuario y contraseña mediante challenge)	Obligatorio (posible smartcard)	Opcional (credenciales de usuario)	Opcional (credenciales de usuario)	PAC No PKI
Validación de certificados	No	No	OCSP TLS	OCSP TLS	OCSP TLS	No
Credenciales soportadas	MD5 hash	Hash similar a MS-CHAP	Certificados de cliente	CHAP, PAP, MS-CHAPv1 y v2	EAP-MSCHAPv2, EAP-GTC, otros	PAC
Soporta cambio de contraseña	No	No	No	Si	Si	Si
Autenticación mútua	No (Solo cliente)	Si (Challenge)	Si	Si	Si	Si
Tunelamiento	No	No	Si, TLS	Si, TLS	Si, TLS	Si, TLS
Entrega de claves dinámicas	No	Si	Si	Si	Si	Si
Reconexión rápida	No	No		Si	Si	Si
Base de datos de autenticación	SQL, en formato MD5	AD, NTLM	AD, NTLM, Token, LDAP, Novell NDS, OTP	AD, NTLM, LDAP, Token	AD, NTLM, Novell NDS, Token	AD, NTLM, LDAP
Desarrollador	Estándar	Solo Cisco	Microsoft	Punk y Certicom	Microsoft, Cisco y RSA	Cisco
Suplicantes que lo soportan	Microsoft, WPA, MacOS Supplicant	Proprietarios MacOS Linux	Microsoft Linux MacOS	Juniper, Oddissey SecureW2, MacOS WPA Supplicant	Microsoft Linux MacOS	Cisco
Muestra nombres de usuario	Si	Si	Si	Anónimo en la Fase 1	Anónimo en la Fase 1	Si
Vulnerable MiTM	Si	Si	No	No	No	No
Vulnerable actualmente	Si Diccionario	Si	No	No	No	No
Usos recomendados	Solo redes cableadas 802.1X	No recomendado	802.1X Alámbrica e Inalámbrica Smartcards	802.1X Alámbrica e Inalámbrica	802.1X Alámbrica e Inalámbrica	Redes con equipos Cisco propietarios

Tabla 1: Comparativa de tipos de EAP

2.2.3 Shared secret

El shared secret o secreto compartido de RADIUS es una contraseña que está definida tanto en el cliente(NAS) como en el servidor RADIUS. Este secreto compartido se usa para el cifrado de las comunicaciones entre el cliente y el servidor para evitar que intercepten el tráfico en texto plano.

"El funcionamiento del shared secret es muy similar al del sistema de desafío o challenge utilizado en muchos protocolos de autenticación como CHAP o NTLM. Muchos de ellos están dejando de utilizarse o utilizan algoritmos de reducción diferentes a MD5 que es el que utiliza RADIUS. MD5 es un algoritmo de reducción del mensaje, no de cifrado de mensaje, y se utiliza para no tener que transportar una clave en texto plano. Esa clave se pasa por una función MD5 que la procesa en una Base generada o hash que es irreversible; no se debe poder obtener la clave inicial a través de ese hash.", Fernández, Ramos & García-Morán(2009).

"En una autenticación mediante RADIUS, el cliente o NAS comparte el mismo secreto que el servidor, por eso se llama secreto compartido, claro. Cuando se produce una solicitud de acceso (Access-Request) el NAS envía esta petición en texto plano, cifrando solamente el campo de password o contraseña de usuario. Para ello el NAS genera una frase de 16 bytes de código aleatorio que se denomina Request Authenticator, y que incluye en el paquete de solicitud. Esta frase se concatena con el secreto compartido y se pasa el resultado por una función MD5 que crea un hash de 16 bytes, al que se le aplica la función .XOR con los primeros 16 bytes de la contraseña del usuario. Si la contraseña midiera más de 16 bytes se repetiría la misma fórmula tantas veces como segmentos de 16 bytes tuviera la contraseña, excepto porque la concatenación de la función MD5 se realiza usando en vez del Request Authenticator el resultado de la primera frase cifrada.", Fernández, Ramos & García-Morán(2009).

Es muy importante tener en mente la creación de shared secrets algo seguras y difíciles de adivinar que además NO estén basadas en diccionarios. Debe ser una contraseña bastante aceptable donde se mezclen letras mayúsculas, letras minúsculas, números y símbolos.

2.2.4 Atributos AVP

AVP (Attribute Value Pairs) son como las variables de programación a las que se les establece un significado o función, un formato o tipo de datos, y unos valores formateados para esos datos. Sus usos son muy diversos, desde definir un nombre de usuario o una contraseña hasta establecer el modo de funcionamiento de una puerta de acceso.

2.2.5 Secuencia de autenticación de RADIUS

Tomado de Fernández, Ramos & García-Morán(2009), en la Ilustración 8 se muestra la secuencia de comunicación entre el suplicante, el NAS y el Servidor de autenticación.

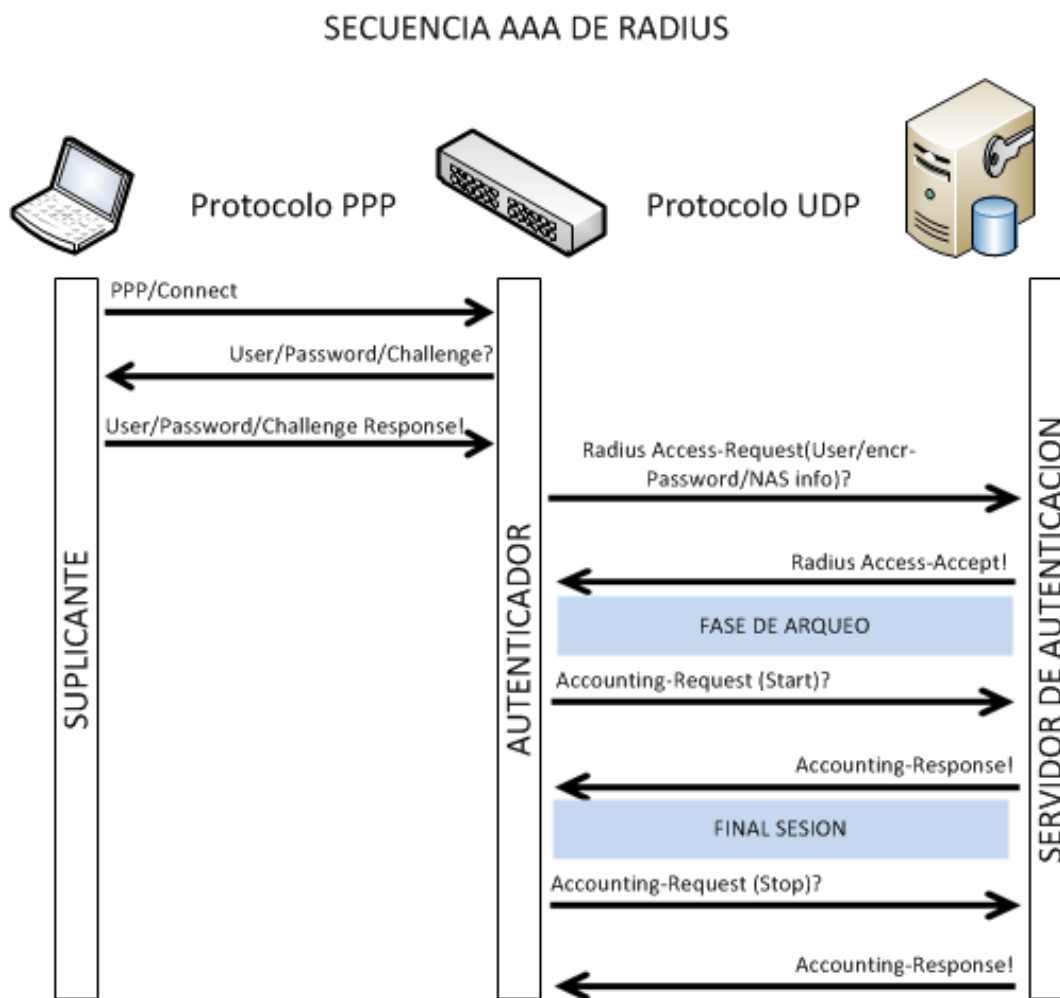


Ilustración 8: Secuencia de comunicación AAA

- El proceso suele comenzar por un Access-Request. La solicitud de acceso es un mensaje que contiene atributos como el nombre de usuario, la contraseña (en caso de PAP) el número de puerto NAS y el ID de cliente. El NAS envía esta solicitud al primer servidor RADIUS que tenga preestablecido en su lista de servidores. Si no recibe respuesta en un tiempo a determinar, reintentará el envío un número de veces designado a ese o a otros servidores RADIUS de la red.
- El servidor RADIUS que recibe la solicitud comprueba si proviene de un equipo NAS autorizado; si no es así la descarta silenciosamente (silently discard). Si el cliente está en su lista y el shared secret es el correcto, comprueba en la(s) base de datos de usuarios el nombre y la contraseña. Si este registro fuera acompañado de otros atributos de servicio (dirección IP, ancho de banda asignado, etc.) se los enviaría al cliente o NAS para que los gestione.
- Si el tipo de autenticación es CHAP o cualquier otra basada en el desafío, se envía al suplicante un mensaje de Access-Challenge con la frase aleatoria que debe calcular. En algunos métodos basados en desafío como OTP (One Time Password o contraseña de un solo uso) como Token, dispositivos de cifrados USB, etc. se solicita en este momento al usuario un PIN, contraseña, la

introducción de una tarjeta o dispositivo de cifrado USB, etc. para volver a enviar un Access-Request con los nuevos datos calculados. Este proceso se puede volver a repetir varias veces dependiendo del método de autenticación utilizado.

- Tras comprobar todos estos datos de autorización y la base de datos de credenciales, decidirá si acepta o deniega la solicitud, enviando un mensaje de Access-Accept o Acces-Reject al NAS con los atributos necesarios para activar o denegar el servicio.
- En ese momento el NAS abrirá el puerto solicitado con los atributos designados, y enviará un mensaje de Accounting-Request [Start] al servidor RADIUS indicando que ha comenzado a registrar los datos de arqueo de la sesión del usuario y pasándole los datos del inicio de sesión. Algunos equipos NAS no soportan el arqueo de cuentas, y se saltan este proceso.
- El servidor RADIUS ratifica la recepción del inicio de sesión, devolviendo al NAS un mensaje Accounting-Response [Start], y guardando en su base de datos o archivo log los datos de inicio de sesión del usuario. Si el servidor no envía este mensaje puede que se deba a que no ha recibido la solicitud de inicio, por lo que el NAS deberá reenviar a este o a otro servidor su solicitud de inicio de arqueo.
- Al desconectar el suplicante o finalizarse la sesión por cualquier causa, el NAS envía un mensaje de Accounting-Request [Stop] al servidor para indicarle el fin de la sesión del usuario, y enviarle los datos de consumo del usuario.
- El servidor (muy educado) ratifica la recepción de esos datos mediante un mensaje Accounting-Response [Stop] que envía al NAS. Si no se produce este mensaje el NAS deberá seguir intentando enviar a este o a otro servidor de arqueo su solicitud de fin de sesión con los datos de la misma. La pérdida de esta información suele conllevar un perjuicio económico, por lo que debe ser tratada con la adecuada atención.

2.2.6 El estándar 802.1X

802.1X se usa como una extensión en el sistema de autenticación RADIUS que tiene como finalidad la securización de la conexión de dispositivos en la infraestructura de red de la organización. Este estándar trabaja sobre la Capa 2 o capa de enlace estableciendo la seguridad de control de acceso.

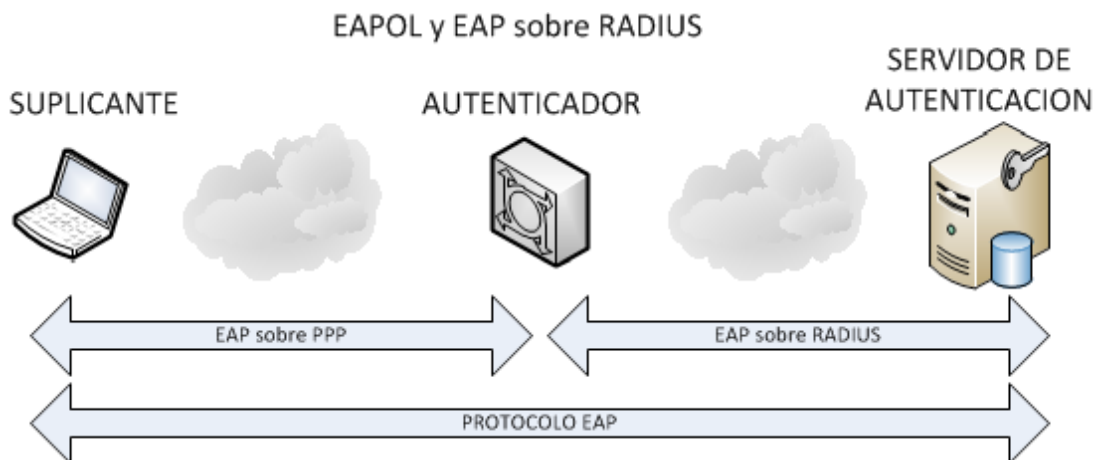
Trabaja tanto para redes cableadas como inalámbricas y el procedimiento de conexión utilizado por este estándar se basa en la provisión de un sistema de autenticación por puertos en los que se establece un canal o puerto blindado de comunicación entre el suplicante y el NAS a fin de permitir únicamente su autenticación.

“La seguridad en la capa de enlace provee de mecanismos necesarios para viabilizar los procesos de autenticación y envío de tramas de control. Esto es lo que se conoce como la seguridad en la capa de enlace que además expande esa seguridad a todas las capas superiores. 802.1X es comúnmente llamado EAPoL (Extensible Authentication Protocol over Lan) o EAP sobre Ethernet. Si es aplicado a tecnologías inalámbricas se llamaría EAPoW (EAP over Wireless). EAP es el protocolo de transporte de la autenticación nativo de 802.1X, ya que trabaja contra servidores de autenticación sobre tramas

Ethernet. También se define como un sistema de autenticación basado en puertos puesto que el control de admisión se realiza a través de puertos virtuales LAN.”, Fernández, Ramos & García-Morán(2009).

2.2.7 Estructura de las comunicaciones EAP

En la Ilustración 9 se muestra, de forma resumida, la estructura de comunicación EAP



2.2.7.1 Secuencias de autenticación EAP

En Fernández, Ramos & García-Morán(2009) indican que es importante conocer la forma en la que se desarrolla una secuencia completa AAA basada en el protocolo de transporte EAP. No hay una fórmula universal para realizar esta secuencia, ya que dependerá de ciertos factores variables, como por ejemplo el tipo de EAP utilizado o de la capacidad de iniciar la conversación o no por parte del equipo NAS. En el caso de un conmutador que haga de autenticador, será éste quien detecte que un equipo se conecta a uno de sus puertos, o que un equipo acaba de activarse o encenderse; pero en el caso de un AP inalámbrico que no tiene puertos físicos, el suplicante debe iniciar la conversación tras asociarse con el punto de acceso mediante un mensaje EAPOL-Start, ya que el AP no es capaz de detectar al nuevo cliente de otra manera.

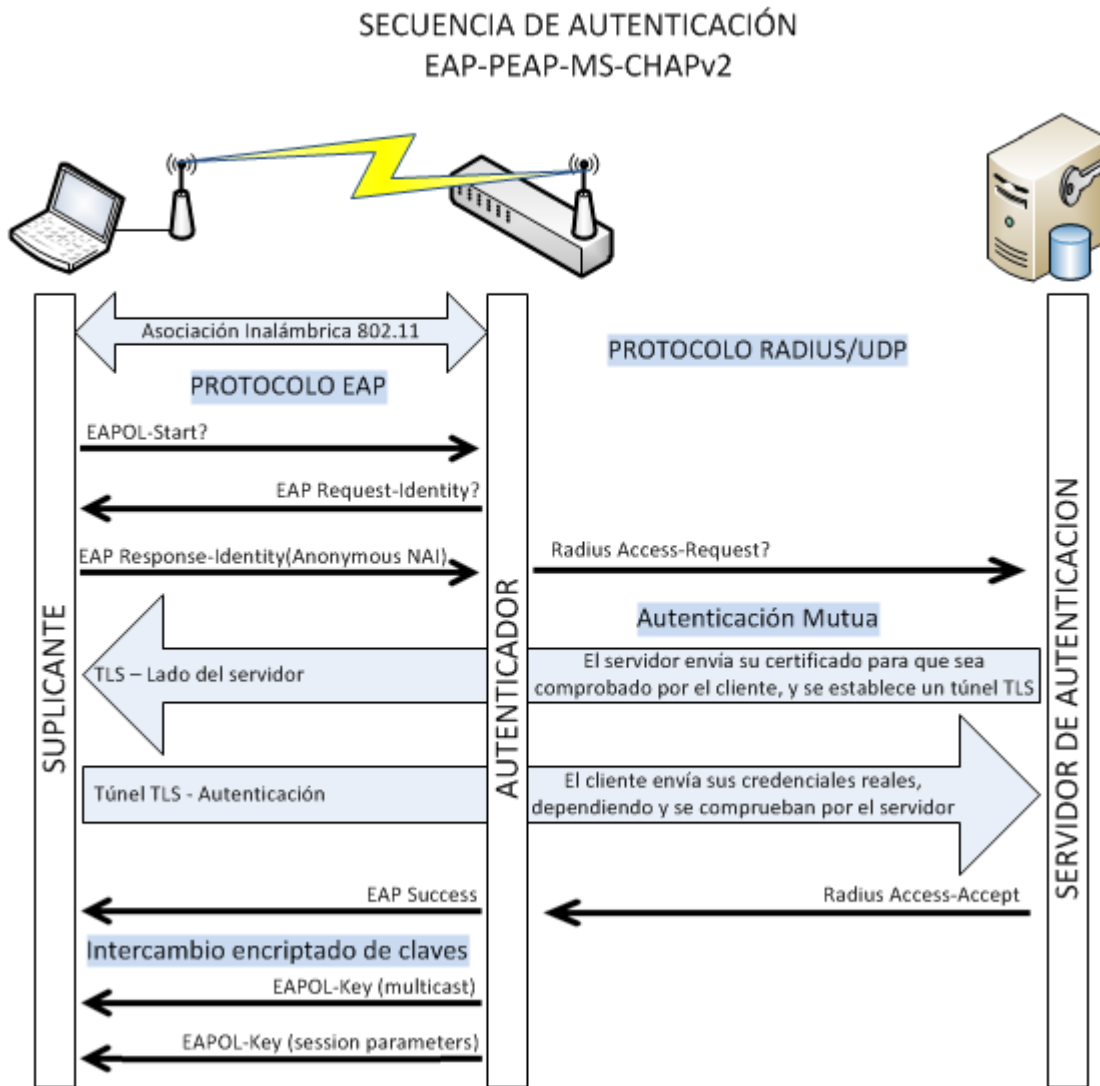


Ilustración 10: Secuencias de autenticación EAP-PEAP/MS-CHAPv2

La Ilustración 10 muestra la secuencia EAP-PEAP-MSCHAPv2 (A continuación una breve descripción)

<Inicio y solicitud de identidad y tipo de EAP >

- 0. EAPOL-Start (Suplicante-NAS)
- 1. EAP -Request/Identity (NAS-Suplicante)
- 2. EAP-Response/Identity (Suplicante-NAS)
- 3. EAP-Response/Identity (NAS-RADIUS)
- 4. EAP-Request/Start PEAP (RADIUS-NAS)

< Fase de establecimiento del canal TLS >

- 5. Intercambios para TLS (RADIUS-Suplicante)
- 6.

Suplicante.....	Autenticador.....
EAP-Response/Identity (Anonymous NAI) ->	<- EAP-Request/Identity
EAP-Response/EAP-Type=BAP-TLS (TLS client_hello)->	<- EAP-Request/IAP-Type=EAP-TLS (TLS Start)
WAP-Response/EAP-Type=EAP-TLS (TLS certificate (no cert), TLS client_key_exchange, TLS change_cipher_spec, TLS finished) ->	<- EAP-Request/EAP-Type=EAP-TLS (TLS server_hello, TLS certificate, [TLS server_key_exchange, TLS certificate_request, TLS server_hello_done)
EAP-Response/EAP-Type=EAP-TLS (TLS client_hello)->	<- EAP-request/EAP-Type=EAP-TLS (TLS change_cipher_spec,finished, hello_request)
EAP-Response/EAP-Type=EAP-TLS (TLS certificate, TLS client_key_exchange, TLS certificate_verify, TLS change_cipher_spec, TLS finished) ->	<- EAP-Request/EAP-Type=EAP-TLS (TLS server_hello, TLS certificate, TLS server_key_exchange, TLS certificate_request, TLS server_hello_done)
EAP-Response/EAP-Type=EAP-TLS ->	<- EAP-Request/EAP-type=EAP-TLS (TLS change_cipher_spec, TLS finished)
	<- EAP-Success

< Fase de autenticación EAP-MS-CHAP-V2 dentro del túnel TLS >

6. EAP- Request/Identity (RADIUS-Suplicante)
7. EAP- Response/Identity (Suplicante-RADIUS)
8. EAP- Request/EAP-MS-CHAP-V2 Challenge (RADIUS-Suplicante)
9. EAP-Response/EAP-MS-CHAP-V2 Response (Suplicante-RADIUS)
10. EAP-Request/EAP-MS-CHAP-V2 Success (RADIUS-NAS)
11. EAP-Response/EAP-MS-CHAP-V2 Ack (Suplicante-RADIUS)
12. EAP-Success (RADIUS-Suplicante)

< Fase de intercambio de claves de cifrado >

13. EAPOL-Key (multicast) (RADIUS-Suplicante)
14. EAPOL-Key (session key) (RADIUS-Suplicante)

Explicado al detalle:

0. Cuando el suplicante o el NAS se han detectado, la secuencia suele comenzar por un paquete EAPOL-Start, para indicar que se desea iniciar una conversación EAP.
1. Cuando el NAS (autenticador) detecta un paquete EAPOL-Start (cuando detecta la actividad del suplicante conectándose a uno de sus puertos, o asociándose a 802.11) envía un paquete EAP tipo “EAP-Request/Identity” al suplicante.
2. El suplicante responde mediante un paquete de tipo “EAP-Response/Identity” con su identidad (en PEAP debe ser anónima en esta fase). El NAS envía este paquete EAP encapsulado en forma de atributo EAP-Message dentro de un paquete UDP estándar de RADIUS.
3. El servidor RADIUS solicita un desafío o challenge al NAS, dependiendo del método de autenticación EAP demandado y disponible. El NAS realiza su misión de desencapsular y encapsular los mensajes tipo RADIUS a EAP y se lo envía al suplicante. Se producen intercambios de paquetes Challenge, dependiendo del método EAP y del tipo de autenticación mutua.
4. El suplicante responde a los mensajes Challenge a través del NAS, que siempre encapsula todos los mensajes EAP en formato RADIUS y se los envía al Servidor RADIUS.
5. Tras los procesos necesarios para la creación del túnel e intercambios de identidades y si además cada uno de los procesos de autenticación, dentro y fuera del túnel, se han superado correctamente, el servidor RADIUS responderá con un mensaje Access-Accept enviado al NAS con todos los parámetros necesarios de autorización para permitir el acceso del puerto del cliente. El NAS responderá al suplicante con un mensaje EAP-Success.
6. Se produce (en caso de VPN o red inalámbrica Wi-Fi) el intercambio de las claves de sesión necesarias para que se establezca un correcto cifrado durante la consiguiente sesión segura del usuario a la red.

Estos procesos descritos anteriormente pueden variar ligeramente o drásticamente, dependiendo del tipo de EAP que se decida implantar; pero la teoría es la misma para cualquiera de los intercambios de paquetes entre los diferentes componentes.

3 CONFIGURACIÓN DE PAQUETES

3.1 Configuraciones para soporte IMQ

3.1.1 Instalación del Sistema Operativo Debian 6

El sistema operativo a utilizar será GNU/Linux Debian 6 debido a su amplio uso y por sobretodo el soporte para una gran variedad de paquetes. Desde la web www.debian.org se puede descargar el instalador.

Cuando arranca el equipo, nos mostrará la primera pantalla de instalación del sistema operativo (ver Ilustración 11) donde se puede escoger entre instalación a modo gráfico o modo texto. Para esta instalación se escogió modo texto.



Ilustración 11: Inicio de instalación Debian

A continuación, se deberá escoger el idioma del proceso de instalación, la ubicación del país para fijar la zona horaria y el idioma o distribución que deberá tener el teclado.

En vista que este sistema dispondrá de dos tarjetas de red (ver Ilustración 12) las cuales se usarán una para la red local y otra para la conexión a Internet. Se deberá escoger la tarjeta que se desee usar como primaria para la conexión a Internet. Se hace la recomendación de desconectar los cables de red del equipo a fin de que no pueda tomar dirección IP automáticamente. Luego se configurará.

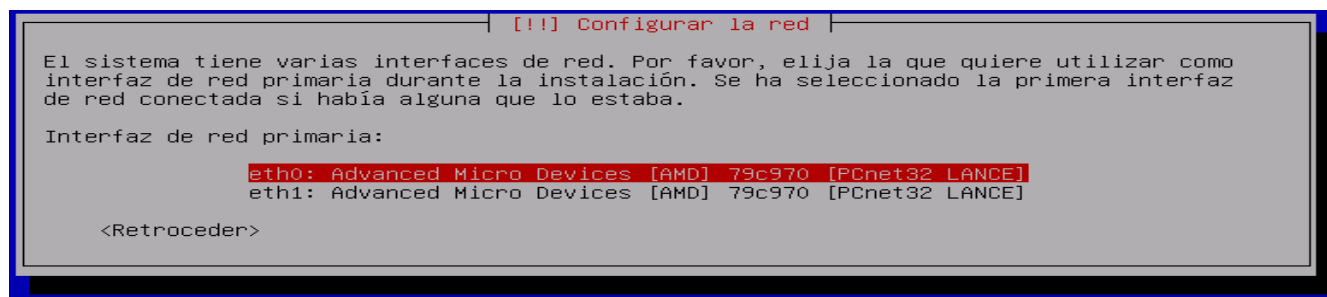


Ilustración 12: Instalación Debian, selección de tarjeta de red primaria

Configuración de paquetes

En caso de tener conectado el cable de red y no contar con un servicio DHCP, mostrará un error (ver Ilustración 13) al no encontrar un servidor DHCP activo en la red. No hay problema con esto puesto que la IP se le asignará de forma manual. También, si se ha olvidado desconectar el cable de red, se puede escoger la opción Retroceder para configurar la IP.

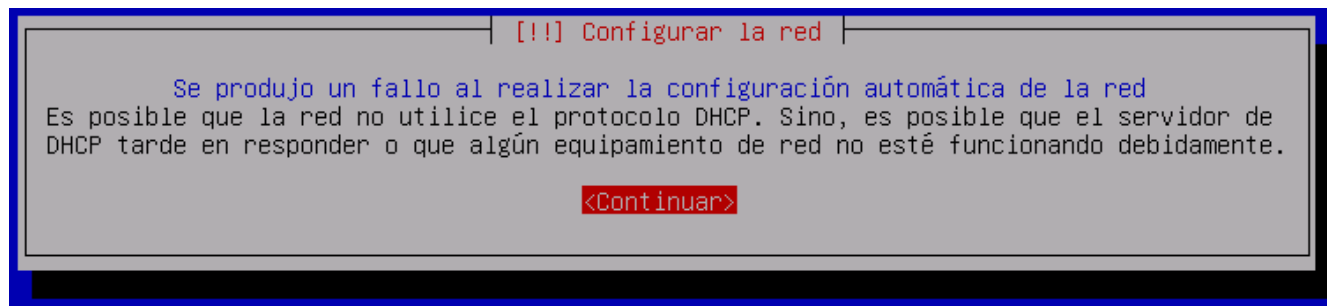


Ilustración 13: Instalación Debian, fallo de configuración IP automática

En la Ilustración 14 se muestra la siguiente opción a escoger: Configurar la red manualmente

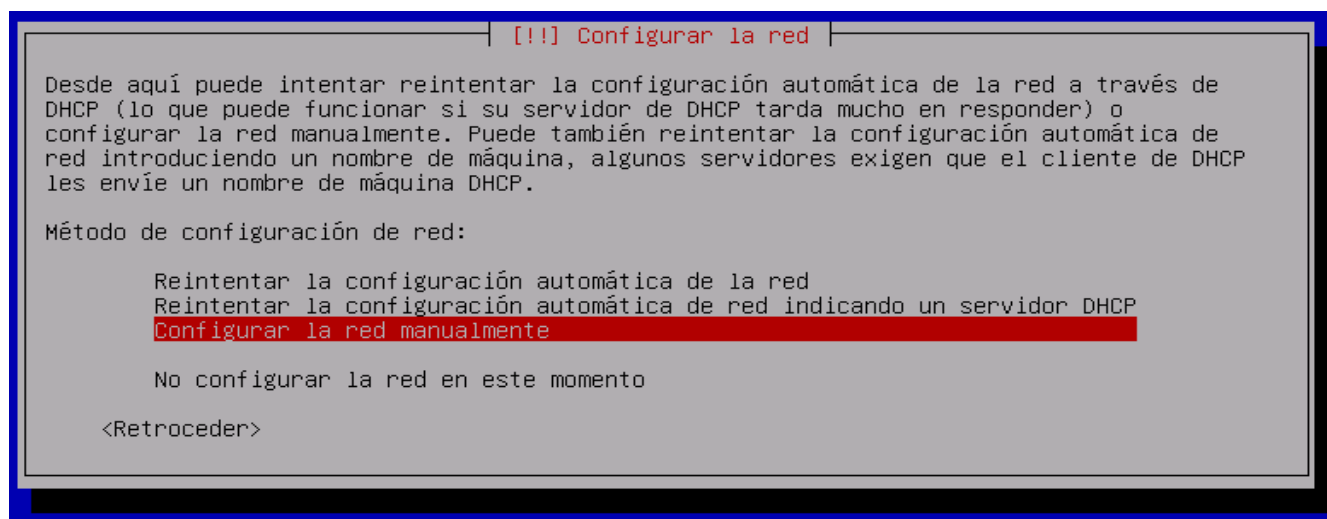


Ilustración 14: Instalación Debian, configurar red manualmente

En las 6 pantallas siguientes, se pedirá la configuración para:

1. dirección IP
2. máscara de red
3. puerta de enlace (o pasarela o gateway en la red)
4. servidor de nombres(DNS)
5. nombre de la máquina con el cual se dará a conocer en la red
6. dominio en el cual trabajará este servidor,

Una vez configurados los parámetros de red, se deberá configurar la contraseña de acceso para el usuario *root* o administrador del sistema, así como también configurar un usuario general que se deberá usar para las tareas no administrativas.

Configuración de paquetes

En vista que se va a dedicar todo el tamaño del disco duro para la instalación del sistema, en la configuración de las particiones del sistema (ver Ilustración 15) bien se puede escoger la opción “Guiado – utilizar todo el disco”, pero en la opción manual se pueden especificar las particiones a gusto.

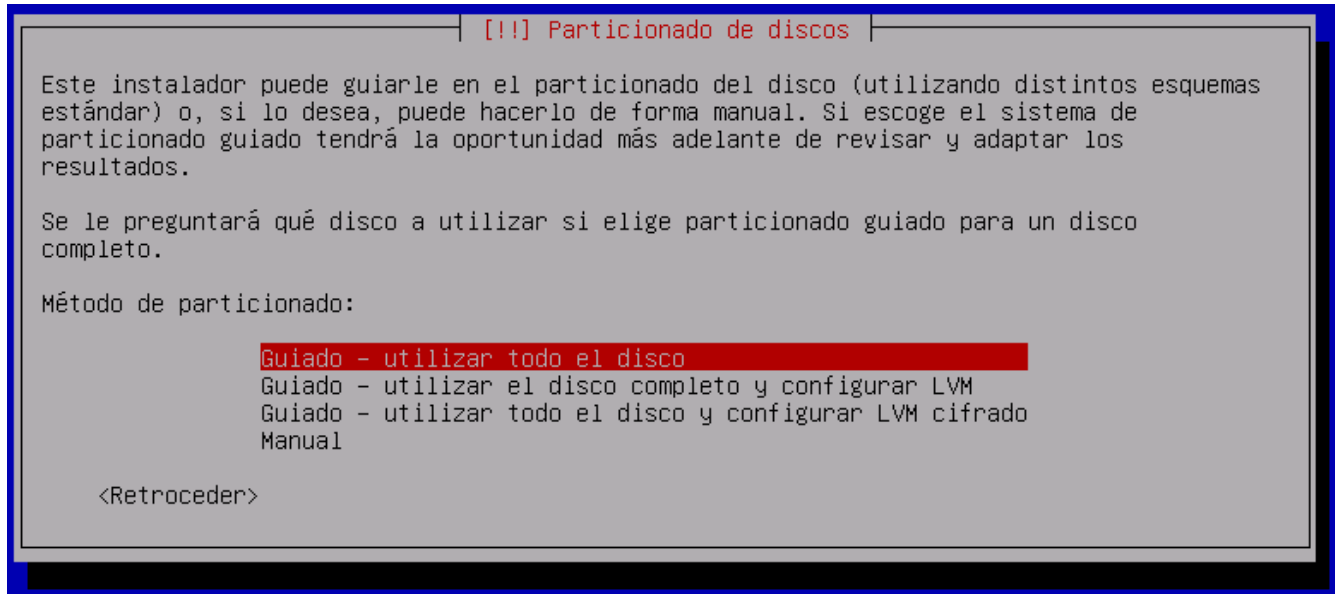


Ilustración 15: Instalación Debian, configuración de particiones

Luego se escogerá el disco duro donde se va a instalar el sistema. Esto por si en algún dado caso existiese más de un disco duro. También se puede separar la partición /home de la raíz. Configurado lo anterior, aparecerá un resumen con las opciones que se han escogido para la partición del disco duro. Se creará una partición / con formato ext3 y una partición de intercambio o la comúnmente llamada partición Swap. Cabe recordar que si se desea establecer esta configuración de forma personalizada se deberá escoger la opción “Manual” en la pantalla del método de particionado.

Configuración de paquetes

El esquema de particiones de la Ilustración 16 es solamente de ejemplo.

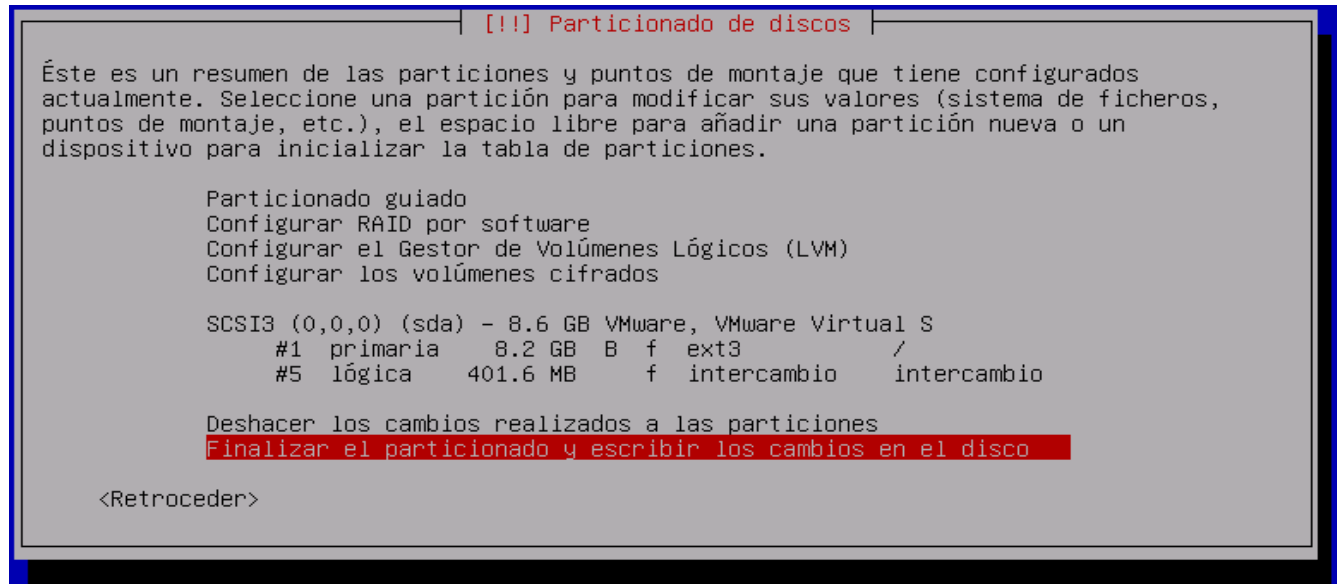


Ilustración 16: Instalación Debian, ejemplo simple de particiones

Luego se verán tres pantallas que son para leer algún otro medio de instalación, si se desea participar en una encuesta y finalmente mostrará algo parecido a la Ilustración 17 donde se deben seleccionar los paquetes a instalar en nuestro servidor.



Ilustración 17: Instalación Debian, selección de programas por defecto

Ya para finalizar, el instalador del sistema pedirá que se le indique donde ubicar el cargador de arranque GRUB a lo cual se elige que se instale en el registro principal de arranque.

3.1.2 Instalación y configuración de herramientas

Para empezar la instalación, activación y configuración del soporte IMQ en el sistema operativo, se debe primero actualizar el o los repositorios mediante el comando

```
# apt-get update
```

Se realiza la instalación de dependencias para la construcción y compilación de paquetes.

```
# aptitude install module-assistant
# m-a prepare
# aptitude install kernel-package quilt autoconf automake libtool libncurses5-dev
pkg-config checkinstall build-essential zlib1g-dev fakeroot wget bzip2 patch time
```

Se hace la desinstalación de las cabeceras del kernel que se está ejecutando

```
# aptitude purge linux-headers-2.6.32-5-amd64 linux-headers-2.6.32-5-common linux-
kbuild-2.6.32
```

Se procede a la descarga de los siguientes paquetes: kernel, iptables y parches.

```
# cd /usr/src/
# apt-get source linux-image-2.6.32-5-amd64
# wget http://www.netfilter.org/projects/iptables/files/iptables-1.4.8.tar.bz2
# wget http://www.linuximq.net/patches/iptables-1.4.10-imq.patch
# wget wget http://www.linuximq.net/patches/iptables-1.4.10-imq.patch
```

Descomprimir

```
# bzip2 -d iptables-1.4.8.tar.bz2
# tar -xvf iptables-1.4.8.tar
```

Crear links simbólicos

```
# ln -s linux-2.6-2.6.32/ linux
# ln -s iptables-1.4.4 iptables
```

Parchar el kernel para agregarle la característica de soporte IMQ

```
# cd /usr/src/linux
# patch -p1 < ../linux-2.6.32-imq-test2.diff
# cp /boot/config-2.6.32-5-amd64 .config
# perl -pi -e 's/CONFIG_VT6656=m/CONFIG_VT6656=n/g' .config
```

Desinstalar iptables para luego parchar las fuentes

```
# aptitude purge iptables iptables-dev
# cd /usr/src/iptables
# patch -p1 < ../iptables-1.4.10-imq.patch
```

Configuración de paquetes

Se deben seleccionar las opciones a utilizar en la configuración del nuevo kernel como son soporte para IMQ

```
# cd /usr/src/linux
# make menuconfig
```

Para soporte de IMQ

```
Device Drivers -> Network device support-> {*} IMQ (intermediate queueing device)
support
  IMQ behavior (PRE/POSTROUTING) (IMQ AB)
  Number of IMQ devices (4)

Networking support --->
-* Networking options --->
  [*] Network packet filtering framework (Netfilter) --->
    Networking options --->
      [*] Network packet filtering framework (Netfilter) --->
        Core Netfilter Configuration --->
          <M> "IMQ" target support
```

Una vez activadas las características de IMQ en el kernel, se procede con la compilación del mismo

```
# make-kpkg clean
# make-kpkg --append-to-version=-imq --initrd kernel-image kernel-headers kernel-
source
# dpkg -i ../linux-*.deb
```

Si se opta por dejar el kernel anterior solamente hay que configurar que el grub arranque desde el nuevo kernel que hemos instalado con soporte para IMQ.

Primero editamos el archivo `/etc/default/grub` donde se cambia la opción por defecto de 0 a 2.

```
# nano /etc/default/grub
```

```
. . .
GRUB_DEFAULT=0
. . .
```

```
. . .
GRUB_DEFAULT=2
. . .
```

Luego se ejecuta el comando

```
# update-grub
```

Finalmente se reinicia con el nuevo kernel

```
# reboot
Probar el módulo IMQ
```

```
# modprobe imq
# ip link set imq0 up
# ifconfig
```

Si todo va bien, se procede con la compilación e instalación del paquete iptables.

```
# cd /usr/src/iptables
# ./configure --with-ksource=/usr/src/linux --prefix=/usr --libdir=/lib --enable-
libipq --enable-devel
# make
# checkinstall
```

3.2 Desarrollo de la aplicación para administración de ancho de banda

3.2.1 Descripción general


La aplicación para administración del ancho de banda de la facultad de ciencias económicas de la UNAN-Managua provee la funcionalidad de configurar mediante una interfaz gráfica los parámetros necesarios para ejecutar la herramienta *tc* que está incluida en el paquete *iproute* de la mayoría de distribuciones Linux.


De forma nativa en el kernel de Linux, la herramienta *tc* no cuenta con una interfaz de configuración gráfica sino que se realiza a modo texto. En este sentido, la aplicación desarrollada muestra las opciones necesarias para su funcionamiento de forma que la configuración sea lo más sencilla y práctica posible.


Para la programación de esta aplicación se llevó a cabo un proceso iterativo de prototipado para llegar a una solución funcional teniendo como apoyo para este fin el entorno de desarrollo Netbeans bajo el sistema operativo Linux. El lenguaje de programación utilizado fue java.


3.2.2 Diagrama de clases

El diagrama de clases es bastante simple y se muestra en la Ilustración 18

 DlgMonitorBW
<i>Attributes</i>
package List areas = new ArrayList() package List IDs = new ArrayList()
<i>Operations</i>
private void btnVistaGeneralBWActionPerformed(ActionEvent evt) private void btnVistaFiltradaActionPerformed(ActionEvent evt) private void btnSalirActionPerformed(ActionEvent evt) private void llenarCombo() private void llenarListaAreas() private void llenarGridDebug(String cadena)

 Principal
<i>Attributes</i>
package String redExterna package String redInterna
<i>Operations</i>
private void mnGeneralSalirActionPerformed(ActionEvent evt) private void mnGeneralCreditosActionPerformed(ActionEvent evt) private void mnOpConfigurarTCActionPerformed(ActionEvent evt) private void mnOpEjecutarTCActionPerformed(ActionEvent evt) private void mnOpMonitorTCActionPerformed(ActionEvent evt) private void configuraRed() package String getRedExterna() package void setRedExterna(String redExterna) package String getRedInterna() package void setRedInterna(String redInterna)

 DlgEjecucionBW
<i>Attributes</i>
<i>Operations</i>
private void btnIniciarActionPerformed(ActionEvent evt) private void btnDetenerActionPerformed(ActionEvent evt) private void btnReiniciarActionPerformed(ActionEvent evt) private void btnEstadoActionPerformed(ActionEvent evt) private void btnSalirActionPerformed(ActionEvent evt)

 DlgConfiguraBW
<i>Attributes</i>
package JComboBox comboMask = new JComboBox()
<i>Operations</i>
public boolean isCellEditable(int row, int col) private void btnNuevoActionPerformed(ActionEvent evt) private void gridConfAreasMouseClicked(MouseEvent evt) private void btnAgregarActionPerformed(ActionEvent evt) private void btnEliminarActionPerformed(ActionEvent evt) private void btnCargarActionPerformed(ActionEvent evt) private void btnGuardarActionPerformed(ActionEvent evt) private void btnSalirActionPerformed(ActionEvent evt) private void btnPrimeroActionPerformed(ActionEvent evt) private void btnSubirActionPerformed(ActionEvent evt) private void btnBajarActionPerformed(ActionEvent evt) private void btnUltimoActionPerformed(ActionEvent evt) private void gridConfAreasFocusGained(FocusEvent evt) private void guardarArchivo() private void llenarArchivo() private void llenarGrid(boolean pit) private void llenarGrid() private int evaluarAreas() private int evaluarBW0(int bw) private boolean evaluarBWTotal() private void evaluarBWTotal(int r, int c) private int evaluarCeil() private void ocultarColGridYOtros() private void validaBotones() private void llenarTarjetasRed() private void llenarCmbMascRed() private void quitarEspacios() private boolean estaEnSubred(String ipaddr, String subnet, String mask) private int traducirDireccionNumerica(String ipaddr) private boolean validarIP() private boolean verificarMascara() private int validarIsDouble(int x)

 Funciones
<i>Attributes</i>
<i>Operations</i>
public String ejecutarComando(String comando) public boolean validarExisteArchivo(String ruta) public boolean verificaServicioBW()

3.2.3 Uso de la aplicación

Dado que el objetivo de la aplicación es configurar las opciones necesarias para el funcionamiento de la herramienta tc del kernel de linux junto con los algoritmos de encolado htb y sfq, las opciones con las cuales cuenta la aplicación son:

Configuración del servicio

Esta opción permite establecer los límites de ancho de banda que debe tener cada subred/usuario. A su vez, presenta las opciones de guardar la configuración en el servidor o cargar la última configuración realizada. La Ilustración 19 muestra una captura de pantalla de esta opción.

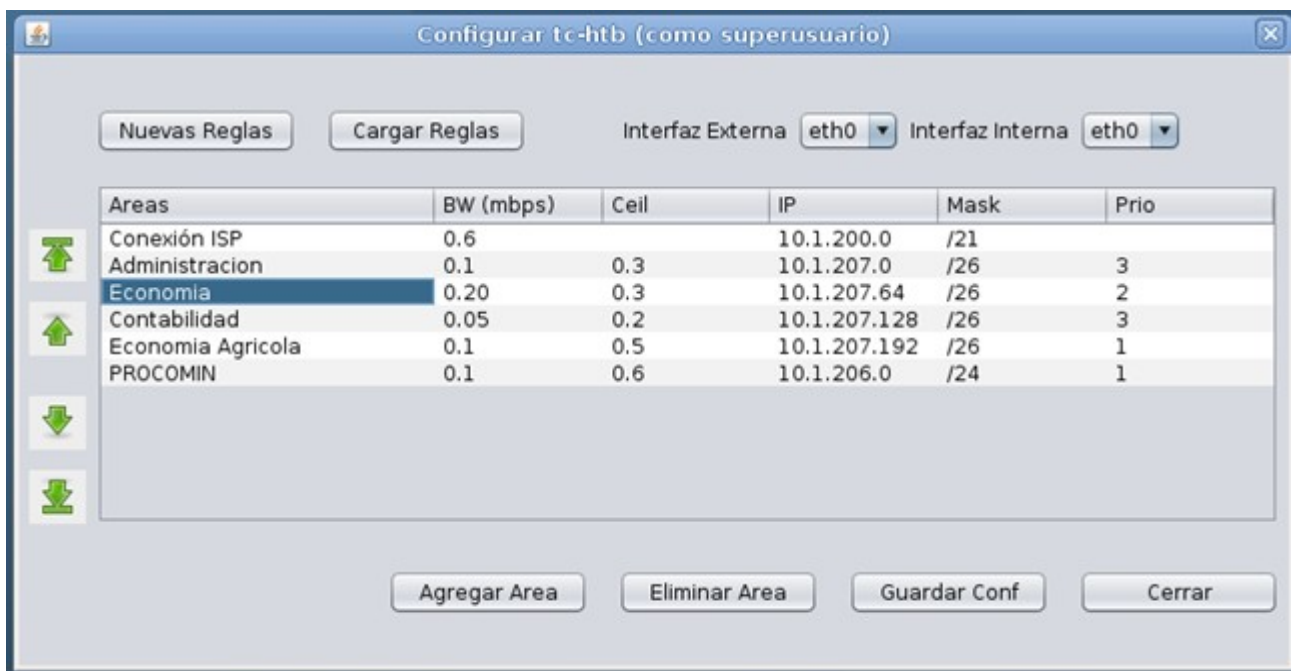


Ilustración 19: Interfaz de configuración tc-htb

Ejecutar el servicio

Básicamente permite poner en marcha el servicio de administración de ancho de banda para las distintas subredes/usuarios que existen en la red de la facultad. Se puede iniciar el servicio, detener el servicio, reiniciar el servicio o ver el estado del servicio. La Ilustración 20 muestra una captura.

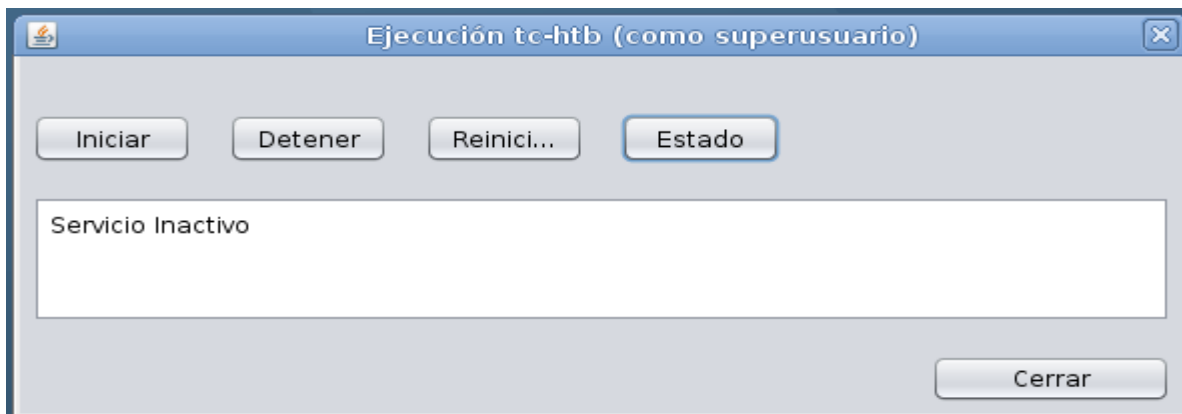


Ilustración 20: Interfaz de ejecución del servicio tc-htb

Monitor de ejecución

En esta opción se muestra información acerca de las configuraciones que tiene cada subred/usuario en tiempo de ejecución de forma que se indican parámetros como la velocidad configurada para esa subred/usuario, velocidad máxima permitida, velocidad actual, entre otros. Hay dos vistas, una general donde se muestran todas las subredes configuradas y una filtrada (ver Ilustración 21) donde se muestra información acerca de cada subred.



Ilustración 21: Interfaz de monitoreo de ejecución de tc-htb

3.2.4 Esquema de administración de ancho de banda

El esquema de prioridades para la administración del ancho de banda de la facultad de ciencias económicas de la UNAN-Managua que se pretende poner en funcionamiento se resume en la Tabla 2.

Configuración de paquetes

Área General	BW	Ceil	Prioridad	Componentes	# Usuarios
Profesores	1.25	3.3	1	Administración	14
				Contabilidad	12
				Economía	17
				Economía Agrícola	18
				Matemática	11
				PROCOMIN	8
				CINET	7
Laboratorios	0.75	3.3	2	Laboratorio A	28
				Laboratorio B	28
				Laboratorio C	39
Administrativos A	0.5	2	2	CECOM	14
				RRHH	2
				Compras	2
				Intendencia	1
				Biblioteca	3
				Contabilidad	6
				SITRA	1
				ATD	1
				Revista	1
Administrativos B	0.25	1	3	Archivo	1
				Recepción	1
				Caja	1
				Bodega	1
				UNEN	2
Secretarias	0.25	1	3	Esc. Economía Agrícola	2
				Esc. Economía	1
				Esc. Contabilidad	2
				Esc. Administración	1
				Matemática	1
				PROCOMIN	1
				Biblioteca	1

Configuración de paquetes

				CINET	1
				Admón Financiera	1
				Decanatura	1
				Vice-Decanatura	1
				Secretaría Académica	1
				CECOM	1
Red Inalámbrica	0.25	1	4	--	--
					234

Tabla 2: Esquema de prioridades para la administración del ancho de banda de UNAN-RUCFA

El campo **BW** indica el ancho de banda base para esa área

El campo **Ceil** indica el ancho de banda máximo al que puede aspirar esa área

El campo **Prioridad** indica, como la palabra lo hace suponer, la prioridad que tendrá esa red para acceder a un recurso. Esto es utilizado cuando dos áreas pretenden hacer uso del ancho de banda que no se está utilizando. Quien tenga mayor prioridad(número menor) será el primero en usar el recurso disponible.

En la Tabla 2 se aprecia un total de 234 usuarios sin meter a los usuarios de la red inalámbrica que no es posible determinar dado que no se lleva control de la cantidad de usuarios a los que se les ha permitido el acceso a la red inalámbrica.

3.3 Instalación de sistema operativo para FreeRADIUS

En primera instancia, para iniciar con la configuración y puesta en funcionamiento de un servidor de autenticación para la red inalámbrica de la UNAN-RUCFA se deberá atravesar el proceso de instalación del sistema operativo para lo cual se usará Ubuntu Server por ser una distribución bastante ligera y con opciones de instalación/configuración no tan complejas.

Ubuntu Server no posee librerías para la ejecución de entorno gráfico y ciertamente no se necesitará. Además, los entornos gráficos tienden a ralentizar la ejecución de un servidor empleando recursos que bien pueden ser destinados al servicio para el cual se ha de configurar.

El proceso de instalación de Ubuntu Server es bastante similar al proceso de instalación de Debian que se describió en el tema 3.1.1 con la salvedad de que usará una configuración de IP distinta por tratarse de la configuración de un servidor independiente.

Además, en la elección de paquetes a instalar se escogerá lo siguiente, ver Ilustración 22:

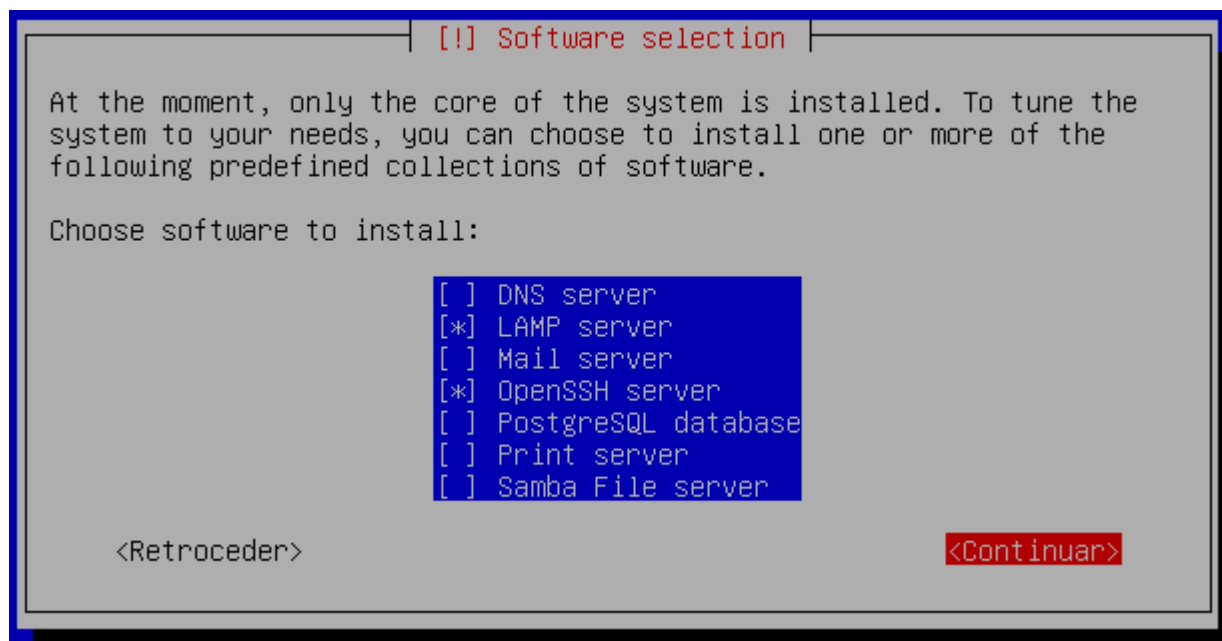


Ilustración 22: Paquetes por defecto para el servidor freeRADIUS

Esto ofrece la facilidad de tener el servidor web (LAMP – Linux Apache Mysql PHP) preconfigurado y listo para usarse

Si todo ha ido bien, se procede a la actualización de los paquetes instalados en el sistema mediante la orden:

```
# apt-get update
```

También se deberá actualizar el sistema operativo mediante la orden:

```
# apt-get upgrade
```

El propósito de estas actualizaciones es para que se lleven a cabo soluciones a algunos problemas de compatibilidad así como instalación de actualizaciones de seguridad.

3.4 Instalación de FreeRADIUS

Antes de iniciar con la configuración del servidor de autenticación, se comentará sobre el paquete a utilizar que es FreeRADIUS.

FreeRADIUS es un software de código abierto que implementa los estándares para RADIUS, pudiéndose instalar bajo cualquier plataforma (Windows, Linux, MacOS, entre otras). Inició su historia a partir de 1999 con sus creadores Alan DeKok y Miquel van Smoorenburg con el proyecto The FreeRADIUS Project.

Entre las muchas características de FreeRADIUS (solo se mencionan algunas) se encuentra que:

- Tiene soporte para muchos módulos de autenticación, entre los cuales están PAP, CHAP, MS-CHAPv1, MS-CHAPv2, etc.
- En el soporte de base de datos y servidores de directorios con FreeRADIUS pueden usarse Oracle, SQL Server, MySQL, LDAP, Unix, Active Directory, etc.
- En el apartado de soporte para lenguajes de programación se tiene a Perl, Python, Java y PHP.
- Se pueden tener los registros de eventos que se llevan a cabo en las fases de Autenticación, Autorización y Arqueo (o Contabilidad).

Para comenzar la configuración de freeRADIUS, se deberá primero instalar algunos paquetes adicionales a php necesarios para la ejecución de la interfaz web de administración del servicio.

```
# aptitude install php5 php-pear php-gd php-db php-pear php5-gd php5-odbc
```

Luego se hará la instalación de los paquetes necesarios para el funcionamiento de freeRADIUS:

```
# aptitude install freeradius freeradius-common freeradius-utils libfreeradius2  
freeradius-mysql
```

FreeRADIUS se ejecuta en modo demonio de la siguiente forma:

```
root@radius:~# /etc/init.d/freeradius start  
root@radius:~# /etc/init.d/freeradius stop
```

Sin embargo, no trae configurado el parámetro *status* para ver el estado del demonio respecto a que si está en ejecución o no, aunque eso también se pudiese verificar viendo los puertos abiertos en el servidor ya que FreeRADIUS brinda su servicio en los puertos 1812, 1813 y 1814. O bien, se puede ver el estado del servicio en la interfaz web de administración que más adelante se comentará.

Para configurar el parámetro *status* del demonio de FreeRADIUS, vamos a modificar el código fuente de dicho demonio de tal manera que se le agregará el siguiente código que aparece en negrita:

```
log_end_msg 0
;;

status)
echo -n "Checking for service freeradius\n"
if [ ! -r "$PIDFILE" ]; then
    echo "$NAME is not running."
    exit 1
fi
if read pid < "$PIDFILE" && ps -p "$pid" > /dev/null 2>&1; then
    echo "$NAME is running."
    exit 1
else
    echo "$NAME is not running but $PIDFILE exists."
    exit 1
fi
;;

*)
echo "Usage: $0 start|stop|restart|reload|force-reload|status"
exit 1
;;
esac

exit 0
```

Para la ejecución del servicio en su fase de pruebas de configuración se hace en modo debug para ver posibles errores. Si ocurrese algún error en la ejecución, en la salida del comando mostrará en la última línea el archivo con el posible error de configuración.

```
root@radius:~# freeradius -X
```

3.5 Guía de configuración de OpenSSL

Antes de continuar con la configuración de freeRADIUS, se procederá a la creación de un certificado de servidor mediante openssl para agregarle seguridad al servidor. OpenSSL ofrece excelentes opciones para la implantación de una infraestructura PKI en Linux. OpenSSL provee funciones que se necesitan para el uso de SSL y TLS.

Para iniciar con la configuración de OpenSSL, se deberá editar el archivo de configuración *openssl.cnf* (se muestran partes del archivo) en el cual se muestran sombreadas las modificaciones que habrá que hacer:


```

root@radius:~# vim /etc/ssl/openssl.cnf
...[código cortado]...
#####
[ CA_default ]

dir          = ./masterCA          # Where everything is kept
certs       = $dir/certs           # Where the issued certs are kept
crl_dir     = $dir/crl             # Where the issued crl are kept
database    = $dir/index.txt       # database index file.

...[código cortado]...

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crlnumber must also be commented out to leave a V1 CRL.
# crl_extensions = crl_ext

default_days = 730                 # how long to certify for
default_crl_days= 30               # how long before next CRL
default_md   = sha1                 # which md to use.

...[código cortado]...

# so use this option with caution!
string_mask = nombstr

# req_extensions = v3_req # The extensions to add to a certificate request

[ req_distinguished_name ]
countryName          = Country Name (2 letter code)
countryName_default  = NI
countryName_min      = 2
countryName_max      = 2

stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Managua

localityName         = Locality Name (eg, city)

0.organizationName   = Organization Name (eg, company)
0.organizationName_default = UNAN-RUCFA

# we can do this but it is not needed normally :-)
#1.organizationName  = Second Organization Name (eg, company)
#1.organizationName_default = World Wide Web Pty Ltd

organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = CECOM

commonName           = Common Name (eg, YOUR name)
commonName_max       = 64

...[código cortado]...

```

Configuración de paquetes

Lo siguiente será ubicarnos en la carpeta donde estarán los certificados a crear. En dicha carpeta se encuentran certificados y archivos que se crearon con la instalación de openssl, deberemos borrarlos para crear los que se usarán de forma personalizada en este servidor.

```
root@radius:# cd /etc/freeradius/certs
root@radius:/etc/freeradius/certs# rm 01.pem ca* index* serial* server.crt
server.csr server.key server.p12 server.pem dh random
```

Para la creación de certificados se dejará el script que viene con la instalación de openssl, el cual es **bootstrap** que sirve para crear certificados de forma automática. También los archivos **Makefile**, **xpextensions**, **server.cnf**, **client.cnf**, **ca.cnf**.

Para la creación de nuestros certificados hay que copiar el script **CA.sh** que incluye la instalación de openssl para la creación de certificados de forma manual.

```
root@radius:/etc/freeradius/certs# cp /usr/lib/ssl/misc/CA.sh ./
```

Habrá que hacer dos modificaciones al script **CA.sh**, las cuales se muestran sombreadas a continuación :

```
root@radius:/etc/freeradius/certs# vim /etc/freeradius/certs/CA.sh

...[código cortado]...

DAYS="-days 730" # 2 year
CADAYS="-days 1095" # 3 years
REQ="$OPENSSL req $SSLEAY_CONFIG"
CA="$OPENSSL ca $SSLEAY_CONFIG"
VERIFY="$OPENSSL verify"
X509="$OPENSSL x509"

CATOP=./masterCA
CAKEY=./cakey.pem
CAREQ=./careq.pem
CACERT=./cacert.pem
```

El primer paso para trabajar con certificados en un sistema PKI es el de crear la Autoridad Certificadora raíz, para lo cual creamos el certificado raíz con el siguiente comando:

(Solo se muestran los campos en que debemos intervenir)

```
root@radius:/etc/freeradius/certs# ./CA.sh -newca

...[código cortado]...

Enter PEM pass phrase: clavePKI2012
Verifying - Enter PEM pass phrase: clavePKI2012

...[código cortado]...
```

```
Locality Name (eg, city) []: Oriental
    ...[código cortado]...
Common Name (eg, YOUR name) []: JMAG
E-mail Address []: jmag@red.local
    ...[código cortado]...
A challenge password []: clavePKI2012
    ...[código cortado]...
Enter pass phrase for ./masterCA/private/./cackey.pem: clavePKI2012
```

Los archivos generados que contienen los certificados se pueden ver con la siguiente estructura mediante el comando tree

```
root@radius:/etc/freeradius/certs# tree
masterCA/
|-- cacert.pem
|-- careq.pem
|-- certs
|-- crl
|-- index.txt
|-- index.txt.attr
|-- index.txt.old
|-- newcerts
`-- 00.pem
-- private
   |-- cakey.pem
   |-- serial
   `-- serial.old
```

Ahora se le deberá solicitar a la nueva Autoridad Certificadora un nuevo certificado para esta máquina con el servicio FreeRADIUS.

Primero, se genera una solicitud de certificado

```
root@radius:/etc/freeradius/certs# ./CA.sh -newreq
    ...[código cortado]...
Enter PEM pass phrase: clavePKI2012
Verifying - Enter PEM pass phrase: clavePKI2012
    ...[código cortado]...
```

Configuración de paquetes

```
Locality Name (eg, city) []:Oriental
    ...[código cortado]...
Common Name (eg, YOUR name) []:radius.red.local
E-mail Address []:soporte@red.local
    ...[código cortado]...
A challenge password []:clavePKI2012
    ...[código cortado]...
```

Ahora se debe firmar esta solicitud de certificado para el servidor radius.

```
root@radius:/etc/freeradius/certs# ./CA.sh -sign
    ...[código cortado]...
Enter pass phrase for ./masterCA/private/cackey.pem: clavePKI2012
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]:y
```

El directorio ha quedado de la siguiente forma:

```
root@radius:/etc/freeradius/certs# tree
|-- CA.sh
|-- Makefile
|-- README
|-- bootstrap
|-- ca.cnf
|-- client.cnf
|-- masterCA
|   |-- cacert.pem
|   |-- careq.pem
|   |-- certs
|   |-- crl
|   |-- index.txt
|   |-- index.txt.attr
|   |-- index.txt.attr.old
|   |-- index.txt.old
|   |-- newcerts
|   |   |-- 00.pem
|   |   |-- 01.pem
|   |-- private
|   |   |-- cakey.pem
|   |-- serial
|   |-- serial.old
|-- newcert.pem
|-- newkey.pem
|-- newreq.pem
|-- server.cnf
|-- xpextensions
```

Configuración de paquetes

Dado que el formato ha quedado en PEM y por si acaso algún sistema Windows necesita del certificado raíz en forma DER, habrá que convertirlo de la siguiente forma:

```
root@radius:/etc/freeradius/certs# cd masterCA/
root@radius:/etc/freeradius/certs/masterCA# openssl x509 -inform PEM -outform DER
-in cacert.pem -out cacert.der
```

Crear la carpeta donde estarán los archivos para el servidor radius y copiar el archivo DER que se acaba de generar

```
root@radius:/etc/freeradius/certs/masterCA# mkdir /etc/certs
root@radius:/etc/freeradius/certs/masterCA# cp cacert.der /etc/certs/
```

Se deben generar los archivos dh (Diffie-Hellman) y random que se utilizan para establecer sesiones TLS y encriptación

```
root@radius:/etc/freeradius/certs/masterCA# cd /etc/freeradius/certs/
root@radius:/etc/freeradius/certs# dd if=/dev/urandom of=random count=2
root@radius:/etc/freeradius/certs# openssl dhparam -out dh 1024
```

Para tener un control centralizado del certificado con la clave pública y privada, se puede crear en un solo fichero el certificado con sus claves privada y pública de la siguiente manera.

```
root@radius:/etc/freeradius/certs# cat newkey.pem newcert.pem > radius_keycert.pem
```

Copiar los archivos necesarios para el servidor radius a una ubicación controlada

```
root@radius:/etc/freeradius/certs# mkdir -p /etc/certs/server
root@radius:/etc/freeradius/certs# mkdir -p /etc/certs/clients
root@radius:/etc/freeradius/certs# cp radius_keycert.pem /etc/certs/server/
root@radius:/etc/freeradius/certs# cp dh random /etc/certs/
root@radius:/etc/freeradius/certs# cp masterCA/cacert.pem /etc/certs/
```

En vista que ya tenemos una ubicación controlada/centralizada para los archivos de certificados del servidor radius, se pueden eliminar los archivos creados originalmente

```
root@radius:/etc/freeradius/certs# rm new* radius_keycert.pem
```

La autenticación para clientes Windows se hará mediante el protocolo 802.1X y para esto hay que generar una solicitud de certificado para luego hacer la firma del mismo. Para asegurarnos de la compatibilidad del certificado que se va a generar con Windows, hay que verificar en el archivo `xpextensions` lo siguiente.

```
root@radius:/etc/freeradius/certs# vim /etc/freeradius/certs/xpextensions
[ xpclient_ext]
extendedKeyUsage = 1.3.6.1.5.5.7.3.2
```

Configuración de paquetes

```
[ xpservers_ext]
extendedKeyUsage = 1.3.6.1.5.5.7.3.1
```

Luego de verificar lo anterior, hay que generar la solicitud de certificado de cliente:

```
root@radius:/etc/freeradius/certs# openssl req -new -keyout newreq.pem -out
newreq.pem -passin pass:clavePKI2012 -passout pass:clavePKI2012

...[código cortado]...
Locality Name: Oriental
Common Name: Cliente Windows
E-mail Address: mailpcwin@red.local

...[código cortado]...
A challenge password: clavePKI2012
```

Ahora, se debe firmar y generar el certificado con las extensiones para soporte de Windows que están indicadas en el archivo `xpextensions`. Se debe responder con "y" a las dos preguntas referente a si deseamos firmar el certificado.

```
root@radius:/etc/freeradius/certs# openssl ca -policy policy_anything -out
newcert.pem -passin pass:clavePKI2012 -key clavePKI2012 -extensions xpclient_ext
-extfile xpextensions -infile newreq.pem

...[código cortado]...
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n] y

...[código cortado]...
```

Luego habrá que generar un certificado de cliente con formato PKCS#12 que lleve incluida la clave privada

```
root@radius:/etc/freeradius/certs# openssl pkcs12 -export -in newcert.pem -inkey
newreq.pem -out clientcert.p12 -clcerts -passin pass:clavePKI2012 -passout
pass:clavePKI2012

root@radius:/etc/freeradius/certs# openssl pkcs12 -in clientcert.p12 -out
clientcert.pem -passin pass:clavePKI2012 -passout pass:clavePKI2012
MAC verified OK
```

El certificado que se ha generado ahora se exporta a formato DER

```
root@radius:/etc/freeradius/certs# openssl x509 -inform PEM -outform DER -in
clientcert.pem -out clientcert.der
```

Para finalizar, se debe copiar el archivo con el certificado a la carpeta donde se van almacenando los

certificados generados a fin de que luego se traslade al equipo Windows que se va a autenticar mediante FreeRADIUS.

```
root@radius:/etc/freeradius/certs# cp clientcert.* /etc/certs/clients/
```

Solamente por mencionar y verificar, se pueden ver los tres certificados que se han emitido a través del siguiente archivo:

```
root@radius:/etc/freeradius/certs# cat masterCA/index.txt
V 150707184553Z 00 unknown /C=NI/ST=Managua/O=UNAN-
RUCFA/OU=CECOM/CN=JMAG/emailAddress=jmag@red.local
V 140707185833Z 01 unknown /C=NI/ST=Managua/L=Oriental/O=UNAN-
RUCFA/OU=CECOM/CN=radius.red.local/emailAddress=soporte@red.local
V 140707214914Z 02 unknown /C=NI/ST=Managua/L=Oriental/O=UNAN-
RUCFA/OU=CECOM/CN=Cliente Windows/emailAddress=mailpcwin@red.local
```

Los principales archivos vistos hasta el momento son:

dh: Archivo de parametros del algoritmo Diffie-Hellman.

random: Archivo de contenido aleatorio para sesiones TLS o SSL.

xpextensions: Archivo de extensiones OID de servidor y cliente para certificados Windows.

cacert.pem: Certificado raíz de la Autoridad Certificadora en codificación PEM.

cacert.der: Certificado raíz de la Autoridad Certificadora en codificación DER.

cakey.pem: Clave privada del certificado de la Autoridad Certificadora.

newreq.pem: Solicitud para nuevo certificado.

newkey.pem: Clave privada del nuevo certificado.

newcert.pem: Nuevo certificado firmado a partir de la solicitud.

keycert.pem: Certificado y clave privada concatenados para mayor comodidad.

3.6 Configuración de los archivos de freeRADIUS.

En este apartado, los archivos a tener en cuenta son: /etc/freeradius/sql.conf, /etc/freeradius/eap.conf, /etc/freeradius/proxy.conf, /etc/freeradius/radiusd.conf y /etc/freeradius/sites-enabled/default

3.6.1 radiusd.conf

Las configuraciones generales del servicio se deben especificar en el archivo radiusd.conf. Este archivo de configuración es el de mayor importancia dado que desde aquí se indica la lectura de los demás archivos de configuración como módulos o servidores virtuales. En general no hay mucho que modificar más que indicar que no se va a leer el archivo de los clientes puesto que esta tarea se hará desde la base de datos.

```
root@radius:/etc/freeradius# vim radiusd.conf
```

```
...[código cortado]...

prefix = /usr
exec_prefix = /usr
sysconfdir = /etc
localstatedir = /var
sbindir = ${exec_prefix}/sbin
logdir = /var/log/freeradius
raddbdir = /etc/freeradius
radacctdir = ${logdir}/radacct

#INCLUDE clients.conf

...[código cortado]...
```

3.6.2 eap.conf

Dado que en el capítulo referente a OpenSSL se crearon los certificados, ahora se le deberá indicar a FreeRADIUS la ruta de donde se encuentran para su utilización en la autenticación con EAP-TLS, EAP-TTLS y EAP-PEAP.

En el archivo **eap.conf** se establecen las rutas exactas hacia los archivos cacert.pem (clave pública de la Autoridad Certificadora), radius_keycert.pem (certificado de servidor con clave pública y clave privada), dh (Diffie-Hellman, para el tunelamiento) y random (usado para el cifrado). Además, se especifica la contraseña de cifrado de la clave privada para que FreeRADIUS compruebe los certificados. También se indica si los métodos EAP a utilizar serán con tunel o sin tunel.

A continuación se muestran algunos de los campos que se deberán revisar

```
root@radius:/etc/freeradius# vim eap.conf

...[código cortado]...

    tls {
        certdir = /etc/certs/server
        cadir = ${confdir}/certs/masterCA

        private_key_password = clavePKI2012
        private_key_file = /etc/certs/server/radius_keycert.pem
        certificate_file = /etc/certs/server/radius_keycert.pem
        CA_file = ${cadir}/cacert.pem

        dh_file = /etc/certs/dh
        random_file = /etc/certs/random
        rsa_key_length = 1024
        dh_key_length = 1024
```



```
...[código cortado]...

    ttls {
        default_eap_type = mschapv2
        copy_request_to_tunnel = yes
        use_tunneled_reply = yes
    }

    peap {
        default_eap_type = mschapv2
        copy_request_to_tunnel = yes
        use_tunneled_reply = yes
    }

...[código cortado]...
```

3.6.3 sites-available/default

Este archivo de configuración es el correspondiente al servidor virtual por defecto que atenderá las solicitudes que lleguen al servidor RADIUS. Esta característica de servidor virtual es importante por que se pueden crear más de un servidor virtual que pueden tener configuraciones distintas para un mismo servicio, así por ejemplo se puede tener un servidor virtual que atienda las solicitudes en puertos y direcciones IP diferentes a los establecidos por FreeRADIUS con políticas de configuración distintas para su proceso AAA.

Para el caso que aborda el presente trabajo, se modificará el archivo por defecto(default) para la configuración de servidor virtual de forma que el servicio FreeRADIUS haga uso de consultas a la base de datos para su proceso AAA. Para esto se deben activar los campos marcados con la etiqueta “sql”, de forma que si aparecen comentados habrá que descomentarlos. También, dado que se hará uso de la característica de autenticación mediante EAP, habrá que revisar las secciones authorize, authenticate y post-proxy para que los campos “eap” aparezcan descomentados(sin #), por defecto están activas.

```
root@radius:/etc/freeradius# vim sites-enabled/default

authorize {

    ...[código cortado]...

    sql

    ...[código cortado]...
```

3.6.4 sql.conf

El servicio freeRADIUS puede hacer uso de base de datos para la autenticación de usuarios y el reconocimiento de clientes. Para esto, hay que editar el archivo *sql.conf* de manera que se especifique el motor de base de datos, driver, servidor, login entre otras cosas. La mayoría de opciones que vienen por defecto son las que se van utilizar y posiblemente solo se modifique la contraseña para la base de datos de “radius” así como también verificar que el campo “readclients” esté en “yes” puesto que los clientes serán leídos desde la base de datos y no desde el archivo de configuración. Además, verificar que el campo *nas_table* está con valor “nas” ya que es en esa tabla de la base de datos donde se almacenarán los clientes NAS.

```
root@radius:/etc/freeradius# vim /etc/freeradius/sql.conf

...[código cortado]...

server = "localhost"
#port = 3306
login = "radius"
password = "password"
radius_db = "radius"
readclients = yes
nas_table = "nas"

...[código cortado]...
```

3.6.5 proxy.conf

Una de las funciones más importantes -y para este trabajo la de mayor importancia- que debe presentar freeRADIUS es la de funcionar como servidor proxy para las peticiones de autenticación de usuarios que no estén activos en su base de datos. Para ello, el archivo *proxy.conf* se encarga de definir cuales son los servidores proxy que se van a utilizar y como se procesarán las solicitudes.

```
root@radius:/etc/freeradius# vim proxy.conf

home_server localhost {
    type = auth+acct
    ipaddr = 127.0.0.1
    port = 1812
    secret = secret123
}

realm NULL {
}

realm LOCAL {
}

realm red1.local {
}
```

```
realm red2.local {
    type = radius
    authhost = 10.1.200.95
    accthost = 10.1.200.95
    secret = secret123
    nostrip
}
```

3.7 Servidor Web - Apache

Dado que se va a agregar un poco de seguridad al servidor web de este equipo, una forma para ello es implementar el uso de SSL para la conexión al servidor web por parte de los clientes de modo que no se puedan conectar al puerto 80(HTTP) sino al 443(HTTPS). Para ello, hay que deshabilitar el acceso web al puerto 80 y dejar solamente el 443.

```
root@radius:~# vim /etc/apache2/ports.conf
<IfModule mod_ssl.c>
    Listen 443
</IfModule>
```

Luego, se debe habilitar el módulo SSL para Apache(nuestro servidor web). Para esto, basta con hacer un enlace simbólico al módulo

```
root@radius:~# ln -s /etc/apache2/mods-available/ssl.* /etc/apache2/mods-enabled/
```

Dado que Apache trabaja con servidores virtuales, se debe editar el archivo relacionado al servidor virtual por defecto y agregarle el soporte SSL. El archivo en cuestión quedará de la siguiente forma:

```
root@radius:~# vim /etc/apache2/sites-enabled/000-default
```

```
NameVirtualHost *:443
<VirtualHost *:443>
    ServerAdmin webmaster@red.local

    ServerName radius.red.local
    ServerAlias radius.red.local www.red.local
    SSLEngine On
    SSLOptions +FakeBasicAuth +ExportCertData
    SSLCertificateFile /etc/certs/server/apache_cert.pem
    SSLCertificateKeyFile /etc/certs/server/apache_key_insecure.pem

    DocumentRoot /var/www/
    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>
```

```
<Directory /var/www/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
</Directory>

ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
<Directory "/usr/lib/cgi-bin">
    AllowOverride None
    Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
    Order allow,deny
    Allow from all
</Directory>

ErrorLog /var/log/apache2/error.log

# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
LogLevel warn

CustomLog /var/log/apache2/access.log combined
ServerSignature On

Alias /doc/ "/usr/share/doc/"
<Directory "/usr/share/doc/">
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
    Allow from 127.0.0.0/255.0.0.0 ::1/128
</Directory>
</VirtualHost>
```

Para las comunicaciones SSL, se debe crear un certificado para el servidor Apache de la forma a como se ha hecho anteriormente, generar una solicitud de certificado y luego firmarla.

```
root@radius:~# cd /etc/freeradius/certs/
root@radius:/etc/freeradius/certs# ./CA.sh -newreq

...[código cortado]...

Enter PEM pass phrase: clavePKI2012
Verifying - Enter PEM pass phrase: clavePKI2012

...[código cortado]...

Locality Name (eg, city) []: Oriental

...[código cortado]...
```

Configuración de paquetes

```
Common Name (eg, YOUR name) []:www.red.local
E-mail Address []:webmaster@red.local

...[código cortado]...

A challenge password []:clavePKI2012
```

```
root@radius:/etc/freeradius/certs# ./CA.sh -sign

...[código cortado]...

Enter pass phrase for ./masterCA/private/cackey.pem:clavePKI2012

...[código cortado]...

Sign the certificate? [y/n]: y

1 out of 1 certificate request certified, commit? [y/n]y
```

Generados los archivos de certificado con la clave pública(newcert.pem) y con la clave privada(newkey.pem), se deben trasladar a la ubicación definida de nuestro depósito de certificados.

```
root@radius:/etc/freeradius/certs# cp newcert.pem /etc/certs/server/apache_cert.pem
root@radius:/etc/freeradius/certs# cp newkey.pem /etc/certs/server/apache_key.pem
```

Luego, eliminar la solicitud de la contraseña para la clave privada que se daría en cada inicio de Apache, esto para no estar digitando la contraseña con cada reinicio de Apache.

```
root@radius:/etc/freeradius/certs# cd /etc/certs/server/
root@radius:/etc/certs/server# openssl rsa -in apache_key.pem -out
apache_key_insecure.pem
```

Y finalmente solo reiniciar el servicio web

```
root@radius:/etc/certs/server# /etc/init.d/apache2 restart
```

Se debe notar que ya no se escucha sobre el puerto 80, sino sobre el puerto 443

```
root@radius:/etc/certs/server# netstat -putan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program
name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*                LISTEN      4820/mysqld
tcp        0      0 0.0.0.0:22             0.0.0.0:*                LISTEN      4722/sshd
tcp        0      0 0.0.0.0:443            0.0.0.0:*                LISTEN      5335/apache2
tcp6       0      0 :::22                  :::*                    LISTEN      4722/sshd
```

3.8 daloRADIUS y MySQL

Para facilitar la gestión de usuarios freeRADIUS, se usará la herramienta daloRADIUS que está escrita en lenguaje PHP y sirve para administrar FreeRADIUS vía web con motor de base de datos MySQL. Esta herramienta es propiedad intelectual de Liran Tal quien la desarrolló en el 2007. Para el uso en el desarrollo del presente proyecto de tesis se ha modificado el original de forma que se adapte a lo que se necesita y además se terminó de traducir al español.

El paquete daloRADIUS tiene como requisito los siguientes paquetes: Apache 1/2, PHP 4/5, PHP GD, PHP DB, PHP Pear, PHP Mail, MySQL 4/5

En la instalación y configuración de daloRADIUS se siguen los siguientes pasos.

1. Descargar y extraer el paquete

```
# wget
http://sourceforge.net/projects/daloradius/files/daloradius/daloradius0.9- \
9/daloradius-0.9-9.tar.gz
# tar xzvf daloradius-0.9-9.tar.gz
# mv daloradius-0.9-9 /var/www/daloradius
```

2. Configurar los permisos

```
chown www-data:www-data /var/www/daloradius -R
chmod 644 /var/www/daloradius/library/daloradius.conf.php
```

3. Instalar la base de datos tanto del paquete daloRADIUS como para freeRADIUS

Para esto, primero se debe crear la base de datos radius

```
# mysqladmin create radius
```

Luego, las tablas tanto de freeRADIUS como de daloRADIUS

```
# mysql -u root -p radius < fr2-mysql-daloradius-and-freeradius.sql
```

4. Establecer parámetros básicos en la configuración de daloRADIUS

Nada más será revisar que estén bien configurados los parámetros correspondientes al motor de base de datos, puerto, host, usuario y nombre de la base de datos

```
# cd /var/www/daloradius/library
# vim daloradius.conf.php
```

5. Visualizar la interfaz

Para visualizar la interfaz bastará con escribir en el navegador

`https://direccionIPdelServidor/daloradius`

La cuenta de administración inicial es:

usuario: **administrator**

contraseña: **radius**

3.9 Test de funcionamiento del servicio

Para la prueba de funcionamiento, se hará con un cliente inalámbrico para lo cual se debe crear un nuevo cliente(NAS) en daloRADIUS que en este caso corresponderá a un router inalámbrico el cual tiene la dirección IP 192.168.1.220. La configuración para que los clientes se autenticuen a través del servidor FreeRADIUS es a como se muestra en la Ilustración 23:

The screenshot displays the configuration page for 'Wireless Security w10'. At the top, there are navigation tabs: 'Basic Settings', 'Radius', 'Wireless Security', 'MAC Filter', and 'Advanced Settings'. Below the tabs, the title 'Wireless Security w10' is shown. The main configuration area is titled 'Physical Interface w10 SSID [testW] HWAddr [00:1D:7E:C5:F0:61]'. It contains several fields: 'Security Mode' is set to 'WPA2 Enterprise'; 'WPA Algorithms' is set to 'TKIP+AES'; 'Radius Auth Server Address' is '192.168.1.222'; 'Radius Auth Server Port' is '1812' with '(Default: 1812)' noted; 'Radius Auth Shared Secret' is 'ddwrt123' with an 'Unmask' checkbox checked; and 'Key Renewal Interval (in seconds)' is '600'. At the bottom of the configuration area, there are two buttons: 'Save' and 'Apply Settings'.

Ilustración 23: Configuración de RADIUS para cliente NAS

Configuración de paquetes

En una portátil, la configuración empieza por detectar y agregar el SSID que está emitiendo el router. Ver Ilustración 24

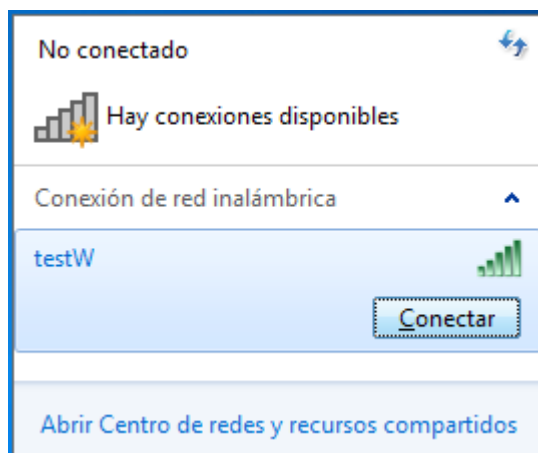


Ilustración 24: Detección de SSID

Luego, en la interfaz de administración de redes inalámbricas se deben configurar los parámetros necesarios para este punto de acceso. Empezando por el tipo de seguridad en el cual se elije “WPA2-Enterprise” y Tipo de cifrado en “AES”. En la Ilustración 25 se muestran estos parámetros.

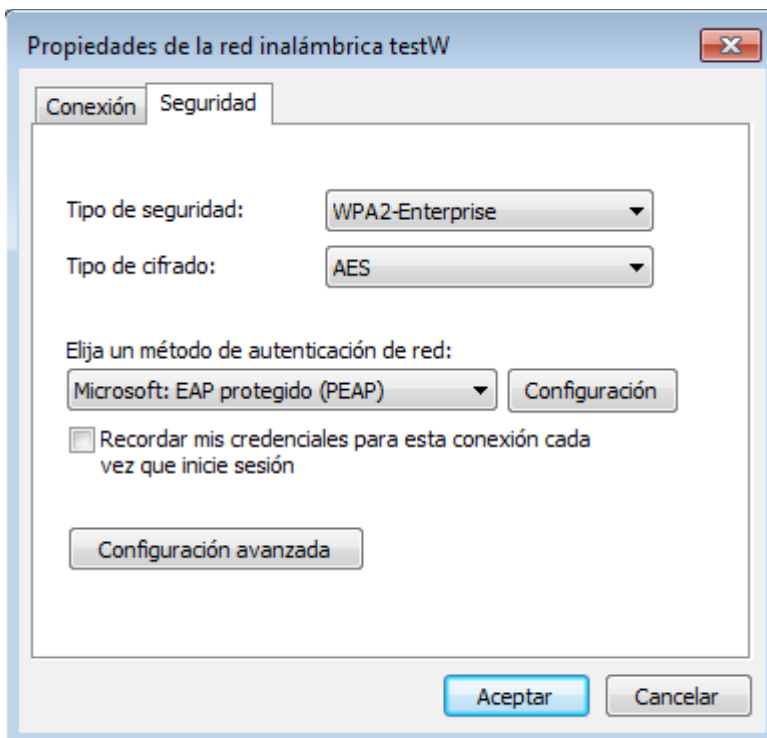


Ilustración 25: Parámetros de configuración para red inalámbrica

Configuración de paquetes

Se da click en el botón Configuración del método de autenticación EAP-PEAP y dejamos los valores tal cual aparecen en la Ilustración 26.

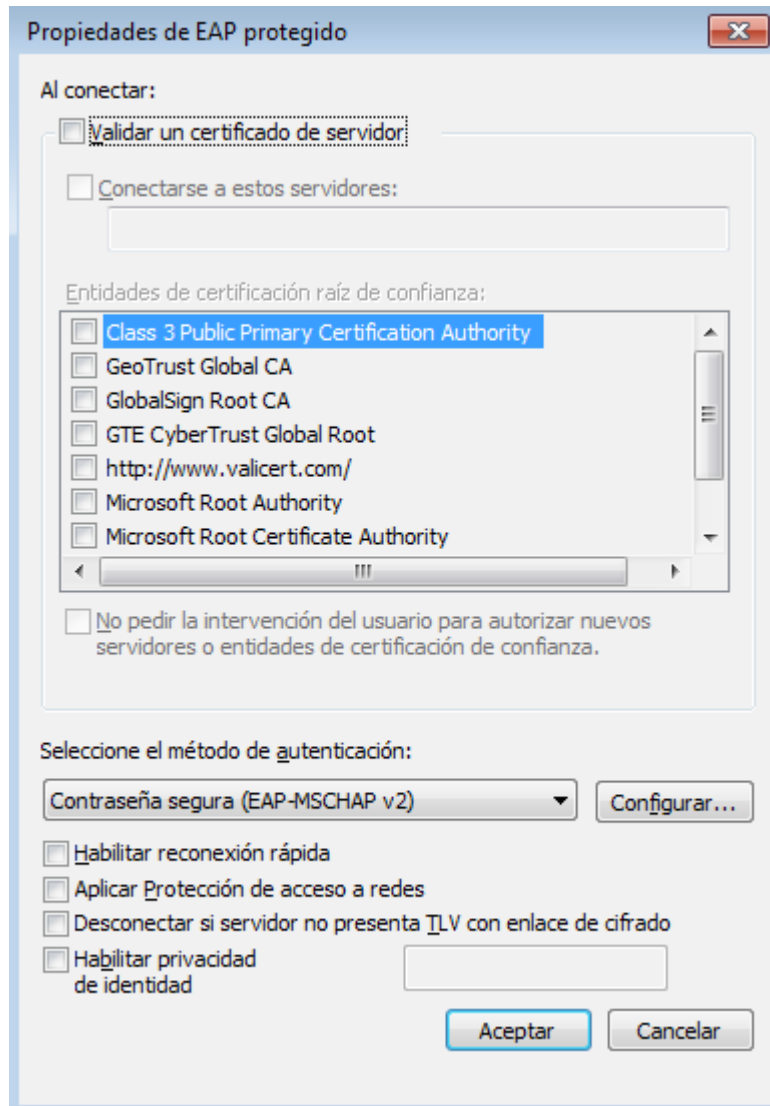


Ilustración 26: Propiedades EAP-PEAP

Configuración de paquetes

Luego, click en Configurar del método de autenticación EAP-MSCHAPv2 y dejamos vacío el check box de las propiedades de EAP-MSCHAPv2 para que no use las credenciales de Windows. Ver Ilustración 27.

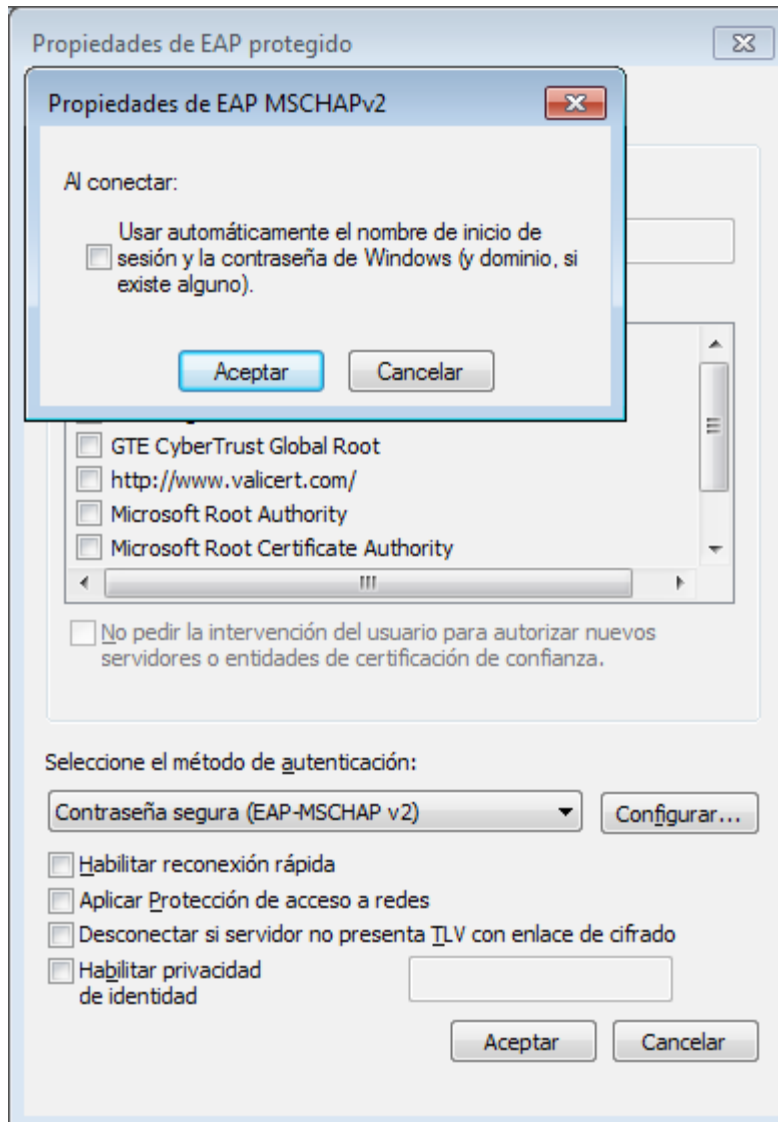


Ilustración 27: Propiedades EAP-MSCHAPv2

Configuración de paquetes

De regreso en la pestaña Seguridad, click en el botón Configuración Avanzada. Aparecerá la ventana de configuración de 802.1X a como se muestra en la Ilustración 28 y se escoge la opción Autenticación de Usuarios.

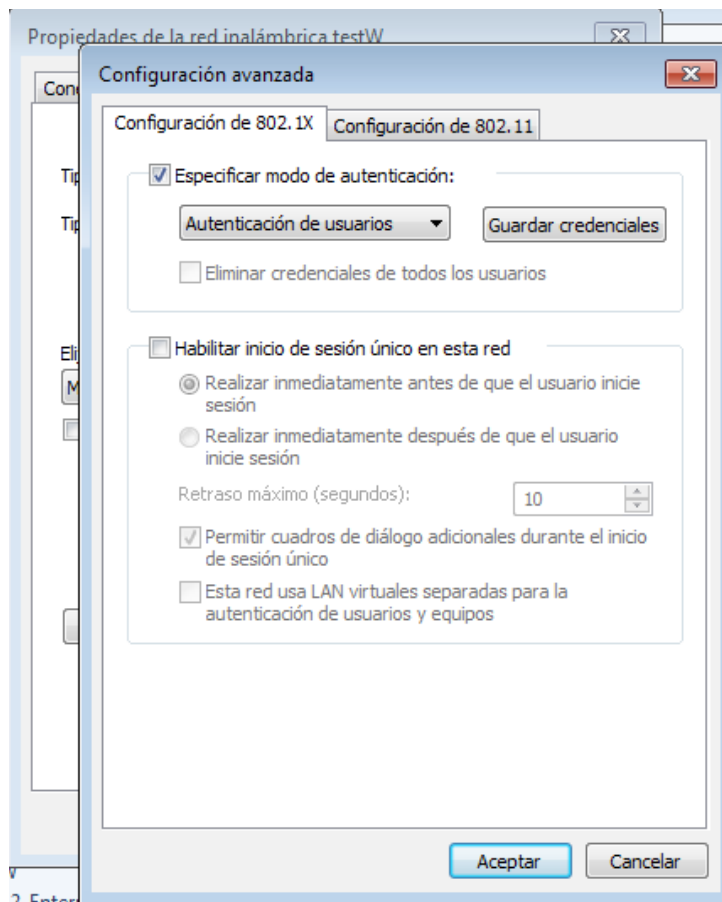


Ilustración 28: Configuración 802.1X

Finalizado lo anterior, se procede a establecer la conexión y se mostrará la ventana de solicitud de credenciales de usuario a como se muestra en la Ilustración 29

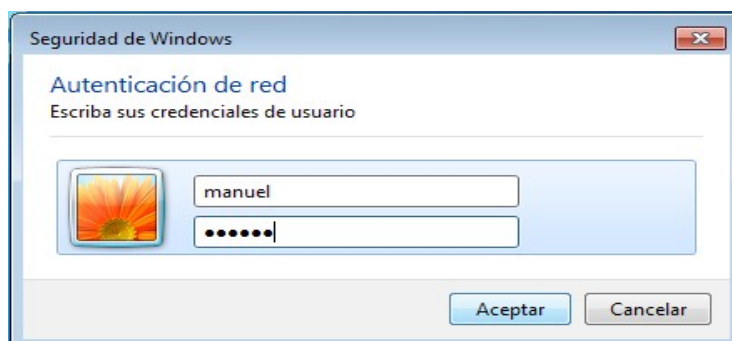


Ilustración 29: Solicitud de credenciales de usuario

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

Una vez que se ha finalizado el proceso descrito en los anteriores apartados, se puede concluir lo siguiente:

- Es posible tener una red segmentada de acuerdo a un esquema de prioridades en cuanto al uso del ancho de banda y esto se logra mediante el uso de herramientas libres sin necesidad de incurrir en altos gastos económicos.
- La interfaz de configuración que se desarrolló, facilita en gran medida la configuración de un esquema de prioridades para uso del ancho de banda. Esta interfaz es de mucha utilidad puesto que el kernel de Linux no trae consigo una herramienta gráfica de configuración.
- Dado que la seguridad es un elemento muy importante para una red, mediante la implementación del protocolo RADIUS se puede establecer una capa adicional de seguridad.
- Es posible tener dos bases de datos de usuarios de la red inalámbrica en servidores distintos y que estos puedan hacer uso del servicio de red inalámbrica independientemente del punto de acceso al cual se están conectando.

4.2 Recomendaciones

Luego de finalizado el presente trabajo y de cara a darle continuidad y mejoras a lo que se ha abordado, se pueden hacer algunas recomendaciones:

1. Interfaz de administración de ancho de banda

- Para que el trabajo de configuración sea completo, se puede agregar un módulo que establezca las configuraciones del sistema operativo así como su compilación para que el usuario no deba meterle mano a dicha fase que es de las más delicadas. Así como también integrar en la aplicación la configuración del firewall de la red si este se encuentra en el mismo servidor que distribuye la conexión a Internet.
- Agregar la funcionalidad de mostrar gráficas de uso del ancho de banda en tiempo real.
- Migrar la aplicación desarrollada a una ejecución vía web para restarle recursos al servidor y darle movilidad al usuario para que realice sus configuraciones.
- Proponer el uso de esta aplicación para los recintos de la UNAN-Managua que no cuentan con un software para administrar el ancho de banda disponible.

2. Protocolo RADIUS

- Proponer el uso del protocolo RADIUS para autenticación de usuarios de la red inalámbrica a nivel de todos los recintos de la UNAN-Managua.
- Agregar la funcionalidad de contabilidad en la interfaz de administración de freeRADIUS.

5. BIBLIOGRAFÍA

[1]Bert Hubert; *et al.*, (2004). "*Linux Advanced Routing & Traffic Control HOWTO*". [Online] Disponible en <http://lartc.org/>

[2]Martin Devera aka devik, (2003). "*HTB Home*". [Online] Disponible en <http://luxik.cdi.cz/~devik/qos/htb/>

Flickenger R., (2006). "*How To Accelerate Your Internet*" INASP/ICTP. Obtenido de <http://bwmo.net/>

Martin A. Brown, (2006). "*Traffic Control HOWTO*". [Online]. Disponible en <http://www.tldp.org/HOWTO/Traffic-Control-HOWTO/>

Alexey N. Kuznetsov, (2001). "tc - show / manipulate traffic control settings". Páginas del manual Linux.

Martin Devera aka devik, (2002). "HTB - Hierarchy Token Bucket". Páginas del manual Linux.

Alexey N. Kuznetsov, (2001). "sfq - Stochastic Fairness Queueing". Páginas del manual Linux.

Leonardo Balliache, (2003). "*Differentiated Service on Linux HOWTO*". [Online] Disponible en <http://opalsoft.net/qos/DS.htm>

Dan Singletary, (2003). "*ADSL Bandwidth Management HOWTO*". [Online] Disponible en <http://tldp.org/HOWTO/ADSL-Bandwidth-Management-HOWTO/index.html>

Lucian Gheorghe, (2006). "*Designing and Implementing Linux Firewalls with QoS using netfilter, iproute2, NAT and L7-filter*". [Online] Capítulos disponibles en <http://books.google.com>

Joel Spolsky, (s.f.). "*Stack Overflow*". [Online] Disponible en <http://stackoverflow.com/>

(s.f.). "*Programming tutorials and source code examples*". [Online] Disponible en <http://java2s.com/>

Oracle and/or its affiliates, (Última actualización Junio/2012). [Online] "*The Java Tutorials*". Disponible en <http://docs.oracle.com/javase/tutorial/>

Nicolas Baradakis, *et al*; (Última actualización Septiembre/2012). [Online] "*FreeRADIUS: The world's most popular RADIUS Server*". Disponible en <http://freeradius.org>

Philippe Hanset, Chad Bauer; (s.f.). "*eduroam-US: THE Global SSID for Education*". [Online] Disponible en <http://www.eduroamus.org/>

Belnet, (s.f.). "*Belnet Eduroam Service*". [Online] Disponible en <http://www.eduroam.be/>

(s.f.). “*How to...! (deploy, promote and support) eduroam*”. [Online] Disponible en <https://confluence.terena.org/display/H2eduroam/>

Yago Fernández, Antonio Ramos y Jean Paul García-Morán. “*AAA/RADIUS/802.1X. Sistemas basados en la autenticación en Windows y Linux/GNU*”, (2009)

Hugo, (s.f.). “*freeRADIUS*”. [Online] Disponible en <http://freeradius.1045715.n5.nabble.com>