Implementación de *appliances* para enrutado de IPv6 desde plataformas *hardware* económicas

Proyecto realizado por:
Jaime Prieto Bermejo,
Eduardo Vegas Calle y
Rafael Martínez Torres como Profesor Director.

Curso académico 2007/2008
Proyecto de Sistemas Informáticos.
Facultad de Informática, Universidad Complutense de Madrid.

Índice

Autorización de difusión y utilización

Los alumnos:

Jaime Prieto Bermejo con DNI 53615147P y

Eduardo Vegas Calle con DNI 70245480F

autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Resumen / *Summary*

Resumen

A comienzos de los años 2000 numerosas compañías y agencias nacionales para la comunicación realizaron un enorme esfuerzo por redefinir y desarrollar las infraestructuras destinadas al transporte nativo de IPv6, conocidas como *backbones*. En la actualidad, la actividad desarrollada en universidades e instituciones de investigación constituye un medio excelente para diseminar esta nueva tecnología. Este documento describe una posible forma de introducir la conectividad IPv6 en la red de nuestro campus universitario. El planteamiento general es conseguirlo de una manera "suave" y escalonada a corto plazo, teniendo en cuenta la estructura actual. En primer término, la conectividad externa es adquirida utilizando una tecnología fiable, conocida como *6to4 tunneling*. Después la conectividad nativa es introducida progresivamente.

Summary

On the early 2000's a vast effort was made in order to redefine and setup IPv6 backbone's infrastructures. Nowadays, universities and research institutions are considered as excellent opportunities to disseminate this new technology. This paper describes a proposal to introduce IPv6 connectivity in our campus network. Underlying rationale is to achieve it in a smooth way: on a short term. External connectivity is acquired using a reliable technology, known as 6to4 tunneling. Then, native connectivity is introduced progressively.

Palabras clave para la búsqueda bibliográfica

IPv6, IPv4, *tunneling, tunnel,* 6to4, TSP, Teredo, VLAN.

1. Introducción

Descripción del proyecto

IPv6 ya no es un protocolo en fase de diseño y experimentación. Tras años de inversión en la creación de *backbones* de carácter pan-europeo, todas las ventajas que introduce este nuevo protocolo (espacio virtualmente ilimitado de direcciones, seguridad a nivel de red, movilidad, *multicast*…) aparecen disponibles ante un mundo que todavía no deja de sorprenderse por el impacto de una tecnología, IPv4, que fue concebida allá por los años 60 con unos objetivos muy distintos de aquellos a los cuales sirve en la actualidad.

La mayoría de sistemas operativos están preparados para el nuevo protocolo y la emigración al mismo no debería suponer problemas al usuario final. No obstante, desde el punto de vista del profesional existe aún cierto recelo a la introducción de esta nueva tecnología, temor que con este trabajo pretendemos disipar describiendo esta sencilla propuesta para la introducción de IPv6 en una compleja red.

En nuestra opinión, la convergencia tecnológica a medio plazo hacia el nuevo protocolo IPv6 es inevitable. En este contexto las universidades y demás centros de investigación deberían desplegar servicios de red como apoyo a la docencia y la investigación, constituyéndose así como un seminario ideal para la innovación y experimentación con tecnologías que en un futuro no lejano encontraremos integradas plenamente en el resto de la sociedad.

Una vez planteados los aspectos más generales de nuestro proyecto, comenzaremos en las siguientes páginas a describir de manera detallada las características de los protocolos de Internet versión 4 (IPv4) y versión 6 (IPv6) y los aspectos que han provocado que IPv6 esté predestinado en un futuro, a sustituir a IPv4. En segundo lugar nos centramos en los mecanismos de transición de IPv4 a IPv6 existentes, entre los cuales nuestro trabajo en este proyecto se ha centrado en las técnicas de *tunneling*. A continuación exponemos la estructura de la Red de Datos de la Universidad Complutense de Madrid, escenario donde llevar a la práctica los planteamientos anteriormente descritos. Para ello nos serviremos de un medio tan común como un PC, corriendo una distribución de Linux, *Slackware,* y dotándole de las herramientas que hemos desarrollado para facilitar su administración. Por último nos centramos en cuál puede ser el trabajo futuro a desarrollar con este proyecto como base, para finalizar con la bibliografía utilizada.

IPv4 a IPv6

En este punto describimos en profundidad los protocolos de Internet versión 4 (IPv4) y versión 6 (IPv6) así como los factores que han provocado la necesaria implantación de IPv6 como futuro protocolo de Internet.

Protocolos de red

La comunicación en las redes de información es posible gracias a los protocolos de red. Un protocolo es un paquete de bits con una cierta estructura que permite que uno o más dispositivos se comuniquen entre sí. Además son los encargados del establecimiento y la liberación de una comunicación, así como del establecimiento del flujo de información entre dos o más nodos. En Internet, el protocolo más utilizado es el TCP/IP (*Transmission Control Protocol/Internet Protocol*).

TCP/IP

TCP/IP fue diseñado inicialmente para cubrir las necesidades del Departamento de Defensa (DoD) de los EE.UU. A finales de los 60 la ARPA del DoD comenzó a establecer acuerdos con universidades de los EE.UU y con la comunidad de investigación para diseñar estándares y protocolos abiertos para su red conocida como ARPANET. La inicial ARPANET, la primera red de conmutación de paquetes, empezó su operación en 1969 conectando cuatro universidades, tres en el estado de California y una en el estado de Utah. Estos primeros cuatro nodos se enlazaron vía circuitos de 56 kbps utilizando el protocolo NCP (el predecesor de TCP/IP). El experimento fue un éxito permitiendo a la ARPANET continuar con la generación de proyectos de investigación de interés militar.

Uno de esos primeros proyectos fue la demanda de una aplicación de transferencia de archivos, acceso remoto y correo electrónico. En 1973 se inició un proyecto donde se desarrollarían protocolos de capas inferiores debido a que las capas existentes se volvieron inadecuadas. En 1974 el diseño para un nuevo conjunto de protocolos, para la ARPANET, fue propuesto por Vinton G. Cerf y Robert E. Kahn. El nombre oficial para ese conjunto de protocolos fue TCP/IP, el cual fue tomado de los nombres del protocolo de capa de red (*Internet Protocol* [IP]) y de unos de los protocolos de la capa de transporte (*Transmission Control Protocol* [TCP]).

Los protocolos propuestos deberían cumplir con las siguientes especificaciones:

- Independencia subyacente de los mecanismos de la red y de la arquitectura del *host*
- Conectividad universal a través de la red
- Reconocimientos extremo a extremo
- Protocolos estandarizados

Así nace el protocolo TCP/IP. La especificación inicial se fue desarrollando en varias versiones, culminando en la versión 4 en 1979, la cual fue finalmente estandarizada en 1981. El protocolo TCP/IP tuvo gran éxito en el mundo UNIX gracias a que la Universidad de California en Berkeley emprendió la implementación de TCP/IP en la versión 4.2 de su sistema operativo UNIX BSD en 1983, así como a la publicación del código fuente como un software de dominio público.

La versión de IP comúnmente usada es la versión 4 (IPv4), la cual no ha sido substancialmente modificada desde que el RFC 791 fue publicado en 1981. Con 32 bits IPv4 ofrecía más de 4.200 millones de direcciones. A principios de los noventa, con la apertura comercial del Internet, la revolución de PCs, las redes de área local (LANs) y el mal reparto de las IPs, se vio claramente que estas direcciones no serían suficientes. Por este motivo se desarrolló la versión 6 del protocolo IP (IPv6) que está destinado a sustituir a IPv4. En este *status quo* se desarrolla nuestro proyecto.

El horizonte que se presenta

La conversión desde IPv4 a IPv6 supondrá un gran esfuerzo para el sector, mayor que la preparación del efecto Año 2000. Afectará a prácticamente todas las aplicaciones que se basan en comunicaciones, a los sistemas finales, sistemas de infraestructura y arquitecturas de red. Es esencial que este cambio pueda llevarse adelante con responsabilidad para evitar costosos pasos en falso que comprometan el desarrollo de tecnologías no útiles. A diferencia de lo sucedido con el efecto Año 2000, la conversión a IPv6 no tiene una frontera temporal definida. No obstante, como se indicaba anteriormente, la tasa de consumo de direcciones de IPv4 está aumentando constantemente.

La simplicidad del paso de IPv4 a IPv6 será la clave para una rápida adopción de esta tecnología.

Al igual que con IPv4 (cuyas primeras instalaciones fueron mayoritariamente transiciones desde redes X.25), el despliegue de IPv6 comenzará en los márgenes de la red, aprovechando las ventajas de la transmisión sobre cualquier tecnología de red disponible. Se adelanta ya que los proveedores de servicios de Internet (ISPs) van a reaccionar ante la demanda de sus clientes y será éste el factor determinante a la hora de decidir cuándo se implantará el *routing* nativo IPv6. En cualquier caso llevará varios años reemplazar los equipos de comunicaciones, será por tanto un proceso lento.

Es de esperar que en el futuro la mayoría de los fabricantes produzcan sistemas con soporte tanto para IPv4 como IPv6, de forma que las conexiones que no se puedan realizar con IPv6 puedan ser reconducidas utilizando IPv4 (dando por supuesto que existe una conectividad con IPv4 previa a la introducción de IPv6). El objetivo final es garantizar una transición e implantación no traumática, donde las aplicaciones ya actualizadas puedan aprovechar las ventajas del nuevo protocolo, sin que ello suponga renunciar a las actuales presentes en IPv4.

Dando los primeros pasos en este sentido desarrollamos parte de nuestro proyecto. A continuación explicamos más detalladamente las características y limitaciones de IPv4, así como las características de IPv6 y la forma de acceder a él mediante diferentes mecanismos de transición. En particular nuestro proyecto se ha centrado en parte en el estudio y desarrollo de túneles IPv6 sobre IPv4, que permitan utilizar las infraestructuras IPv4 mientras la red IPv6 está siendo implantada.

IPv4

IPv4 es la versión 4 del Protocolo IP (*Internet Protocol*). Esta fue la primera versión del protocolo que se implementó extensamente, y forma la base de Internet. Fue publicado en 1981 en RFC 791. Tiene la capacidad de ofrecer más de 4.200 millones de direcciones, pero no está capacitado para cubrir la demanda actual existente, no sólo por el número de direcciones que soporta, sino también por la forma en que agrupa los bits en su sistema de numeración *network/host* (que desperdicia direcciones y sufre excesiva sobrecarga en el *routing*).

Limitaciones de IPv4

IPv4 usa direcciones de 32 bits, limitándose así a 2^32 = 4.294.967.296 direcciones. Esto, que en un principio puede parecer más que suficiente, en realidad no lo es tanto. Se estima que en la actualidad se encuentran en uso aproximadamente unos 2/3 de estas combinaciones.

Además hay que tener en cuenta que no todas las combinaciones están disponibles para el protocolo IP público. Hay una serie de combinaciones reservadas para utilizarse como máscaras de subred, una serie de combinaciones que se reservan para los servidores DNS, una serie de combinaciones reservadas para comunicaciones específicas, etc. Esto hace que el número real de IP's disponibles no sea tan elevado.

En principio hay que asegurar una serie de combinaciones diferentes para las conexiones a Internet, por lo que a cada proveedor ISP (proveedor de servicios de Internet o ISP por la sigla en inglés de *Internet Service Provider*) se le asigna un determinado número de direcciones IP, asignándolas estos a su vez entre sus clientes. Para optimizar este número de conexiones los proveedores ISP recurren al sistema de IP dinámica. Este sistema hace posible que con un número limitado de IP's se atienda a un número bastante superior de usuarios, a condición de que el número de conexiones simultaneas no supere el número de IP's asignadas (por poner un ejemplo, con 1.000 direcciones IP asignadas un proveedor ISP puede dar servicio a 2.000 clientes que se conecten en tandas de 12 horas cada uno).

Además hay que tener en cuenta que este tipo de conexiones es cada vez más empleado. No sólo por ordenadores, sino también por dispositivos de otro tipo, tales como, por ejemplo, cámaras IP, comunicaciones de voz del tipo VoIP, teléfonos móviles, PDA, etc., lo que ha provocado, junto al cambio de hábitos en las conexiones (hemos pasado de conexiones por un corto periodo de tiempo cuando conectábamos por RTB a tener conectado el ordenador las 24 horas, o al menos 8 horas diarias) y el incremento en el número de usuarios (que prácticamente se duplica cada año desde 1.988), que el número de conexiones disponibles no sólo no sea exagerado, sino que no se encuentre lejos de su agotamiento.

Una de las consecuencias de este sistema es la necesidad de utilizar para conectarse a la red (Internet) un sistema que permita una sola IP por conexión (independientemente de los ordenadores que luego se conecten a través de esta conexión). Este sistema es el denominado NAT (*Network Address Translator*), y permite mediante un *router* (o enrutador) tener una red interna (direcciones IP privadas) apuntando a una sola dirección de Internet (IP pública).

Esta serie de limitaciones han ayudado a estimular el impulso hacia IPv6, que está actualmente en las primeras fases de implementación, y se espera que termine reemplazando a IPv4. Pasamos a describir IPv6 con detalle en las siguientes páginas.

IPv6

IPv6 es la versión 6 del Protocolo de Internet (Internet Protocol). Está destinado a sustituir al estándar IPv4. Su implantación no sólo ampliará el número de direcciones IP disponibles, sino que mejorará el servicio globalmente; por ejemplo, proporcionando a futuras celdas telefónicas y dispositivos móviles direcciones propias y permanentes. Recordemos que a día de hoy se calcula que más de las dos terceras partes de las direcciones que ofrece IPv4 ya están asignadas.

Nacimiento de IPv6

A principios de la década de los 90, en Julio de 1991, el *Internet Engineering Task Force* (IETF) comenzó a trabajar para desarrollar un nuevo protocolo que resolviera en primer lugar el problema de saturación de direcciones de IPv4 y que además aportara mejoras en las comunicaciones.

En noviembre de 1992 surgió un nuevo área de investigación llamada *Internet Protocol Next Generation* (IPng) comisionada por el IETF para estudiar formalmente las diferentes propuestas para el desarrollo de este nuevo protocolo. En diciembre de 1993 fue distribuido el RFC 1550, el cual invitaba a todas las partes interesadas a participar dando sus comentarios acerca de cualquier requerimiento específico que consideraran pertinente incluir durante el proceso de selección de IPng. Veintiuna respuestas fueron recibidas, las cuales contenían puntos de vista de diferentes tipos de industrias, tales como la industria celular, televisión por cable y seguridad, sólo por mencionar algunas.

Ya en el RFC 1726 , el grupo de investigación IPng definió un conjunto de criterios que serían usados para el proceso de evaluación del IPng. Fueron los siguientes:

• Escalabilidad: El nuevo protocolo debería ser capaz de identificar y direccionar por lo menos 10^12 sistemas finales y 10^9 redes individuales.
• Flexibilidad topológica: La arquitectura de enrutamiento y protocolos para IPng debía permitir utilizar una gran cantidad de distintas topologías de red.
• Rendimiento: Para IPng los *host* deberían ser capaces de transferir datos a tasas comparables a las alcanzadas con IPv4 utilizando niveles similares de recursos máquina.
• Servicio robusto: Servicio de red y protocolos de control y enrutamiento robustos.

• Transición: Debían existir mecanismos para realizar la transición de IPv4 hacia IPng de manera transparente para los protocolos y aplicaciones de las capas superiores.
• Independencia del medio: El protocolo debe de trabajar con diferentes medios LAN, WAN y MAN, así como distintas velocidades de conexión, (desde bits/segundo hasta giga bits/segundo.
• Configuración, Operación y Administración: Este nuevo protocolo también debía permitir conexiones fáciles, además de operación y configuración ampliamente distribuida. También debía permitir la configuración automática de *hosts* y enrutadores.
• Operaciones seguras: IPng también debía proveer una capa de red segura (IPSec).
• Acceso y documentación: Los protocolos que definen a IPng, deberían ser publicados en los RFC's, así como estar disponibles libremente y no requerir licencia para su implementación.
• Nombrado único: IPng debía asignar nombres únicos a todos los objetos de la capa IP.
• Multicast: IPng debía soportar transmisión de paquetes Unicast y Multicast.
• Extensibilidad: IPng debía ser capaz de evolucionar para cubrir las necesidades futuras de Internet, permitiendo desarrollar nuevas versiones de él que pudieran coexistir sobre la misma red.
• Servicio de red: IPng debía permitirle a la red asociar paquetes con clases de servicio.
• Movilidad: El protocolo debía soportar huéspedes, redes e inter-redes móviles.
• Protocolo de control: El protocolo debía incluir soporte elemental para probar y depurar redes.
• Redes privadas: Por último, IPng debía permitir a los usuarios construir redes privadas sobre la infraestructura básica de red, soportando ambas, redes basadas o no basadas en IP.

En base a estos criterios varias propuestas fueron revisadas y en enero de 1995, el RFC 1752 fue publicado. En él se resumían las evaluaciones hechas en tres propuestas para el IPng:

- Arquitectura Común para el Protocolo de Internet de la Siguiente Generación (CATNIP).
- Protocolo de Internet Simple Plus (SIPP).
- TCP/IP con Direcciones más Grandes (TUBA).

CATNIP proponía una concordancia entre Internet, OSI y los protocolos Novell. Para lograrlo integraba protocolos de red tales como IP, Novell's *Internetwork Packet Exchance* (IPX) e ISO *Connectionless Network Protocol* (CLNP). El diseño de CATNIP permitía un gran número de protocolos de transporte, tales como el ISO *Transport Protocol class* 4 (TP4), *Connectionless Transport Protocol* (CLTP), TCP, UDP, y Novell's *Sequenced Packet Exchange* (SPX). Sin embargo los revisores de esta propuesta sintieron que CATNIP sólo cumplía con cinco de los criterios establecidos, dos más no eran cumplidos y no tenían una conclusión acerca de los criterios restantes.

SIPP, por su parte proponía una evolución a IPv4, por esto, todas las funciones de IPv4 que les parecieron buenas fueron mantenidas en su nueva propuesta, también fue aumentado el tamaño de las direcciones de 32 a 64 bits de longitud y lo mejor de todo, su instalación sería como una actualización de software. SIPP además sería interoperable con IPv4. En cuanto a esta propuesta, los revisores decidieron que SIPP cumplía con diez de los criterios clave, dos criterios no eran cumplidos y no tenían una conclusión acerca de los criterios restantes.

TUBA proponía remplazar IPv4 con CLNP, lo cual traía consigo dos beneficios inmediatos: incremento en el espacio de direcciones y permitir a protocolos de la capa de transporte operar de manera transparente. Los revisores de TUBA determinaron que esta propuesta cumplía con cinco de los criterios clave, no cumplía un criterio y no tenían una conclusión acerca de los criterios restantes.

Como resultado de las revisiones a estas tres propuestas se decidió elegir como base el Protocolo de Internet Simple Plus (SIPP) e incorporarle direcciones de 128 bits de longitud además de alguna otra modificación. El resultado final es lo que se conoce actualmente como IPv6 o versión 6 del Protocolo de Internet y que pasamos a explicar detalladamente a continuación.

Características IPv6

La primera y más importante mejora que IPv6 presenta frente a IPv4, como ya hemos indicado anteriormente, es el incremento en el tamaño de las direcciones que este maneja, 128 bits de IPv6 frente a 32 bits de IPv4. Este incremento permite soportar más niveles de direccionamiento jerárquico y tener muchos más nodos direccionables. Otra mejora en el diseño de IPv6 es la simplificación del formato del encabezado. Se eliminaron, y en otros casos se hicieron opcionales, algunos de los campos del encabezado, reduciendo así las tablas de enrutamiento y mejorando el rendimiento de los enrutadores. Otra mejora aparece en el soporte para extensiones y opciones en el encabezado, gracias al uso de encabezados opcionales. Se incrementa así la flexibilidad del protocolo IP permitiendo añadir nuevas características en un futuro, sin que sea necesario rediseñar por completo toda la estructura del paquete IP. Otra mejora destacable en IPv6 son las capacidades en la Calidad de Servicio, ya que este permite etiquetar paquetes pertenecientes a un flujo particular para el cual el emisor desea un trato especial por parte de los enrutadores IPv6 que intervengan en la comunicación. Mejoran también la autentificación y privacidad gracias a dos encabezados opcionales que en conjunto nos brindan integridad de datos y confidencialidad.

IPv6 usa direcciones de 128 bits, ofreciendo así 2^128 = 340.282.366.920.938.463.463.374.607.431.768.211.456 (ó 340 sextillones) direcciones — disponiendo así de miles de direcciones por pulgada cuadrada de la superficie de La Tierra, que parecen ser ya sí más que suficientes para cubrir las necesidades de comunicación existentes y futuras.

No obstante la adopción de IPv6 ha sido frenada por la traducción de direcciones de red (NAT), que alivia parcialmente el problema de la falta de direcciones IP. Pero NAT hace difícil o imposible el uso de algunas aplicaciones P2P, como son la voz sobre IP (VoIP) y juegos multiusuario. Además, NAT rompe con la idea originaria de Internet donde todos pueden conectarse con todos. Actualmente, el gran catalizador de IPv6 es la capacidad de ofrecer nuevos servicios, como la movilidad, Calidad de Servicio (QoS), privacidad, etc.

Se espera que IPv4 se siga soportando hasta por lo menos el 2011, dado que hay muchos dispositivos heredados que no se migrarán a IPv6 nunca y que seguirán siendo utilizados por mucho tiempo. IPv6 es la segunda versión del Protocolo de Internet que se ha adoptado para uso general. También hubo un IPv5, pero no fue un sucesor de IPv4. Fue un protocolo experimental orientado al flujo de *streaming* que intentaba soportar voz, video y audio. En la actualidad se sigue experimentando y desarrollando versiones superiores de Protocolos de Internet.

Direccionamiento IPv6

De acuerdo como está definido en el RFC 1884 existen tres tipos de direcciones IPv6 y son:

• *Unicast*: Definidas en el RFC 1887 identifican una sola interfaz. Cuando un paquete es enviado a una dirección *unicast*, éste solamente es entregado a la interfaz que tenga dicha dirección. Existen tres tipos de direcciones *Unicast* y estos son:
 Global: Las direcciones *Unicast* Globales son direcciones de Internet, es decir, tienen significado y pueden ser enrutadas por Internet, ya sea de manera nativa si así lo permite la infraestructura de red, o por medio de túneles.
 Sitio: Este tipo de direcciones identifica una interfaz dentro de un dominio IPv6, pero no pueden ser enrutadas fuera de él, ya que pierden significado.
 Local: Este tipo de direcciones sirven para identificar una interfaz únicamente dentro de un mismo segmento de red (LAN), fuera de él pierden totalmente su valor.

• *Multicast*: Identifica a un conjunto de interfaces. Este tipo de direcciones son muy parecidas a las direcciones de difusión que maneja IPv4, es decir, un paquete que es enviado a una dirección *Multicast* es entregado a todas las interfaces identificadas por dicha dirección.

• *Anycast*: Identifica a un conjunto de interfaces. A diferencia de las direcciones *multicast*, un paquete que es enviado a una dirección *anycast* es entregado a una de las interfaces identificadas por dicha dirección (la más cercana de acuerdo al protocolo de enrutamiento). El RFC 1884 da una referencia sobre posibles usos para este tipo de direcciones, entre ellos están:

 Identificación de un conjunto de enrutadores pertenecientes a un Proveedor de Servicio de Internet (ISP).

 Identificación de un conjunto de enrutadores agregados a una subred particular.

 Identificación de un grupo de enrutadores que sirven como entrada a un dominio en particular.

Seguridad en IPv6

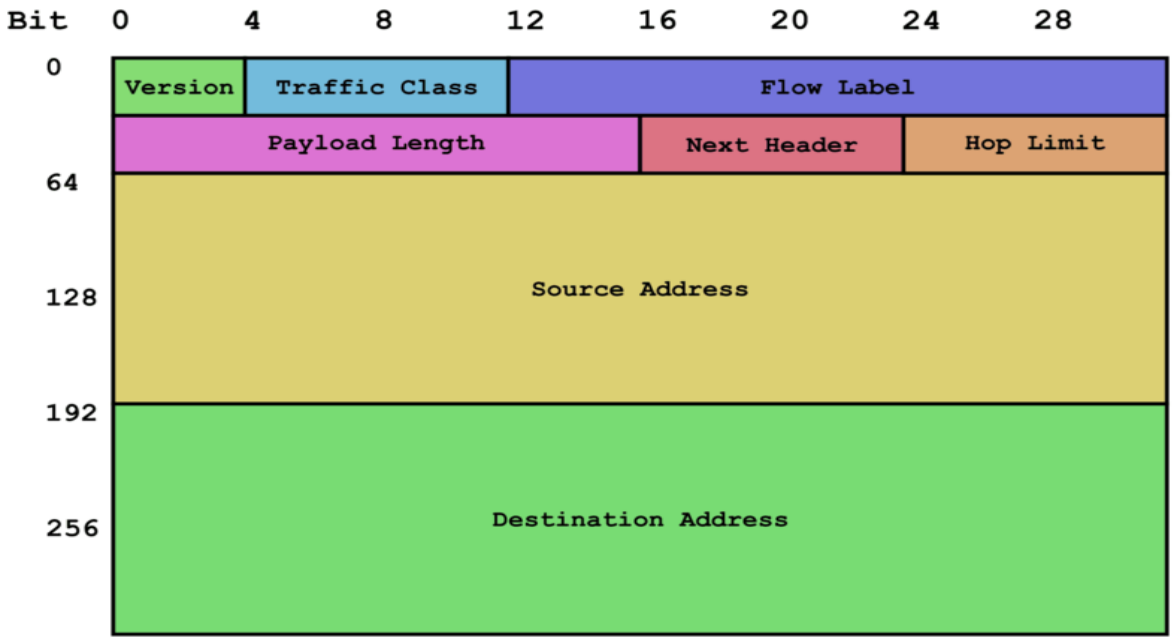
Los diseñadores de IPv4 en un principio no tuvieron que preocuparse por implementar mecanismos de seguridad en IP ya que Internet era utilizado por muy pocas personas; sin embargo, en la actualidad es uno de los medios más recurridos para realizar no sólo transacciones monetarias, sino también utilización de información delicada. Debido a esto, se han desarrollado protocolos para tratar de asegurar los datos que son transmitidos por Internet, algunos ejemplos son SSL y S/MIME. Dichos protocolos no aseguran del todo la información transmitida ya que su efectividad se limita a las capas superiores a la capa de red. Esto hace que la información transmitida sea vulnerable a los ataques que se realizan en la capa de red. IPv6 incorpora los servicios de seguridad de IPSec (*Internet Protocol Security*) definido en el RFC 1825 mediante dos encabezados de extensión:

• Encabezado de Autenticación (AH): El Encabezado de Autenticación como lo define el RFC 1826 provee integridad de datos y autentificación del origen de los datagramas IP, obteniendo así protección contra reenvío de paquetes.

• Encapsulado de seguridad de la carga (ESP): ESP, definido por el RFC 1827, está diseñado para proveer confidencialidad, autentificación del origen de los datos, integridad sin conexión y servicio contra reenvío de paquetes.

Paquetes IPv6

En la siguiente figura se muestra la estructura de un paquete IPv6.



Un paquete en IPv6 está compuesto principalmente de dos partes: la cabecera y los datos.

La cabecera está en los primeros 40 bytes del paquete y contiene las direcciones de origen y destino (128 bits cada una), la versión de IP (4 bits), la clase de tráfico (8 bits, Prioridad del Paquete), etiqueta de flujo (20 bits, manejo de la Calidad de Servicio), longitud del campo de datos (16 bits), cabecera siguiente (8 bits), y límite de saltos (8 bits, Tiempo de Vida).

Después está el campo de datos, con los datos que transporta el paquete, que puede llegar a 64k de tamaño en el modo normal.
Hay dos versiones de IPv6 levemente diferentes. La ahora obsoleta versión inicial, descrita en el RFC 1883, difiere de la actual versión propuesta de estándar, descrita en el RFC 2460, en dos campos: 4 bits han sido reasignados desde "etiqueta de flujo" (*flow label*) a "clase de tráfico" (*traffic class*). El resto de diferencias son menores.

Actualmente el protocolo IPv6 está soportado en la mayoría de los sistemas operativos modernos, en algunos casos como una opción de instalación. Linux, Solaris, Mac OS, NetBSD, OpenBSD, FreeBSD, Windows (2000, XP y VISTA de forma nativa) y Symbian (dispositivos móviles), etc.

2. Mecanismos de transición

Como ya hemos indicado anteriormente el cambio de IPv4 a IPv6 ya ha comenzado y es en esta línea de trabajo donde se ha desarrollado nuestro proyecto. Durante veinte años se espera que convivan ambos protocolos y que la implantación de IPv6 sea paulatina.

Existen una serie de mecanismos que permitirán la convivencia y la migración progresiva tanto de las redes como de los equipos de usuario. Los podemos clasificarse en tres grupos:

- Pila dual o *Dual Stack*
- Traducción
- Túneles

Dual Stack

La pila dual (*Dual Stack)* hace referencia a una solución de nivel IP con pila dual (RFC 2893), que implementa las pilas de ambos protocolos, IPv4 e IPv6, en cada nodo de la red. Cada nodo de pila dual en la red tendrá dos direcciones de red, una IPv4 y otra IPv6.

Traducción

La traducción es necesaria cuando un nodo sólo IPv4 intenta comunicar con un nodo sólo IPv6. Los mecanismos de traducción pueden ser divididos en dos grupos basándonos en si la información de estado está guardada con estado (*NAT-PT, TCP-UDP Relay, Socks-based Gateway*) o sin estado (*Bump-in-the-Stack, Bump-in-the-API*).

Túneles

Los túneles permiten conectarse a redes IPv6 "saltando" sobre redes IPv4. Estos túneles trabajan encapsulando los paquetes IPv6 en paquetes IPv4. De esta manera, los paquetes IPv6 pueden ser enviados sobre una infraestructura IPv4. Hay muchas tecnologías de túneles disponibles. La principal diferencia está en el método que usan los nodos encapsuladores para determinar la dirección a la salida del túnel.
Entre los mecanismos de transición descritos nuestro proyecto se ha centrado en los túneles.

Túneles

Definición

Los túneles proporcionan un mecanismo para utilizar las infraestructuras IPv4 mientras la red IPv6 está siendo implantada. Este mecanismo consiste en enviar datagramas IPv6 encapsulados en paquetes IPv4. Los extremos finales del túnel siempre son los responsables de realizar la operación de encapsulado de paquetes. Estos túneles pueden ser utilizados de formas diferentes:

- *Router* a *router*. *Router*s con doble pila (IPv6/IPv4) se conectan mediante una infraestructura IPv4 y transmiten tráfico IPv6. El túnel comprende un segmento que incluye la ruta completa, extremo a extremo, que siguen los paquetes IPv6.
- *Host* a *router*. *Hosts* con doble pila se conectan a un *router* intermedio (también con doble pila), alcanzable mediante una infraestructura IPv4. El túnel comprende el primer segmento de la ruta seguida por los paquetes.
- *Host* a *host*. *Hosts* con doble pila interconectados por una infraestructura IPv4. El túnel comprende la ruta completa que siguen los paquetes.
- *Router* a *host*. *Router*s con doble pila que se conectan a *hosts* también con doble pila. El túnel comprende el último segmento de la ruta.

Los túneles se clasifican según el mecanismo por el que el nodo que realiza el encapsulado determina la dirección del nodo extremo del túnel. En los dos primeros casos (*router* a *router* y *host* a *router*), el extremo final es un *router* intermedio que debe desencapsular el paquete IPv6 y reenviarlo a su destino final. La dirección del extremo final del túnel ha de ser determinada a través de información de configuración en el nodo que realiza el túnel. Es lo que se denomina “túnel configurado”, describiendo aquel tipo de túnel cuyo extremo final es explícitamente configurado.
En los otros dos casos (*host* a *host* y *router* a *host*), el extremo final del túnel es el nodo destino del paquete, y por tanto, la dirección IPv4 está contenida en la dirección IPv6, este caso se denomina “túnel automático”.

Hemos trabajado sobre tres tipos de túneles

- 6o4
- TSP
- Teredo

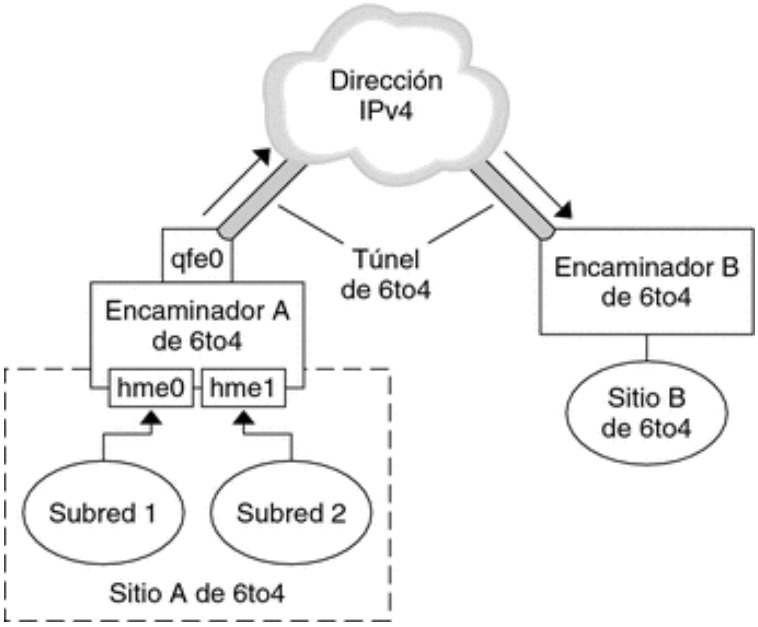
que a continuación describimos detalladamente.

6to4

Los túneles 6to4 permiten que ubicaciones IPv6 aisladas se comuniquen mediante un túnel automático a través de una red IPv4 que no admite IPv6. Para utilizar túneles 6to4 se debe configurar un *router* o encaminador de límite de sistema en la red IPv6 como un punto final del túnel automático 6to4. Después, el encaminador 6to4 puede participar en un túnel hasta otra ubicación 6to4, o, si es necesario, hasta un ubicación IPv6 nativa, no 6to4.

Configuración de un túnel 6to4

La siguiente figura muestra un túnel 6to4 entre dos ubicaciones 6to4.



La figura muestra dos redes 6to4 independientes, Site A y Site B. Cada ubicación tiene configurado un encaminador con una conexión externa a una red IPv4. Un túnel 6to4 a través de la red IPv4 conecta las dos ubicaciones 6to4.

Antes de que una ubicación IPv6 pueda convertirse en 6to4, debe configurar al menos una interfaz de encaminador para que admita 6to4. Esta interfaz debe proporcionar la conexión externa a la red IPv4. La dirección configurada en qfe0 debe ser única globalmente. En esta figura, la interfaz qfe0 del encaminador de límite de sistema *Router* A conecta la ubicación Site A con la red IPv4. La interfaz qfe0 ya debe estar configurada con una dirección IPv4 antes de que sea posible configurar qfe0 como una pseudo-interfaz 6to4.

En la figura, la ubicación 6to4 Site A está compuesta de dos subredes, que están conectadas a las interfaces hme0 y hme1 del encaminador *Router* A. Todos los *hosts* IPv6 de ambas subredes de la ubicación Site A se reconfiguran automáticamente con direcciones derivadas 6to4 al recibir el anuncio del encaminador *Router* A. La ubicación Site B es el punto final del túnel opuesto a Site A. Para recibir tráfico de Site A, debe haber un encaminador de límite de sistema en Site B configurado para admitir 6to4. De no ser así, los paquetes que reciba el encaminador de Site A no se reconocen y se descartan.

A continuación describimos el flujo de paquetes entre un *hosts* en una ubicación 6to4 y un *host* en una ubicación 6to4 remota. Este ejemplo utiliza la configuración de la figura anterior, considerando que los encaminadores y *hosts* 6to4 ya están configurados.

- Un *host* en la subred 1 (Subnet 1) de la ubicación 6to4 Site A envía una transmisión, con un *host* de la ubicación 6to4 Site B como destino. El encabezado de cada paquete del flujo tiene una dirección de origen derivada de 6to4 y una dirección de destino derivada de 6to4.

- El encaminador 6to4 *Router* A recibe los paquetes salientes y crea un túnel a través de una red IPv4 hasta la ubicación 6to4 Site B.

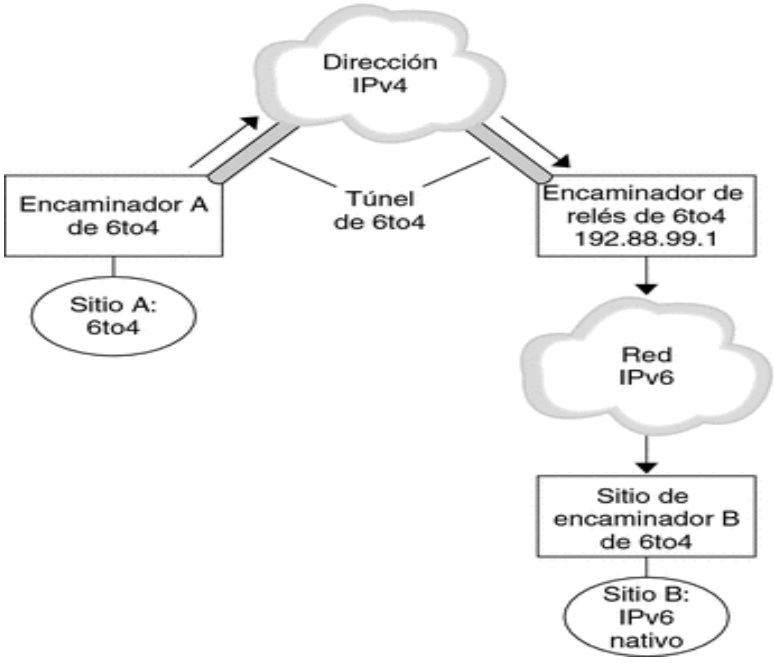
- En encaminador de la ubicación Site A encapsula cada paquete 6to4 dentro de un encabezado IPv4. Después, el encaminador utiliza procedimientos estándar IPv4 para reenviar los paquetes a través de la red IPv4.

- Cualquier encaminador IPv4 que encuentren los paquetes en su camino utilizará la dirección de destino IPv4 del paquete para reenviarlo. Esta dirección es la dirección IPv4 globalmente única de la interfaz del encaminador *Router* B, que también funciona como pseudo-interfaz 6to4.

- Los paquetes de Site A llegan al encaminador *Router* B, que desencapsula los paquetes IPv6 del encabezado IPv4.

- A continuación, *Router* B utiliza la dirección de destino del paquete IPv6 para reenviar los paquetes al receptor en Site B.

En la siguiente figura vemos un túnel desde una ubicación 6to4 hasta una ubicación IPv6 nativa.



La figura muestra la ruta de tráfico desde Site A hasta un túnel 6to4 a través de una red IPv4. Los puntos finales del túnel son el encaminador 6to4 *Router* A y un encaminador de reenvío 6to4. Más allá del encaminador de reenvío 6to4 se encuentra la red IPv6, a la que está conectada la ubicación IPv6 Site B.

A continuación a su vez describimos el flujo de paquetes desde una ubicación 6to4 hasta una ubicación IPv6 nativa, utilizando la configuración de la anterior figura.

- Un *host* en la ubicación 6to4 Site A envía una transmisión que especifica como destino un *host* en la ubicación IPv6 nativa Site B. El encabezado de cada paquete del flujo tiene una dirección derivada 6to4 como dirección de destino. La dirección de destino es una dirección IPv6 estándar.

- El encaminador 6to4 *Router* A recibe los paquetes salientes y crea un túnel a través de una red IPv4 hasta un encaminador de reenvío 6to4.

Los encaminadores de reenvío 6to4 que forman parte del grupo de difusión por proximidad de encaminador de reenvío 6to4 tienen la dirección IP 192.88.99.1 . Esta dirección de difusión por proximidad es la dirección predeterminada de encaminadores de reenvío 6to4. Si necesita utilizar un encaminador de reenvío 6to4 específico, puede anular la dirección predeterminada y especificar la dirección IPv4 del encaminador.

- El encaminador 6to4 de la ubicación Site A encapsula cada paquete dentro de un encabezado IPv4, que tiene la dirección IPv4 del encaminador de reenvío 6to4 como destino. El encaminador 6to4 utiliza procedimientos IPv4 estándar para reenviar el paquete a través de la red IPv4. Cualquier encaminador IPv4 que encuentren los paquetes en su camino los reenviará al encaminador de reenvío 6to4.

- El encaminador de reenvío 6to4 de difusión por proximidad más cercano físicamente a la ubicación Site A recibe los paquetes destinados al grupo de difusión por proximidad 192.88.99.1.

- El encaminador de reenvío desencapsula el encabezado IPv4 de los paquetes 6to4 y, de este modo, revela la dirección de destino IPv6 nativa.

- A continuación, el encaminador de reenvío envía los paquetes, que ahora son sólo IPv6, a la red IPv6, donde los recibe un encaminador de la ubicación Site B. El encaminador reenvía los paquetes al nodo IPv6 de destino.

Los túneles 6to4 sólo son válidos con direcciones IPv4 públicas. Durante el desarrollo del proyecto hemos dispuesto de una IP pública en uno de los laboratorios de la Facultad de Informática de la UCM para desarrollar los *script*s que nos dan acceso a IPv6 mediante túneles 6to4. Tras las numerosas pruebas realizadas hemos podido comprobar que la electrónica de nivel-2 funciona correctamente. Gracias a ello hicimos una visita al Centro de Proceso de Datos de la UCM para mostrar nuestro trabajo. Explicaremos en profundidad la visita al CPD al final de la memoria.

Por último destacar que el requisito imprescindible de tener una dirección IP pública hace que la mayoría de usuarios no puedan trabajar con túneles 6to4 ya que sólo disponen de IP's privadas, las proporcionadas por los proveedores ISP (*Internet Service Provider*). Serán estos usuarios los que se accedan a IPv6 mediante los túneles TSP o Teredo desarrollados también en el proyecto y que describimos detalladamente en las siguientes páginas.

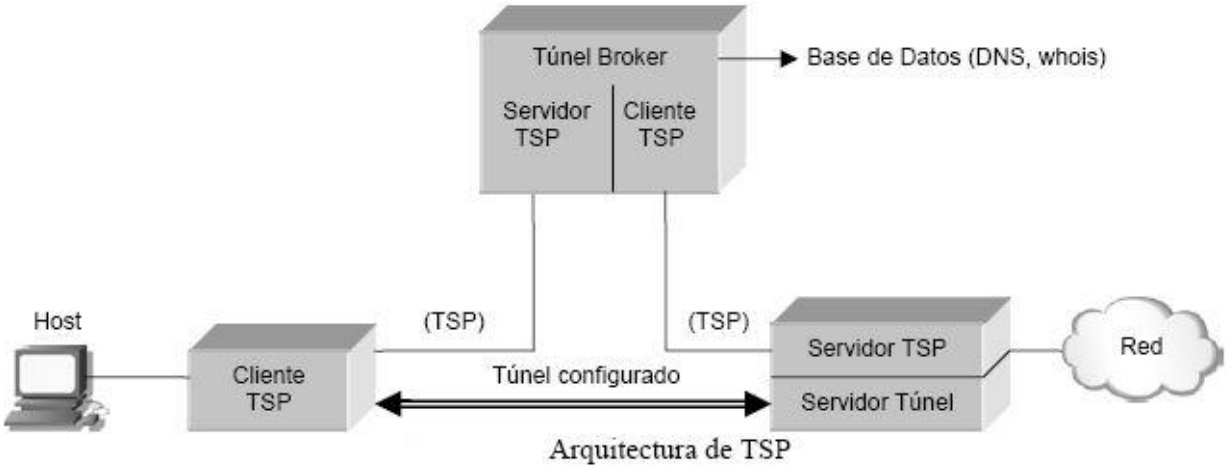
TSP

Tunnel Setup Protocol (TSP) es un protocolo de señal usado para establecer los parámetros de configuración de un túnel IPv6. TSP está basado en una arquitectura cliente-servidor. En nuestro proyecto hemos orientado la investigación sobre TSP hacia Freenet6.

Freenet6 es un servicio de acceso IPv6 gratuito. Este servicio permite a cientos de usuarios de todo el planeta experimentar con el protocolo IPv6. Los usuarios de Freenet6 pueden obtener conectividad IPv6 desde cualquier punto, incluyéndose aquellos que se encuentran *detrás* de una NAT. Es un proyecto que fue desarrollado por una compañía canadiense llamada *Viagénie,* la cual se unió a los esfuerzos por estandarizar IPv6 a partir de 1996 y cuya meta principal es desarrollar IPv6 a gran escala por medio de túneles configurados. Actualmente está disponible en *http://go6.net*

Freenet6 en lugar de ofrecer una interfaz Web, como usualmente hacen otros *tunnel brokers*, utiliza un modelo innovador basado en una arquitectura cliente-servidor. El cliente Gateway6 es software que habitualmente funciona correctamente en un PC y que implementa TSP (*Tunnel Setup Protocol*). El cliente Gateway6 es usado para negociar automáticamente un túnel configurado entre un PC o *router* y el *tunnel broker* de Freenet6, facilitando así la instalación y mantenimiento de IPv6. El código fuente del cliente Gateway6 tiene licencia pública bajo GPL (*General Public License*). Existe también una versión con licencia comercial.

En la siguiente figura mostramos la arquitectura cliente – servidor por medio de un modelo de *broker* IPv6 de los túneles configurados.



Arquitectura de TSP

En un modelo de *Tunnel Broker*, es el *broker* el responsable de toda la comunicación entre los servidores de túnel y los clientes de túnel. Los clientes solicitan túneles al *broker* y éste encuentra un servidor de túnel apropiado, realiza la petición para el establecimiento de un túnel al servidor y envía la información de vuelta al cliente.

Freenet6 puede ser utilizado en dos modos: registrado o anónimo. Para obtener acceso al modo registrado hemos de crear una cuenta a través del portal *http://go6.net*. Así obtendremos una serie de datos de configuración necesarios para acceder al servicio Freenet6. También es posible acceder sin realizar registro alguno, accederemos así a Freenet6 en modo anónimo. En los *script*s desarrollados en el proyecto hemos incluido un sencillo interfaz que permitirá al usuario elegir entre ambos modos de manera cómoda y sencilla y acceder así al servicio ofrecido por Freenet6. Se explicará con más detalle en los siguientes puntos de la memoria.

Teredo

Teredo es una tecnología de transición IPv6 que habilita la direccionabilidad directa de equipos detrás de una NAT (*Network Address Traducer*). Así en presencia de dispositivos NAT no compatibles con IPv6 donde sólo están disponibles direcciones privadas, hay disponible una tecnología, llamada Teredo, también conocida como "NAT Traversal", que permite hacer *tunneling* de tráfico IPv6 sobre NAT incluyendo una cabecear UDP (*User Datagram Protocol*) que se puede emplear para establecer un mecanismo de túnel sobre IPv4 entre sistemas finales. Será la situación (IP privada detrás de una NAT) en la que se encuentren la mayoría de usuarios.

Teredo se basa en el uso de un servidor que puede ser *enrutado* públicamente y globalmente para trabajar con conexiones potenciales. El servidor de Teredo da al servidor y cliente de la aplicación un punto de reunión común en el que pueden intercambiar información de conexión. Los equipos solicitan a continuación una dirección Teredo temporal y los paquetes se pasan mediante túneles a través de la red existente. La compatibilidad de Teredo en WCF requiere la habilitación de la compatibilidad con Teredo e IPv6 en el sistema operativo. Windows XP y los sistemas operativos posteriores admiten Teredo. Windows Vista y los sistemas operativos posteriores admiten IPv6 de forma predeterminada y sólo requieren que el usuario habilite Teredo. Windows XP SP2 y Windows Server 2003 requieren que el usuario habilite IPv6 y Teredo. En el caso de UNIX, entorno en el que hemos desarrollado nuestro proyecto, se desarrolla Miredo. Es una implementación *open-source* de Teredo con la que hemos trabajado.

Miredo fue originalmente desarrollado, y es activamente mantenido por Rémi Denis-Courmont. Incluye todos los componentes de la arquitectura Teredo que a continuación explicaremos detalladamente. Actualmente es distribuido bajo los términos GNU (*General Public License*) y es ejecutable en GNU/Linux kernel, FreeBSD, NetBSD y MacOS X.

Arquitectura Teredo

La arquitectura Teredo está formada por los siguientes componentes

- Cliente Teredo
- Servidor Teredo
- *Teredo relays*
- *Teredo host-specific relays*

Cliente Teredo

Un cliente Teredo es un nodo IPv6/IPv4 que soporta un interfaz de túnel Teredo cuyos paquetes son enviados con otros clientes Teredo o con nodos IPv6 (a través de un *Teredo relay*). El cliente Teredo envía una petición a un nodo IPv6, éste envía la petición encapsulado en UDP/IPv4 al servidor Teredo, quien lo dirige a IPv6 nativo y envía la respuesta al cliente.

Servidor Teredo

Un servidor Teredo es un nodo IPv6/IPv4 que se conecta tanto a IPv4 como a IPv6. Soporta una interfaz de túneles sobre la cual son recibidos los paquetes. La función principal del servidor Teredo es asistir en la configuración de direcciones de los clientes Teredo, así como facilitar la comunicación inicial entre varios clientes Teredo o entre clientes Teredo y *hosts* IPv6.

Teredo relay

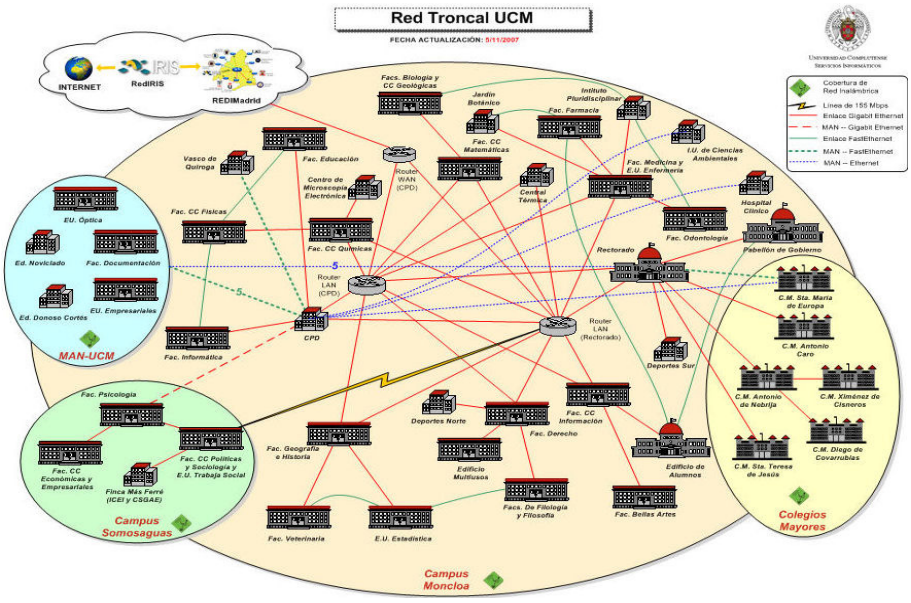
Por *Teredo relay* nos referimos a un *router* IPv6/IPv4 capaz de enviar paquetes entre los clientes Teredo de IPv4 y los *hosts* IPv6. En algunos casos interactúa con el servidor Teredo para facilitar la comunicación inicial entre los clientes Teredo y los *hosts* IPv6.

Teredo host-specific relay

La comunicación entre clientes Teredo y *hosts* IPv6 que son configurados con una dirección global deben pasar necesariamente a través de un *Teredo relay*. Un *Teredo host-specific relay* es un nodo IPv6/IPv4 que tiene un interfaz que proporciona conectividad tanto con nodos IPv4 como con nodos IPv6.

3. Estructura de la Red de Datos de la U. Complutense.

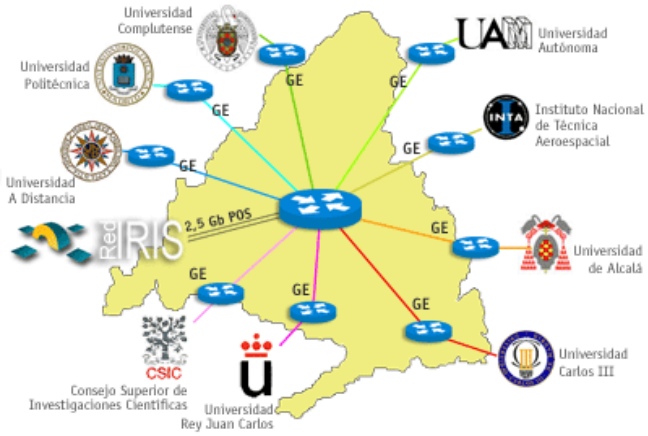
Estructura



Infraestructura de la Red de Datos de la UCM

En la anterior figura se puede observar *grosso modo* la estructura de la Red de Datos (RR.DD) de la Universidad Complutense.

La Universidad recibe un prefijo de red de tipo B (147.96.0.0/16) del ente REDIRIS, proveedor de servicios de Internet para las Instituciones Universitarias y de Investigación en España, a través de su concesionaria regional RedIris-Madrid. REDIMadrid se conecta a Internet a través de la Red Académica Nacional (RedIRIS) mediante un enlace de 2,5Gb de ancho de banda. Y cuya topología de conexión se muestra en la siguiente figura.



La RR.DD se trata de una red fuertemente centralizada, a diferencia de otras estructuras de carácter más federal en otras instituciones similares. Como se puede apreciar, el núcleo de la red lo constituye el Campus de Moncloa, al que se conectan eventualmente otras redes de campus satélite (Somosaguas, Colegios Mayores…) interconectando todos sus edificios mediante tecnología Gigabit Ethernet, también conocida como GigE.

GigE es una ampliación del estándar Ethernet (concretamente la versión 802.3ab y 802.3z del IEEE) que consigue una capacidad de transmisión de 1 gigabit por segundo, correspondientes a unos 1000 megabits por segundo de rendimiento contra unos 100 de Fast Ethernet (También llamado 100-Base/T); en cuanto a las dimensiones de red que ofrece esta tecnología, no hay límites respecto a extensión física o número de nodos. Gigabit Ethernet soporta diferentes medios físicos, con distintos valores máximos de distancia. Se han identificado tres objetivos específicos de distancia de conexión:

- Conexión de fibra óptica multimodo con una longitud máxima de 500m;
- Conexión de fibra óptica monomodo con una longitud máxima de dos kilómetros;
- Conexión basada en cobre con una longitud de al menos 25m. Además, se está trabajando para soportar distancias de al menos 100m en cableado UTP de categoría 5.

Los enlaces verticales que unen los centros de distribución de cableado dentro de cada edificio o grupo de edificios conexos forman una estrella que parte del centro de distribución de cableado principal de cada edificio (el que se une con la red troncal), siempre con capacidad mínima de Gigabit Ethernet sobre fibra óptica.

Se proporciona a cada puesto de trabajo una capacidad de acceso a la red de 100 Mbps, siempre conmutados. Se establece el requisito de que un punto de conexión a la red sólo puede servir a un único equipo de usuario.

En la siguiente tabla se ofrecen las cifras de los puntos de conexión a la red y puertos de electrónica de red instalados, y de los puntos que actualmente están activos dando servicio a los usuarios para los dos campus y centros remotos.

: Noviembre 2007	Puntos de cableado estructurado	Puertos electrónica de red instalados	Puntos activos
TOTAL	**43867**	**23230**	**20749**

Subneting lógico de nivel 3.

El nivel 3 se corresponde a la capa de red, es una capa que proporciona conectividad y selección de ruta entre dos sistemas de hosts que pueden estar ubicados en redes geográficamente distintas, cuya misión es conseguir que los datos lleguen desde el origen al destino aunque no tengan conexión directa.

Existe un único punto de entrada y salida por el que el tráfico es derivado hacia Internet. Este punto es un sistema *router-firewall* con un interfaz que sirve de enlace WAN hacia RedIris-Madrid. El equipo técnico suele referirse a él como 'troncal'.

A continuación se encuentra un *router*, referido como *CPD*, en el que se practica a nivel 3 una fragmentación o *subneting* lógico con más de 30 redes de clase C, (prefijo **/24**) facilitando la gestión y monitorización de las redes. En una red de clase C, se asignan los tres primeros octetos para identificar la red, reservando el octeto final (8 bits) para que sea asignado a los hosts, de modo que la cantidad máxima de hosts es 2^8 - 2, o 254 hosts. Estas redes tienen su medio de transmisión en las correspondientes estructuras de nivel 2, de la que se habla en el siguiente apartado.

VLAN por puertos: fragmentación a nivel 2.

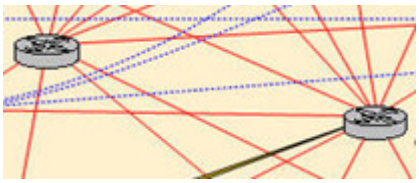
El objetivo del nivel de enlace (nivel 2) es conseguir que la información fluya, libre de errores, entre dos máquinas que estén conectadas directamente (servicio orientado a conexión).
Para lograr este objetivo tiene que montar bloques de información (llamados tramas en este nivel), dotarles de una dirección de nivel de enlace, gestionar la detección o corrección de errores y ocuparse del control de flujo entre equipos (para evitar que un equipo más rápido desborde a uno más lento). Cuando el medio de comunicación está compartido entre más de dos equipos es necesario arbitrar el uso del mismo

Una red tan grande como la que nos ocupa, con más de 20.000 nodos, forzosamente ha de producir un elevado número de colisiones en su intento por acceder al medio compartido, lo que ha de redundar en los tiempos de latencia. A tal efecto, en cada edifico existen equipamientos de *switching,* capaces de fragmentar toda la red de nivel 2 en VLAN por puertos, aislando el tráfico redundante. Paralelamente a este aumento del rendimiento se consigue una incipiente forma de movilidad, permitiendo a dispositivos de una misma VLAN conectarse desde edificios distintos.

Una VLAN es un método de crear redes lógicamente independientes dentro de una misma red física. Varias VLANs pueden coexistir en un único conmutador físico o en una única red física. Son útiles para reducir el dominio de colisión y ayudan en la administración de la red separando segmentos lógicos de una red de área local (como departamentos de una empresa) que no deberían intercambiar datos usando la red local (aunque podrían hacerlo a través de un enrutador).

Sistema de routing tolerante a fallos.

Se puede observar como las conexiones se agrupan en torno a dos polos o estrellas, cada una de ellas con el símbolo *router*. El segundo de estos polos lo constituye el *router* identificado como 'Rectorado', planteado para entrar en acción en caso de eventual caída o inoperatividad del *CPD.* El protocolo empleado para coordinarse entre sí los dos *router*s es el VRRP (*Virtual Routing Redundancy Protocol*).



Detalle conexiones

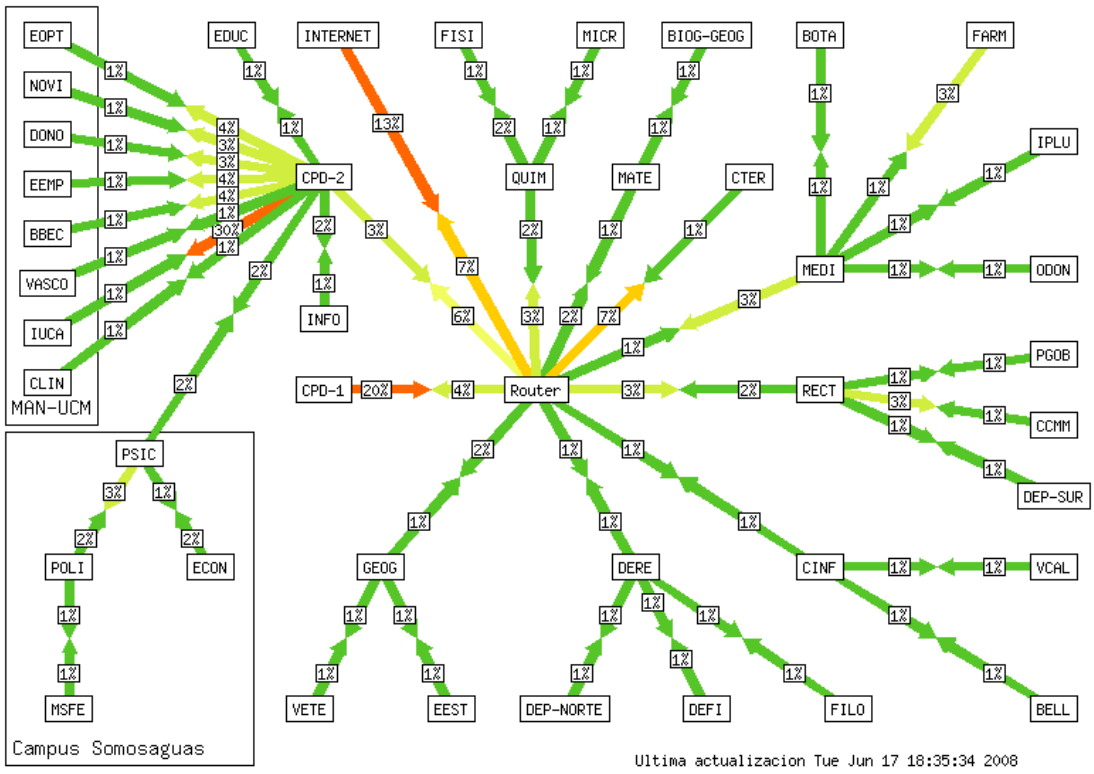
El VRRP, es un protocolo de redundancia no propietario definido en el RFC 3768, diseñado para aumentar la disponibilidad de la puerta de enlace por defecto dando servicio a máquinas en la misma subred. El aumento de fiabilidad se consigue, mediante el anuncio de un *router* virtual, como una puerta de enlace por defecto en lugar de un *router* físico. Dos o más *router*s físicos se configuran representando al *router* virtual, con sólo uno de ellos realizando realmente el enrutamiento. Si el *router* físico actual que está realizando el enrutamiento falla, el otro *router* físico negocia para sustituirlo. Se denomina *router* maestro al *router* físico que realiza realmente el enrutamiento, y *routers* de respaldo a los que están en espera de que el maestro falle.

Características de la electrónica de red

- Toda la red se basa en elementos modulares, compatibles e intercambiables entre sí, lo cual permite una gran facilidad de adaptación a los cambios que requieran ampliación o actualización de las conexiones. Además, esta característica mejora sustancialmente las condiciones de mantenimiento de la red en su conjunto.
- La electrónica cuenta con elementos redundantes, como las fuentes de alimentación, con capacidad de reparto de la carga, y todos los módulos son reemplazables "en caliente", sin interrupción del servicio.
- La capacidad de los mecanismos de conmutación es suficiente para soportar el uso de al menos el 80% del ancho de banda agregado del máximo número de puntos instalables en cada módulo y chasis, sin congestión.
- Los chasis modulares aceptan la inclusión de módulos que soportan los siguientes protocolos y estándares:
    - Ethernet, Fast Ethernet, Gigabit Ethernet, FDDI.
    - Gestión SNMP y RMON.
    - Redes virtuales (802.1 Q).
    - Calidad de servicio (802.1 p).
    - Control de flujo (802.3 x).
    - Soporte *multicast*.
- Además, el equipamiento ofrece las siguientes funcionalidades relevantes:
    - Agregación de líneas o *port trunking*.
    - *Port mirroring*, que permita redirigir el tráfico en una puerta a otra puerta para ser observado.
    - Establecimiento de filtros de acceso sobre la información de niveles 2, 3 y 4.
    - Establecimiento de políticas de seguridad asociadas a cada puerto.

- Control de los umbrales de tráfico de *broadcast*.
- Limitación del uso del ancho de banda por puerto.

Información del uso de la red



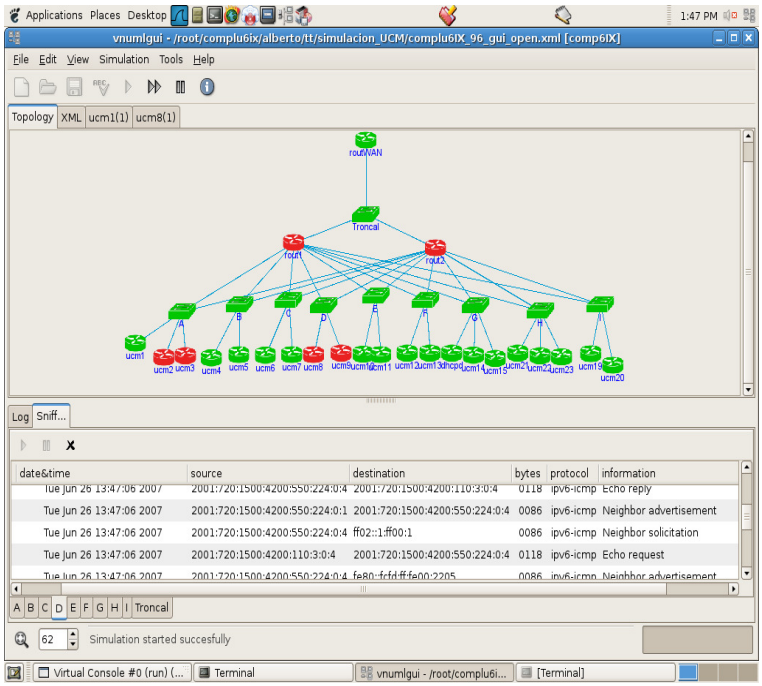
El mapa anterior muestra la topología lógica de la red de datos de la UCM. Están representados los diferentes nodos y los enlaces que los unen. Cada enlace entre dos nodos se representa por dos flechas que indican el sentido del tráfico y que están coloreadas en función del estado de ocupación del mismo. Cada flecha tiene un indicador numérico que representa el porcentaje actual de ocupación del enlace en ese sentido.

Simulación de un entorno IPv4-IPv6.

Pese a que el objetivo final del protocolo sea la total sustitución de la vigente versión IPv4, los expertos sugieren que a corto-medio plazo el escenario más común para facilitar la transición es aquel en el que los dos protocolos coexisten, siendo una minoría las redes que desplieguen IPv6 exclusivamente, adoptando el resto una postura 'híbrida', gestionando cada nodo de red una doble pila de protocolos IPv4 e IPv6.

Este último enfoque es el que hemos adoptado en nuestro planteamiento de transición a IPv6 de la RR.DD de la UCM, valorando que el usuario final no vea perturbado su actividad cotidiana usando los servicios tradicionales sobre IPv4 (Web, correo, ftp, mensajería, etc.)
Como soporte a la toma de decisión, el equipo técnico valoró la posibilidad de realizar simulaciones sobre la topología especificada, y dentro de las limitaciones que un simulador ofrece, verificar la compatibilidad y coexistencia sin interferencia de los dos protocolos.

Para ello empleamos la conocida herramienta de simulación de redes desarrollada por Escuela Técnica Superior de Ingeniería Telecomunicación (ETSIT) de la Universidad Politécnica de Madrid: VNUML.



Simulación de la RR.DD de la UCM en el programa VNUML

VNUML es una colección de *script*s en *Perl* que permite arrancar y configurar una serie de máquinas virtuales de acuerdo a una topología de red indicada previamente en un fichero de configuración XML.

En la figura anterior se puede observar una escena de la simulación, con una muestra representativa de 25 nodos alojados en cada una de las VLAN.

Análisis de la red IPv6 propuesta.

En nuestro análisis, presuponemos que el Ente REDIRIS-Madrid ha de proveernos con un enlace WAN dedicado a IPv6 con un prefijo delegado de /48; una vez satisfecho este requisito, el diseño de la futura red IPv6 pasa por formular una estrategia que contemple los pilares básicos de la infraestructura de red, de los que hablamos a continuación:

Plan de direccionamiento.

Al igual que se hace en IPv4, procedemos a realizar el *subneting* lógico en redes de clase C. Nuestro criterio fue asignar a cada VLAN de nivel 2 una red de prefijo /64.

A continuación vemos el esquema de direccionamiento aplicado.

```
2002:9360:1926:0006::0/64 # VLAN A
2002:9360:1926:0007::0/64 # VLAN B
2002:9360:1926:0008::0/64 # VLAN C
2002:9360:1926:0009::0/64 # VLAN D
```

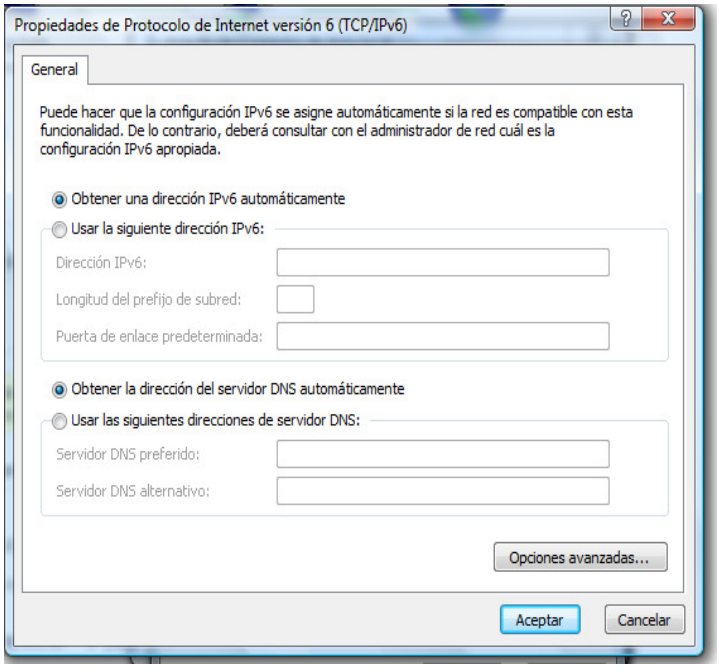
Autoconfiguración RADVD.

El direccionamiento que persiste en la actual infraestructura IPv4 es estático, lo que quiere decir que cada usuario es responsable, en coordinación con el personal del CPD, de configurar manualmente su equipo con una dirección previamente asignada.

Este planteamiento no es asumible con direcciones IPv6 de 128 bits, donde la especificación de direcciones IPv6 tales como 2002:9360:1926:00fe:e409:fe05:c21b:c670 puede ser tediosa y proclive a errores tipográficos.

Afortunadamente, este factor fue tenido en cuenta por los diseñadores del protocolo, dotando al mismo de posibilidades de *autoconfiguración*, según se dicta en la especificación RFC 2461. Gracias a ello, la asignación de una dirección se realiza automáticamente al conectar un equipo a la red, sin perjuicio de la dirección estática que por su papel adquieren algunos nodos dedicados a funciones específicas (servidores Web, agentes SNMP de monitorización, etc.).

En este sentido, en nuestro planteamiento, el mismo agente encargado de enrutar el tráfico IPv6 se erige como *anunciante de prefijos de red.* Exactamente cada uno de los prefijos anunciados coincide con los descritos anteriormente sobre los planes de direccionamiento.
Actualmente, el sistema operativo Windows Vista, las distintas distribuciones de sistemas Linux e incluso plataformas MAC, vienen preparadas por defecto para reaccionar a estos mensajes.



Windows Vista habilitado por defecto para IPv6.

Servicios básicos de aplicación.

Una vez cubierta la infraestructura de los servicios de red, el primer servicio de aplicación más urgente es el de DNS, encargado de trasladar los nombres de dominio a sus correspondientes direcciones IPv6. Las direcciones de IPv6 tienen reservadas un tipo especial de registro (AAAA), pero el mismo protocolo de aplicación DNS puede estar definido utilizando como transporte TCP/IPv4 o TCP/IPv6.

De este modo, en un entorno híbrido como el que nos afecta, no es imprescindible a corto plazo la implementación de un servidor de nombres sobre TCP/IPv6: la consulta al DNS se hace por IPv4.
A medio-largo plazo consideraremos la migración del resto de servicios de aplicación (SMPP, IMAP4, POP3, HTTP…) al nuevo protocolo, pero siempre teniendo en cuenta que en una red de estas características los servidores efectúan un balanceo de carga sobre la misma IP. Por tanto no siempre será posible contar con versiones habilitadas para el nuevo protocolo.

Implementación y despliegue.

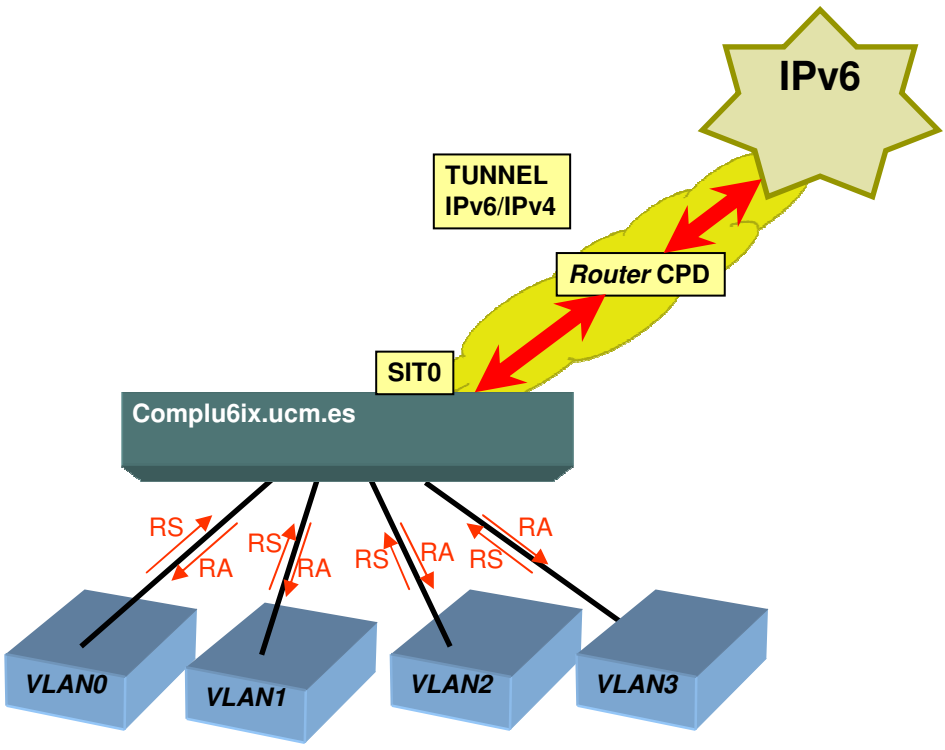
A la hora de plantearse la inyección de conectividad IPv6 en la actual infraestructura, corresponde al equipo técnico la valoración del impacto que puede tener en la red en producción la sustitución de elementos troncales, como *routers*, por otros capaces de procesar el nuevo protocolo. En este sentido, no parece *realista* la propuesta de cambiar estos elementos de tanta importancia de forma inmediata. El escepticismo de esta práctica hace que muchas organizaciones con una política de seguridad conservadora posterguen de forma reiterada la introducción del protocolo IPv6.

En el caso de la RR.DD de la UCM, este es un punto de significativa importancia, habida cuenta de la gran inversión que supone la adquisición de equipamiento electrónico. Afortunadamente, no sucede lo mismo con los equipos de *switching*; dado que estos elementos procesan el nivel 2, para ellos es transparente el uso de un protocolo de nivel 3 como IPv6. Tampoco parece razonable desplegar la conectividad IPv6 sobre algunos sectores VLAN estratégicos, a falta de la elaboración de un plan de seguridad perimetral más sofisticado.

Una primera solución de compromiso consiste en ofrecer conectividad IPv6 a través de equipo plataforma-hardware muy común -PC- corriendo el sistema operativo Linux, al punto muy optimizado (compilación de kernel, configuración de servicios…) para desempeñar funciones de *enrutamiento*, y excepcionalmente dotado de 6 tarjetas de Red Ethernet convencionales conectadas a las VLAN que se desea autoconfigurar según el planteamiento anteriormente expuesto.

En esta fase no adoptaremos conectividad nativa, sino que como único requisito plantearemos que una de estas tarjetas esté dotada de una dirección IPv4 pública, factor indispensable para poder experimentar con técnicas de *tunneling*.

La siguiente figura ilustra el planteamiento anteriormente descrito: `complu6ix.ucm.es` representa el *appliance* que sirve de agente *enroutador* de IPv6. Cada una de sus tarjetas aparece conectada a una red VLAN.



Esquema de túnel empleado para adquirir conectividad IPv6

Por último, una serie de consideraciones generales sobre seguridad.

Teniendo en cuenta que nuestro sistema adquiere conectividad mediante técnicas de *tunneling*, compromete el control de dos focos: el *router* virtual, encargado de gestionar el *enrutado* interior y la codificación de paquetes IPv6 usando IPv4, y el *router* CPD, desde donde se gobierna el tráfico IPv4 hacia o desde el exterior.

En el nodo que actúa como *router* virtual, podemos controlar mediante las reglas de filtrado del cortafuegos todo lo relativo al tráfico IPv6 de entrada-salida originado o con destino a la Red de Datos de la Universidad Complutense, en sus diversas modalidades, ya sea por puertos, o por direcciones. Esto se consigue mediante la conocida herramienta `iptables`, que se describe ligeramente en el capítulo siguiente.

Por otro lado, al objeto de impedir que cualquier nodo del espacio IPv4 de la RR.DD puede erigirse como *router* virtual, pudiendo de este modo burlar la anterior política de seguridad, corresponde al *router* central del CPD filtrar todo intento de tráfico IPv6 por otros túneles distintos al que se ha propuesto como “oficial”.

4. Herramientas para la administración del sistema

Slackware

Nuestro entorno de producción ha sido la distribución Linux Slackware creada por Patrick Volkerding. Su principal característica es la simplicidad y su meta la estabilidad del sistema. Nosotros hemos trabajado con la versión 12.0. Hemos desarrollado nuestros *scripts,* que a continuación describimos, mediante el editor *vi* tratando siempre de seguir el esquema de los *scripts* ya existentes en Slackware, con el objetivo de que estos fuesen fácilmente integrables en el sistema.

Netconfig

Netconfig es un complejo *script*. Es muy utilizado para la administración de sistemas UNIX. Está escrito en lenguaje *shell*. La definición que podemos dar de *Netconfig* sería la de un instrumento a base de texto para la configuración simple de dispositivos *ethernet*. La función desempeñada por este *script* es la de recopilación de información de usuario, para la configuración de una tarjeta (IP, *gateway*, *dhcp*, caso estático, etc.). *Netconfig* original solicita información para una tarjeta (eth0). Esta información es guardada en el fichero de configuración */etc/rc.d/rc.inet1.conf*., el cual también puede ser editado manualmente

A continuación vemos un ejemplo del posible contenido del archivo.

```
# Primary network interface card (eth0)
 IPADDR[0]=""
 NETMASK[0]=""
 USE_DHCP[0]=""
 DHCP_HOSTNAME[0]=""
```

En este caso, nuestra tarea es meramente poner la información correcta entre las comillas (por ejemplo para habilitar el DHCP escribiremos "*YES*" en *USE_DHCP)*. Estas variables son llamadas por */etc/rc.d/rc.inet1* al inicio para configurar las interfaces de red.

El *script /etc/rc.d/rc.inet1* tiene una serie de órdenes que se ejecutan sobre líneas de comandos.

Los comandos disponibles son:

- *start, up*: levanta todas las interfaces configuradas.
- *stop, down*: detiene todas las interfaces.
- *restart*: levanta de nuevo las interfaces.
- *ethi_start, ethi_up*: levanta la interfaz *ethi* especificada.
- *ethi_stop, ethi_down*: detiene la interfaz *ethi*.
- *ethi_restart*: detiene *ethi* y la inicia de nuevo.

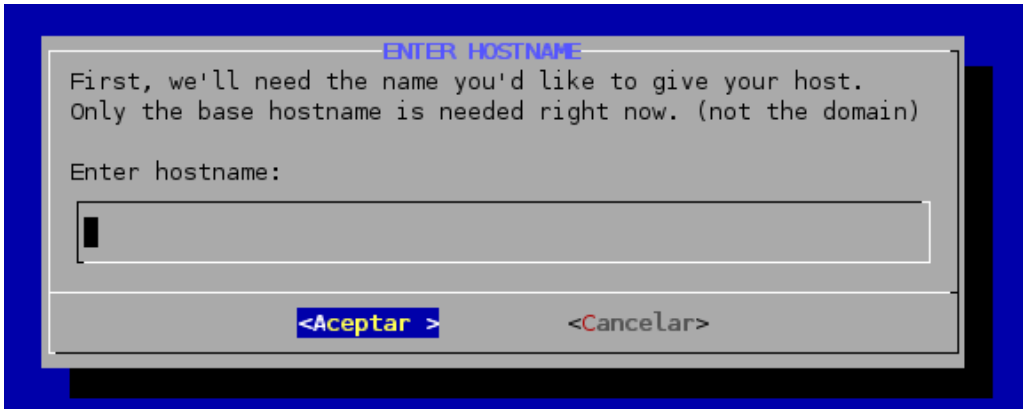
Netconfig nos ofrece distintos esquemas de direccionamiento IP para configurar las tarjetas.

- *IP Estático*: Las direcciones IP estáticas son direcciones fijas, que sólo cambian cuando se modifican manualmente. Éstas se utilizan cuando el administrador no desea que la información IP cambie, ya sea en servidores internos de una LAN, cualquier servidor conectado a Internet, y enrutadores en red.
- *DHCP*: Es el medio a través del cual las direcciones IP pueden ser asignados a una computadora al arrancar. Cuando el cliente DHCP arranca, pone una solicitud de la Red de Área Local (LAN) para el servidor DHCP, para que le asigne una dirección IP. El servidor DHCP tiene un fondo de direcciones IP disponibles. El servidor va a responder a esta solicitud con una dirección IP del fondo, junto con un 'tiempo de arrendamiento'. Una vez este tiempo para una dirección IP dada haya expirado, el cliente debe contactar al servidor de nuevo y repetir la negociación. El cliente entonces aceptará la dirección IP que le brinda el servidor y configurará la interfaz de red que lo solicitó con la dirección IP.
- *Disable*: esta opción se elegirá para dejar la tarjeta deshabilitada.

Con *Netconfig* sólo tenemos la posibilidad de configurar una tarjeta. Como parte de nuestro proyecto hemos ampliado el *script* original para que se detecten todas las tarjetas existentes en la máquina. Éstas se almacenarán en un *array*, el cual será recorrido pidiendo para cada tarjeta los datos de configuración necesarios al usuario.

A continuación describimos las acciones que un usuario podrá realizar sobre *Netconfig* para configurar las tarjetas que desee sobre su máquina.

Primeramente debemos introducir el nombre que vamos a asignar a nuestro *host*.

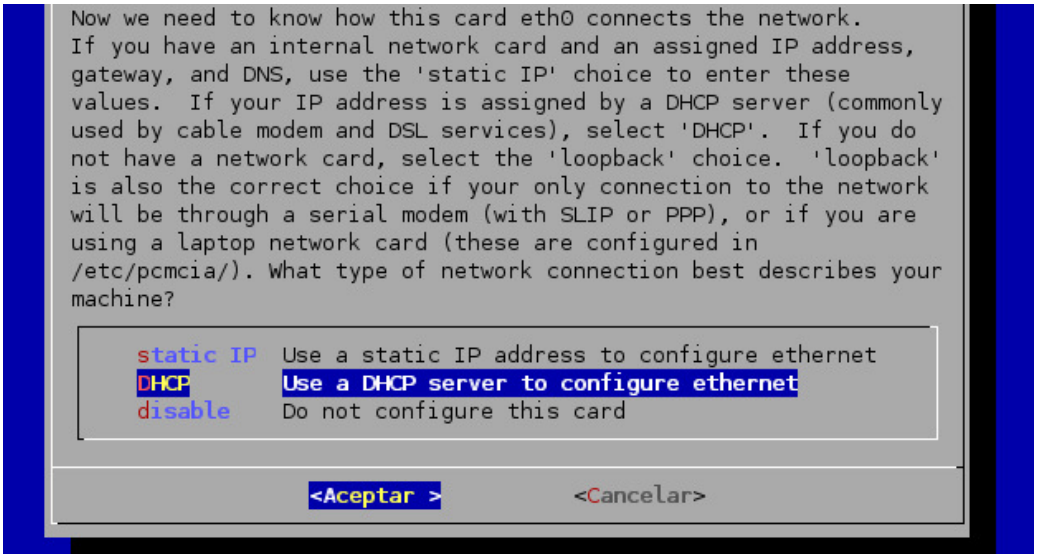


A continuación debemos introducir el nombre de dominio.

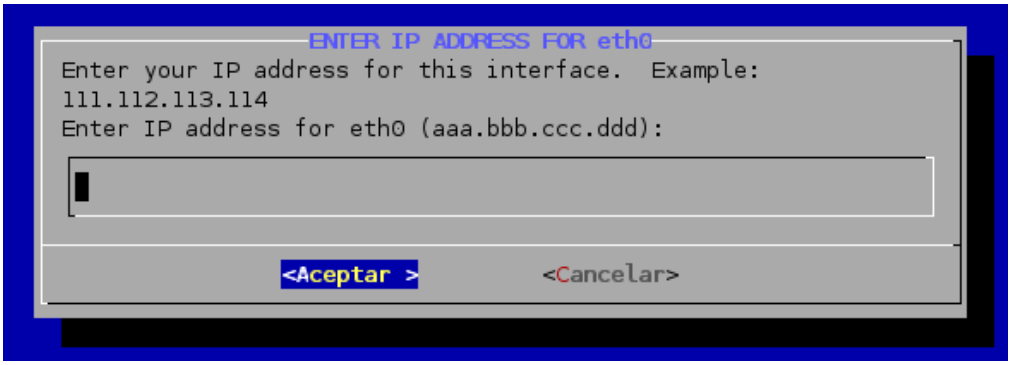
Después, una vez reconocidas todas las tarjetas que tiene la máquina, el *script* pregunta una por una la configuración que queremos dar a cada tarjeta. Las opciones de configuración que se ofrecen son las explicadas con anterioridad:

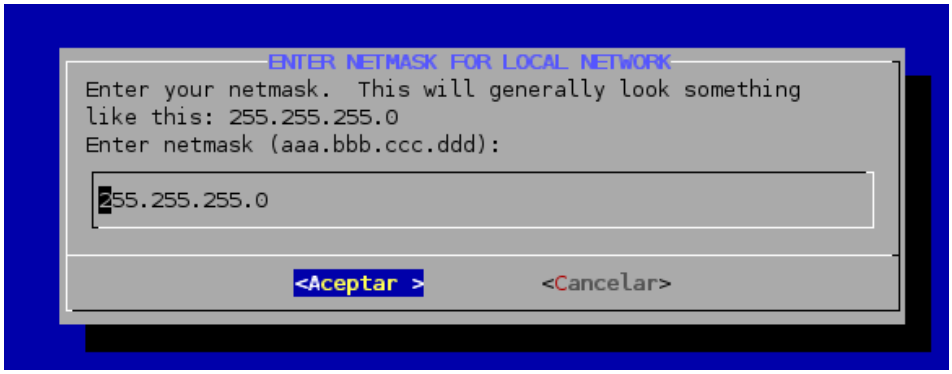
- *Static IP*: caso estático, en el que debemos introducir una IP y la máscara.
- *DHCP*: mecanismo de auto asignación, o auto configuración en IPv4.
- *Disable*: si queremos dejar deshabilitada esa tarjeta.

El menú mostrado es el siguiente:

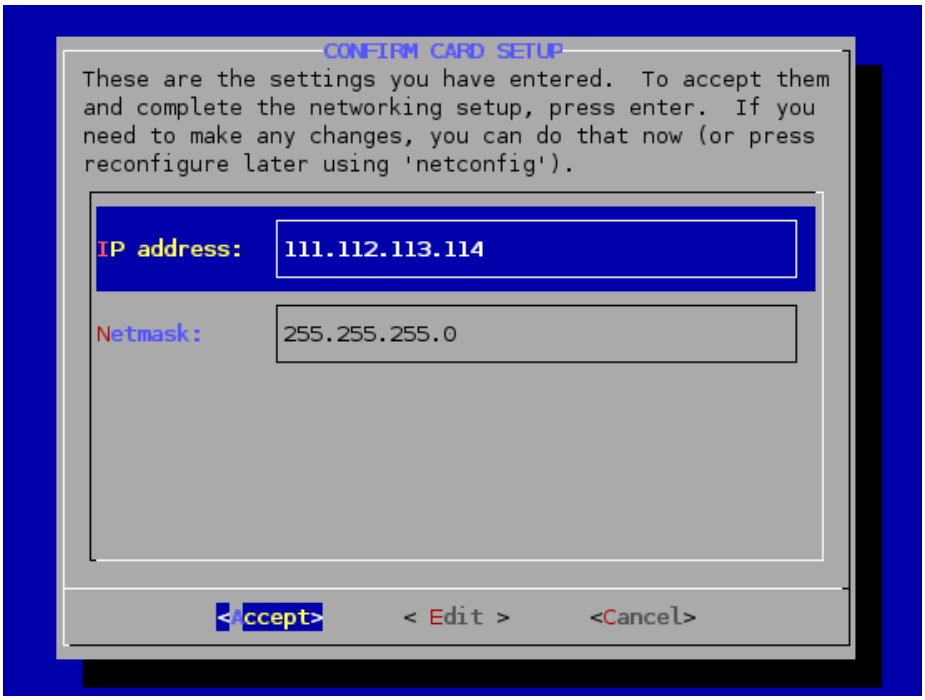


Si la opción elegida es la estática, el usuario ha de introducir la IP, y seguidamente la máscara.



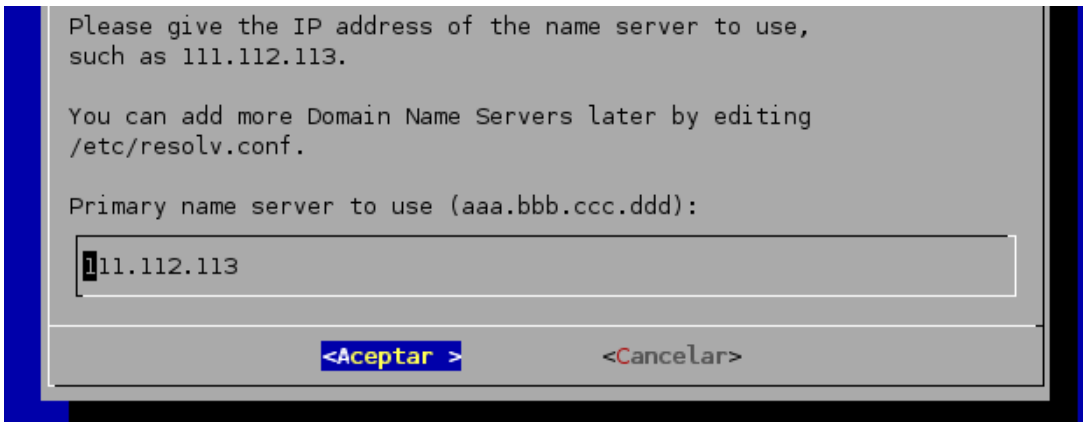


Una vez concluida la configuración de esa tarjeta nos mostrará un cuadro con la información que hemos introducido, en el que podremos editar algún cambio de última hora.



Una vez configuradas todas las tarjetas nos pedirá información para el *NameServer*.

En caso de elegir la opción *NameServer* introduciremos una serie de datos.



Seguidamente nos pedirá la información de *gateway*.

El proceso de configuración finaliza mostrando el siguiente mensaje de éxito en la configuración.

Netconfig6

*Netconfig*6 es un nuevo *script* homólogo a *Netconfig*. Tiene su misma funcionalidad (la configuración de tarjetas) orientada en este caso a IPv6. Su estrucutra es similar a la de *Netconfig* original, para así poder implantarlo en un futuro en la distribución de Slackware. Consta de un fichero de configuración donde se almacenan los datos de las tarjetas (*rc.inet1v6.conf)*. Y un *script rc.inet1v6* de comandos que ejecutará las órdenes sobre las tarjetas que decida el usuario leyendo los datos de configuración necesarios del fichero citado anteriormente (*rc.inet1v6.conf)*.

*Netconfig*6 se ejecuta de la misma forma que su hermano mayor. Primero detecta todas las tarjetas conectadas a la máquina, (almacenándolas en *arrays*), para después pedir la información de configuración para cada una de las tarjetas existentes.

Las opciones que da este *script* para la configuración de las tarjetas son:

- *Static IP*: similar al caso en IPv4, pero con los datos en IPv6. El usuario tendrá que introducir la IP y la máscara.
- *AUTOCONF*: es un mecanismo de auto asignación basado en *radvd* utilizado para la auto configuración de IPv6. Es un demonio que trata solicitudes de *router* (RS) de los host y responde con anuncios de *router* (RA). Estos anuncios contienen información, que es usada por los host para configurar sus interfaces.
- *DHCPv6*: análogo al mecanismo de auto asignación de IPv4 pero para IPv6.
- *Disable*: si queremos dejar la tarjeta deshabilitada.

Una función que cabe destacar en *Netconfig*6 es la de verificación de direcciones y máscaras, ya que es capaz de detectar si una dirección o máscara introducida se corresponde con el formato correcto en IPv6. De no ser así se comunicará al usuario la incorrección detectada y se pedirán los nuevos datos correctos.

Fichero de configuración rc.inet1v6.conf

El fichero de configuración donde se recopila la información es *rc.inet1v6.conf*. La información recopilada puede ser editada manualmente.

Consta de los siguientes campos:

IPADDR[i]-> Contiene la IP en IPv6 de la tarjeta i, sólo se necesita en el caso estático.

NETMASK[i]-> La máscara en IPv6 de la tarjeta i.

USE_AUTOCONF[i]-> Sólo cuando expresamente se ponga a *'no'* estará desactivada, si no es la opción que está por defecto (RADVD).

USE_DHCPv6[i]-> Otro mecanismo de auto asignación, sólo cuando sea *'yes'* estará activada.(No está implementado).

DHCP_HOSTNAME[i]-> Nombre del *Host* para DHCP.(No está implementado).

Una vez explicados todos los campos del fichero de configuración mostramos un ejemplo de cómo sería dicho fichero si todas las tarjetas estuvieran configuradas a *autoconf (radvd)*.

```
# Config information for eth0:
IPADDR[0]=""
NETMASK[0]="64"
USE_AUTOCONF[0]=""
USE_DHCPv6[0]=""
DHCP_HOSTNAME[0]=""
# Config information for eth1:
IPADDR[1]=""
NETMASK[1]="64"
USE_AUTOCONF[1]=""
USE_DHCPv6[1]=""
DHCP_HOSTNAME[1]=""
# Config information for eth2:
IPADDR[2]=""
NETMASK[2]="64"
USE_AUTOCONF[2]=""
USE_DHCPv6[2]=""
DHCP_HOSTNAME[2]=""
```

```
# Config information for eth3:
IPADDR[3]=""
NETMASK[3]="64"
USE_AUTOCONF[3]=""
USE_DHCPv6[3]=""
DHCP_HOSTNAME[3]=""
```

Existe la posibilidad de que no todas las tarjetas estén configuradas del mismo modo. A continuación mostramos un ejemplo en el que las tarjetas 0 y 1 están configuradas como estáticas, y las restantes con *autoconf*.

```
# Config information for eth0:
IPADDR[0]="ipv6"
NETMASK[0]="64"
USE_AUTOCONF[0]="no"
USE_DHCPv6[0]=""
DHCP_HOSTNAME[0]=""
# Config information for eth1:
IPADDR[1]="ipv6"
NETMASK[1]="64"
USE_AUTOCONF[1]="no"
USE_DHCPv6[1]=""
DHCP_HOSTNAME[1]=""
# Config information for eth2:
IPADDR[2]=""
NETMASK[2]="64"
USE_AUTOCONF[2]=""
USE_DHCPv6[2]=""
DHCP_HOSTNAME[2]=""
# Config information for eth3:
IPADDR[3]=""
```

NETMASK[3]="64“

USE_AUTOCONF[3]=""“

USE_DHCPv6[3]=""“

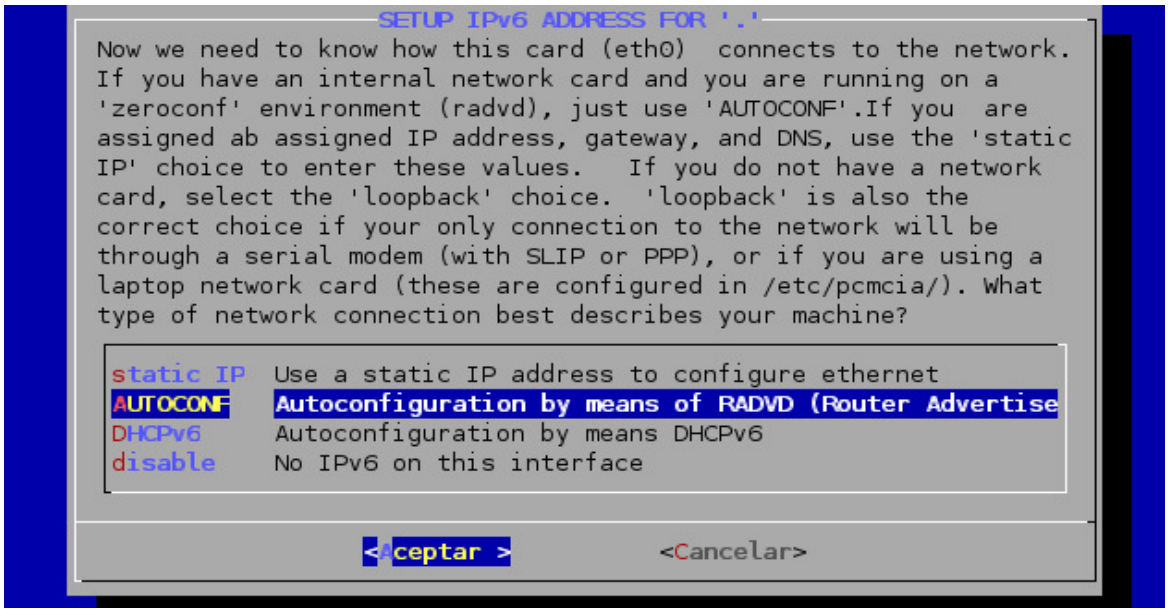
DHCP_HOSTNAME[3]=""“

Ficheros de ordenes rc.inet1v6

Este *script* fue creado para poder ejecutar distintas acciones sobre las tarjetas previamente configuradas en IPv6. Sigue la estructura de *rc.inet1*, el cual trabaja sobre tarjetas en IPv4. Está adaptado a las necesidades y requisitos existentes en IPv6. Los comandos disponibles son:

- *start,up*: levanta todas las interfaces configuradas.
- *Stop,down*: detiene todas las interfaces.
- *restart*: levanta de nuevo las interfaces.
- *ethi_start, ethi_up*: levanta la interfaz *ethi* especificada.
- *ethi_stop, ethi_down*: detiene la interfaz *ethi*.
- *ethi_restart*: detiene *ethi* y la inicia de nuevo.

A continuación describimos las acciones que un usuario podrá realizar sobre *Netconfig*6 para configurar las tarjetas que desee sobre su máquina.

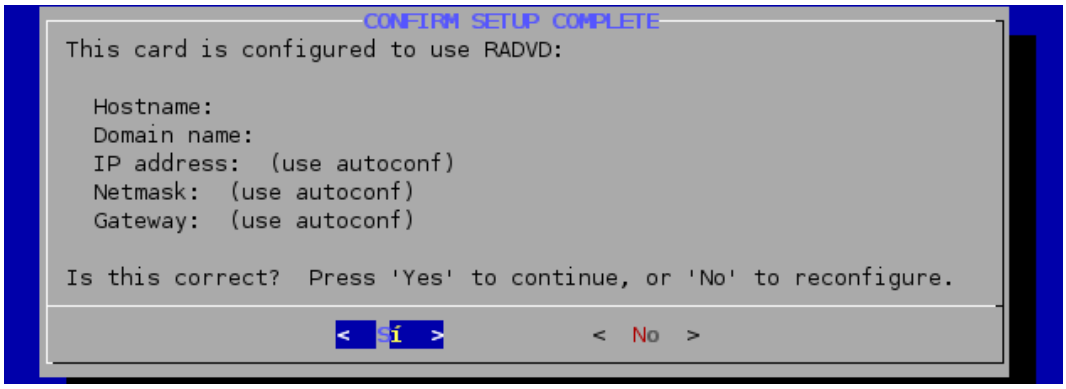


Como vemos, nos pide la configuración para la tarjeta0 (*eth0*). Después se repetirá el proceso para las siguientes tarjetas que sean detectadas.

Las posibles configuraciones que podemos seleccionar son las ya citadas en la descripción del *script*, y son las siguientes:

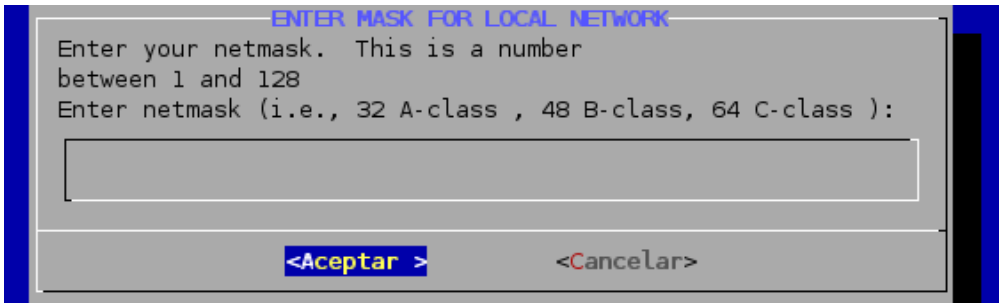
- *Static IP*: similar al caso en IPv4, pero con los datos en IPv6. El usuario tendrá que introducir la IP y la máscara.
- *AUTOCONF*: es un mecanismo de auto asignación basado en radvd.
- *DHCPv6*: análogo al mecanismo de auto asignación de IPv4 pero para IPv6.
- *Disable*: si queremos dejar la tarjeta deshabilitada.

Dependiendo de cuál de las siguientes opciones elijamos se nos pedirán unos datos u otros.



Eligiendo *radvd* todo ya está auto configurado, y sólo tendremos que confirmar la configuración.

Si por el contrario eligiéramos el caso estático tendríamos que introducir los datos del mismo modo que veíamos en IPv4, aunque ahora trabajando en IPv6 también deberemos introducir la máscara.



Por último nos mostrará la configuración que hemos elegido. En este punto tendremos la posibilidad de editar los datos de configuración introducidos.

Finalmente se mostrará una confirmación de que el proceso ha concluido correctamente.

eas6ytunnel

Como parte de nuestro proyecto y con el objetivo de configurar y levantar los túneles descritos en el punto 2 de la memoria de una manera sencilla y eficaz hemos desarrollado los siguientes *script*s:

- eas6ytunnel
- rc.eas6ytunnel

que a continuación describimos detalladamente.

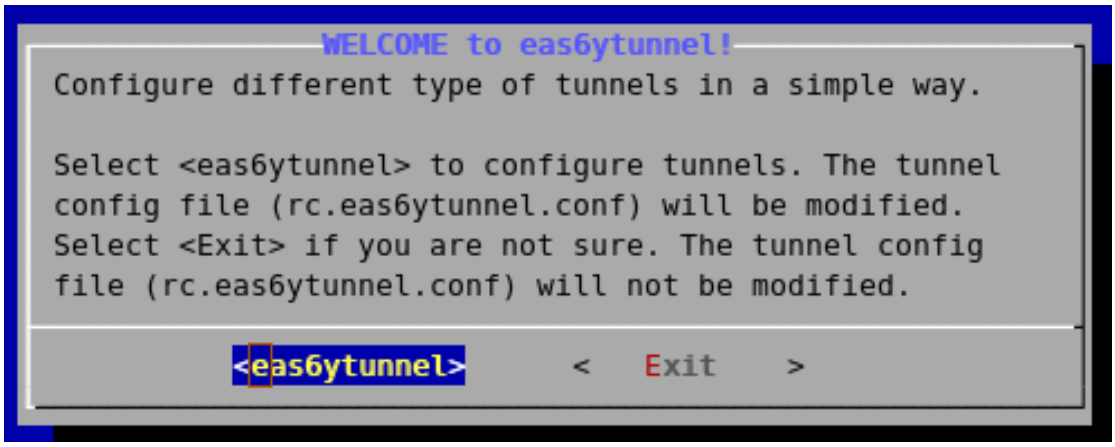
eas6ytunel

El *script* eas6ytunnel permite configurar distintos tipos de túneles de manera rápida y sencilla. Con él el usuario puede consultar el estado actual del archivo de configuración de túneles (*/etc/rc.d/rc.eas6ytunnel.conf* que describimos más adelante) e ir añadiendo los datos de configuración de nuevos túneles (6to4, TSP y Teredo) de una manera cómoda gracias al uso de *Dialog* que brinda gran flexibilidad al lenguaje de *script*ing por ser una interfaz de ventanas.

A continuación describimos las acciones que un usuario podrá realizar sobre eas6ytunnel para configurar los túneles que desee sobre su máquina.

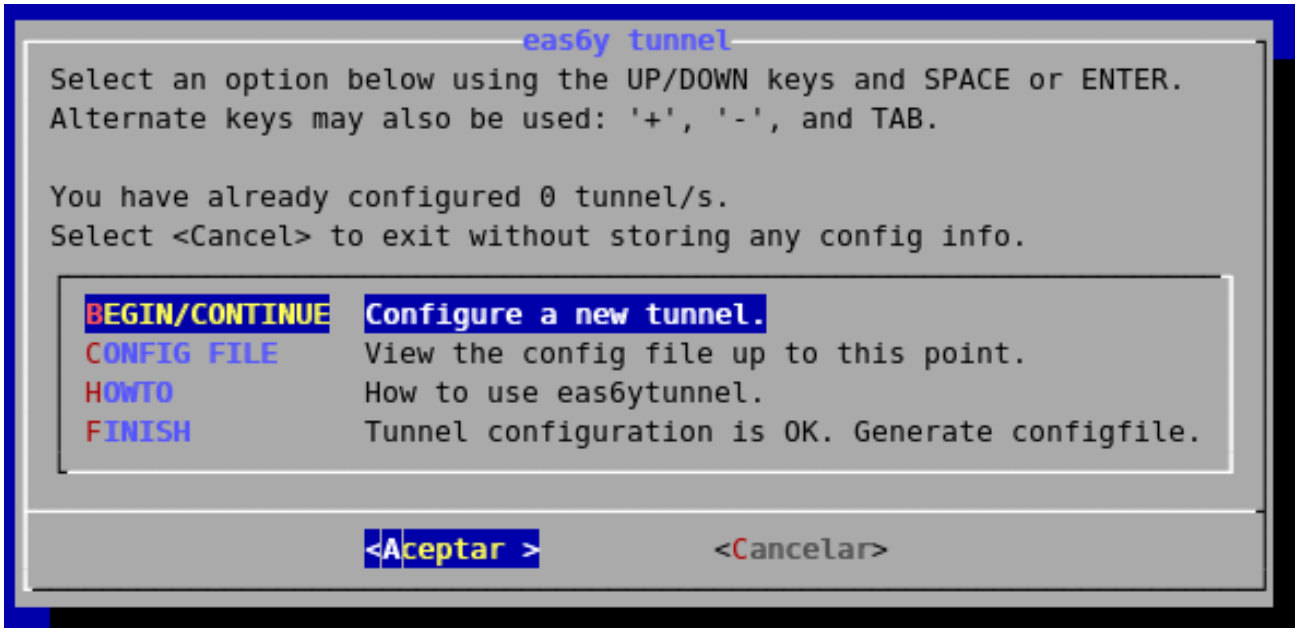
Al ejecutar eas6ytunnel se muestra en primer lugar el siguiente mensaje.

En este mensaje de bienvenida se da la opción al usuario de ejecutar el propio *script* eas6ytunnel para configurar nuevos túneles (este proceso de configuración es explicado en las siguientes páginas) o de salir del mismo (en el caso de que el usuario no esté seguro de las acciones que quiere realizar, en este caso el archivo de configuración existente no se verá modificado).

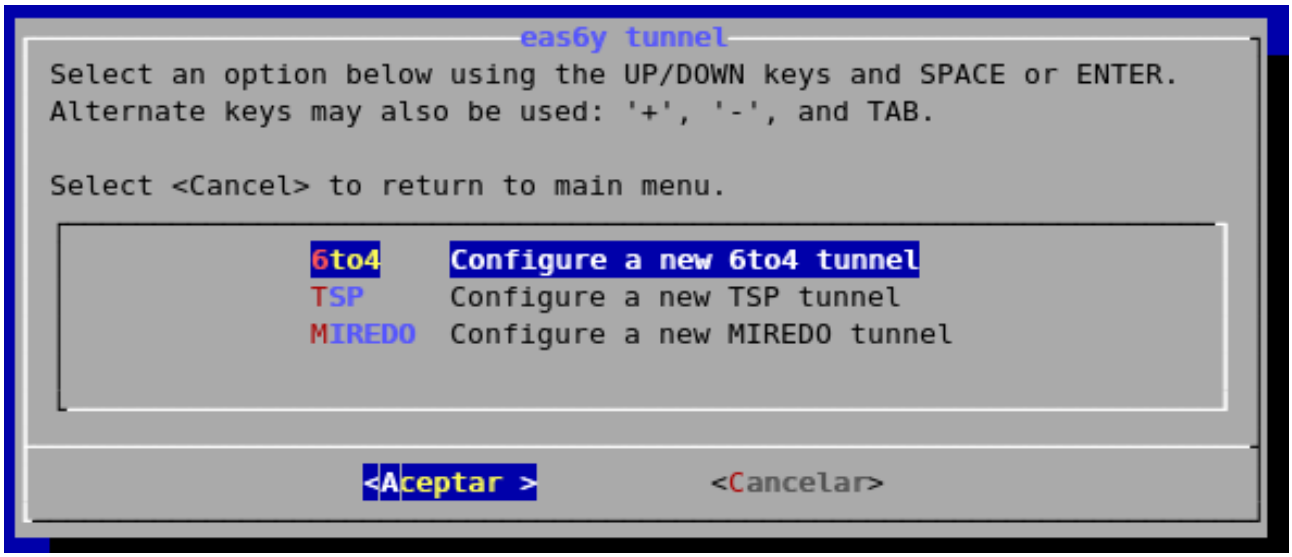


Para configurar nuevos túneles el usuario accederá seleccionando <eas6ytunnel>. Si los túneles ya estaban configurados y por tanto el archivo de configuración */etc/rc.d/rc.eas6ytunnel.conf* ya estaba generado correctamente, el usuario seleccionará <Exit> para terminar la ejecución sin hacer ningún cambio.

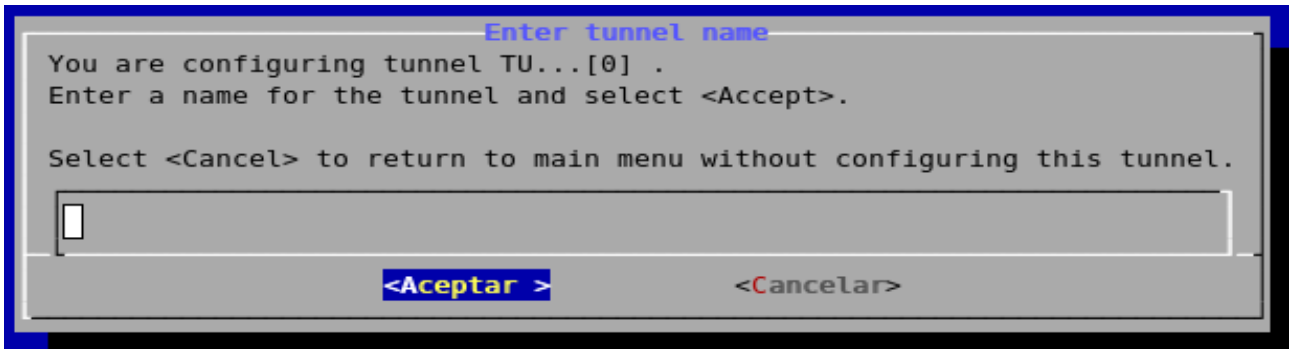
Una vez que accedemos se muestra el menú principal en el que podemos seleccionar entre configurar un nuevo túnel, consultar el estado del archivo de configuración, consultar la ayuda y finalmente generar el archivo de configuración de los túneles.



Para empezar a configurar nuevos túneles seleccionaremos BEGIN/CONTINUE, accediendo así a un menú en el que el usuario elegirá el tipo de túnel que desea configurar.



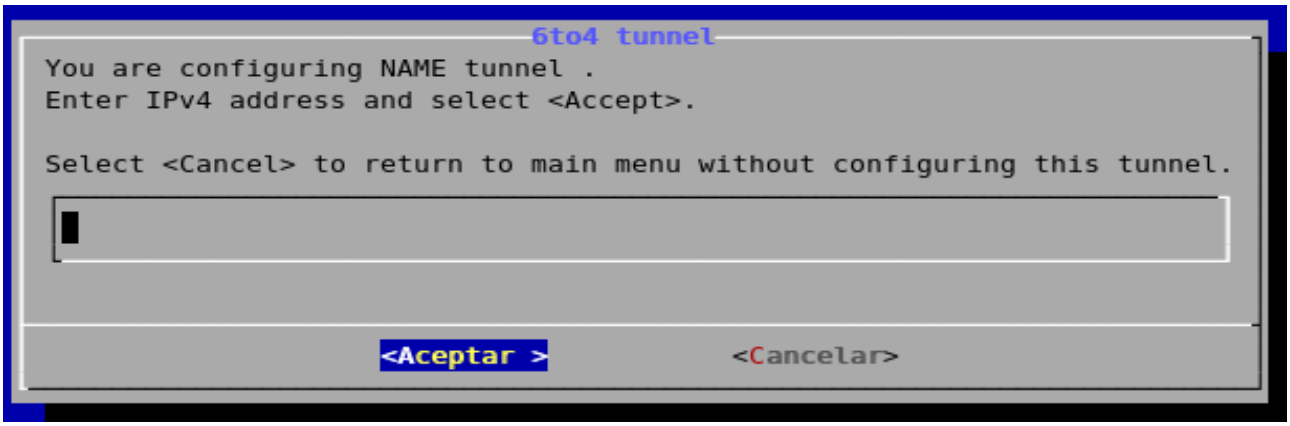
Cada tipo de túnel necesita unos datos de configuración concretos. Un dato común que se pide es el nombre que el usuario desea dar al túnel. Deberá ser un nombre válido (no vacío) y que no haya sido elegido anteriormente. Si no es así se mostrará un ventana de error.



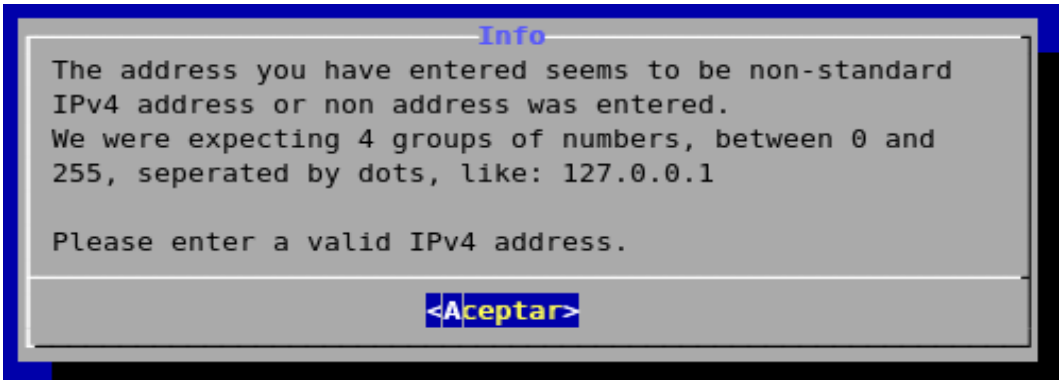
Tras introducir el nombre del túnel se pedirán los datos de configuración necesarios para cada túnel.

En el caso del 6to4:

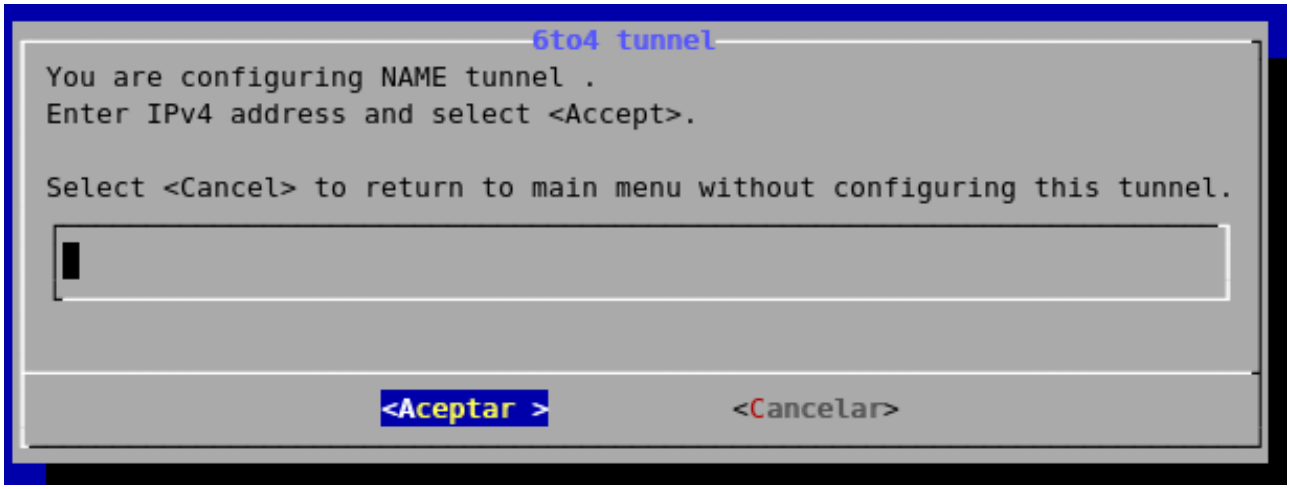
Primero se ha de introducir una dirección IPv4 (recordemos que como indicábamos al describir los túneles 6to4 ésta ha de ser una dirección pública).



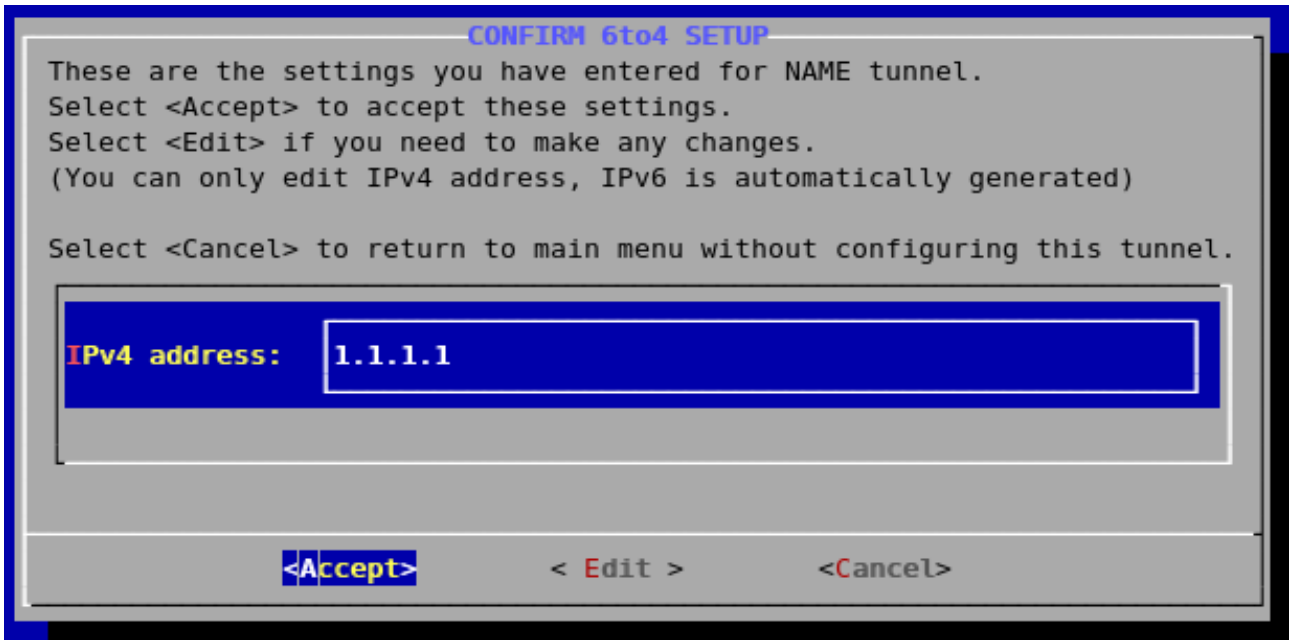
En caso de no introducir ninguna dirección o de que ésta no sea válida se mostrará una advertencia por pantalla.



Para continuar con la configuración del túnel el usuario deberá introducir una IPv4 válida o bien seleccionar <Cancelar> para volver al menú principal sin configurar el actual túnel.

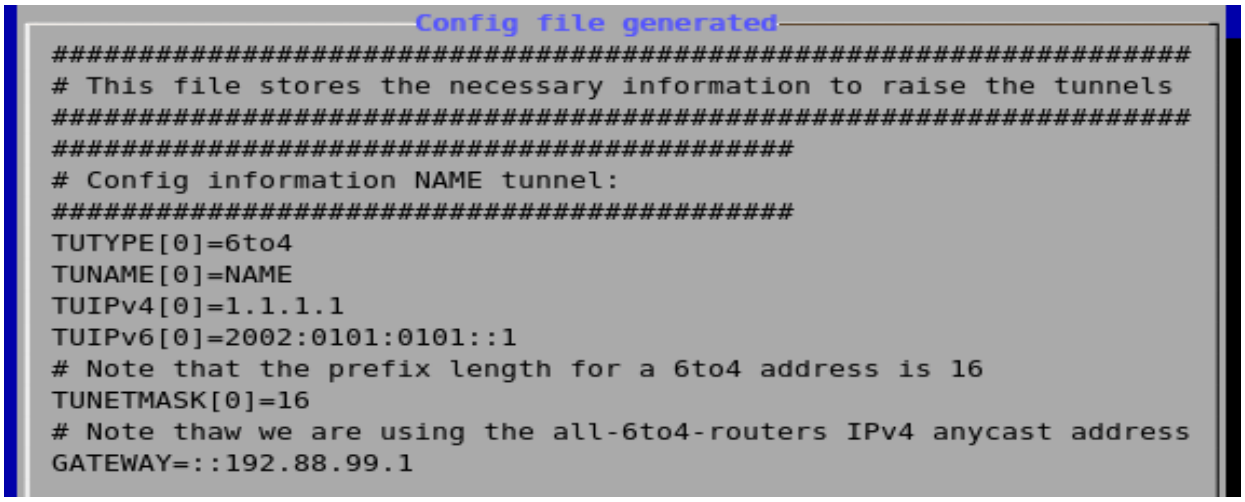


Una vez introducida una IPv4 válida se mostrarán estos datos de configuración por pantalla. En este punto se permite la posibilidad de editar esta información (seleccionando la opcion <Edit>) por si se hubiera cometido algún error en el proceso de configuración.
Vemos un ejemplo de posible configuración.



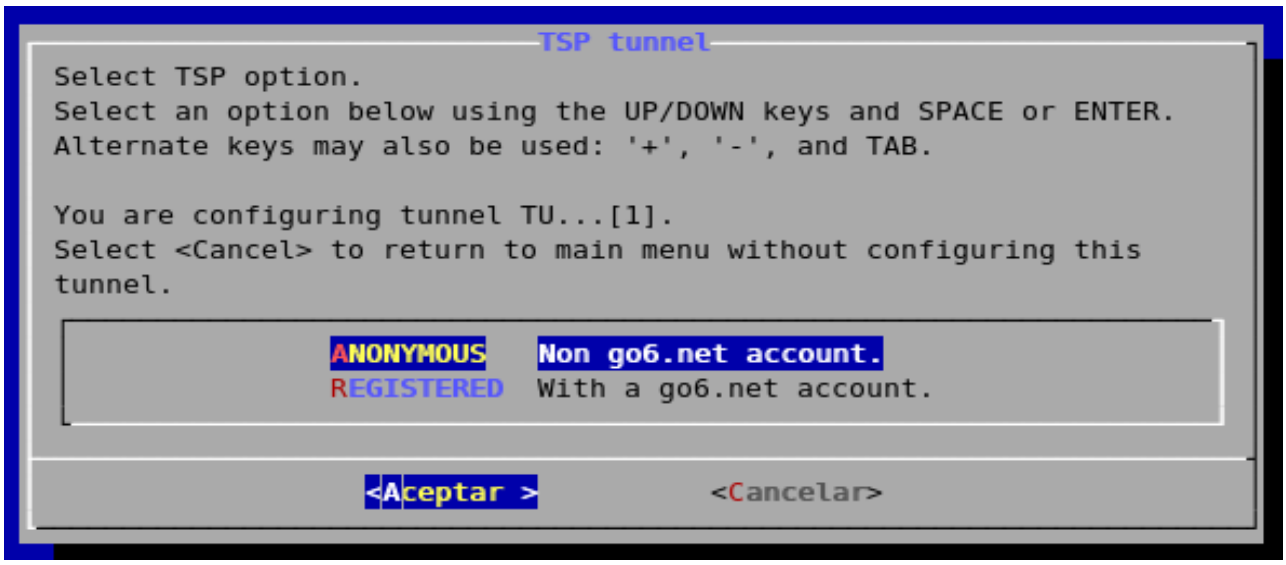
Seleccionando <Aceptar> confirmamos que la información de configuración es correcta y por tanto será almacenada de manera temporal (hasta que el usuario decida generar finalmente el archivo de configuración) y podrá ser consultada desde el menú principal seleccionando CONFIG FILE.

En este punto tendríamos:



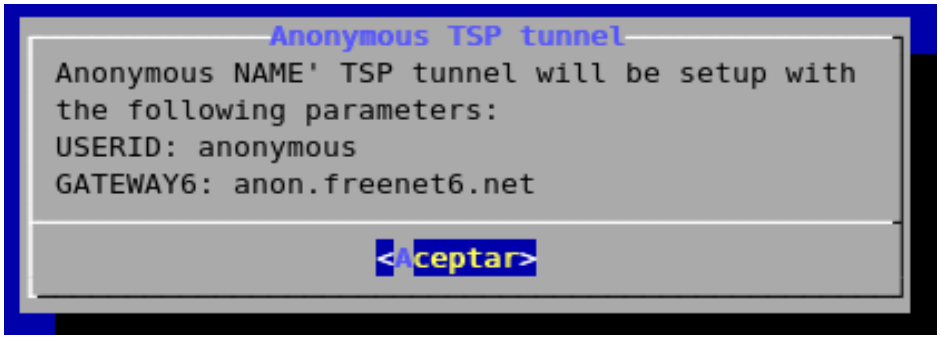
El archivo contiene los datos de configuración del túnel que ya hemos configurado. Esta información es almacenada en distintos *arrays* (para túneles 6to4: TUTYPE para el tipo de túnel, TUNAME para el nombre del túnel, TUIPv4 para la dirección IPv4, TUIPv6 para la dirección IPv6 que nuestro *script* traduce de forma automática a partir de la dirección IPv4 introducida y TUNETMASK para la máscara) en la posición 0 ya que es el primer túnel configurado.
El siguiente túnel que configuremos será almacenado en la posición 1 de los distintos *arrays*, el siguiente en la posición 2 y así sucesivamente. Estamos trabajando por tanto con *arrays* de N posiciones dispuestas de 0 a N-1.

Esta información es para configurar un túnel 6to4. En el caso de configurar un túnel TSP la información que el usuario deberá introducir será pedida también por pantalla. En primer lugar el usuario debe seleccionar entre los dos tipos de túneles TSP disponibles (anónimo o registrado).



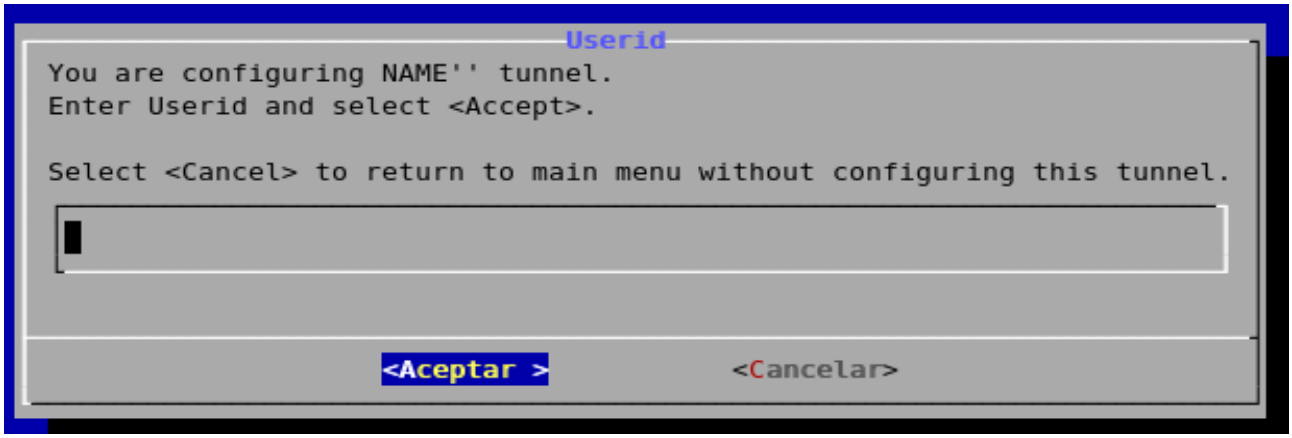
A continuación se introducirá el nombre del túnel (del mismo modo explicado para el 6to4).

En el caso de que deseemos configurar un túnel TSP anónimo (seleccionando <ANONYMOUS>) los datos de configuración están disponibles en el propio *script* y simplemente se mostrarán por pantalla.

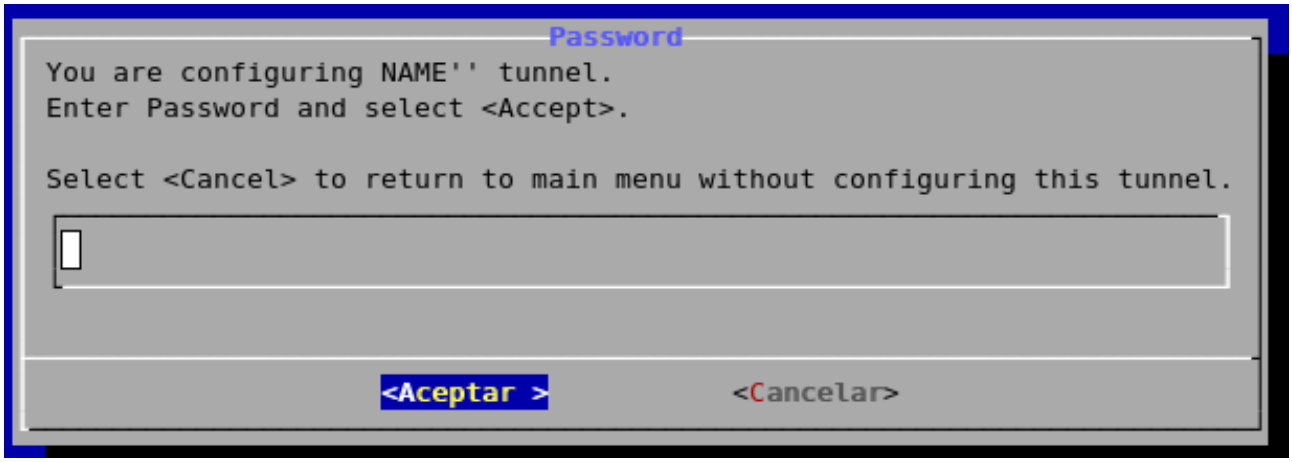


En el caso de que tengamos una cuenta en *http://go6.net* y queramos configurar un túnel TSP con registro (seleccionando <REGISTERED>) los datos de configuración obtenidos tras el registro deberán ser introducidos. Son UserId, Password y Gateway6.

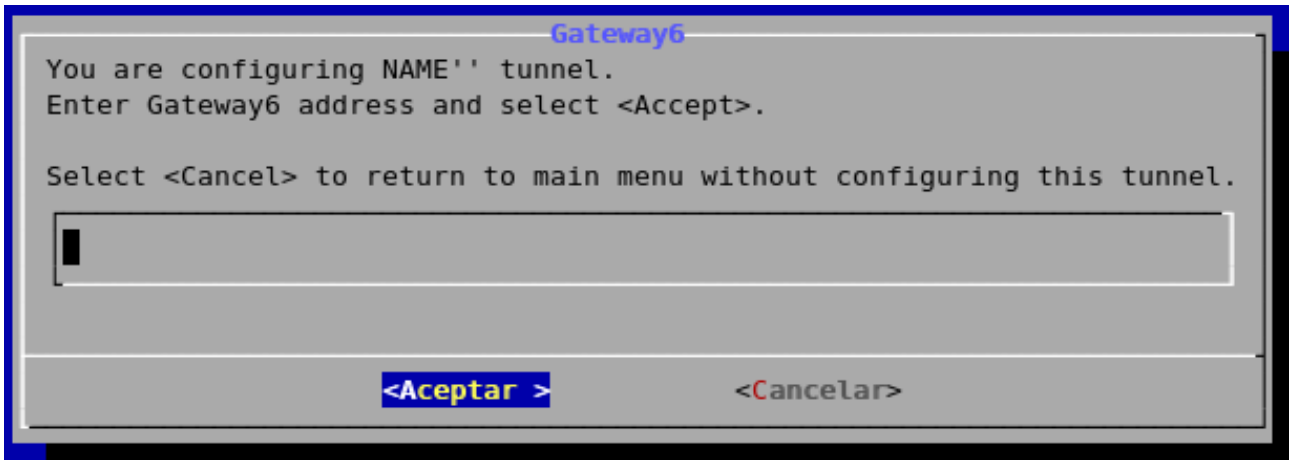
UserId



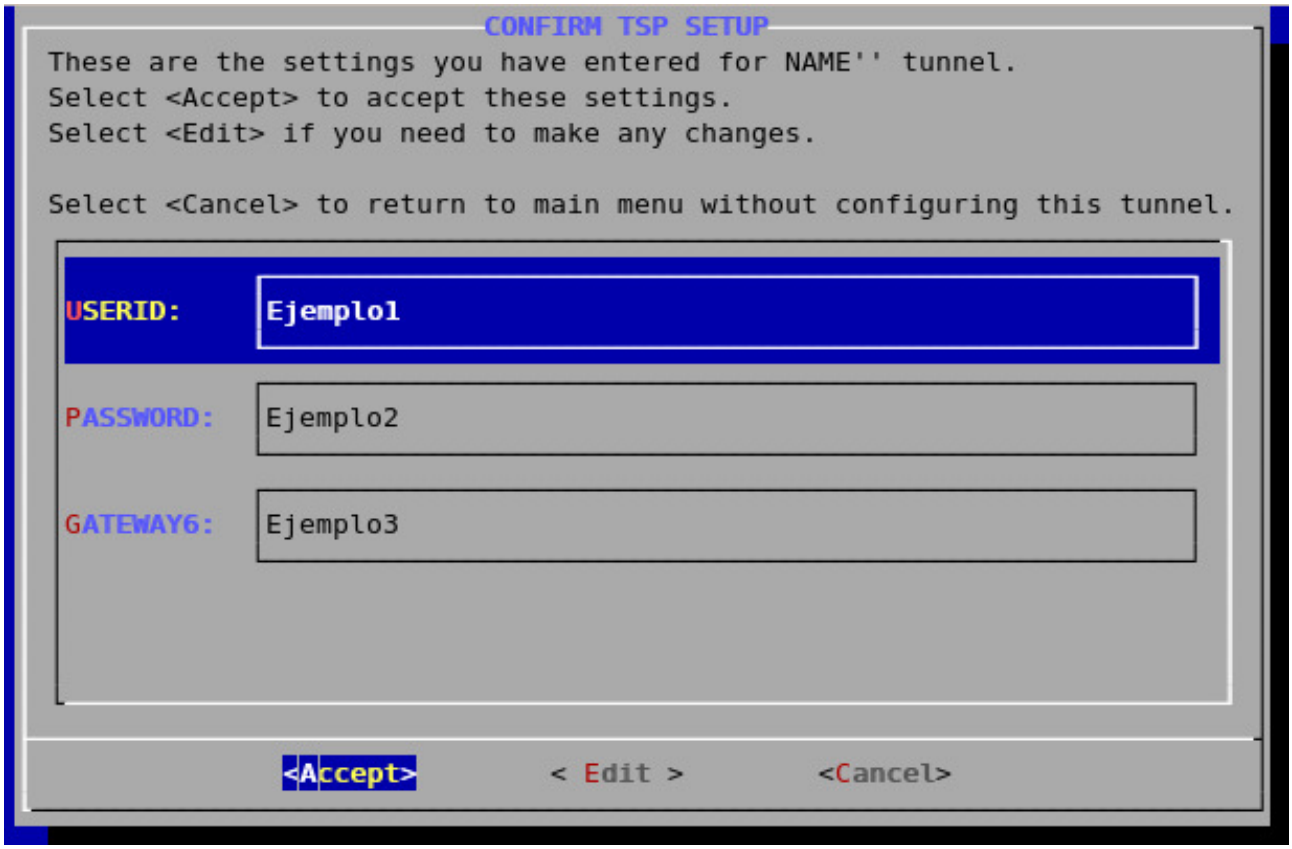
Password



Gateway6



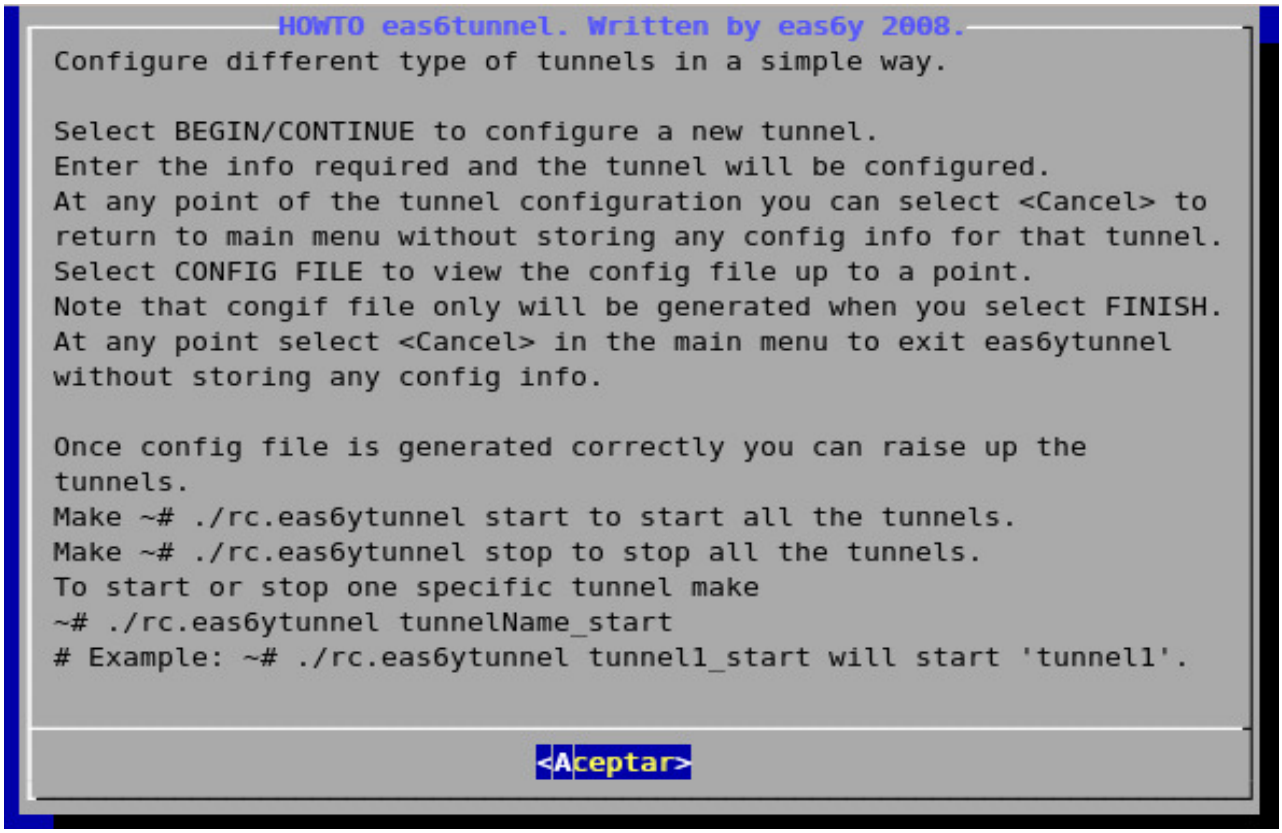
Una vez introducidos los datos de configuración, serán mostrados por pantalla. En este punto se permite la posibilidad de editar esta información (seleccionando la opcion <Edit>) por si se hubiera cometido algún error en el proceso de configuración. Vemos un ejemplo de posible configuración.



Seleccionando <Aceptar> confirmamos que esta información de configuración es correcta.

En este punto el archivo de configuración contendría la información introducida por el usuario para cada túnel configurado. Como ya hemos indicado anteriormente esta información será almacenada en el archivo de configuración */etc/rc.d/rc.eas6ytunnel.conf*.

Otra opción disponible en el menú principal nos muestra unas sencillas y útiles instrucciones para utilizar el *script* eas6ytunnel. Podemos acceder a ello seleccionando <HOWTO>.



Por último indicar que en cualquier punto del proceso de configuración el usuario podrá salir bien de la configuración de un túnel concreto o bien de la configuración general ya realizada seleccionando <Cancelar>.

La información introducida sólo será almacenada correctamente en el archivo de configuración de túneles rc.eas6ytunnel.conf cuando el usuario seleccione FINISH en el menú principal. En este punto se mostrará el contenido del archivo de configuración generado.

En este punto el usuario ya ha configurado los túneles deseados. Es el momento de levantar los túneles. Para ello hemos desarrollado el *script rc.eas6ytunnel* que accediendo a los datos de configuración almacenados en */etc/rc.d/rc.eas6ytunnel.conf* levantará los túneles en la máquina. Explicamos el *script rc.eas6ytunnel* más detalladamente en las siguientes páginas.

rc.eas6ytunnel

Este *script* accede a los datos de configuración de los túneles almacenados en el archivo */etc/rc.d/rc.eas6ytunnel.conf*. Lee los datos de configuración de los distintos túneles y procede a ejecutar las acciones necesarias para levantar los túneles indicados por el usuario.

Para levantar todos los túneles configurados haremos:
~# ./rc.eas6ytunnel start
Para levantar un túnel en particular haremos:
~# ./rc.eas6ytunnel nombreTúnel_start

A modo informativo se mostrarán por línea de comandos una descripción de las acciones realizadas al levantar los distintos túneles.

Por ejemplo:
/etc/rc.d/rc.eas6ytunnel: /sbin/ifconfig sit0 up
/etc/rc.d/rc.eas6ytunnel: /sbin/ifconfig sit0 add 2002:0102:0304::1/16
/etc/rc.d/rc.eas6ytunnel: /sbin/route -A inet6 add 2000::/3 gw ::192.88.99.1 dev sit0

Para comprobar que los túneles han sido bien levantados sólo hay que hacer:
~# ifconfig
que desplegará numerosos parámetros de los interfaces de redes existentes en la máquina.

En el caso de túneles 6to4 aparecerá el interfaz *sit0.* Por ejemplo:

```
sit0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-65-74-00-00-00-00-00-00-00-00
          inet6 addr: 2002:9360:5144::1/16 Scope:Global
          inet6 addr: ::147.96.81.68/96 Scope:Compat
          inet6 addr: ::127.0.0.1/96 Scope:Unknown
          UP RUNNING NOARP  MTU:1480  Metric:1
          RX packets:435016 errors:0 dropped:0 overruns:0 frame:0
          TX packets:870414 errors:20590 dropped:0 overruns:0 carrier:20581
          collisions:0 txqueuelen:0
          RX bytes:492112835 (469.3 MiB)  TX bytes:92503487 (88.2 MiB)
```

En el caso de túneles TSP aparecerá el interfaz *sit1* o el interfaz *tun*.

En el caso de túneles Teredo aparecerá el interfaz *teredo*.

Una vez configurados y levantados los túneles sólo queda acceder al mundo IPv6.
Así por ejemplo haremos:
~# ping6 www.euro6ix.org
y estaremos accediendo a través de nuestro túnel a la página en IPv6 www.euro6ix.org
En la consola se mostrará la información de la conexión que se está produciendo:

PING www.euro6ix.org(ns1.euro6ix.com) 56 data bytes
64 bytes from ns1.euro6ix.com: icmp_seq=1 ttl=62 time=156 ms
64 bytes from ns1.euro6ix.com: icmp_seq=2 ttl=62 time=123 ms
64 bytes from ns1.euro6ix.com: icmp_seq=3 ttl=62 time=129 ms
64 bytes from ns1.euro6ix.com: icmp_seq=4 ttl=62 time=139 ms

Para concluir la conexión pulsar *ctrl+c.* Se mostrará entonces un informe de la conexión:

--- www.euro6ix.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 123.140/137.161/156.304/12.446 ms

En caso de fallo en la conexión (bien porque el túnel no está bien configurado, bien porque la página no está disponible, etc.) se nos indicará por pantalla de las siguientes formas:

Network is unreachable,
unknown host , etc.

Si esto ocurriese hay que repasar las acciones anteriores para ver que todo está correcto y volver entonces a establecer una conexión.

Para detener un túnel ya existente sólo es necesario escribir por línea de comandos
~# ./rc.eas6ytunnel nombreTúnel_stop
o bien
~# ./rc.eas6ytunnel stop que detendrá todos los túneles existentes.

5. Trabajo futuro.

Conectividad nativa, movilidad.

Este documento sólo es una propuesta para valorar la introducción de conectividad IPv6 en la RR.DD de la Universidad Complutense utilizando técnicas de *tunneling,* sin acometer grandes cambios en la infraestructura actual. Pasado un tiempo de evaluación, juzgando el uso y valorando posibles problemas de configuración por parte del usuario, el paso siguiente sería la negociación con el actual proveedor de Internet para la Universidad, el ente RedIris, para obtener un enlace nativo.

A partir de ahí se abre un mundo de posibilidades para experimentar con IPv6: movilidad, seguridad a nivel de red, integración con otros dispositivos alternativos al computador, como teléfonos móviles, y aprovechamiento de la optimización de *multicast*, en escenarios como la video-conferencia.

IPTABLES

Para ejercer un control sobre el tráfico IPv6, podemos utilizar la conocida herramienta *IPTABLES.*

Iptables permite al administrador del sistema definir reglas acerca de qué hacer con los paquetes de red. Las reglas se agrupan en *cadenas*: cada cadena es una lista ordenada de reglas. Las cadenas se agrupan en *tablas*: cada tabla está asociada con un tipo diferente de procesamiento de paquetes.
Cada regla especifica qué paquetes la cumplen (*match*) y un *destino* que indica qué hacer con el paquete si éste cumple la regla. Cada paquete de red que llega a una computadora o que se envía desde una computadora recorre por lo menos una cadena y cada regla de esa cadena se comprueba con el paquete.

Si la regla cumple con el datagrama, el recorrido se detiene y el destino de la regla dicta lo que se debe hacer con el paquete. Si el paquete alcanza el fin de una cadena predefinida sin haberse correspondido con ninguna regla de la cadena, la *política* de destino de la cadena dicta qué hacer con el paquete. Si el paquete alcanza el fin de una cadena definida por el usuario sin haber cumplido ninguna regla de la cadena o si la cadena definida por el usuario está vacía, el recorrido continúa en la cadena que hizo la llamada (lo que se denomina *implicit target RETURN* o RETORNO de destino implícito). Solo las cadenas predefinidas tienen políticas.

En *iptables*, las reglas se agrupan en cadenas. Una cadena es un conjunto de reglas para paquetes IP, que determinan lo que se debe hacer con ellos. Cada regla puede desechar el paquete de la cadena (cortocircuito), con lo cual otras cadenas no serán consideradas. Una cadena puede contener un enlace a otra cadena: si el paquete pasa a través de esa cadena entera o si cumple una regla de destino de retorno, va a continuar en la primera cadena. No hay un límite respecto de cuán anidadas pueden estar las cadenas. Hay tres cadenas básicas (*INPUT*, *OUTPUT* y *FORWARD*: ENTRADA, SALIDA y REENVÍO) y el usuario puede crear tantas como desee.

Un trabajo futuro interesante, seria desarrollar lo expuesto anteriormente, debido a que la seguridad es una herramienta fundamental.
Iptables es una buena herramienta, pero se tiene que configurar por línea de comandos, una labor tediosa y sobretodo difícil para usuarios inexpertos. En este sentido un posible trabajo futuro sería desarrollar una herramienta similar a *Netconfig*, o a *Netconfig*6, que obtenga y almacene información de usuario que pueda ser utilizada para configurar *iptables*.

Visita al Centro de Proceso de Datos de la UCM

Como parte de nuestro proyecto solicitamos una reunión con los miembros del Centro de Proceso de Datos (CPD) de la UCM para explicar nuestro trabajo y ofrecer nuestro proyecto como posible sistema de producción, permitiendo preparar en los años siguientes la plena migración a IPv6 con el proveedor de Internet para la Universidad Complutense de Madrid. Accedieron a nuestra petición y nos reunimos en el CPD el jueves 29 de mayo.

A continuación mostramos el manual que les entregamos para trabajar de manera rápida y sencilla con nuestro proyecto.

MANUAL eas6y CPD

El siguiente manual explica de una manera rápida y sencilla las acciones necesarias para trabajar de una forma adecuada con la máquina. Se divide en tres puntos principales (RADVD, Tarjetas y Túneles) que seguidamente pasamos a explicar.

- RADVD

A continuación se dan una serie de procedimientos a seguir para poder manejar RADVD. Indicando en cada caso, los pasos necesarios para acciones básicas como su arranque y su detención. Así como los pasos a seguir para comprobar su correcto comportamiento y la ruta y contenido de sus archivos de configuración.

1. ¿Cómo arrancar RADVD?
RADVD se arranca automáticamente al iniciar el sistema mediante el *script* que se encuentra en la ruta /etc/rc.d/rc.local. Si se necesita arrancar otra vez, basta con ejecutar dicho *script* asegurándose de que no está ya en ejecución o volver a iniciar la máquina.

2. ¿Cómo parar RADVD?
RADVD se para como otro proceso cualquiera, mediante el comando ps.

3. ¿Cómo comprobar que se ha ejecutado bien RADVD?
Se puede comprobar el correcto comportamiento de RADVD utilizando el comando *Netconfig.* En cada tarjeta eth, tiene que aparecer la dirección IPv6 de dicha tarjeta, según la configuración que se ha dado a RADVD. Para nuestro caso, estas son las direcciones que tienen que aparecen en cada tarjeta:

ETH0=>inet6 addr: 2002:9360:5144:**6**:204:75ff:fef5:8fa8/64 Scope:Global
ETH1=> inet6 addr: 2002:9360:5144:**7**:204:75ff:fef5:8fa8/64 Scope:Global
ETH2=> inet6 addr: 2002:9360:5144:**8**:204:75ff:fef5:8fa8/64 Scope:Global
ETH3=> inet6 addr: 2002:9360:5144:**9**:204:75ff:fef5:8fa8/64 Scope:Global

4. ¿Cómo funciona el archivo de configuración?
El archivo de configuración se encuentra en /etc/radvd.conf y contiene la información necesaria para que se *autoconf*igure cada una de las 4 tarjetas. Para cada una de ellas, el contenido de archivo es el siguiente:

```
interface ethY
{
  AdvSendAdvert on;
  prefix 2002:9360:5144:X::0/64
  {
          AdvOnLink on;
          AdvAutonomous on;
  };
};
```

Dónde la Y es el número de cada tarjeta y la X es el número de cada subred.
Si se necesita añadir otra tarjeta, únicamente hay que elegir la Y del número de tarjeta y cualquier X que no estuviera ya utilizada. Para eliminar una tarjeta, simplemente basta con eliminar la parte del archivo dedicada a esta tarjeta.

– Tarjetas

En este manual se dan una serie de procedimientos a seguir para poder manejar y configurar las tarjetas de la máquina.

- **Cómo ver las tarjetas de la máquina**
  Para ver las tarjetas que dispone la máquina, es necesario ejecutar la instrucción o comando ifconfig y nos mostrará todas las tarjetas de la maquina.

- **Configuración de las tarjetas**

  Para configurar cada tarjeta hay que ir al archivo de configuración */etc/rc.d/rc.inet1v6.conf*, en él se podrá configurar manualmente cada tarjeta.

  CASO CONFIGURACIÓN TARJERTA i COMO ESTÁTICA

  - **IPADDR[i]->** Contiene la IP en IPv6 de la tarjeta i, sólo se rellenará en el caso de que configuremos la tarjeta i como estática.

  NETMASK[i]-> La máscara en IPv6 de la tarjeta i. Sólo se rellenará si la configuración para la tarjeta i es estática.

  CASO CONFIGURACIÓN TARJETA i COMO *AUTOCONF*(RADVD)

  - **USE_*AUTOCONF*[i]->** Sólo cuando expresamente se ponga a 'no' estará desactivada, en caso contrario será la que esté por defecto, estando así esa tarjeta configurada con *autoconf* (radvd).

  OTROS CASOS

  – **USE_DHCPv6[i]->** (No está implementado).

  – **DHCP_HOSTNAME[i]->**(No está implementado).

  Operaciones posibles

  - **start, up**: levanta todas las interfaces configuradas.
  - **Stop, down**: para todas las interfaces.
  - **restart**: levanta de nuevo las interfaces.
  - **ethi_start, ethi_up**: levanta la interfaz ethi especificada.
  - **ethi_stop, ethi_down**: para o tira la interfaz ethi.
  - **ethi_restart**: primero para ethi y luego es levantada de nuevo.

– Túneles

1. Configuración túneles

La información de configuración de los distintos túneles se almacena en el archivo de configuración */etc/rc.d/rc.eas6ytunnel.conf*

Este archivo puede ser editado manualmente o por medio del *script* desarrollado *eas6ytunnel.*

En concreto la máquina entregada ya contiene este archivo editado con la información necesaria para levantar un túnel 6to4 que nos permitirá acceder a IPv6.

En concreto el archivo de configuración proporcionado es el siguiente:

```
##########################################
Config information complu6ix tunnel:
##########################################
 TUTYPE[0]=6to4
  TUNAME[0]=complu6ix
  TUIPv4[0]= 147.96.81.68
  TUIPv6[0]= 2002:9360:5144::1
  # Note that the prefix length for a 6to4 address is 16
  TUNETMASK[0]=16
  # Note thaw we are using the all-6to4-router's IPv4 anycast address
  GATEWAY=::192.88.99.1
```

El atributo TUNAME contiene el nombre del túnel. En este caso el archivo contiene los datos de configuración de un túnel de nombre *complu6ix.*

Sólo queda levantar el túnel.

2. Levantar túneles configurados

Para levantar un túnel cuyos datos de configuración ya están almacenados sólo es necesario escribir por línea de comandos *~# ./rc.eas6ytunnel nombreTúnel_start*

En el caso de la máquina entregada levantaremos el túnel *complu6ix* haciendo:

~# ./rc.eas6ytunnel complu6ix_start

o bien

~# ./rc.eas6ytunnel start

que levantará todos los túneles configurados en rc.eas6ytunnel.conf (en definitiva *complu6ix*).

3. Comprobación túneles levantados

Para comprobar que los túneles han sido bien levantados sólo hay que hacer:

~# ifconfig

que desplegará numerosos parámetros de los interfaces de redes existentes en la máquina.

Para el caso de los túneles 6to4 nos interesa ver que contamos con el interfaz *sit0* que para el caso concreto en el que levantamos el túnel *complu6ix* tendrá los siguientes parámetros:

```
sit0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-65-74-00-00-00-00-00-00-00-00
          inet6 addr: 2002:9360:5144::1/16 Scope:Global
          inet6 addr: ::147.96.81.68/96 Scope:Compat
          inet6 addr: ::127.0.0.1/96 Scope:Unknown
          UP RUNNING NOARP  MTU:1480  Metric:1
          RX packets:435016 errors:0 dropped:0 overruns:0 frame:0
          TX packets:870414 errors:20590 dropped:0 overruns:0 carrier:20581
          collisions:0 txqueuelen:0
          RX bytes:492112835 (469.3 MiB)  TX bytes:92503487 (88.2 MiB)
```

Vemos que el interfaz *sit0* muestra la dirección IPv4 147.96.81.68 y la dirección IPv6 2002:9360:5144::1 lo que nos indica que ha sido levantado con los parámetros del túnel *complu6ix*.
La información que se muestra en último término (RX packets, TX packets) mostrará los paquetes enviados, recibidos, perdidos, etc. por lo que irá variando en función de las conexiones que se hagan con la máquina.

4. Acceder a IPv6

Una vez configurados y levantados los túneles sólo queda acceder al mundo IPv6.
Sólo hay que hacer *ping6* (recordamos que con *ping* accedemos a IPv4 y con *ping6* a IPv6).
Así por ejemplo haremos:
~# ping6 www.euro6ix.org
y estaremos accediendo a través de nuestro túnel a la página en IPv6 www.euro6ix.org
En la consola se mostrará la información de la conexión que se está produciendo:

```
PING www.euro6ix.org(ns1.euro6ix.com) 56 data bytes
64 bytes from ns1.euro6ix.com: icmp_seq=1 ttl=62 time=156 ms
64 bytes from ns1.euro6ix.com: icmp_seq=2 ttl=62 time=123 ms
64 bytes from ns1.euro6ix.com: icmp_seq=3 ttl=62 time=129 ms
```

Para concluir la conexión pulsar *ctrl+c.* Se mostrará entonces un informe de la conexión:

```
--- www.euro6ix.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 123.140/137.161/156.304/12.446 ms
```

En caso de fallo en la conexión (bien porque el túnel no está bien configurado, bien porque la página no está disponible, etc.) se nos indicará por pantalla de las siguientes formas:
Network is unreachable,
unknown host , etc.
Si esto ocurriese hay que repasar los pasos anteriores para ver que todo está correcto y volver entonces a establecer una conexión.

5. Eliminar túneles existentes

Para eliminar un túnel ya existente sólo es necesario escribir por línea de comandos
~# ./rc.eas6ytunnel nombreTúnel_stop
En el caso de la máquina entregada al CPD eliminaremos el túnel *complu6ix* haciendo:
~# ./rc.eas6ytunnel complu6ix_stop
o bien
~# ./rc.eas6ytunnel stop
que eliminará todos los túneles existentes (en definitiva *complu6ix*).
Una vez eliminado podríamos volver a levantarlo volviendo al punto 2.

De forma más resumida mostramos un manual rápido para los túneles de la máquina.

1- *~# ./rc.eas6ytunnel start* -----> (levantar túneles)

2- *~# ifconfig* -----> (comprobar que han sido levantado correctamente)

3- *~# ping6 ...* -----> (acceder al mundo IPv6)

4-*~# ./rc.eas6ytunnel stop* ------> (detener túneles)

6. Bibliografía

[1] IPv6 core protocols implementation / Qing Li, Tatuya Jinmei, Keiichi Shima

San Francisco : Morgan Kaufmann, cop. 2007

[2] IPv6 in practice : a Unixer's guide to the next generation Internet / Benedikt Stockebrand.

New York : Springer, 2006

[3] Understanding IPv6 / Joseph Davies

Redmond : Microsoft Press, cop. 2003

[4] IPv6 networks / Marcus Goncalves, Kitty Niles

New York [etc.] : McGraw-Hill, cop. 1998

[5] ”Application Performance Analysis In Transition Mechanism From Ipv4 to IPv6”, *Ettikan Kandasamy Karuppiah, Gopi Kurup, Takefumi Yamazaki, Proceedings APAN Conf. 2000, Tsukuba, Japan, Feb. 14-18, 2000.*

[6] “Transition Mechanism Between IPv4 & IPv6 and deciding your choice”, *Ettikan Kandasamy, Tong Hui Tee, Seow Chen Yong, APAN Conf. 2002, Jan. 22-25.*

[7] "IPv4 Address Lifetime Expectations in IPng: Internet Protocol Next Generation" *, S. Bradner, A. Mankin, ed. Adddison Wesley, 1996.*

[8] “The Recommendation for the IPng Protocol”, *S. Bradner, A. Mankin, RFC 1752.*

[9] “IP Versión 6 Addressing Architecture”, *Hinden, R. and S. Deering, RFC 1884*.

[10] “Get IPv6 Now with Freenet6 “,*Peter Todd, Linux Journal , 2003.*

[11] “Security Architecture for the Internet Protocol”, *R. Atkinson, RFC 1825.*

[12] “IP Authentication Header”, *R. Atkinson, RFC 1826.*

[13] “Format for Literal IPv6 Addresses in URL's”, *R. Hinden, B. Carpenter, RFC 2732.*

[14] “Transition Mechanisms for Ipv6 Hosts and *Router*s ”, *R. Gilligan, E.Nordmark, RFC 2893*.

[15] “Network Address Translation – Protocol Translation (NAT-PT)” *G. Tsirtsis, P. Srisuresh.*

[16] “IPv6 Tunnel Broker”, *A. Durand, P. Fasano, I. Guardini, D. Lento, RFC 3053.*

[17] “Internet Protocol Version 6 (Ipv6) Specification”, *S. Deering, R. Hinden, RFC 2460.*

[18] “Connection of IPv6 Domains via IPv4 Clouds”, *B. Carpenter, K. Moore, RFC 3056.*

Además de las siguientes páginas web:

www.ipv6.org

www.go6.net

www.euro6ix.org

www.remlab.net/miredo/

www.linuxhq.com/IPv6/

www.wikipedia.org