CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN DE COMPUTACIÓN

IPv6 Interoperabilidad y robustez

Tesis que presenta
Axel Ernesto Moreno Cervantes

para obtener el grado de
Maestro en Ciencias

en la especialidad de
Ingeniería Eléctrica

con opción en Computación

Director de tesis:
Guillermo Benito Morales-Luna

México, D.F. Julio 2004

Resumen

En esta investigación se presenta la evaluación del Protocolo de Internet versión 6 (IPv6) en relación a su interoperabilidad con plataformas tales como Windows XP, Solaris 9, Linux Red Hat (versiones 8.0 y 9.0) y Mac OS X 10.2, 10.3, también con protocolos tales como el Protocolo Ligero de Acceso a Directorio (LDAP) y el Protocolo de Seguridad IP (IPSec), y sus mecanismos de transición como son los Túneles *6to4*, el doble *Stack* y el *Tunnel-broker*. Además se proporciona una guía a través de ejemplos para configurar IPv6 sobre las plataformas Windows XP, Solaris (versiones 8.0 y 9.0), Linux RedHat (versiones 8.0 y 9.0) y Mac OS X (versión 10.2 "Jaguar") y sobre los enrutadores CISCO 2620 XM.

Asimismo se da una guía para configurar IPv6 sobre los servidores: DNS BIND 9, FTP WSFTP, OpenLDAP y Apache. Para finalizar se presenta la configuración de una Red Privada Virtual sobre IPv6 utilizando los servicios de seguridad *Encabezado de Autenticación* (AH) y *Encapsulación de Seguridad de la Carga* (ESP) provistos por IPv6 que permiten el aseguramiento de la información.

Las principales contribuciones de esta investigación son las siguientes: El desarrollo de un *benchmark* para medir el desempeño de un servidor LDAP, tomando como métricas la latencia y el rendimiento del mismo al utilizar como protocolos de comunicación IPv4 e IPv6. También desarrollamos un modelo matemático capaz de reproducir el comportamiento del mismo servidor LDAP, basándonos en los resultados obtenidos en el *benchmark* desarrollado anteriormente. Además realizamos pruebas de interoperabilidad de IPv6 con la implementación de IPSec en el sistema operativo IOS (versión 12.2(11)T3) de los enrutadores CISCO 2620 XM.

Abstract

The aim of this research is to introduce the Internet Protocol version 6 (IPv6) evaluation related to its interoperability with platforms as Windows XP, Solaris 9, Linux Red Hat 8,9 and Mac OS X 10.2, 10.3, also with other protocols as the Lightweight Directory Access Protocol (LDAP) and the Internet Security Protocol (IPSec), as well as its transition mechanisms such as 6to4 Tunnels, the Double Stack and the Tunnel-Broker. Also we give a guide to provide IPv6 on Windows XP, Solaris 8,9, Linux RedHat 8,9 and MacOSX 10.2 "Jaguar" platforms together with the routers CISCO 2600 series. All procedures are introduced through examples.

Likewise, the procedures for configuring IPv6 on the DNS BIND 9, FTP WSFTP, OpenLDAP and Apache servers are supplied. The configuration of a Private Virtual Network on IPv6 using AH and ESP safety services provided for IPv6, is presented.

The main contributions in this research are the development of a benchmark to measure the performance of a LDAP server, having as metrics its latency and throughput, while operations are carried out over IPv4 and IPv6. Also we develop a mathematical model capable of reproducing the behavior of the LDAP server taking as base the results obtained in the benchmark developed previously, besides we made a series of IPv6 interoperability tests with the IPSec implementation on the IOS 12.2(11)T3 operating system for the CISCO 2620 XM routers.

Agradecimientos

En primer lugar agradezco a CONACYT por el apoyo económico brindado durante el programa de maestría.

Agradezco a mi mamá por ser mi todo.

Agradezco a mi familia por mantenerme y soportar mi mal humor durante los días de estrés.

A Alicia por ser mi adoración.

Agradezco a mi asesor por la paciencia y el interés mostrado durante el desarrollo de mi tesis, así como por poner a mi disposición el equipo necesario para el desarrollo de esta investigación.

A los profesores Luis Gerardo de la Fraga y Francisco José Rodríguez Henríquez, así como a Emilio Espinosa por sacarme de apuros y resolver mis dudas.

A Sofía Reza Cruz por su ayuda incondicional.

Al Dr. Arturo Díaz Pérez, Adrián Ramírez Cabrera y Andrés Huerta García por las facilidades otorgadas y su valioso apoyo.

A Valerio Torres Tovar, Gustavo Becerril y al Dr. José Antonio Moreno Cadenas por brindarme su apoyo aún sin conocerme.

A Graciela Meza C., Raúl Montaño M. y David Torres F. por sus atenciones brindadas en todas las ocasiones que acudí a la Biblioteca de Ingeniería Eléctrica.

A todos los profesores y compañeros de la Sección de Computación por su amistad durante más de dos años.

Índice general

Glosario

Apéndices

Relación de tablas y figuras

Pág.

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Apéndice A

Apéndice B

Apéndice E

Capítulo 1

Introducción

1.1 Motivación y antecedentes

La presente investigación está motivada por una serie de deficiencias que se han observado en el diseño de IPv4, las cuales se describen a continuación.

El creciente número de máquinas conectadas al Internet, en conjunto con la deficiente manera en que las direcciones IP son asignadas, han provocado una pronta escasez en el espacio de direcciones disponibles del protocolo de red actual (IPv4). La asignación de direcciones mediante CIDR, así como la Traducción de Direcciones de Red (NAT) han intentado resolver este problema sin obtener mucho éxito, ya que en primera instancia no todos los enrutadores soportan CIDR y además hay muchas aplicaciones que requieren utilizar direcciones estáticas, por lo que NAT no ayuda en gran medida en estos casos. Otro aspecto a destacar es la seguridad misma que ha tomado un papel muy importante en el manejo de la información que es enviada a través de los medios electrónicos. IPv4 no fue diseñado para ser un protocolo seguro y muchas de las aplicaciones creadas para solucionar este problema de seguridad solo protegen la información en las capas de comunicación más altas (Aplicación y Transporte) haciendo vulnerable la información a ataques en la capa de Red. IPv6 por su parte se presenta como la solución más viable a todos los problemas anteriormente mencionados porque incluye entre otras mejoras, direcciones IP de 128 bits, en vez de las direcciones de 32 bits de IPv4 y seguridad sobre la capa de Red.

Aún con todas las mejoras en IPv6 se tiene que seguir utilizando IPv4, porque IPv6 aún se encuentra en su fase de pruebas. Las implementaciones existentes de este protocolo aún presentan problemas de interoperabilidad con algunos otros protocolos de capas superiores, cada plataforma tiene su propia implementación, por lo que se tienen problemas de incompatibilidad entre plataformas y muchos protocolos aún no han sido probados sobre IPv6 el cual es el punto principal de nuestro trabajo de tesis.

Aún hay protocolos que no han sido estudiados sobre IPv6, tal es el caso del Protocolo Ligero de Acceso a Directorio (LDAP) y el protocolo de seguridad IP (IPSec). LDAP ha ganado mucha popularidad sobre las bases de datos en un número creciente de aplicaciones como son:

- Base de datos de personal
- Almacenamiento de reglas de autenticación de usuarios en escenarios de seguridad, donde se administran certificados digitales
- Base de datos para almacenamiento de configuraciones de red
- Base de datos de traducción de direcciones IP para telefonía IP
- Almacenamiento de reglas de políticas de servicio

La base de la popularidad de LDAP radica en su simplicidad de implementación y uso, así como en la velocidad de acceso a la información en comparación con una base de datos. IPSec por su parte viene incluido obligatoriamente en el estándar IPv6 para proveer integridad, autenticación y confidencialidad de la información en el nivel de capa de Red.

Existen numerosos grupos de trabajo que contribuyen en el desarrollo de IPv6, cada uno de ellos enfocado en una plataforma en particular como es el caso de USAGI, el cual se encarga de desarrollar implementaciones de IPv6 para la plataforma Linux o MSR-IPv6 cuyas investigaciones se enfocan en las implementaciones de IPv6 sobre Windows. También hay grupos de trabajo encargados de realizar pruebas de interoperabilidad de IPv6 como es el caso de KAME.

Existen varios grupos de trabajo dedicados a estandarizar IPv6 además de publicaciones que muestran evaluaciones de este protocolo en conjunto con otros protocolos de la capa de transporte [1-3], sin embargo hay muchos protocolos que corren sobre IPv4/IPv6 en los que no se ha puesto mucha atención y que día con día toman más importancia, como es el caso de LDAP e IPSec. Nuestro trabajo trata de contribuir en el desarrollo de IPv6 enfocándonos precisamente en estos dos protocolos y su interoperabilidad con IPv6.

1.2 Principales objetivos de este trabajo

El objetivo general de esta investigación fue realizar un estudio comparativo de IPv6 vs. IPv4 para observar los beneficios reales obtenidos al migrar al protocolo de la nueva generación.

Los objetivos específicos fueron los siguientes: configurar el protocolo IPv6 en las plataformas Windows XP, Linux Red Hat 9, Solaris 9 y Mac OS X. Una vez hecho esto construir una *intranet* típica montada en IPv6. Estos primeros dos objetivos se realizaron debido a la necesidad de evaluar a IPv6 en un ambiente heterogéneo ya que es así como están conformadas la gran mayoría de las redes. La elección de las plataformas se basó en que estas son muy comúnmente utilizadas. Por último se implementaron los protocolos LDAP e IPSec en la red de prueba y se intentó diseñar un *benchmark* para medir el rendimiento de IPv6 utilizando el protocolo LDAP. También se realizaron algunas pruebas de interoperabilidad entre implementaciones de IPSec sobre IPv6.

1.3 Organización de la tesis

El capítulo 2 contiene los conceptos básicos sobre IPv6 y la descripción de cada uno de los campos que forman el encabezado, así mismo se explica la arquitectura de las direcciones IPv6 y la introducción a los servicios de seguridad incluidos en él. En el capítulo 3 mostramos la arquitectura general de la red de pruebas y se describen las primeras pruebas realizadas para comprobar el buen funcionamiento de IPv6 en la red.

Es importante destacar que en los capítulos 4 y 5 describimos las aportaciones a la presente investigación a través del diseño de un *benckmark,* un modelo matemático que reproduce el comportamiento de un servidor LDAP en cuanto a su rendimiento y algunas pruebas de interoperabilidad con algunas implementaciones de IPSec sobre IPv6.

En el capítulo 4 presentamos los conceptos de LDAP y los servicios de directorio, así como el desarrollo de un *benchmark* que se realizó con la finalidad de evaluar que tanto repercute el uso de IPv6 en el rendimiento de un servidor LDAP. En el capítulo 5 describimos detalladamente la arquitectura de IPSec y mostramos un pequeño estudio de interoperabilidad de la implementación de IPv6 en los enrutadores CISCO 2620 XM con su implementación de IPSec. Por último, el capítulo 6 expresamos nuestras conclusiones de esta investigación así como el trabajo futuro sobre esta misma línea de investigación.

.

Capítulo 2

Antecedentes de IPv6

Debido al crecimiento explosivo en el uso de Internet nos ha llevado a aprender nuevas tecnologías, así como palabras en nuestro vocabulario como es el IP, es el protocolo encargado de transportar paquetes de información de una máquina a otra sobre la capa de Red su versión actual es la cuatro (IPv4).

IPv4 está lleno de muchas carencias en su diseño. Como ejemplo, el *American Registry for Internet Number* (ARIN) [4]establece que el total de las direcciones de la clase A, el 62% de la clase B y el 37% de las direcciones de la clase C ya han sido asignadas desde 1996. La firma *Network Wizards* [5] realiza una revisión periódica de ambos, *hosts* y dominios conectados a Internet y detectó que el crecimiento del número de estos, se ha dado a una taza exponencial, por lo que el IETF NGTrans [6] ha previsto que las direcciones IPv4 serán agotadas aproximadamente entre los años 2005 y 2011 [7], Es por eso que surgen algunas interrogantes de como solucionar los problemas que trae consigo un protocolo mal diseñado como es el caso de IPv4 y como hacerlo sin tener que cambiar toda la infraestructura actual. Una posible solución que el IETF ha propuesto es el Protocolo de Internet versión 6 (IPv6), que además de solucionar el problema del agotamiento de direcciones IPv4, también cuenta con nuevas características como es la seguridad incorporada por medio de dos nuevos campos en el encabezado del paquete IP, reducción de las tablas de enrutamiento ya que se redujo el número de campos en el encabezado del paquete IP y muchos otros se hicieron opcionales. Calidad de servicio (QoS), por medio de etiquetado de paquetes, además se cuenta con otras características adicionales que trataremos más detalladamente en la sección 1.3. En este capítulo hablaremos primeramente de la historia de IPv6, después trataremos las mejoras que IPv6 tiene sobre IPv4. Describiremos también el encabezado de un paquete IPv6 y cada uno de sus campos. Por último, hablaremos de la estructura de direcciones IPv6.

2.1 Surgimiento de IPv6

A principios de la década de los 90's, cuando 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 además adicionar a este nuevo protocolo algunas características que no se contemplaron en el diseño de IPv4. En noviembre de 1992 surgió una nueva área de investigación llamada Internet *Protocol Next Generation* (IPng) comisionada por el IETF para formalmente estudiar las diferentes propuestas para el desarrollo de este nuevo protocolo.

En diciembre de 1993 fue distribuido el RFC 1550 [8], 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. Veintiún respuestas fueron recibidas, las cuales contenían puntos de vista de diferentes tipos de industrias, tales como la industria celular, televisión por cable y seguridad, solo por mencionar algunas. En el RFC 1726 [9], el grupo de investigación IPng definió un conjunto de 17 criterios que serían usados para el proceso de evaluación del IPng y eran 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 debían permitir utilizar muchas topologías distintas 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: El servicio de red junto con los protocolos de control y enrutamiento para IPng deberían ser suficientemente 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: Este nuevo protocolo debe de trabajar a través de una Internet con diferentes medios LAN, WAN y MAN, así como distintas velocidades de conexión, que van desde algunos bits/segundo hasta cientos de giga bits/segundo.

- Servicio de datagramas no confiables: En nuevo protocolo debía soportar un servicio no confiable de entrega de datagramas.

- 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 host y enrutadores.

- Operación segura: IPng también debía proveer una capa de red segura (IPSec).

- Acceso y documentación: Los protocolos que definen a IPng, sus protocolos asociados y protocolos de enrutamiento deberían ser publicados en los RFC's, así como estar disponibles libremente y no requerir licencia para su implementación.

- Nombrado único: IPng debía asignar a todos los objetos de la capa IP de manera global nombres de Internet únicos.

- Multicast: IPng debía soportar transmisión de paquetes Unicast y Multicast.

- Extensibilidad: IPng debía ser capaz de evolucionar para cubrir las necesidades futuras del Internet. Así mismo, conforme este evolucione, debería permitir diferentes versiones de él, que puedan coexistir sobre la misma red.

- Servicio de red: IPng debía permitirle a la red asociar paquetes con clases de servicio en particular y proveerlas con los servicios especificados por esas clases.

- 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 ó no basadas en IP.

En base a este criterio varias propuestas fueron revisadas y en enero de 1995, el RFC 1752 [10] fue publicado. En él se resumían las evaluaciones hechas a 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 solo 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 a SIPP, incorporarle direcciones de 128 bits de longitud y hacer algunas otras modificaciones. El resultado final a todas estas modificaciones es lo que se conoce actualmente como IPv6 ó IPng.

2.2 Mejoras en IPv6

La primera y más importante mejora que IPv6 presenta frente a IPv4 es el incremento en el tamaño de las direcciones que este maneja, es decir 128 bits de IPv6 vs. 32 de IPv4, esto le permite soportar más niveles de direccionamiento jerárquico y tener muchos más nodos direccionables. La siguiente mejora que se ha tomado en cuenta en el diseño de IPv6 es la simplificación del formato del encabezado, ya que se eliminaron y en otros casos se hicieron opcionales algunos de los campos del encabezado, ganando con esto la reducción de las tablas de enrutamiento y mejorando el rendimiento de los enrutadores al ocupar menos tiempo analizando los campos de los encabezados, por lo tanto, con Ipv6 se mantienen bajos los costos del ancho de banda a pesar de que se cuadriplicaron las direcciones.

La tercera mejora es el soporte para extensiones y opciones en el encabezado, debido a la manera en que los campos del encabezado de IPv6 están organizados y al uso de encabezados opcionales, con esto es posible incrementar la flexibilidad del protocolo IP añadiendo nuevas características en un futuro, sin que sea necesario rediseñar por completo toda la estructura del paquete IP. La cuarta mejora 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. La quinta mejora es la autenticación y privacidad gracias a dos encabezados opcionales que en conjunto nos brindan autenticación, integridad de datos y confidencialidad.

2.3 Encabezado del paquete IPv6

El encabezado de un paquete IPv6 consta de dos partes: Un encabezado IPv6 base y una extensión de encabezados opcionales, tal y como puede verse en la figura 2.1.

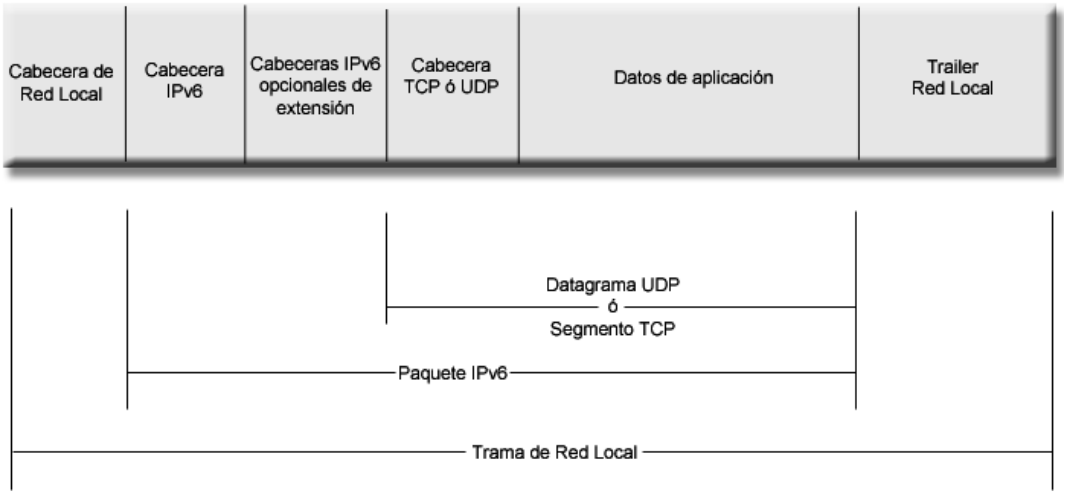


Figura 2.1 Formato de una trama IP

Un paquete IPv6 puede tener cero, uno o más encabezados opcionales. Cada encabezado opcional tiene una longitud múltiplo de 8 bits y deben ser colocadas en el orden mostrado en la tabla 2.1 por cuestiones de rendimiento:

Orden	Cabecera	Características
1	Cabecera IPv6	Encabezado del paquete IPv6
2	Cabecera de opciones *Hop by Hop*	Contiene información opcional que debe ser analizada por cada nodo a lo largo de la ruta del paquete
3	Cabecera de opciones de Destino	Contiene opciones que serán procesadas por el primer destino que aparezca en el campo de dirección destino, más destinos subsecuentes que aparezcan listados en la cabecera de enrutamiento
4	Cabecera de Enrutamiento	Este encabezado opcional lista todos los nodos intermedios que deben ser visitados por el paquete en su trayecto hacia el nodo destino
5	Cabecera de Fragmentación	Es utilizada por un emisor IPv6 para enviar un paquete que es más grande que la Unidad Máxima de Transmisión (MTU) más pequeño de los nodos intermedios hacia el destino
6	Cabecera de Autenticación	Provee integridad de datos y autenticación del origen de los datagramas IP, con esto se logra tener protección contra reenvío de paquetes
7	Cabecera de Encapsulación de seguridad de la carga	Está diseñado para proveer confidencialidad, autenticación del origen de los datos, integridad sin conexión y servicio anti-reenvío
8	Cabecera de opciones de Destino (2)	Contiene opciones que serán procesadas solamente por el destino final
9	Cabecera de protocolos de capas superiores	Encabezados de protocolos de transporte tales como TCP ó UDP deben ir aquí

Tabla 2.1 Orden en el que deben ser colocados los encabezados de extensión IPv6

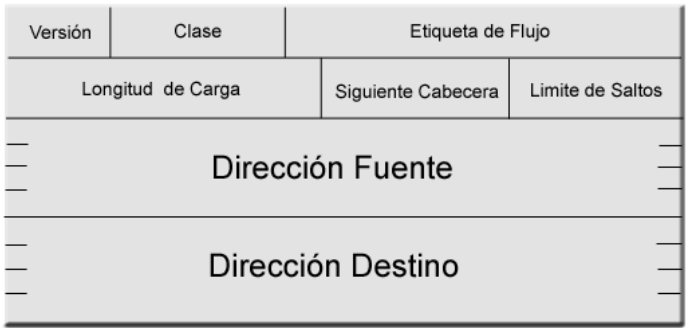


Figura 2.2. Encabezado IPv6

El encabezado IPv6 tiene una longitud de 40 octetos y tal como se ve en la figura 2.2 consta de los siguientes campos:

- Versión: Con 4 bits de longitud, este campo sirve para identificar la versión del protocolo IP, es decir IPv4 ó IPv6.

- Clase: Con 8 bits de longitud, este campo está diseñado para que enrutadores de envío y nodos originadotes de paquetes identifiquen y distingan entre diferentes clases ó prioridades de paquetes IPv6.

- Etiquetado de Flujos: Con 20 bits de longitud, este campo puede ser utilizado por un huésped para solicitar un trato especial a ciertos paquetes, tales como aquellos que requieran una calidad de servicio no por defecto ó una calidad de servicio de tiempo- real.

- Longitud de Carga: Con 16 bits de longitud, este campo se encarga de medir, dado en octetos, la longitud de la carga del paquete, la cual consta de todo lo que sigue después del encabezado IPv6, incluyendo los encabezados opcionales y protocolos de nivel superior, tales como TCP, FTP, etc. Este campo es similar al llamado *Longitud Total* en IPv4, pero a diferencia de este, Longitud de carga solo mide los datos después del encabezado, mientras que *Longitud Total* mide los datos y el encabezado.

- Siguiente Encabezado (*next header*): Con 8 bits de longitud, sirve para identificar al encabezado que sigue inmediatamente después del encabezado IPv6. La tabla 2.2 muestra los valores posibles que puede tomar este campo. Un paquete IPv6 además puede incluir cero, uno o más encabezados opcionales, por lo que dependiendo del número de encabezados que se contengan será el número de siguientes encabezados más uno extra que también deberá incluir. La figura 2.2 muestra un ejemplo en donde en primer instancia se puede ver un paquete IPv6 sin encabezados opcionales, ya que el siguiente encabezado después del encabezado IPv6 es el de los protocolos de l capa superior (en este caso TCP). En el siguiente caso, se muestra un paquete IPv6 al cual se agregó un encabezado de enrutamiento como encabezado opcional y en el tercer ejemplo podemos apreciar un paquete IPv6 al cual se le agregaron dos encabezados opcionales: Encabezado de Enrutamiento y Encabezado de Fragmentación.

- Limite de Saltos: Con 8 bits de longitud, este campo es análogo al campo *time to live* (TTL) en IPv4, pero a diferencia de este expresa el número de saltos y no de segundos que un paquete puede permanecer en la red antes de ser destruido. Cada nodo que reenvíe el paquete decrementa este número en uno.

- Dirección fuente: Con 128 bits de longitud, este campo contiene la dirección IPv6 del nodo que originó el paquete.

- Dirección destino: Con 128 bits de longitud, este campo contiene la dirección IPv6 del nodo que se espera sea el destino final del paquete. Las palabras “se espera” son utilizadas ya que la dirección destino puede no ser el último destino del paquete si está presente un encabezado de enrutamiento.

	Cabecera
0	Opciones Hop-by-Hop
1	ICMPV4
4	IP en IP (encapsulación)
6	TCP
17	UDP
43	Enrutamiento
44	Fragmento
50	Encapsulado de seguridad de la carga (ESP)
51	Autenticación (AH)
58	ICMPV6
59	Ninguno (no hay siguiente cabecera)
60	Opciones de Destino

Tabla 2.2. Posibles valores del campo Siguiente Encabezado

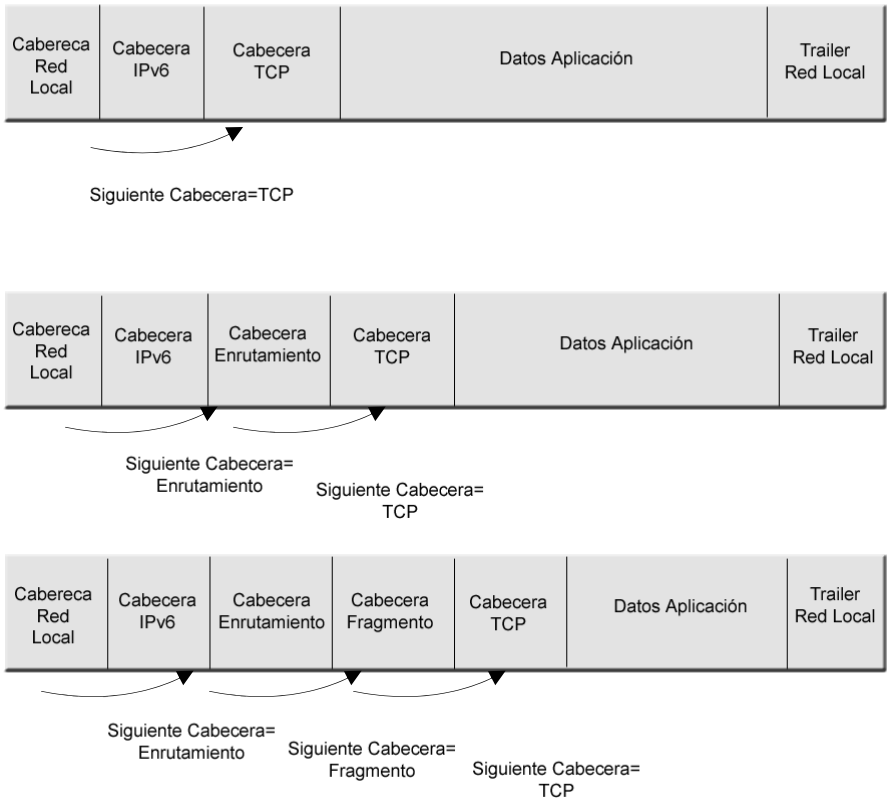


Figura 2.3. Encabezados de Extensión de IPv6

2.4 Estructura de direcciones IPv6

De acuerdo como está definido en el RFC 1884 [11] existen tres tipos de direcciones IPv6 y son los siguientes:

- Unicast: Definidas en el RFC 1887 [12] identifican una sola interfaz. Cuando un paquete es enviado a una dirección unicast, este 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, ó 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.

    La figura 4 presenta la arquitectura de las direcciones unicast, así como un ejemplo de ellas.

- 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 [11] 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.

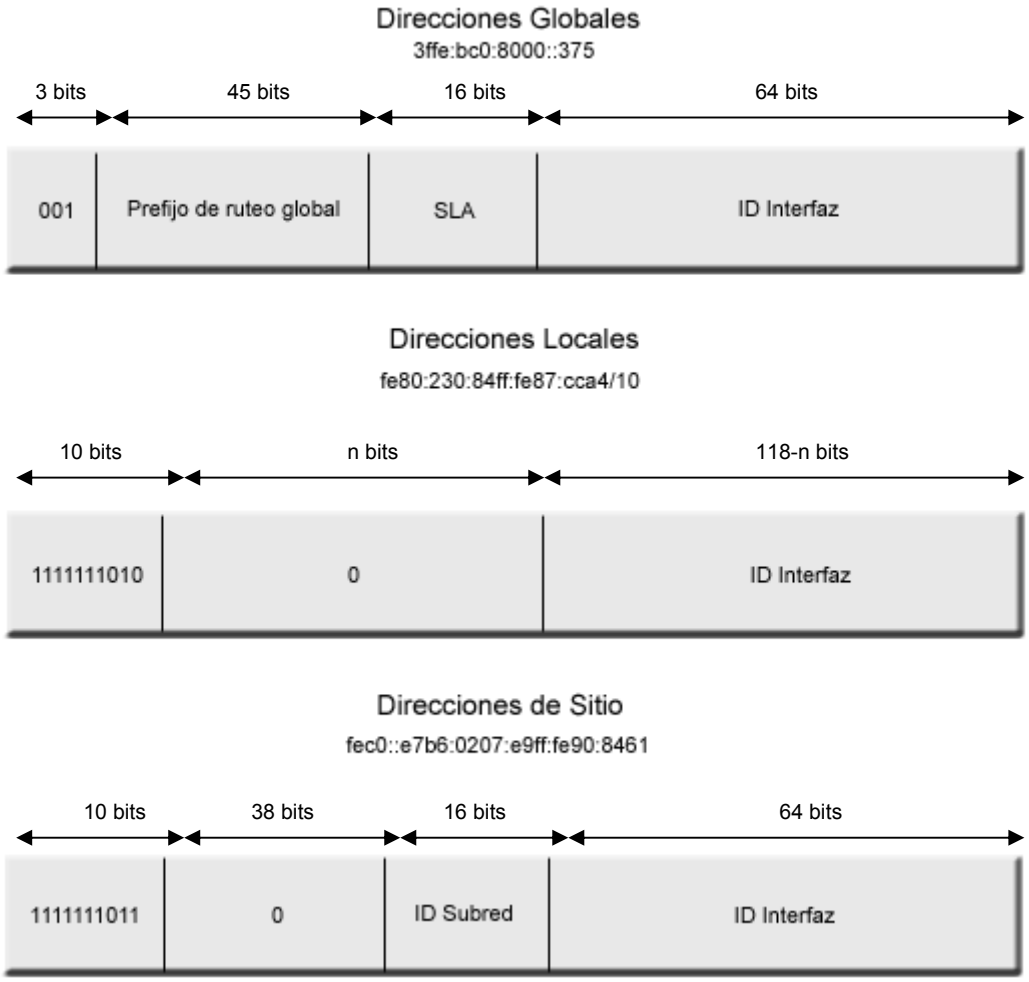


Figura 2.4. Arquitectura de las Direcciones Unicast

Existen también algunos prefijos ya establecidos para las direcciones Unicast como los que se muestran a continuación:

- 2001::/16 utilizado en redes de producción y en Internet6
- 2002::/16 utilizado para configurar túneles 6to4
- 3FFE::/16 utilizado en el *backbone* de pruebas 6Bone

En esta investigación se utilizaron los prefijos 2002 para realizar túneles *6to4* entre enrutadores y evaluar el comportamiento de IPSec sobre IPv6, así como el prefijo 3FFE para realizar túneles mediante la arquitectura *tunnel-broker* con Freenet6 [13] y evaluar el comportamiento de LDAP sobre IPv6.

2.5 Seguridad en IPv6

Los diseñadores de IPv4 en un principio no tuvieron que preocuparse por implementar mecanismos de seguridad en IP ya que el Internet era utilizado por muy pocas personas; sin embargo, en la actualidad es uno de los medios más recurridos para realizar no solo transacciones monetarias, sino también manejo 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 como los presentados en la tabla 3. IPv6 resuelve estos problemas de vulnerabilidad de la información incorporando los servicios de seguridad de IPSec (*Internet Protocol Security*) definido en el RFC 1825 [14] mediante dos encabezados de extensión:

- Encabezado de Autenticación (AH): El Encabezado de Autenticación como lo define el RFC 1826 [15] provee integridad de datos y autenticación del origen de los datagramas IP, con esto se logra tener protección contra reenvío de paquetes.

- Encapsulado de seguridad de la carga (ESP): ESP, definido por el RFC 1827 [16] está diseñado para proveer confidencialidad, autenticación del origen de los datos, integridad sin conexión y servicio contra reenvío de paquetes.

Protocolo	Ataque
ICMP	Inundación de paquetes ICMP
ICMP	Ping de la muerte
IP	Denegación de servicio por envío de fragmentos de paquetes IP
IP	*IP spoofing*
IP	*IP Timestamp*
IP	*IP Address Sweep Scan*
IP	*IP Source Route*

Tabla 2.3. Ataques más frecuentes en la capa de Red

2.5.1 Encabezado de Autenticación

El Encabezado de Autenticación provee, como ya se mencionó antes, integridad y autenticación del origen de los datos para datagramas IP, además de proveer protección contra ataques de re-envío de paquetes. La presencia del Encabezado de Autenticación es identificada por un valor de 52 en el campo Siguiente Encabezado y su encabezado tiene el formato mostrado el la figura 2.5

Figura 2.5. Formato del Encabezado de Autenticación

Como vemos en la figura 2.4, el encabezado de Autenticación consta de seis campos, los cuales describimos a continuación:

- Siguiente Encabezado: Tiene 8 bits de longitud e identifica el encabezado que sigue inmediatamente después del Encabezado de Autenticación.

- Longitud de Carga: Tiene 8 bits de longitud y provee la longitud del campo de autenticación en palabras de 32 bits, menos dos (los primeros 64 bits del Encabezado de Autenticación no son contados). El valor mínimo que puede tomar este campo es 1 el cual equivale a 3 palabras de 32 bits y es solamente utilizado para propósitos de depuración.

- Reservado: Tiene una longitud de 16 bits y está reservado para uso futuro. Es inicializado con un valor de cero.

- Índice de Parámetro de Seguridad: Tiene 32 bits de longitud e identifica la Asociación de Seguridad aplicada para este datagrama.

- Número de Secuencia: Contiene un número de 32 bits de longitud, el cual es incrementado monotónicamente. Los contadores tanto del emisor, como del receptor son inicializados a cero cuando una Asociación de Seguridad es establecida.

- Datos de Autenticación: Es un campo de longitud variable que contiene el Valor de Chequeo de Integridad (ICV o checksum) para este paquete. Este campo debe ser un múltiplo de 32 bits en longitud.

Para el cálculo del *checksum* criptográficamente seguro (también conocido como mensaje digerido o *hash*), así como para la selección de los campos del encabezado IP y encabezados de extensión se toman en cuenta las siguientes reglas:

- En el encabezado IP son excluidos los campos: **Versión, Clase y Etiqueta de Flujo**. El valor del campo **Cuenta de Saltos** se asumirá como cero.

- Todos los encabezados de extensión con el *bit* **Cambio-en-Ruta** encendido serán computados como secuencias de ceros.

- Si un Encabezado de Extensión de Enrutamiento está presente (lo cual indica que se deben intercambiar la dirección destino IPv6 y la siguiente dirección listada en el Encabezado de Enrutamiento, así como incrementar el valor del campo **Siguiente Dirección**) el valor del campo Destino IPv6 es llenado con la dirección del destino final IPv6.

El resultado de tomar todas estas consideraciones es un checksum relativamente corto (128 bits para MD-5 y 160 bits para SHA-1) el cual puede entonces ser truncado ligeramente para permitir que el Encabezado de Autenticación tenga un tamaño múltiplo de 64 bits y con esto se consiga una optimización en el uso de memoria en los enrutadores.

Algoritmos de *checksum*

En la especificación de IPv6 se cuenta con dos algoritmos para realizar el checksum y estos son: *Keyed Message Digest No. 5* (MD-5) y *Secure Hash Algorithm No. 1* (SHA-1).

MD-5 tal como lo define el RFC 1321 [17-18], toma como entrada un mensaje de longitud arbitraria y produce como salida una representación condensada de 128 bits de la entrada.

SHA-1 toma como entrada un mensaje de longitud menor a 2^{64} bits y produce como salida una representación condensada de 160 bits de la entrada tal como lo explica el FIPS 180 [19]. Este algoritmo a pesar de ser un poco más lento que MD5, también es más seguro contra ataques de fuerza bruta debido a que produce un mensaje de salida más grande que MD5.

2.5.2 Encapsulación de Seguridad de la Carga

La Encapsulación de Seguridad de la Carga está diseñada para proveer confidencialidad, autenticación del origen de los datos, integridad, un servicio anti re-envío de paquetes y una limitada confidencialidad en tráfico de flujos. La presencia del Encabezado de ESP es identificada por un valor de 50 en el campo Siguiente Encabezado y su encabezado tiene el formato mostrado el la figura 2.6.

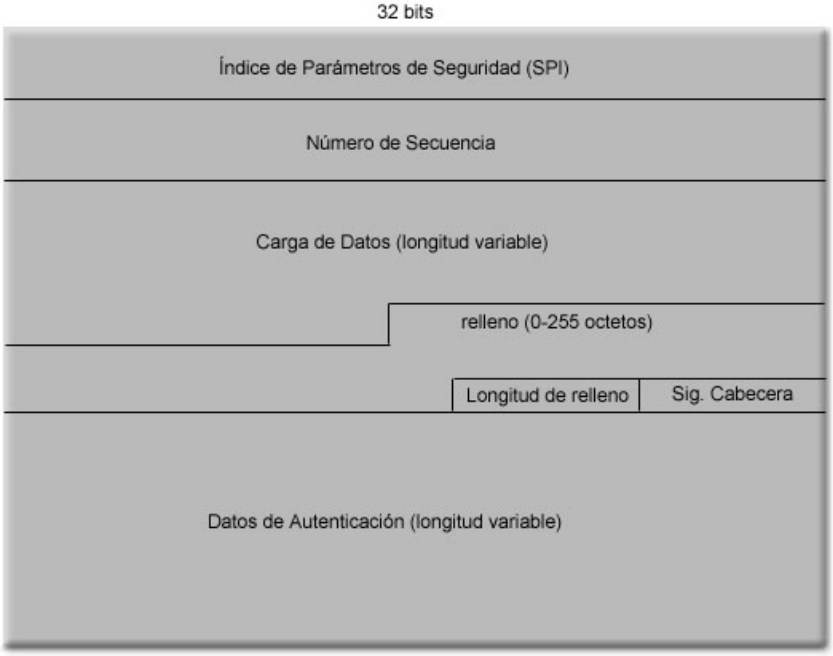


Figura 2.6. Formato del encabezado ESP

Este encabezado consta de 7 campos, mismos que describimos a continuación:

- Índice de parámetros de Seguridad: Con 32 bits de longitud identifica la Asociación de Seguridad aplicada parta este datagrama.

- Número de Secuencia: Contiene un número de 32 bits de longitud, el cual es incrementado monotónicamente. Los contadores tanto del emisor, como del receptor son inicializados a cero cuando una Asociación de Seguridad es establecida.

- Carga de Datos: Tiene una longitud variable y contiene los datos descritos por el campo Siguiente Encabezado.

- Relleno: Puede opcionalmente tener de 0 a 255 octetos de datos de relleno.

- Longitud de relleno: Indica el número de octetos de relleno (0-255) que son agregados en el campo Relleno.

- Siguiente Encabezado: Con 8 bits de longitud este campo identifica el encabezado que sigue inmediatamente después del Encabezado de Encapsulación de Seguridad de la Carga.

- Datos de Autenticación: Tiene longitud variable y contiene el Valor de Chequeo de Integridad (ICV o *checksum*) para este paquete. La longitud de este campo depende de la función de autenticación que sea seleccionada. Este campo es opcional y es incluido solamente si la asociación de Seguridad ha elegido n servicio de autenticación.

En el capítulo 5 tratamos con mayor detalle ambos encabezados (AH y ESP).

2.6 Mecanismos de transición

La conversión de redes IPv4 a IPv6 tardará un largo período de tiempo, por lo que en el diseño de IPv6 se han tomado en cuenta mecanismos que permitan la coexistencia y comunicación de ambos protocolos. Estos mecanismos de transición se dividen en tres clases principales:

- *Dual Stack*: Este mecanismo de transición permite a un enrutador, *host* o servidor utilizar un *stack* IPv4 y un *stack* IPv6 simultáneamente, lo que trae consigo dos grandes ventajas: por un lado un nodo con *dual stack* puede comunicarse con nodos que solo tienen un stack IPv4 de manera nativa y por el otro también puede comunicarse con nodos que solo tengan habilitado el stack IPv6 de manera nativa. Su principal desventaja es la necesidad de contar con una infraestructura de red que soporte el tráfico IPv6 de manera nativa.

- Túneles: Este mecanismo de transición permite a un enrutador IPv6, *host* IPv6 o servidor IPv6 comunicarse con otras redes IPv6 a través de la infraestructura IPv4 actual. Esta técnica consiste en encapsular los paquetes IPv6 dentro de paquetes IPv4 y entonces enviarlos sobre una red IPv4 a un nodo IPv4 destino el cual se encargará de extraer los paquetes IPv6 y entregarlos a su destino final. La principal ventaja de éste mecanismo de transición es que solo es necesario tener un Dual Stack en los nodos que servirán como extremos del túnel. Su principal desventaja es el retardo adicional ocasionado por el encapsulado y desencapsulado de paquetes IPv6 en datagramas IPv4, así como el tráfico de un mayor número de paquetes ocasionado por la reducción de espacio para datos en los datagramas IPv4 que contienen dentro paquetes IPv6.

- Traducción de protocolos: Este mecanismo de transición permite a un nodo que solo cuenta con el stack IPv6 habilitado dentro de una red IPv6 comunicarse con otro nodo que solo tiene el stack IPv4 habilitado dentro de una red IPv4. Sin embargo, ésta técnica requiere tener habilitados mecanismos de traducción entre IPv4 e IPv6 en las orillas de ambas redes (enrutadores). La principal desventaja es que todo el peso de este mecanismo de transición recae en los dispositivos encargados de hacer dicha traducción, a los que no siempre se tiene acceso.

A continuación trataremos con más detalle los mecanismos de transición Dual Stack y Túneles ya que éstos son los que se utilizaron para la realización de esta investigación por ser los más ampliamente difundidos.

2.6.1 Dual Stack

Este mecanismo de transición como ya se había mencionado anteriormente permite a un nodo utilizar un *stack* IPv4 y un *stack* IPv6 simultáneamente teniendo dos grandes ventajas: por un lado un nodo con Dual Stack puede comunicarse con nodos que solo tienen Stack IPv4 de manera nativa y por el otro también puede comunicarse con nodos que solo tengan habilitado el Stack IPv6 de manera nativa Esta técnica no es nueva, ya que en el pasado fue muy utilizada para realizar transiciones entre otros protocolos, tales como el desarrollo de IPv4 dentro de redes como: *Internet Packet Exchange* (IPX) y *Digital Equipment Corporation Network* (DECnet) entre otros.

Antes de que poder utilizar la capacidad del Dual Stack sobre un nodo es necesario modificar las aplicaciones basadas en IPv4 para que éstas también soporten IPv6, ya que el API de las aplicaciones basadas en IPv4 está codificado para utilizar únicamente direcciones de 32 bits. Como se muestra en la

figura 7 (a) Las aplicaciones que soportan únicamente el Stack de IPv4 pueden utilizar TCP o UDP como capa de transporte para entregar los datos, después estos datos llegan al Stack IPv4, en donde son puestos dentro de paquetes IPv4. Estos paquetes IPv4 más tarde son llevados a la interfaz de red. El valor del identificador del protocolo de red usado por tramas Ethernet para paquetes IPv4 es 0x0800. Cuando las aplicaciones son modificadas para soportar IPv6 tal como se ve en la figura 7 (b), éstas pueden llamar la función del API correcta que pueda manejar direcciones de 128 bits. Así los datos que llegan al dual stack pueden seleccionar cual de ellos utilizar para generar los paquetes.

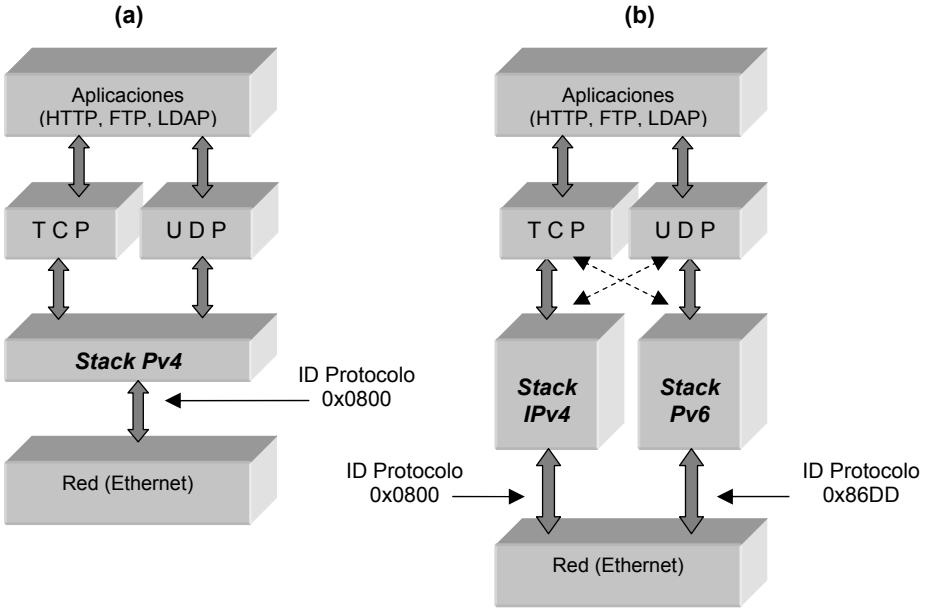


Figura 2.7. Arquitectura de Dual Stack:
(a) Aplicaciones que solo utilizan el Stack IPv4 para enviar paquetes
(b) Aplicaciones que soportan ambos Stack's para enviar paquetes

Esta selección se puede realizar de dos maneras:

- Manual: Cuando el usuario conoce la dirección IPv6 del nodo destino. Para aplicaciones Web es necesario utilizar el formato para direcciones en un URL tal como está definido en el RFC 2732 [20]. El uso de direcciones manualmente establecidas solo es recomendable para propósitos de depuración, en lo posible debe utilizarse un servicio de nombrado.

- Utilizando un servicio de nombrado: Se puede configurar un Nombre de Dominio Completamente Calificado (FQDN) en un servidor de nombrado DNS con ambas direcciones IPv4 e IPv6 y eventualmente este puede ser consultado para proveer información acerca de la disponibilidad de un nodo sobre IPv4 o IPv6. Una aplicación que soporta ambos stack's IPv4 e IPv6 solicitará al servicio de nombrado le resuelva FQDN en ambos tipos de direcciones, pero generalmente dará preferencia a las direcciones IPv6.

2.6.2 Túneles

La principal función de los túneles es llevar protocolos incompatibles o datos específicos sobre una red, por ejemplo, los túneles del Protocolo de Enrutamiento *Multicast* Vector Distancia (DVMRP) llevan datagramas *multicast* sobre redes *unicast*. IPSec en modo túnel lleva datos protegidos por un algoritmo de cifrado. Para el desarrollo de IPv6 sobre una infraestructura existente IPv4 los túneles proveen una manera básica de comunicación entre *hosts* o islas de *hosts* IPv6 utilizando IPv4 como medio de transporte. En la figura 8 un túnel es creado para comunicar dos islas de *hosts* IPv6 sobre el Internet. Los enrutadores encargados de administrar el túnel deben tener configurado un dual *stack* para poder encapsular los paquetes IPv6 en datagramas IPv4 y viceversa.

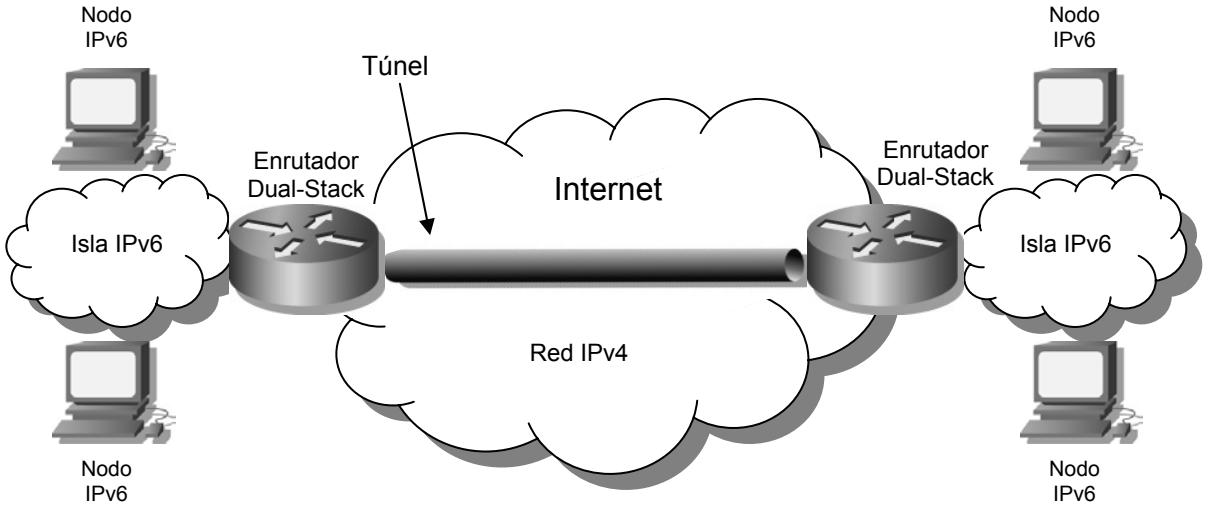


Figura 2.8. Túnel establecido entre dos islas IPv6 a través de la infraestructura IPv4

Para poder configurar un túnel primero es necesario tomar en cuenta los siguientes aspectos:

- Habilitar el protocolo 41: Si se tiene configurado un cortafuegos sobre IPv4, es necesario establecer una regla que permita el acceso y salida al protocolo 41. Como está descrito en el RFC 2893 "*IPv6 Transition Mechanisms*"[21] el número de protocolo asignado a la encapsulación de paquetes IPv6 en IPv4 es el 41. Este valor es utilizado en el campo "Número de Protocolo" en el encabezado de IPv4 para especificar la encapsulación de un paquete IPv6 en un paquete IPv4.

- Manejo de mensajes de error (ICMPv4): Algunos viejos enrutadores en caso de error solo regresan ocho octetos de datos, sin embargo, los nodos emisores de los paquetes IPv6 necesitan conocer los campos de direcciones IPv6 en el error y cada uno de ellos ocupa 16 octetos.

- Traducción de Direcciones de Red (NAT): No es posible establecer túneles IPv6 en IPv4 a través de NAT cuando éste está habilitado en modo traducción dinámica de puerto y redirección de puerto. Por otra parte, es posible establecer dichos túneles si NAT es configurado en modo estático como lo muestra el RFC 2766 [22].

Una vez visto esto es necesario definir un escenario en el cual se usará el túnel. Existen tres posibles escenarios para la creación de un túnel:

- *Host* a *Host*: Esta arquitectura requiere que ambos *hosts* tengan un *Dual Stack* configurado y solo permite el establecimiento de sesiones IPv6 extremo a extremo entre ellos.

- *Host* a Enrutador: *Hosts* con un *Dual Stack* pueden establecer un túnel con un enrutador que también cuente con un Dual Stack. El enrutador puede tener conectividad IPv6 nativa sobre otra interfaz por lo que esta arquitectura permite el establecimiento de sesiones IPv6 extremo a extremo entre cualquier *host* de la isla IPv6 y el *host* aislado a través del enrutador.

- Enrutador a Enrutador: Enrutadores con un Dual Stack sobre una red IPv4 pueden establecer un túnel hacia otro enrutador con Dual Stack. Estos enrutadores pueden ser utilizados para interconectar islas de *hosts* IPv6, por lo que cualquier *host* puede establecer sesiones IPv6 extremo a extremo con otro *host* de la otra isla IPv6.

En la figura 9 se muestran los tres escenarios posibles para la creación de túneles, el caso (1) muestra la generación de un túnel *host* a *host*. El caso (2) presenta la generación de un túnel *host* a enrutador y por último, el caso (3) presenta la generación de un túnel enrutador a enrutador.

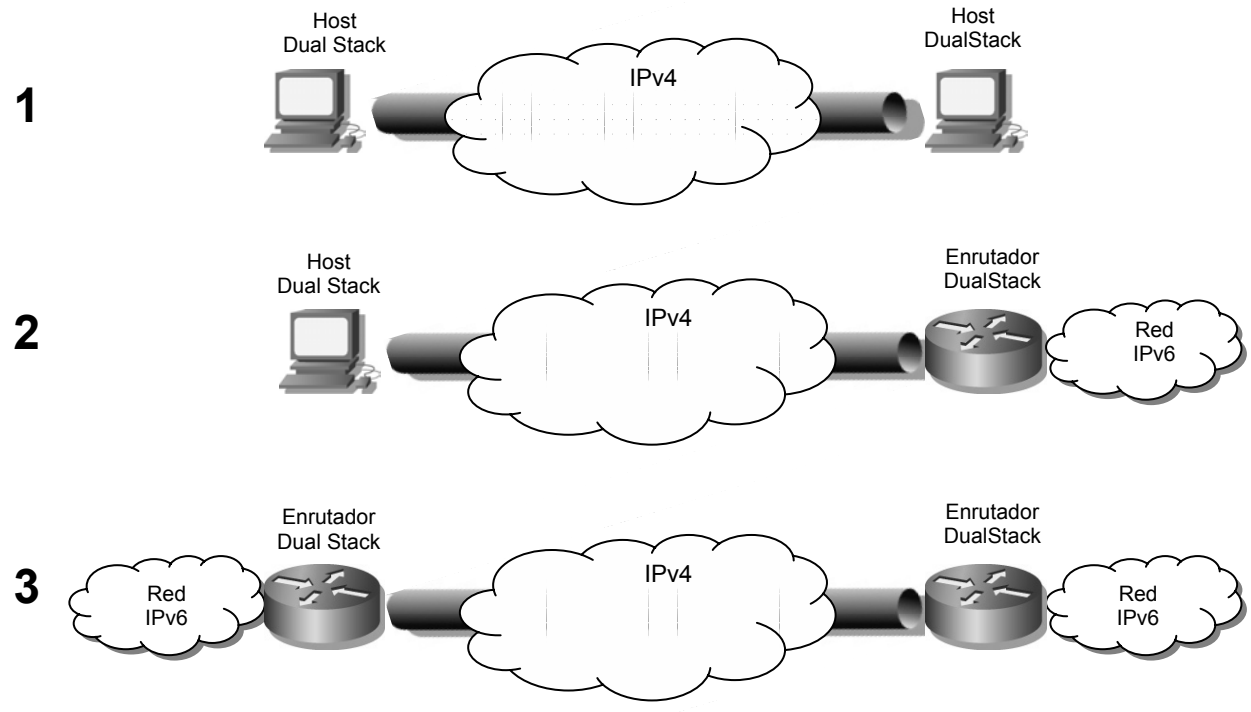


Figura 2.9. Escenarios para la creación de un túnel

2.6.2.1 Técnicas para establecer Túneles

El IETF definió protocolos y técnicas para establecer túneles entre nodos con dual-stack, entre estas técnicas se encuentran las siguientes:

- Túneles 6to4: En esta técnica los extremos del túnel están determinados por las direcciones globales IPv4 embebidas dentro de direcciones IPv6 *6to4*. Las direcciones IPv6 6to4 están formadas por la combinación de un prefijo de enrutamiento global 2002::/16 y una dirección IPv4 globalmente única. Los túneles 6to4 pueden ser configurados entre dos enrutadores en la orilla de sus respectivas redes, o entre un enrutador y un host. El único inconveniente de esta técnica para establecer túneles es que solo permiten enviar tráfico IPv6 entre hosts con prefijos de enrutamiento 2002. Para poder comunicarse con nodos con otros prefijos de enrutamiento tales como 2001::/16 y 3FFE::/16 es necesario utilizar un enrutador de reenvío (*relay router*) del 6bone el cual se encargará de proporcionar un servicio de enrutamiento global 6to4.

- Intransite Automatic Tunnel Addressing Protocol (ISATAP): Esta técnica permite crear túneles IPv6-in-IPv4 automáticamente dentro de un sitio IPv4. Cada *host* solicita a un enrutador dentro del sitio IPv4 una dirección IPv6 e información de enrutamiento, de esta manera, los paquetes enviados al Internet IPv6 son enrutados a través del enrutador ISATAP y los paquetes destinados hacia otros hosts dentro del mismo sitio son entregados directamente mediante túneles ISATAP. Las direcciones IPv6 se configuran automáticamente mediante el protocolo "descubrimiento de enrutador" ISATAP, aunque también pueden ser configuradas de manera manual. Una dirección ISATAP al igual que una dirección 6to4 está formada por la concatenación de un prefijo global de agregación *unicast* IPv6 y el identificador de interfaz. El prefijo utilizado por ISATAP para habilitar una dirección ISATAP en un host es FE80::/10 (dirección local). El identificador de interfaz es formado agregando los 32 bits de la dirección IPv4, después se concatena el valor 0000:5EFE (reservado por IANA para identificar direcciones ISATAP). Ejemplo: para una dirección IPv4 148.247.54.122 su dirección IPv6 ISATAP sería FE80::5EFE:94F7:367A. Las direcciones ISATAP también pueden utilizar prefijos *unicast* Globales de 64 bits, los cuales son asignados por los enrutadores. Cuando un nodo ISATAP desea comunicarse con otro nodo ISATAP sobre IPv6 el paquete IPv6 es encapsulado dentro de un datagrama IPv4 al igual que como sucede con el mecanismo 6to4.

- Tunnel Broker: El IETF definió este mecanismo para facilitar el desarrollo de túneles configurados sobre redes IPv4 ya que mediante esta técnica no se tiene que configurar manualmente cada extremo del túnel. Tal como está establecido en el RFC 3053 "IPv6 *Tunnel Broker*" [23] el *tunnel broker* es un sistema externo que actúa como un servidor sobre la red IPv4 y recibe peticiones de nodos con dual stack para configurar túneles automáticamente (modelo cliente-servidor). Estas peticiones son enviadas vía HTTP sobre IPv4 por el nodo que desea configurar dicho túnel. El *tunnel* broker entonces envía de vuelta al cliente información tal como la dirección IPv4 del servidor del túnel, la dirección IPv6 del servidor del túnel, la nueva dirección IPv6 que será asignada a este host con dual stack y las rutas IPv6 default para la configuración del túnel. Algunos *tunnel* broker's ya proporcionan *scripts* de configuración para los hosts clientes. Finamente el *tunnel* broker aplica comandos de manera remota sobre un enrutador con dual stack y que está conectado a un dominio IPv6 para habilitar el túnel configurado. Para poder hacer uso de esta técnica es necesario utilizar los servicios de alguna entidad que ofrezca el servicio de *tunnel* broker tales como:
    - Freenet6
    - Dolphins *tunnel* broker
    - British Telecom *tunnel broker*
    - Hurricane Electric

- SixXS

La gran mayoría de los tunnel brokers ofrecen el servicio de manera gratuita, el único requisito es registrarse mediante el llenado de un pequeño formulario.

- *Generic Routing Encapsulation* (GRE): Esta técnica fue desarrollada originalmente por Cisco para transportar tráfico Multicast sobre redes *unicast* y protocolos como IPX y Appletalk sobre IP, pero también puede transportar tráfico IPv6 sobre redes IPv4. GRE no utiliza TCP o UDP, en su lugar trabaja directamente con la capa IP, utilizando el protocolo número 47. Este incluye sus propios mecanismos para verificar la entrega e integridad de los paquetes. La carga de un paquete GRE incluye un paquete de capa 3 completo con su encabezado y carga intactos. El enrutador en la entrada del túnel GRE toma los paquetes IP y los envuelve en un nuevo paquete con un encabezado GRE, después los envía por la red hasta que alcanzan el enrutador de la salida del túnel. Este extrae el paquete contenido dentro del paquete GRE y lo entrega al nodo destino

- TEREDO: La meta principal de esta técnica es entregar paquetes IPv6 a nodos con dual stack que se encuentran detrás de un dispositivo NAT sobre dominios IPv4. TEREDO fue diseñado por dos principales razones: la primera es que el buen funcionamiento de los túneles 6to4 recae en la configuración de una dirección pública IPv4 y la implementación de enrutamiento 6to4. Debido a que en muchas ocasiones se tienen configuraciones de NAT de varios niveles no sería posible asignar a cada uno de estos dispositivo NAT una dirección pública IPv4. La segunda razón por la que se creó TEREDO es debido a que los paquetes IPv6 encapsulados en paquetes IPv4 utilizan el valor 41 en el campo de protocolo en el encabezado del paquete IPv4 y la mayoría de los dispositivos NAT solamente son capaces de traducir TCP y UDP. Como el protocolo 41 no es común entre los dispositivos NAT este tipo de paquetes no podrían fluir a través de ellos para alcanzar a los nodos destino. TEREDO utiliza como medio de transporte a UDP para la creación de túneles ya que los dispositivos NAT pueden manejar bien este protocolo a múltiples niveles de anidación. Utilizando una sola dirección IPv4 y mapeos UDP del dispositivo NAT es posible establecer túneles para diferentes hosts con dual stack detrás de un mismo dispositivo NAT, para ello este mecanismo consta de tres componentes principales:

  - Servidor TEREDO: Este servidor está conectado al Internet y cuenta con una dirección global IPv4. Se encarga de administrar la señalización y tráfico con los clientes TEREDO.
  - Cliente TEREDO: este se encuentra detrás de un dispositivo NAT y solicita conectividad IPv6 al servidor TEREDO mediante paquetes UDP IPv4.
  - TEREDO de reenvío: Está conectado a Internet IPv6 y actúa como enrutador IPv6 para brindar conectividad a los clientes TEREDO mediante el uso de paquetes UDP.

Cada una de estas técnicas está desarrollada para un escenario distinto de aplicación túneles *6to4* está diseñado para interconectar islas IPv6, ISATAP está diseñado para interconectar hosts a enrutadores don dual-stack, tunnel-broker está diseñado para conseguir conectividad IPv6 sobre nodos aislados en redes IPv4, GRE permite no solo encapsular tráfico IPv6 sino también IPX y Appletalk y TEREDO está diseñado para conseguir conectividad IPv6 en hosts que están detrás de dispositivos NAT.

Capítulo 3

Pruebas básicas sobre IPv6

En este capítulo describiremos la arquitectura general de nuestra red IPv6 de pruebas, así como también daremos ejemplos de los primeros ensayos que realizamos para constatar el buen funcionamiento de IPv6 en todas las plataformas de prueba. :

3.1 Arquitectura de la red de pruebas

Conformamos la red de nuestras pruebas mediante 4 PC's, un *switch* y un enrutador cuyas especificaciones son mostradas en las tablas 3.1 y 3.2.

Características	Máquina1 (H1)	Máquina2 (H2)	Máquina3 (H3)	Máquina4 (H4)
Procesador	Pentium 4 1.60 GHZ	Pentium 4 1.60 GHZ	Celeron 1.70 GHZ	Power PC G4 700 MHZ
Memoria RAM	256 MB	256 MB	256 MB	128 MB
Disco Duro	80 GB	80 GB	40 GB	40 GB
Tarjeta de red	FastEthernet 100Mbps	FastEthernet 100Mbps	FastEthernet 100Mbps	FastEthernet 100Mbps
Plataforma	Windows XP Home Edition	Red Hat Linux 8.0	Red Hat Linux 8.0 / Solaris 9.0	Mac OS X 10.2

Tabla 3.1. Características de las máquinas utilizadas en la red de pruebas

Características Enrutador	
Modelo	Cisco 2620
Interfaces	Fast Ethernet 100Mbps
Sistema Operativo	Cisco IOS 12.2(11)T10
Procesador	Motorola Power QUICC MPC860 50MHz
Memoria Flash	16 MB
Memoria no volátil	32 KB
Características Switch	
Modelo	3COM Office Connect 8
Núm. Puertos	8 (100Mbps)

Tabla 3.2. Características de Enrutador y Switch

Decidimos utilizar las plataformas Windows XP, Solaris 9, Linux Red Hat 8 / 9 y Mac OS X 10.2 ya que creemos que son las plataformas más utilizadas actualmente. Como servidor DNS utilizamos la versión 9.2.1-9 de BIND para la plataforma Linux porque tiene soporte para IPv6 y cuenta con mucha información disponible para su configuración. En la figura 3.1 podemos ver la arquitectura de la red de pruebas, en ella se muestran clientes de Freenet6 configurados en cada uno de los *hosts* de la red. Estos clientes son esenciales, ya que es por medio de ellos que obtenemos las direcciones IPv6 Globales utilizadas para poder comunicarnos sobre IPv6 fuera de la red. En el apéndice C tratamos con más detalle el uso de Freenet6 y su protocolo de túnel TSP.

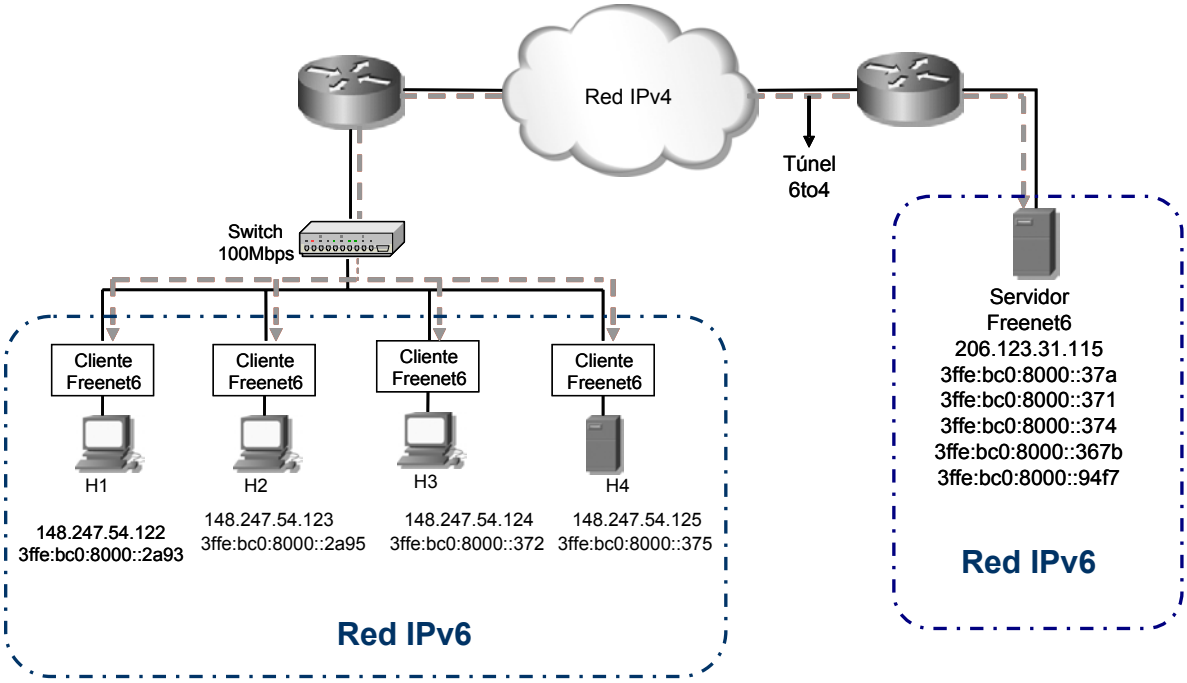


Figura 3.1 Arquitectura actual de la red de pruebas

3.2 Tipo de pruebas realizadas

Antes de intentar probar IPv6 con protocolos complejos como es el caso de LDAP o IPSec los cuales abordamos en los capítulos 4 y 5 primero decidimos realizar ensayos con pruebas simples como son:

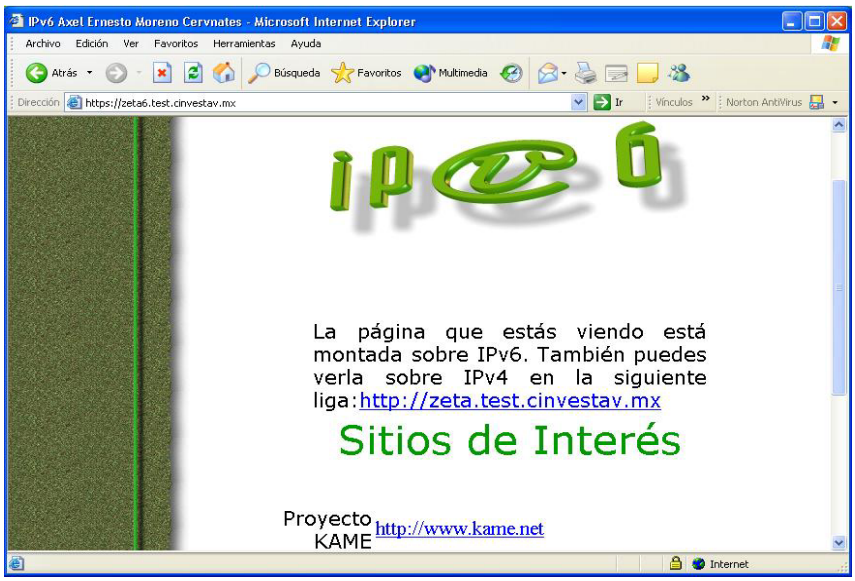
- Uso de navegador para acceder a sitios IPv6
- Envío de paquetes ICMPv6
- Envío de paquetes UDP
- Transferencia de archivos SFTP, SCP

De esta manera pudimos asegurarnos del buen funcionamiento de la red de pruebas tanto con IPv6 nativo, así como con sus mecanismos de transición *dual-stack* y *tunnel-broker.*

3.2.1 Pruebas con el navegador

Para realizar las pruebas con el navegador fue necesario configurar un servidor web y levantar un sitio IPv6; en nuestro caso utilizamos un servidor Apache por ser compatible con IPv6 y además mostrar un buen rendimiento, tal como lo podemos ver en [24]. Por cuestiones de seguridad el servidor web fue configurado con SSL, esto nos también nos dio la posibilidad de probar la interoperabilidad de IPv6 con SSL en las plataformas de prueba. El servidor web IPv6 se configuró en la plataforma Red Hat 9 con la dirección https://zeta6.test.cinvestav.mx (nombre completamente calificado) o https://[3ffe:bc0:8000::375] (formato hexadecimal para usar mediante mecanismo *tunnel-broker*), https://[2002:94f7:5e01:1:207:e9ff:fe90:8461] (formato hexadecimal usando dirección *6to4*), las cuales pueden ser visitadas desde cualquier maquina que tenga habilitado el protocolo IPv6. Realizamos pruebas para acceder a dicho servidor utilizando los mecanismos de transición antes mencionados y utilizando IPv6 de manera nativa mediante direcciones locales tal como se puede ver en la figura 3.2 En la figura 3.2 (b) capturamos una traza de paquetes y en esta podemos ver como efectivamente se está realizando la comunicación sobre IPv6 mediante el mecanismo de *tunnel-broker*, asimismo podemos ver la manera en que se forma el canal de comunicación seguro por medio de SSL Sobre IPv6.

a)



b)

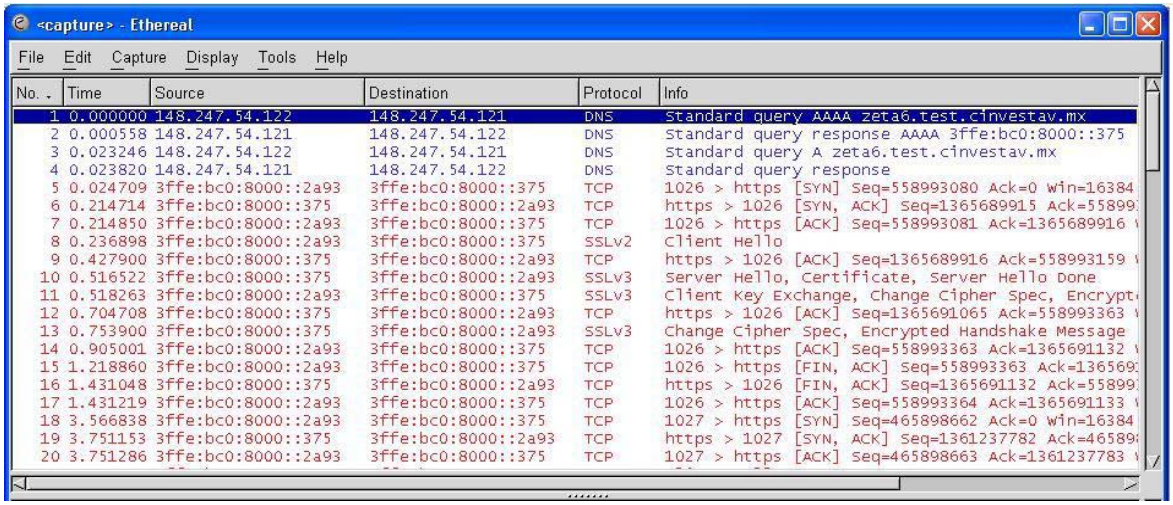


Figura 3.2. Portal IPv6
a) Página sobre un servidor IPv6 vista desde un navegador en la plataforma Windows
b) Captura de paquetes IPv6 mediante ***ethereal***

Todos los accesos al Sitio IPv6 se realizaron de dos maneras:

- Escribiendo la dirección IPv6 como nombre completamente calificado (Tabla 3.3)
- Escribiendo la dirección IPv6 en su formato hexadecimal (Tabla 3.3). utilizando IPv6 nativo, así como también ambos mecanismos de transición

Dirección IPv6 válida desde Windows (Nombre de dominio completamente calificado)	Dirección IPv6 NO válida desde Windows (Formato hexadecimal)
zeta6.test.cinvestav.mx	3ffe:bc0:8000::370 2002:bc0::1

Tabla 3.3 Formato de direcciones IPv6 válido e inválido sobre Windows

En esta primera prueba Windows XP presentó problemas para acceder páginas IPv6 mediante el formato hexadecimal, ya que no es capaz de entender este formato, únicamente entiende nombres completamente calificados ó direcciones IPv4, así que para poder acceder al portal sobre IPv6 desde Windows XP es necesario mediante el nombre completamente calificado del servidor web.

3.2.2 Envío de paquetes ICMPv6

La siguiente prueba que llevamos a cabo fue el envío de paquetes ICMPv6 con diferentes tamaños de carga. En el RFC 2463 [25] se encuentran definidos dos tipos de mensajes ICMPv6 que nos son de interes: "*Echo Request*" y "*Echo Reply*" los cuales son utilizados para una de las utilidades más comunes

de TCP/IP: el Packet INternet Groper (ping). Ping es utilizado para determinar si un host específico está disponible en la red y listo para comunicarse. El nodo emisor envía mensajes del tipo "*Echo Request*" al destino especificado y si este se encuentra disponible envía de vuelta mensajes del tipo "*Echo Reply*". En la figura 3.3 podemos ver se muestra el envío de paquetes ICMPv6 entre dos nodos utilizando el mecanismo de transición tunnel 6to4 desde el host con dirección 2002:94f7:5e01:1:207:e9ff:fe90:8461 hacia el host con dirección 2002:94f7:5e01:1:207:e9ff:fe90:8461.

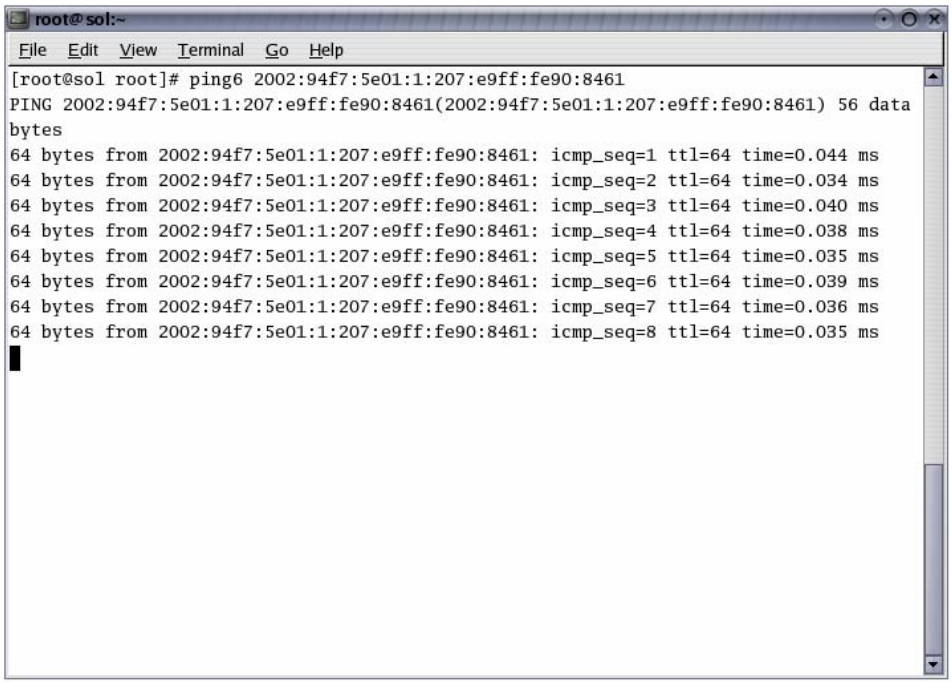


Figura 3.3 Envío de paquetes ICMPv6 utilizando direcciones IPv6 Globales

Mirando con un poco más de detalle podemos ver en la figura 3.4 el contenido del paquete IPv6 en un mensaje ICMPv6 "Echo Reply". En este paquete identificamos a ICMPv6 por tener un valor de 58 (0x3A en hexadecimal) en el campo NextHeader. Dentro del encabezado de ICMPv6 podemos identificar el tipo de mensaje "Echo Reply" por tener un valor de 129 en el campo Type. Así pues, con esto aseguramos que los paquetes que se están transmitiendo corresponden a paquetes ICMPv6 y como vemos en la figura 3.3 fueron exitosos al no reportar perdida de paquetes.

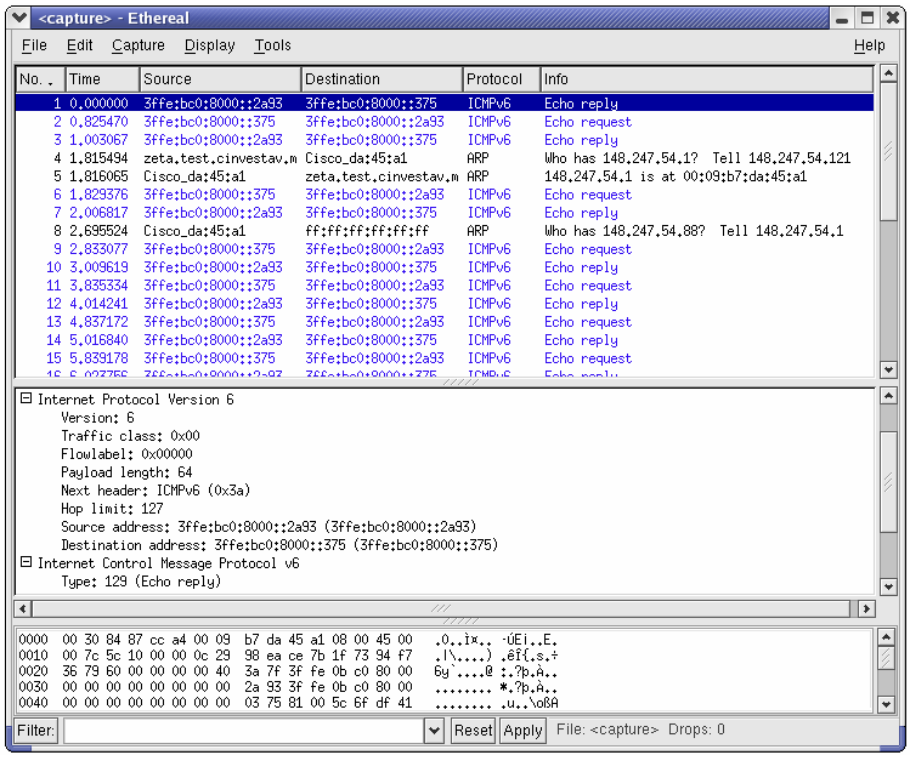


Figura 3.4 traza de paquetes ICMPv6 utilizando la aplicación Ethereal

3.2.3 Envío de paquetes UDP

User Datagram Protocol (UDP) al igual que TCP es un protocolo que trabaja sobre la capa de transporte, pero a diferencia de este, UDP/IP provee muy pocos servicios de detección de error, sin embargo ofrece una manera directa de enviar y recibir datagramas sobre una red IP por lo que su principal uso es para enviar mensajes *broadcast* .

Nuestra prueba consistió en enviar ping's UDP a puertos consecutivos esperando recibir de vuelta mensajes de tipo "PORT UNREACHABLE" desde el host destino. Esta prueba se realizó utilizando direcciones IPv6 Locales y Globales. En la figura 3.5 se aprecia una traza de paquetes utilizando la herramienta *Ethereal*. En ella podemos ver el barrido de puertos y los mensajes que son devueltos por parte del host destino. Con esto pudimos validar que UDP si se estaba comportando muy bien sobre IPv6.

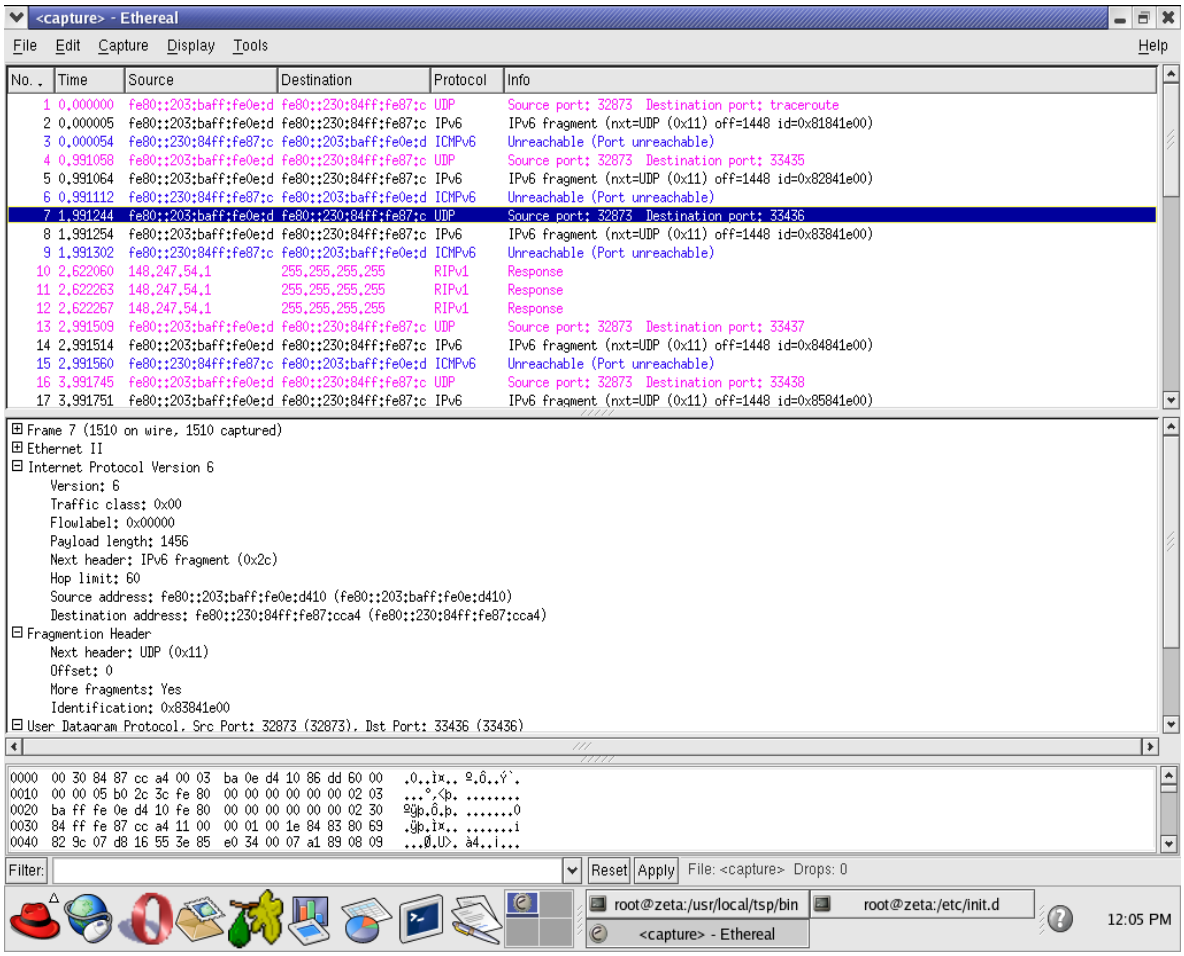


Figura 3.5 Envío de paquetes UDP sobre IPv6

3.2.4 Envío de paquetes a través de SFTP y SCP

Secure File Transfer Program (SFTP) es un protocolo utilizado para transferir archivos tal como lo hace FTP, pero a diferencia de FTP, SFTP realiza todas sus operaciones sobre un canal SSH encriptado. *Secure Copy* (SCP) por su parte, trabaja de manera muy similar a SFTP, la única diferencia entre ellos es que SFTP permite elegir de manera interactiva los archivos a transferir. Nuestra elección por SFTP y SCP sobre FTP queda sobreentendida por lo anteriormente mencionado. En nuestras pruebas realizamos transferencias de archivos de diferentes tamaños utilizando direcciones IPv6 Locales y Globales.

En la figura 3.6 mostramos una traza de paquetes mientras se utilizaba SFTP para transferir archivos utilizando direcciones IPv6 Globales.

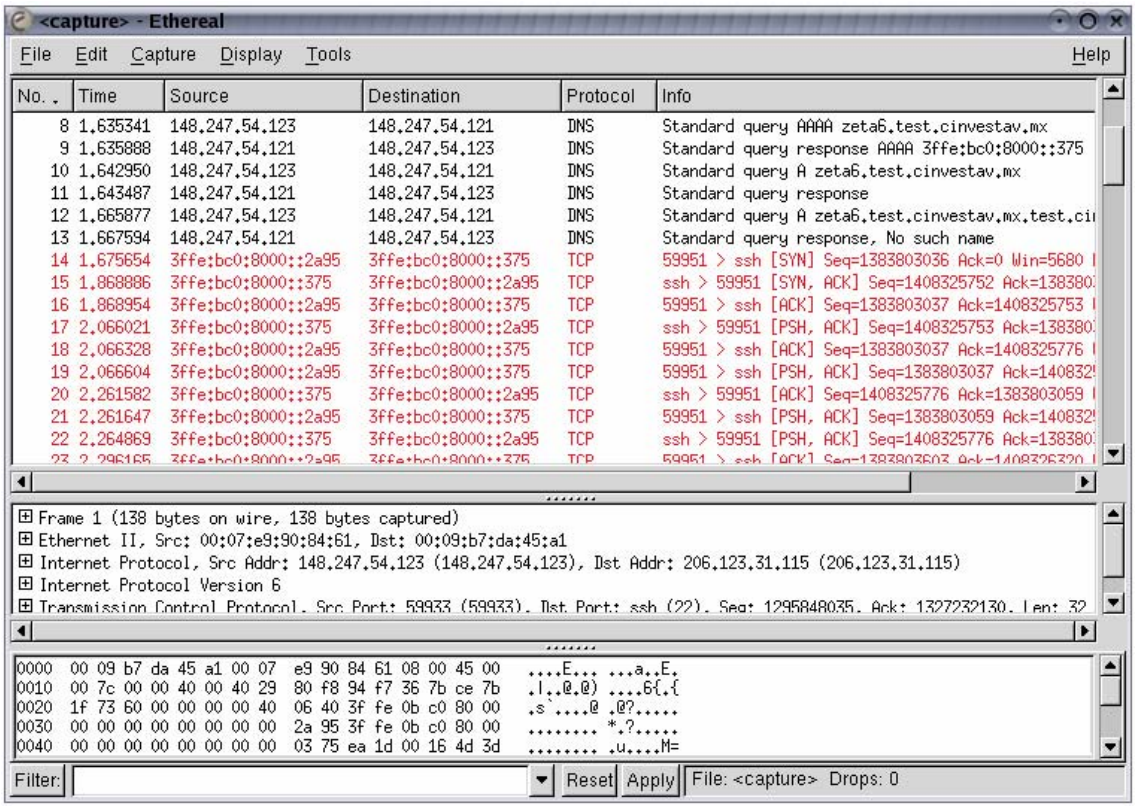


Figura 3.6 Transferencia de archivos mediante SFTP sobre IPv6

Asimismo, en la figura 3.7 mostramos una traza de paquetes mientras se utilizaba SCP para transferir archivos utilizando direcciones IPv6 Globales.

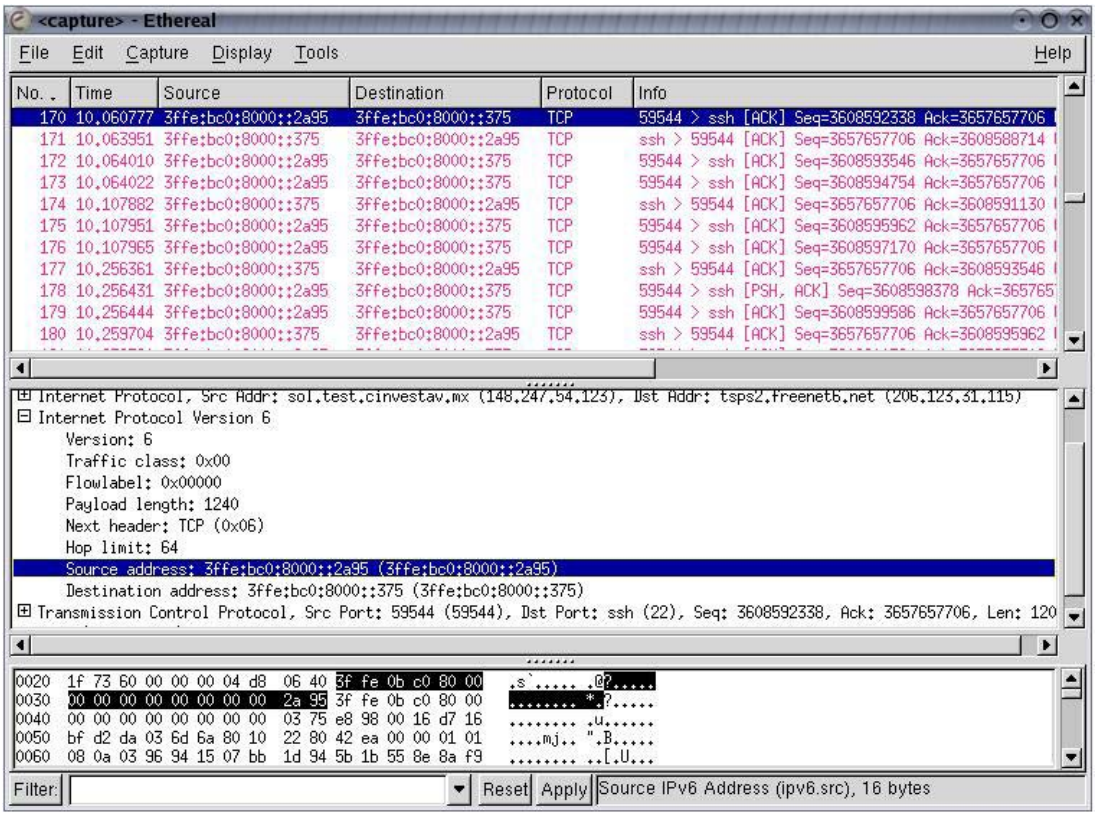


Figura 3.7 Uso de scp para la transferencia de archivos sobre IPv6

Capítulo 4

Protocolo Ligero de Acceso a Directorio (LDAP)

4.1 Introducción

Un directorio puede ser visto como un repositorio en donde podemos colocar información que después consultaremos para su procesamiento. Un directorio también puede ser visto como una pequeña base de datos, pero a diferencia de esta, el directorio está diseñado y optimizado para realizar operaciones de consulta, más que para realizar otro tipo de operaciones como lo es la modificación. Además, un directorio no tiene soporte para realizar operaciones de tipo transacción, comit, rollback, etc., pero si hacemos una analogía con un directorio de páginas blancas, este solo sirve para consultar información en el, no para modificarla.

4.1.1 Servicio de Directorio

Un Servicio de Directorio se encarga de asociar nombres con objetos, y además dichos objetos pueden tener atributos, con esto se consigue el poder realizar búsquedas basándonos en los nombres de los objetos y así obtener sus atributos, o también podemos realizar búsquedas basándonos en los atributos. En otras palabras, un Servicio de Directorio nos permite realizar operaciones de búsqueda, modificación, inserción y borrado de atributos asociados con objetos en un directorio.

4.2 LDAP

El protocolo Ligero de Acceso a Directorio (*Lightweight Directory Access Protocol*) fue desarrollado por la Universidad de Michigan en 1993 [26] para remover parte de la carga excesiva del acceso de X.500 desde los clientes del directorio, obteniéndose un servicio de directorios que corre sobre TCP/IP u otro transporte, además, existen diversas versiones del directorio para diversas plataformas y aplicaciones de cómputo. Al protocolo se le simplificaron muchas operaciones de X.500, retirando características poco usadas y emulando algunas operaciones con otras. LDAP usa una codificación simple para cadenas de caracteres para la mayoría de los atributos, dando como resultado un método de acceso eficiente para el directorio de acceso X.500.

4.3 Características y usos

Las principales características de LDAP son:

- Está basado en el modelo cliente-servidor,
- organiza la información de modo jerárquico utilizando directorios,
- es capaz de propagar sus directorios a otros servidores LDAP independientemente de su ubicación geográfica, proporcionando así acceso global a la información,

- tiene un API de programación bien definido, utiliza diversos esquemas de bases de datos, los cuales trabajan como procesos *back-end*, para el almacenamiento, la consulta y la actualización de su propia información.

Un directorio LDAP puede contener cualquier tipo de información, desde imágenes, direcciones de correo electrónico, passwords y referencias html, hasta certificados digitales, direcciones IP, etc.

La gran diversidad de información que puede ser almacenada en estos directorios los hace aptos para utilizarse en aplicaciones como:

- Directorios de páginas blancas o amarillas
- Servidores de correo electrónico
- Servidores de Nombres de Dominio (DNS)
- Repositorios para certificados digitales
- Repositorios de cuentas de usuario

Solo por mencionar algunas.

4.3.1 Réplicas

Un servidor LDAP puede, ya sea por cuestiones de seguridad ó de rendimiento tener copias de si mismo en otros servidores LDAP, claro, siempre respetando una jerarquía Maestro-Esclavo, de esta manera se reducen en gran medida los cuellos de botella que pudieran generarse si solo se cuenta con un servidor LDAP para atender a todas las solicitudes. La figura 1 muestra un ejemplo de este tipo de configuración de servidores LDAP.

Debe tenerse en cuenta que solo el servidor maestro podrá realizar modificaciones al directorio, es decir, los servidores esclavos únicamente pueden ejecutar operaciones de consulta sobre la información contenida en ellos. Si en algún momento un usuario solicita una operación de modificación sobre un directorio esclavo, este enviará una petición al servidor maestro para que realice tal modificación, una vez hecho esta, el servidor maestro actualizará la información contenida en todos los directorios esclavos a su cargo.

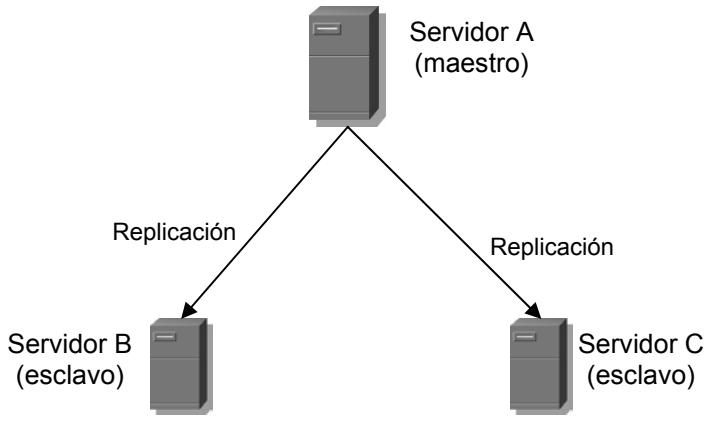


Figura 4.1. Servidores LDAP de réplica

4.3.2 Referencias a otros Directorios LDAP

Una referencia es una liga que une dos servidores LDAP y es utilizada para apuntar al servidor que debe continuar una búsqueda cuando el servidor que lo referenció no es capaz de satisfacer una petición.

Existen dos tipos de referencia: referencias hacia servidores de conocimiento superior y referencias hacia servidores subordinados. La figura 4.2 muestra un ejemplo en donde se pueden ver estos dos tipos de referencias Una referencia a un servidor de conocimiento superior es realizada cuando un directorio, el cual está contenido en otro de mayor jerarquía no produjo ningún resultado en una búsqueda, entonces, ya que posiblemente el directorio con mayor jerarquía sea capaz de satisfacer la petición de consulta, le es enviada la petición de búsqueda en forma de una referencia (entrada de tipo referral).

Por otro lado, una referencia a un servidor subordinado es el caso contrario al que mencionamos, cuando tenemos un directorio configurado de manera distribuida es necesario organizar la información contenida en cada uno de los servidores LDAP de tal manera que se respete una jerarquía, la forma de hacerlo es por medio de referencias que van desde los servidores de mayor jerarquía hacia los de menor jerarquía. Con esto conseguimos tener control sobre toda la información y al mismo tiempo un mejor rendimiento ya que al no estar concentrada toda la información en un solo servidor la carga de trabajo es repartida entre todos los servidores que conforman el directorio.

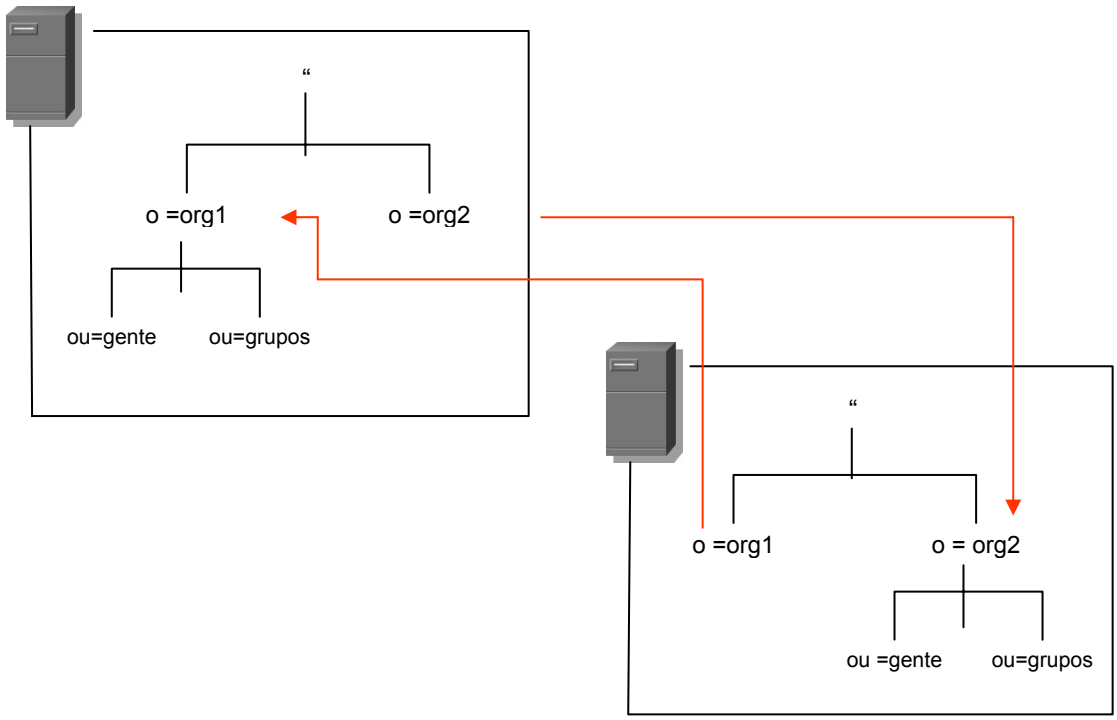


Figura 4.2. Referencias entre dos Servidores LDAP

4.4 Distribuciones

Se cuenta con un gran número de distribuciones de LDAP, tanto clientes como servidores, y podemos agruparlas de la siguiente manera:

- Dominio público:
  - OpenLDAP, http://www.openldap.org/
  - IBM SecureWay Directory, http://rsasecurity.agora.com/rsasecured/detail.asp?product_id=828
  - MVault, http://www.isode.com/products/ic-6097.html
  - InJoin, http://www.cp.net/solutions/enterprise/dataDirectoryIntegration/directoryServer.jsp
  - Netscape Directory Server, http://wp.netscape.com/directory/v4.0/index.html
  - EUDORA Free LDAP Server, http://www.eudora.com/techsupport/worldmail/ldap.html

- Bajo Licencia:
  - iPlanet, http://www.solaris4you.dk/ldapservSS.html
  - Oracle Internet Directory, http://otn.oracle.com/products/oid/content.html
  - ClickMail Central Directory, http://www.gracion.com/server/info.html

De todas estas distribuciones de LDAP nosotros escogimos OpenLDAP, ya que es la más ampliamente difundida, está bien diseñada de acuerdo a los estándares, es fácil de utilizar y se cuenta con mucha información técnica al respecto.

4.5 Configuración del servidor LDAP

En nuestro caso elegimos como servidor LDAP la plataforma RedHat Linux 8.0 y la versión 2.0.27 de OpenLDAP debido a la gran disponibilidad, tanto de información, como de los paquetes necesarios para su configuración. Los rpm's los obtuvimos de la página http://rpm.pbone.net/ y son los siguientes:

- openldap-2.0.27-2.8.0.i386.rpm
- openldap-clients-2.0.27-2.8.0.i386.rpm
- openldap-devel-2.0.27-2.8.0.i386.rpm
- openldap-servers-2.0.27-2.8.0.i386.rpm

En el apéndice D damos una descripción detallada acerca de la configuración del servidor LDAP.

4.6 Configuración de los clientes LDAP

La configuración de las máquinas cliente se hizo de la siguiente manera:

- Solaris 9: solo fue necesario añadir el paquete para el cliente openldap, así como el cliente ssl y el *back end* para la base de datos utilizado por OpenLDAP. Estos paquetes uno los puede descargar del sitio de SUN microsystems.

- Windows XP: se instaló el Service Pack1 y se instaló el JDK

- MacOS X 10.2: Esta plataforma ya soporta LDAP, solo fue necesario instalar el JDK

- Linux RedHat 9: se instalaron los siguientes paquetes:

  - ✓ OpenLDAP-clients
  - ✓ OpenLDAP-devel
  - ✓ OpenLDAP

En el apéndice D mostramos con más detalle la configuración de los clientes LDAP

4.7 Java Naming and Directory Interface (JNDI)

JNDI provee una interfaz común para poder comunicarnos con muchos servicios de nombrado, entre ellos se encuentran:

- COS (*Common Object Services*) Naming: Es el servicio de nombrado para aplicaciones de CORBA y permite a las aplicaciones almacenar y acceder referencias a objetos CORBA.

- DNS (*Domain Name System*): Es el servicio de nombrado de Internet y sirve para mapear nombres completamente calificados (fáciles de recordar) en direcciones IP.

- LDAP (*Lightweight Directory Access Protocol*): Estándar para servicios de directorio. Es una versión ligera de DAP (*Directory Access Protocol*) que forma parte de X.500 y está construido para correr sobre TCP/IP.

- NIS (*Network Information System*): Desarrollado por SUN y permite a los usuarios acceder archivos y aplicaciones sobre muchos hosts mediante un solo ID y Password.

En nuestro caso únicamente lo utilizamos para comunicarnos con los servicios LDAP. JNDI ya viene incluido en las distribuciones del JSDK de SUN, sin embargo, las versiones anteriores al J2SDK 1.5 beta tienen problemas de interoperabilidad con IPv6 sobre la plataforma Windows. La versión 1.5 beta del JSDK ya incluye un soporte completo para la implementación de IPv6 en Windows XP, por lo que esta fue la versión utilizada para la realización de estas pruebas sobre las 4 plataformas.

4.8 Caso de estudio

El objetivo de esta prueba fue por una parte evaluar la interoperabilidad de IPv6 con el protocolo LDAP y por otra comparar si existe alguna diferencia significativa en el rendimiento de un servidor LDAP al utilizar IPv6 mediante sus mecanismos de transición dual stack y *tunnel broker* en lugar de IPv4.

Comenzamos por definir los parámetros de evaluación. Una transacción sobre un servidor LDAP se compone de cuatro pasos: conexión con el servidor LDAP, autenticación del cliente con el servidor LDAP, ejecución de la operación (lectura, búsqueda, modificación, borrado) sobre el directorio LDAP y desconexión del servidor LDAP. El tiempo que tarda un cliente en realizar estos cuatro pasos se le conoce como tiempo de respuesta. Para medir en desempeño del servidor LDAP y evaluar diferencias entre IPv4 e IPv6 medimos el rendimiento del servidor LDAP la latencia de respuesta del mismo con ambos protocolos mientras realizamos operaciones de consulta y modificación en el directorio LDAP. Se configuró un directorio centralizado en un host Linux y se realizaron dos tipos de operaciones en él: Búsqueda y Modificación.

4.8.1 Integración de la prueba

Esta prueba se integró de la siguiente manera. El directorio LDAP se compuso por 100 entradas organizadas de la forma siguiente: 84 entradas pertenecientes a la unidad organizacional Alumnos, 13

pertenecientes a la unidad organizacional Profesores y 3 pertenecientes a la unidad organizacional Personal Administrativo, la agrupación de dichas entradas se hizo en base a la forma en la que están organizados los alumnos, profesores y personal administrativo de la Sección de Computación del CINVESTAV. Las unidades organizacionales ou= Profesores y ou = Alumnos se encuentran contenidos en el servidor zeta (148.247.94.123, 2002:94f7:5e01:1:207:e9ff:fe90:8461) y la unidad organizacional dc = Administrativo se encuentra contenida en el servidor sol (148.247.94.121, 2002:94f7:5e01:1:230:84ff:fe87:cca4) en una configuración de directorio distribuida. En las figuras 4.3 y 4.4 se muestra la forma del directorio LDAP.

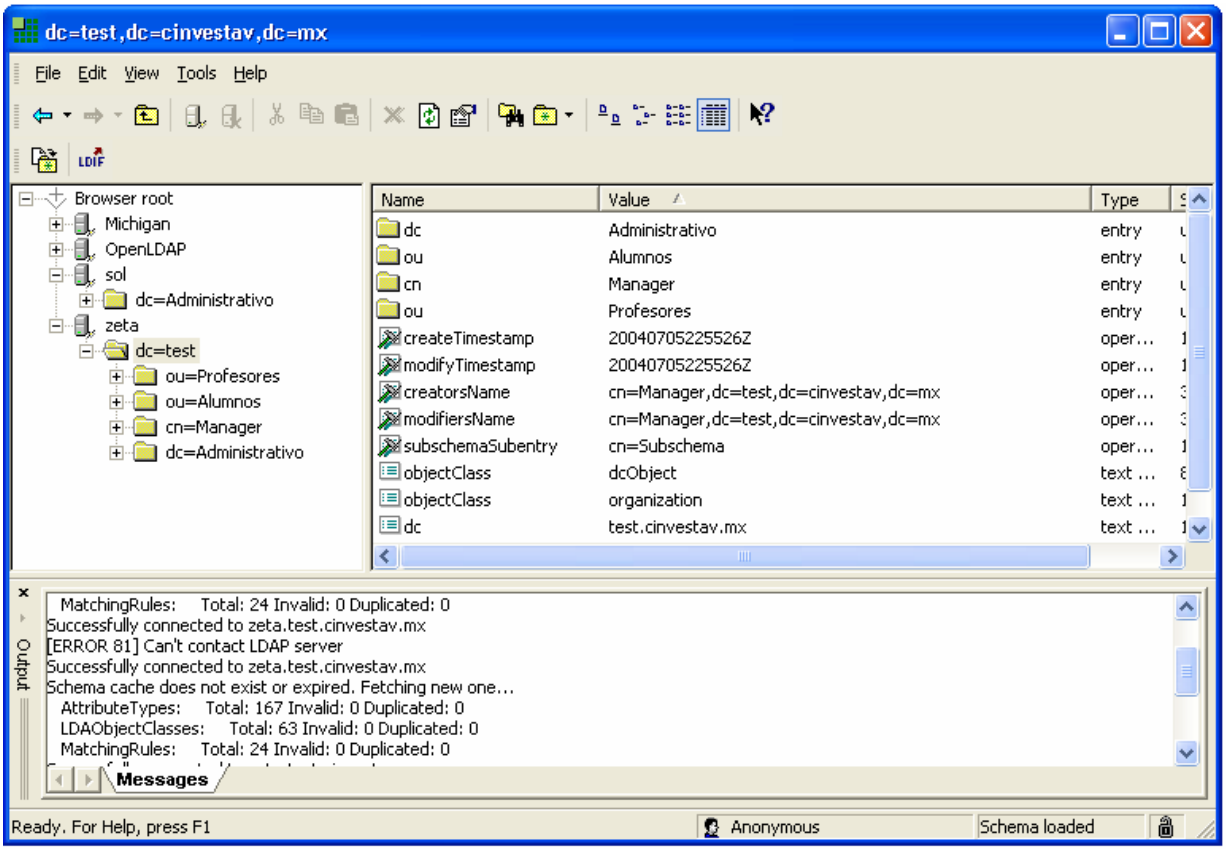


Figura 4.3 Directorio Distribuido LDAP

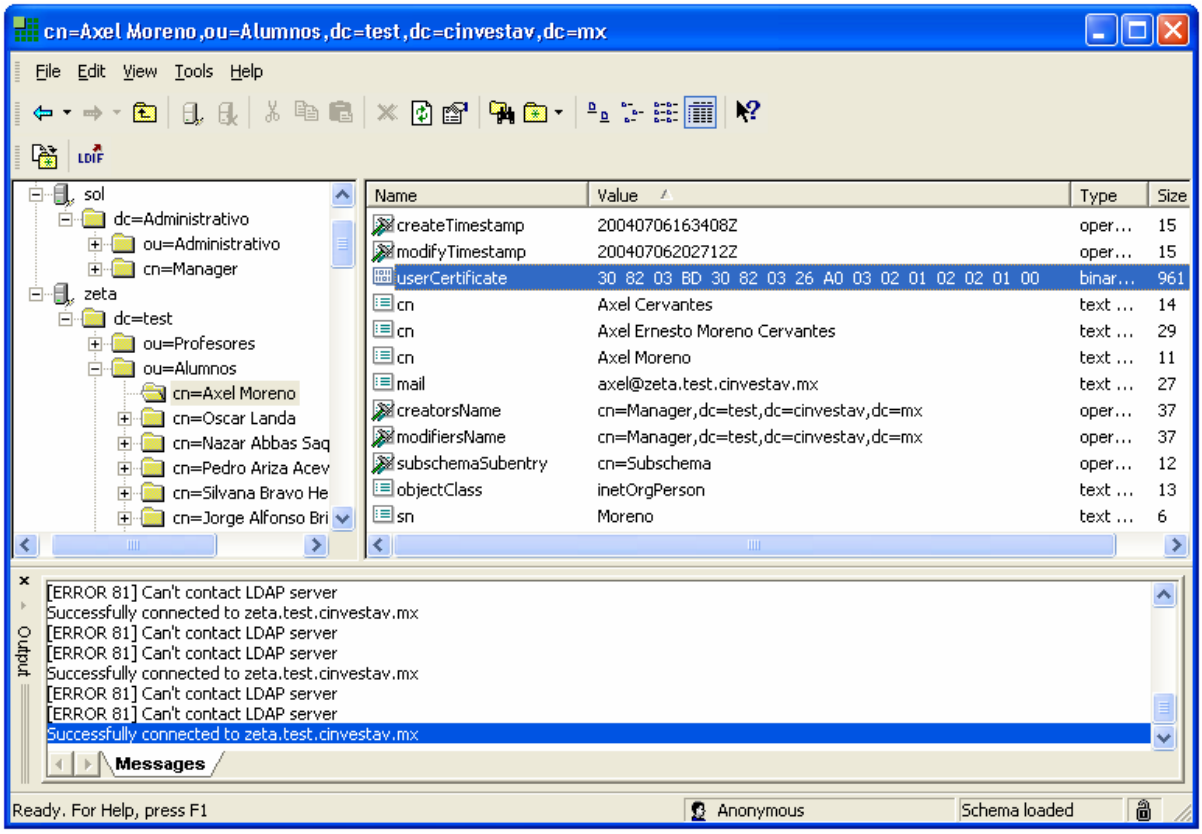


Figura 4.4 Uso de certificados digitales como atributos

A continuación se describen los demás componentes de esta prueba:

- Número de máquinas: 4
- Número de clientes por máquina: 30
- Plataforma servidor: Linux RedHat 8.0, Linux RedHat 9.0
- Plataforma clientes: Solaris (versión 9.0), Windows XP, Mac OS X (versión 10.2), Linux Red Hat (versión 9)
- Rango de transacciones: desde 100 hasta 1000 por cada cliente
- Incremento entre transacciones: 100
- Servidor LDAP: OpenLDAP versión 2.0.27
- Versión LDAP: LDAPv3
- API: JNDI (*Java Naming and Directory Interface*) del J2SDK 1.5 beta
- Clase de objetos utilizada en el directorio LDAP: inetOrgPerson. Utilizamos esta clase de objeto ya que nos permite manejar el tipo de atributo “Certificado Digital”
- Tipo de atributos utilizados en el directorio LDAP: nombre común (cn), apellido (sn), e-mail (mail), certificado de usuario (usercertificate).

4.8.2 Arquitectura de la prueba LDAP

En la figura 4.5 podemos ver la arquitectura de la red para nuestras pruebas con LDAP. Cada una de las máquinas conectadas a la red (clientes y servidores) cuenta con una dirección IPv4, una dirección IPv6-local la cual es proporcionada al configurar el *dual-stack* y otra IPv6-global que es asignada al configurar el *tunnel-broker*. Para poder hacer uso del mecanismo de transición *tunnel-broker* fue necesario configurar un cliente de Freenet6 en cada *hos*t. También podemos ver que el directorio se encuentra centralizado en el servidor LINUX

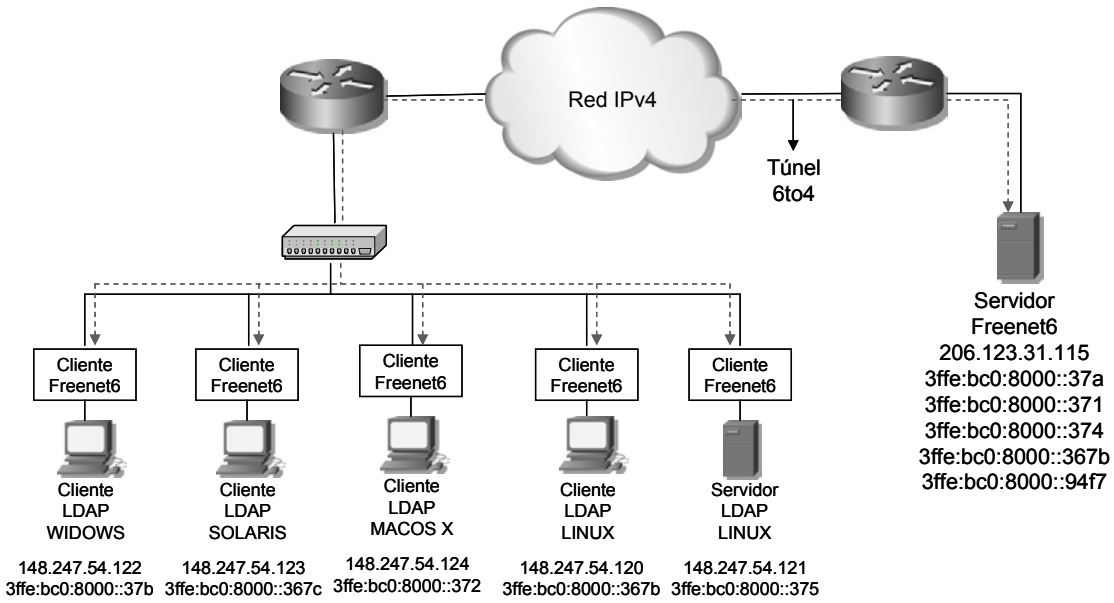


Figura 4.5. Arquitectura para la realización de pruebas utilizando IPv6 y LDAP

Cuando un cliente desea comunicarse con el servidor LDAP utilizando el mecanismo de transición *dual stack* el tratamiento que se le da a los datagramas IP es nativo sobre IPv6. En cambio, cuando la comunicación se realiza por medio del mecanismo *tunnel broker* es necesario utilizar túneles 6to4 para encapsular los datagramas IPv6 en paquetes IPv4 y que así puedan ser enrutados sobre una infraestructura IPv4.

4.8.3 Resultados obtenidos

Rendimiento

En la figura 4.6 observamos que para la operación de consulta al servidor LDAP, el rendimiento mostrado por el servidor cuando se utiliza el mecanismo de *tunnel-broker* está muy por debajo (cerca de 3.24% menor) de los mostrados con direcciones IPv4 e IPv6 con el mecanismo *dual-stack*, esto es porque para utilizar el mecanismo *tunnel-broker* es necesario agregar el encabezado de IPv4 al paquete IPv6. Lo anterior tiene dos efectos negativos: primero, disminuye el tamaño de carga útil de datos; segundo, agrega un factor adicional de tiempo, ya que es necesario hacer una desencapsulación del paquete IPv4 a uno del tipo IPv6. Cabe señalar, que cuando se utiliza el mecanismo *dual-stack* el tratamiento de los paquetes se hace de manera nativa.

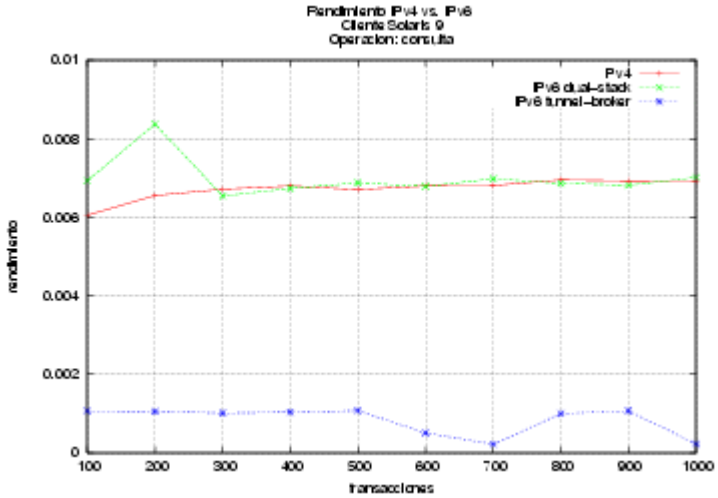


Figura 4.6. Rendimiento del servidor LDAP al realizar operaciones de consulta sobre IPv4 e IPv6 en Solaris 9

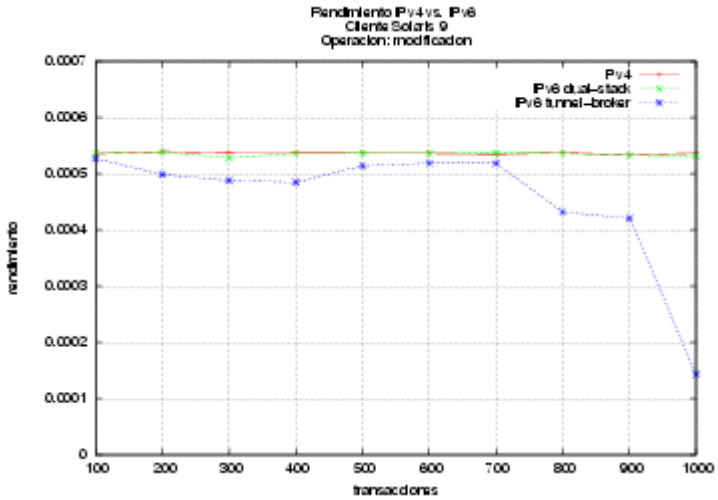


Figura 4.7. Rendimiento del servidor LDAP al realizar operaciones de modificación sobre IPv4 e IPv6 en Solaris 9

La figura 4.7 muestra que para la operación de modificación al servidor LDAP el rendimiento mostrado por dicho servidor usando el mecanismo *tunnel-broker* está muy cercano a los mostrados por IPv4 e IPv6 mediante *dual-stack*, esto es debido a que cuando se realizan operaciones de modificación al directorio, éste debe ejecutar mecanismos de control para mantener la consistencia de los datos contenidos en él, lo cual agrega un factor de retraso, mismo que se ve reflejado de mayor forma en el rendimiento mostrado con IPv4 e IPv6 mediante *dual-stack*.

En la figura 4.8 observamos que no hay una diferencia tan pronunciada en cuanto a rendimientos para la operación de consulta como en el caso de Solaris, debido a la manera en que Windows XP y Solaris implementan los *sockets*. Sin embargo, es necesario realizar mediciones más detalladas para confirmar plenamente lo anterior.

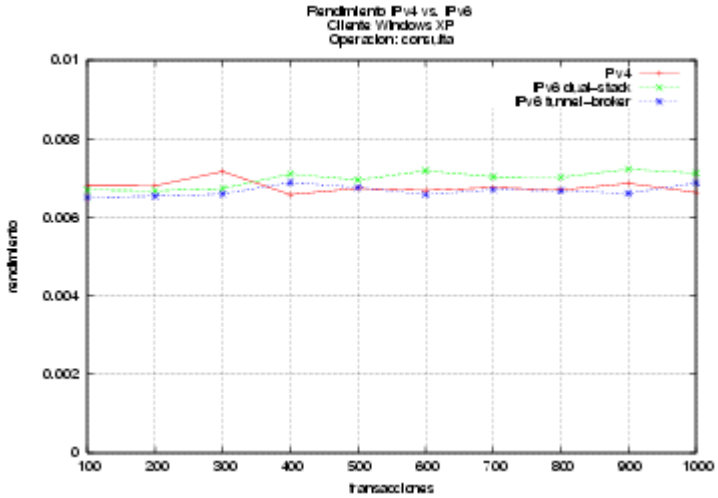


Figura 4.8. Rendimiento del servidor LDAP al realizar operaciones de consulta sobre IPv4 e IPv6 en Windows XP

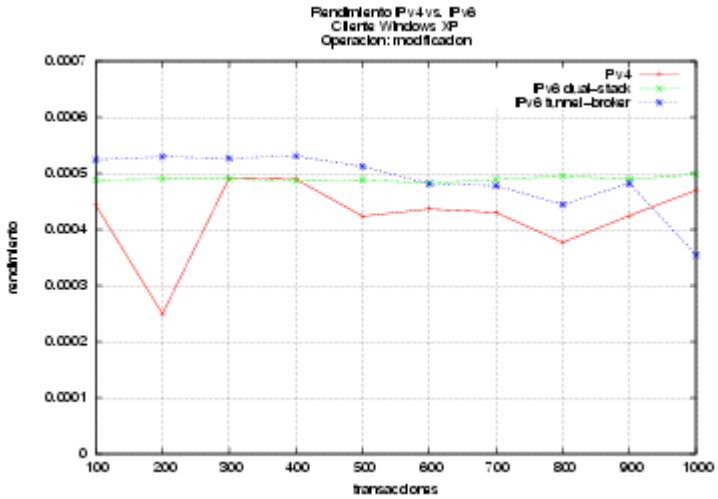


Figura 4.9. Rendimiento del servidor LDAP al realizar operaciones de modificación sobre IPv4 e IPv6 en Windows XP

La figura 4.9 muestra al igual que la figura 4.7, que para la operación de modificación al servidor LDAP el rendimiento mostrado por el servidor mediante el mecanismo *tunnel-broker* está muy cercano a los mostrados por IPv4 e IPv6 usando direcciones locales, lo cual confirma que cuando se realizan operaciones de modificación al directorio, el retardo adicional repercute de mayor manera en el rendimiento mostrado con IPv4 e IPv6 mediante el mecanismo *dual-stack*. Por último, en las figuras 4.10 y 4.11 solo se realizaron pruebas utilizando IPv4 e IPv6 mediante el mecanismo *tunel-broker* ya que al utilizar el mecanismo *dual-stack*(con direcciones IPv6 locales) surgieron problemas de interoperabilidad con JNDI; a pesar de esto, los resultados presentados muestran patrones de comportamiento muy similares a los presentados con las otras plataformas.

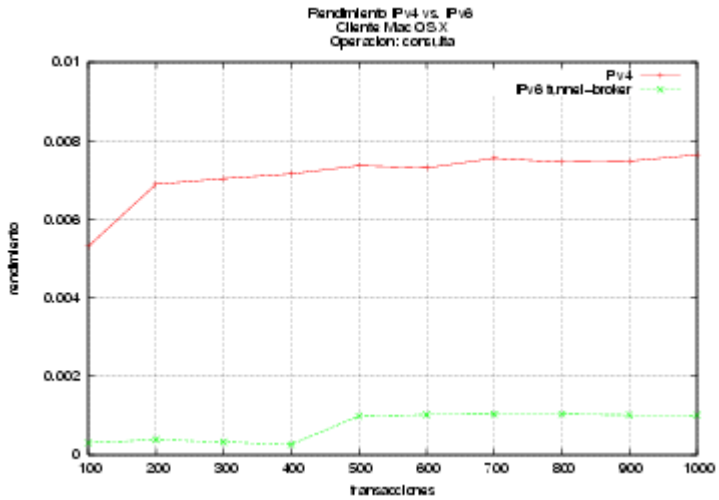


Figura 4.10. Rendimiento del servidor LDAP al realizar operaciones de consulta sobre IPv4 e IPv6 en Mac OS X 10.2

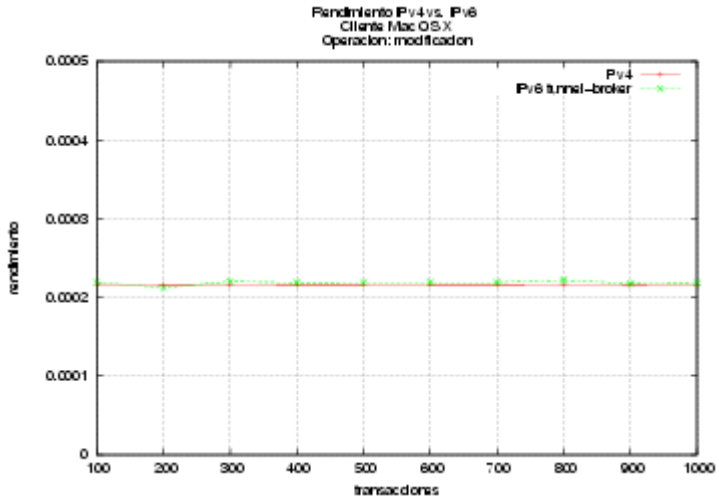


Figura 4.11. Rendimiento del servidor LDAP al realizar operaciones de modificación sobre IPv4 e IPv6 en Mac OS X 10.2

Latencia

Las figuras 4.12 – 4.17 muestran resultados de las mediciones de la latencia de respuesta en función de de la carga del servidor LDAP. En la figura 4.12 la latencia producida cuando se consulta el servidor LDAP utilizando como protocolo de red IPv4 así como el protocolo IPv6 mediante el mecanismo de dual-stack son muy similares y permanecen casi constantes, sin embargo cuando se utiliza el mecanismo de *tunnel-broker* la latencia aumenta considerablemente (cerca de 2900% más cuando se realizan 700 consultas) debido entre otras cosas al trafico existente en Internet, así como el factor de retardo agregado por el proceso de encapsulado y desencapsulado de paquetes IPv6 en IPv4. El mecanismo de *tunnel broker* agrega además un factor de retardo adicional provocado por el algoritmo de Nagle definido en el RFC 896 [27] y que está implementado en todas las redes TCP/IP con la finalidad de controlar la congestión de tráfico con la desventaja de agregar un retardo adicional. Las repercusiones del algoritmo de Nagle son más contundentes utilizando este mecanismo de transición ya que el cliente y el servidor tienen que dar más saltos para alcanzarse uno al otro y en cada salto se aplica el algoritmo de Nagle.

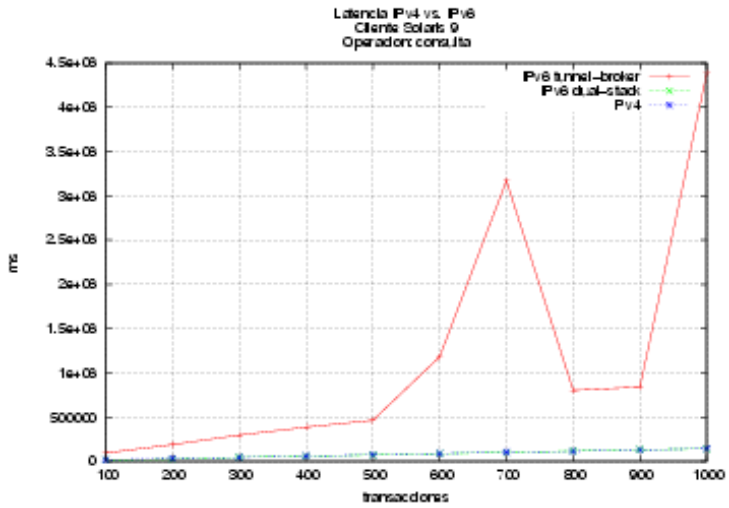


Figura 4.12. Latencia de respuesta del servidor LDAP al realizar operaciones de consulta sobre IPv4 e IPv6 en Solaris 9

En la figura 4.13 al igual que en la figura 4.12 las latencias utilizando IPv6 mediante el mecanismo de *dual stack* e IPv4 son muy similares, pero a diferencia del caso anterior sus latencias se incrementan conforme aumenta la carga del servidor LDAP. El incremento en la latencia se debe a que se genera un cuello de botella en el *backend* de la base de datos utilizada por el servidor LDAP (LDBM) cuando se modifican entradas en el directorio a medida que aumenta la carga en éste. LDBM implementa mecanismos de seguridad para mantener la consistencia de la información en las entradas del directorio mediante el bloqueo de los archivos LDIF que contienen la información. Este método de bloqueo es el principal causante del cuello de botella.

En la figura 4.13 también se puede ver que la latencia es más uniforme cuando se realizan operaciones de modificación al servidor LDAP que cuando se realizan operaciones de consulta porque como la comunicación IPv6 mediante el mecanismo de *tunnel-broker* es más lenta que en los otros dos casos este tiempo adicional da pie a que se tenga un espaciamiento suficiente entre mensajes disminuyendo el número de colisiones en la red.

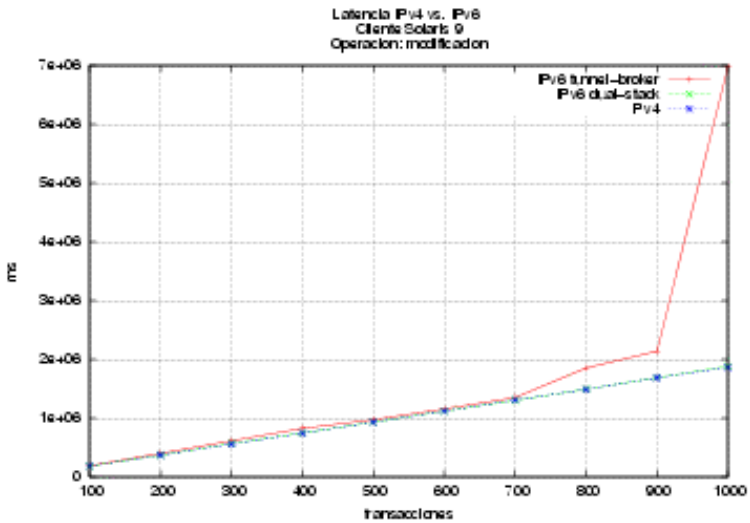


Figura 4.13. Latencia de respuesta del servidor LDAP al realizar operaciones de modificación sobre IPv4 e IPv6 en Solaris 9

La figura 4.14 muestra las latencias generadas cuando se realizan consultas al servidor LDAP desde clientes Windows. Las latencias en este caso son muy uniformes a diferencia de las producidas con los

clientes Solaris y esto es principalmente por la manera en que están implementados los *sockets* en ambas plataformas, sin embargo podemos ver que al igual que en la figura 4.12 la latencia generada cuando se usa una mecanismo de *tunnel-broker* es mayor que utilizando el mecanismo de dual-stack ó utilizando IPv4.

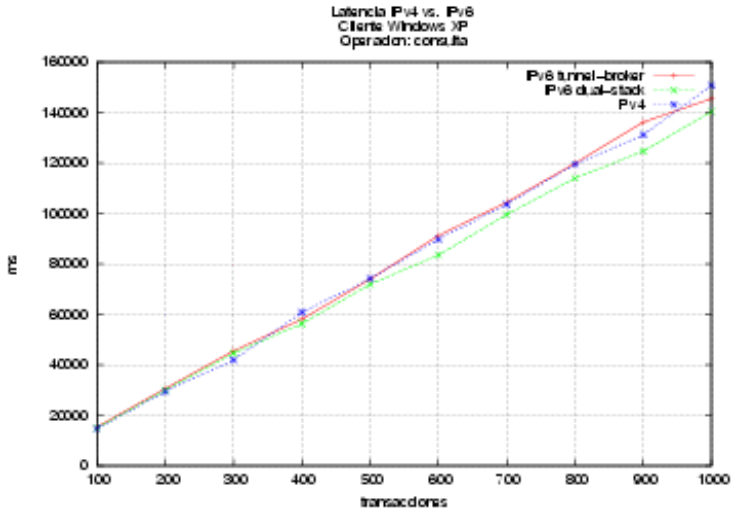


Figura 4.14. Latencia de respuesta del servidor LDAP al realizar operaciones de consulta sobre IPv4 e IPv6 en Windows XP

La figura 4.15 muestra cierta cantidad de picos en las mediciones de la latencia utilizando IPv4 las cuales atribuimos a la carga del servidor al momento de realizar las pruebas e indican que también es importante la cantidad de tráfico dentro de la red de pruebas para aumentar o disminuir la latencia en las mismas, a pesar de esto no notamos una gran diferencia en la latencia al utilizar como protocolo de red IPv6 o IPv4 mientras el número de transacciones es menor de 900 por cliente, en cuanto se supera este número el impacto de los factores mencionados en la figura 4.13 nuevamente toman efecto aquí.

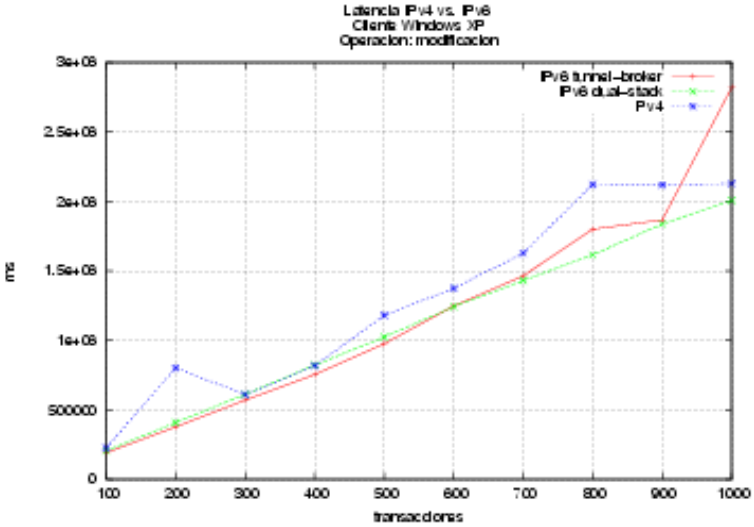


Figura 4.15. Latencia de respuesta del servidor LDAP al realizar operaciones de modificación sobre IPv4 e IPv6 en Windows XP

Para los resultados mostrados en las figuras 4.16 y 4.17 solo fue posible evaluar el comportamiento del servidor LDAP utilizando como protocolos de red IPv6 (mediante *dual-stack*) e IPv4, ya que la forma en la que está implementado el stack IPv6 en Mac OS X10.2 no nos permitió la comunicación LDAP

mediante el mecanismo *dual-stack* con direcciones locales. En la figura 4.16 podemos apreciar una gran diferencia entre las latencias generadas por el uso de un stack IPv4 o uno IPv6 esto nos hace pensar que la implementación del stack IPv6 de Mac OS X 10.2 la cual desciende de un stack desarrollado por el grupo KAME para BSD de es muy similar a la implementación de Solaris.

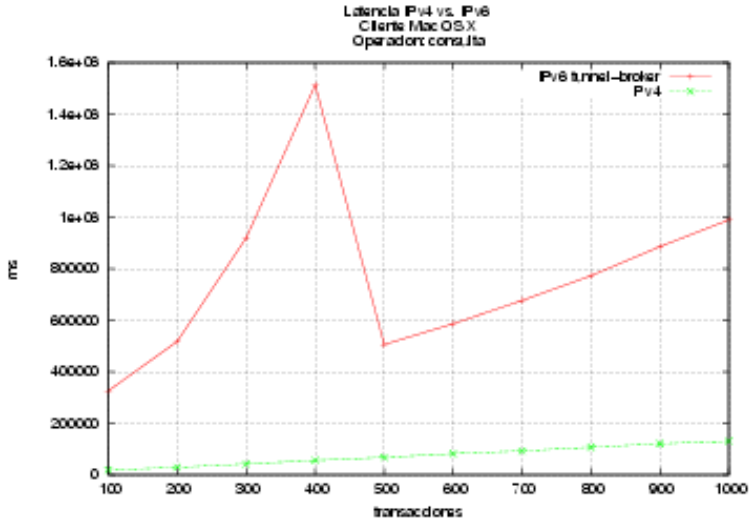


Figura 4.16. Latencia de respuesta del servidor LDAP al realizar operaciones de consulta sobre IPv4 e IPv6 en Mac OS X 10.2

Por último en la figura 4.17 al igual que en la figura 4.13 se muestra una latencia uniforme entre los clientes utilizando cualquiera de los dos protocolos de red, excepto que a una escala más alta.

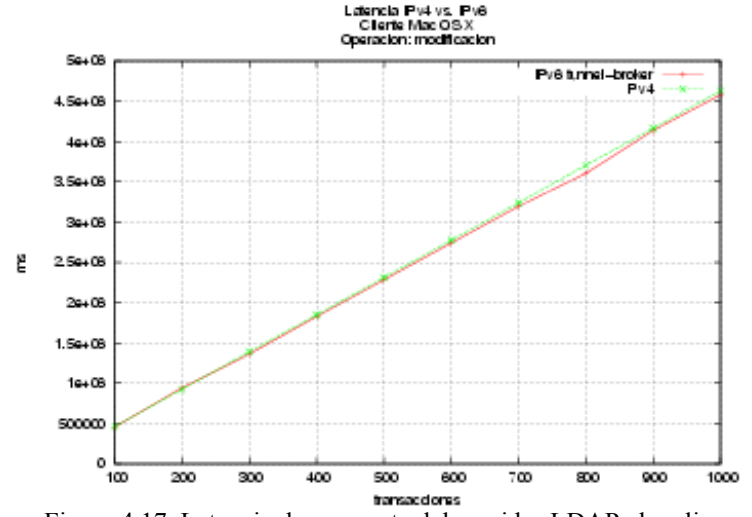


Figura 4.17. Latencia de respuesta del servidor LDAP al realizar operaciones de modificación sobre IPv4 e IPv6 en Mac OS X 10.2

Después de analizar estas gráficas se desarrolló un modelo que se aproximara al comportamiento del servidor LDAP en base a su rendimiento, con el fin de poder predecir el punto en el que el servidor se satura y deja de atender las peticiones LDAP sobre IPv4 e IPv6.

4.8.4 Modelo de comportamiento del servidor LDAP

Como mencionamos en la sección anterior tratamos de obtener un modelo capaz de reproducir el comportamiento del servidor LDAP en cuanto a su rendimiento, con la finalidad de poder predecir la carga máxima soportada por éste y evaluar si acaso influye el protocolo de red utilizado para llevar a cabo la comunicación entre un cliente y el servidor LDAP con la carga máxima soportada por éste último.

Comenzamos definiendo el conjunto de muestras obtenidas de las evaluaciones como una lista de parejas $T = \{(t_i, r_i)\}_{i=1}^{n}$ en donde, para cada *i*, t_i es el *i*-ésimo número de transacciones y r_i es el rendimiento medido asociado a t_i. Precisamente r_i puede expresarse como:

$$r_i = t_i \left[\frac{1}{k} \sum_{j=1}^{k} u_j \right]^{-1} \tag{1}$$

donde *k* es el número de hilos realizando operaciones sobre el directorio LDAP de manera concurrente y cada u_j es el tiempo de respuesta obtenido al realizar t_i operaciones sobre el directorio LDAP mediante el hilo *j*. Como un modelo general para representar la dependencia entre los rendimientos medidos (r_i)del servidor LDAP y las cargas del servidor (t_i), según se ve en las figuras 4.4-4.9, proponemos una función de la forma:

$$r(t) = Ae^{[p_3(t)]} + b \tag{2}$$

donde $p_3(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0$ es un polinomio cúbico real y su coeficiente principal a_3 ha de ser negativo , A > 0 es una constante y b > 0 es un límite asintótico del rendimiento cuando el número de transacciones se incrementa arbitrariamente.

La función en (2) tiene un carácter más bien arbitrario y solo tiene la intención de describir analíticamente la dependencia del rendimiento respecto a las transacciones.

Al imponer que el coeficiente principal en el polinomio $p_3(t)$ sea negativo, esto es $a_3 < 0$, se garantiza que $Ae^{p_3(t)}$ tienda a ser nulo cuando *t* se incremente arbitrariamente, es decir, se tendrá $r(t) \to b$ cuando $t \to +\infty$, lo que corresponde a la observación de que el servidor ha de alcanzar un rendimiento constante a la larga y ese valor es precisamente *b*. La constante A hace las veces de "factor de escala", y queda determinada por el valor del máximo rendimiento obtenido en nuestras mediciones

Para ajustar una muestra de datos $T = \{(t_i, r_i)\}_{i=1}^{n}$ al modelo propuesto en la ecuación (2) vamos a considerar a *b* como un parámetro de control. Tomando logaritmos en la ecuación (2) el problema de ajuste queda reducido ahora a ajustar el siguiente conjunto de datos:

$$S_b = \{(t_i, s_i = \ln(\frac{r_i - b}{A_b}))\}_{i=1}^{n} \tag{3}$$

a un polinomio real de grado 3. Así entonces, sea $A_b = (\max\{r_i \mid i \le n\}) - b$ y sea $p_{3b}(t) = a_{3b} t^3 + a_2 t^2 + a_{1b} t + a_{0b}$ el polinomio cúbico que ajusta a los valores en la ecuación (3) mediante una aproximación por el método de Mínimos Cuadrados (LSA).

Ya que hemos utilizado al parámetro b como uno de control, experimentalmente hemos encontrado que hay un valor umbral $b_0 > 0$ tal que

$$b > b_0 \Rightarrow a_{3b_0} < 0$$

$$b < b_0 \Rightarrow a_{3b_0} > 0$$

y además

$$\lim_{b \to b_0} a_{3b} = +\infty \quad \text{y} \quad \lim_{b_0 \leftarrow b} a_{3b} = -\infty \tag{4}$$

En otras palabras, b_0 es un polo en el cual el ajuste de la muestra de datos T mediante la función definida en la ecuación (2) no es posible. Así que tomando en cuenta que cualquier valor que tome el parámetro b menor a b_0 dará un valor positivo del coeficiente a_{3b} mientras que cualquier valor que tome b mayor a b_0 dará un valor negativo de a_{3b}. Realizando un procedimiento de bisección, pudimos determinar el valor del umbral b_0. Debido a lo antes mencionado el valor obtenido por el método de bisección solo aproximará a un valor cercano a b_0, volviendo por consiguiente el método LSA más inestable en esas cercanías. De aquí que el método de bisección es detenido cuando la aproximación de b está suficientemente cercana a b_0 basándonos en el cálculo de un error propuesto •.

$$\varepsilon = \sqrt{\sum_{i=1}^{n} |r_i - r(t_i)|^2} \tag{5}$$

donde $r(t_i)$ es el rendimiento obtenido mediante la aproximación por el modelo propuesto.

En las figuras 4.18 - 4.25 mostramos ejemplos de los resultados obtenidos en las aproximaciones para cada una de las plataformas (la línea clara con puntos claros muestra los rendimientos medidos, la línea obscura con puntos negros muestra los rendimientos obtenidos mediante el modelo propuesto), asimismo mostramos los diferentes tamaños de carga b que de acuerdo al modelo propuesto es el mínimo rendimiento medible en el servidor LDAP.

En esos resultados observamos que los primeros dos coeficientes a_0, a_1 del polinomio $p_3(t)$ son los que tienen valores significativos por lo que una aproximación similar al de la ecuación (2) pero utilizando sólo un polinomio lineal podría ser también conveniente. Sin embargo, las aproximaciones que obtenemos con un polinomio cúbico logran una mejor aproximación. Insistimos en que el modelo en la ecuación (2) es puramente formal, nos permite predecir y aproximar valores intermedios en nuestras pruebas experimentales y seguramente podría ser susceptible de una interpretación física de los parámetros involucrados. Por el momento nos abstenemos de buscar tal interpretación y nos restringimos a considerarlo de manera puramente formal.

b= 5.445312500000001E-4
A= 5.3929E-4
a_0= 0.6577689682111982
a_1= -0.003168285556226816
a_2= 7.518465274737212E-6
a_3= -4.548400822134843E-9
e= 1.6177045038707467E-11

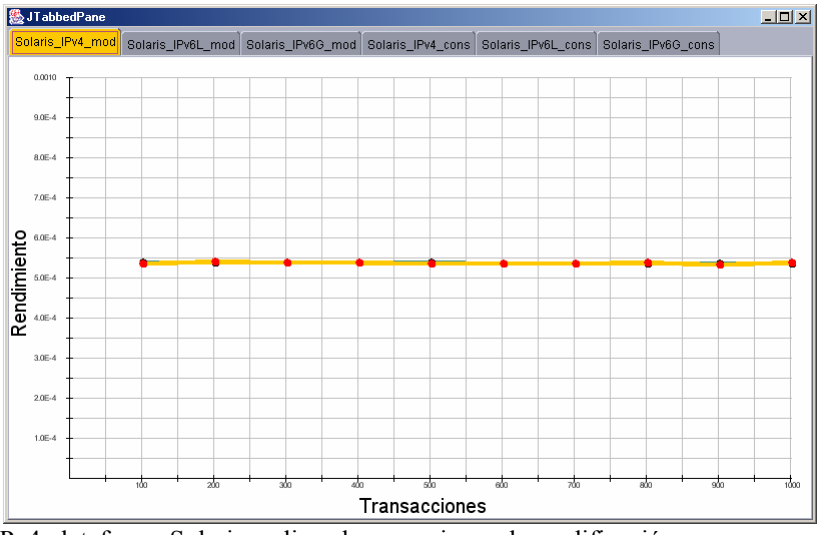


Figura 4.18 Rendimiento de IPv4 plataforma Solaris realizando operaciones de modificación

b= 0.0010
A= 5.2679E-4
a_0= -0.17639276626517242
a_1= 0.0021975645529514515
a_2= -5.644976423705252E-6
a_3= 4.191461731387585E-9
e = 4.629498804536092E-9

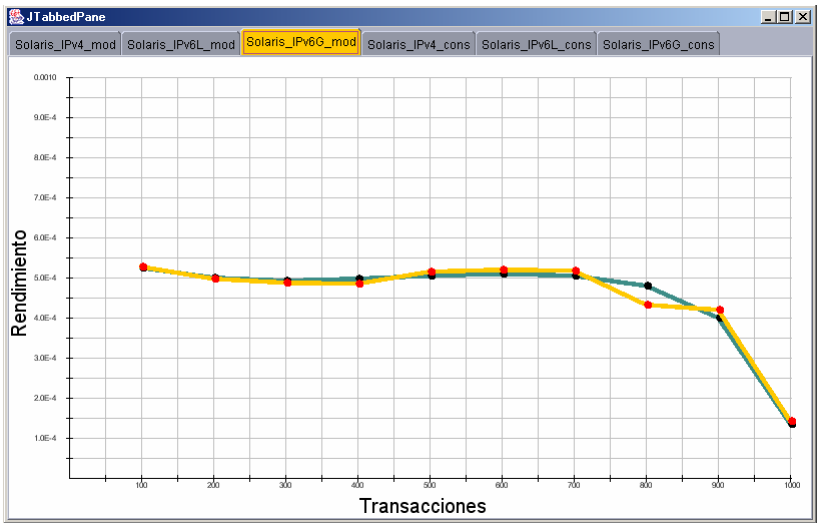


Figura 4.19 Rendimiento de IPv6 plataforma Solaris realizando operaciones de modificación mediante el mecanismo tunnel-broker

b= 0.0010
A= 5.3852E-4
a_0= -0.008000725896544816
a_1= 1.274330404605126E-4
a_2= -2.832709599897695E-7
a_3= 1.8048723666952627E-10
e = 5.722980650122386E-11

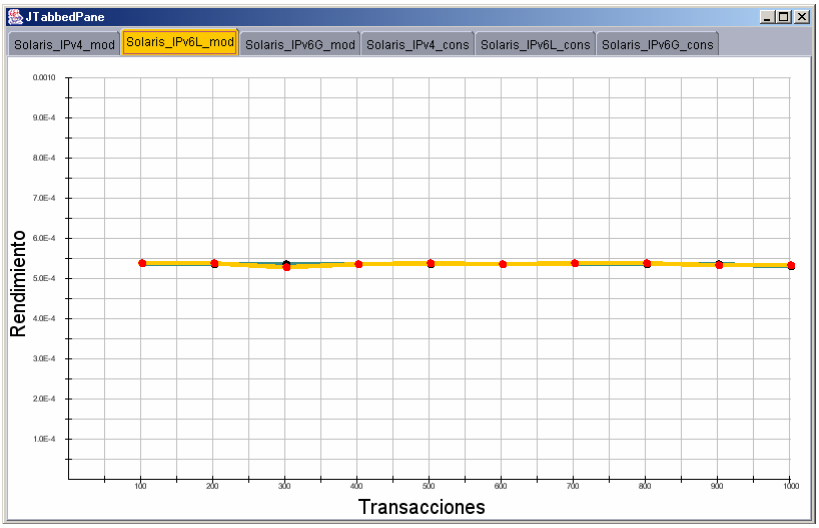


Figura 4.20 Rendimiento de IPv6 plataforma Solaris realizando operaciones de modificación mediante el mecanismo dual-stack

b= 3.5312500000000003E-4
A= 4.91721E-4
a_0= 3.7129580550195502
a_1= -0.019943120584817 61
a_2= 4.212859319170818E-5
a_3= -2.491481163051428E-8
e = NaN

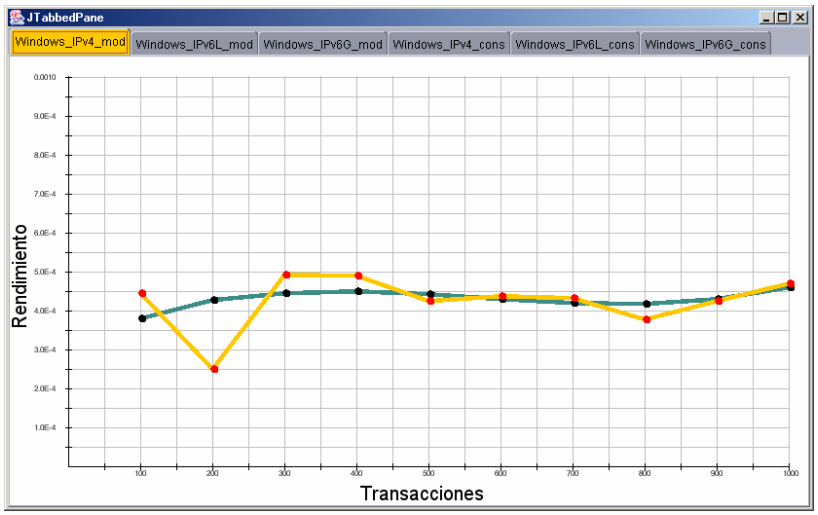


Figura 4.21 Rendimiento de IPv4 plataforma Windows realizando operaciones de modificación

b= 6.000000000000001E-4
A= 5.30816E-4
a_o= 0.13499847364618692
a_1= -0.0010962687780983406
a_2= 3.1281907889260908E-6
a_3= -1.0751231900923874E-9
e = NaN

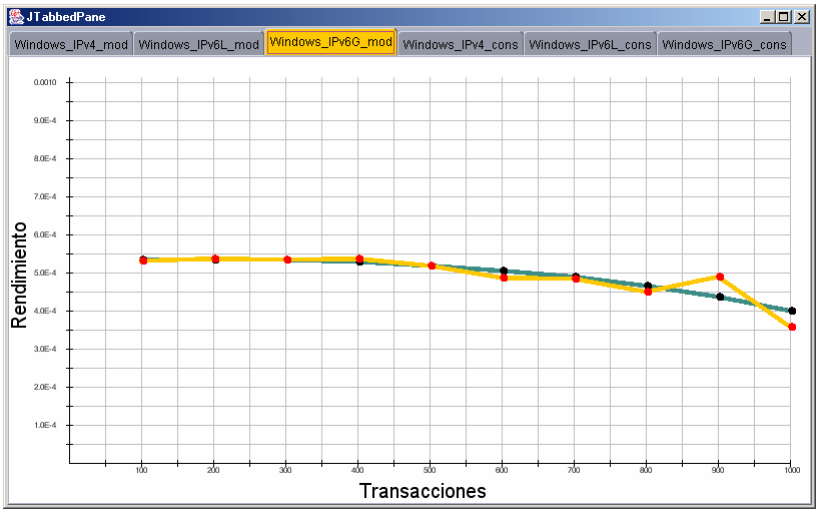


Figura 4.22 Rendimiento de IPv6 plataforma Windows realizando operaciones de modificación mediante el mecanismo tunnel-broker

b= 5.0E-4
A= 4.98205E-4
a_o= 2.124470247493843
a_1= -0.00395049713117963
a_2= 1.2263030261430844E-5
a_3= -1.0220532640482892E-8
e = NaN

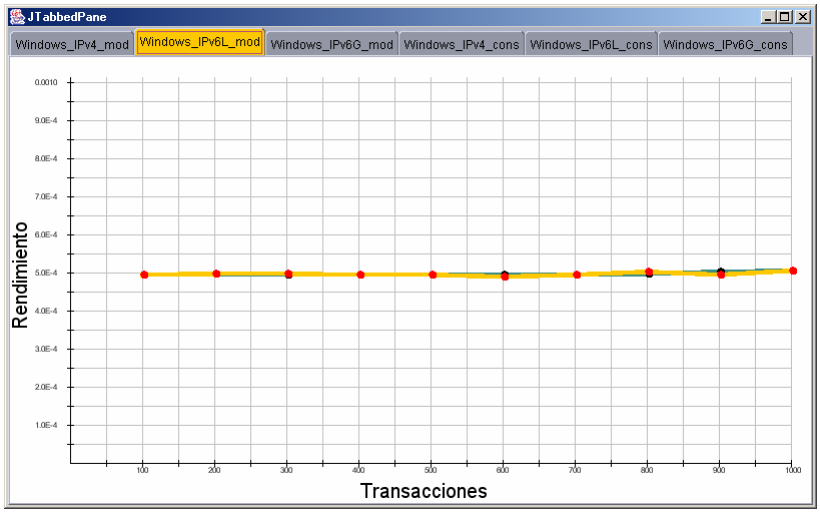


Figura 4.23 Rendimiento de IPv6 plataforma Windows realizando operaciones de modificación mediante el mecanismo dual-stack

b= 0.00755
A= 0.007640612
a_0= 0.9841418376250158
a_1= -0.004129470738692142
a_2= 6.288522452225461E-6
a_3= -3.1536619192409786E-9
e = NaN

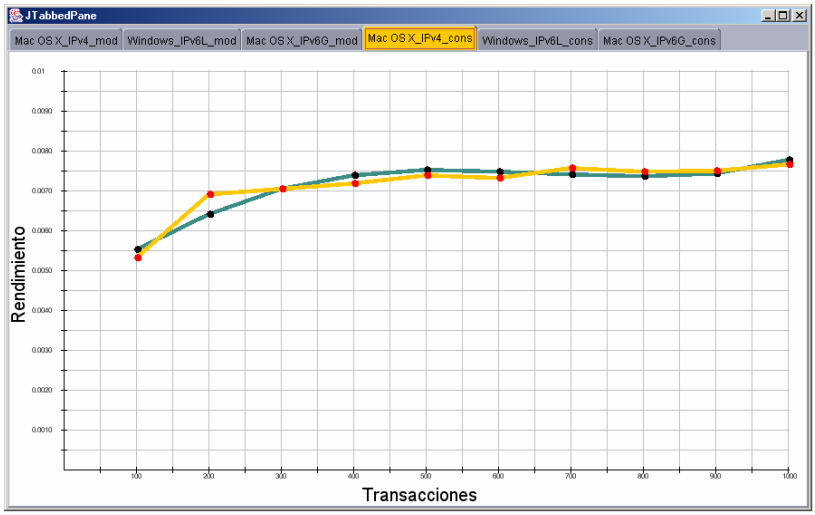


Figura 4.24 Rendimiento de IPv4 plataforma Mac OS X realizando operaciones de consulta

b= 0.009961718750000001
A= 0.001035514
a_0= 0.06404302146776966
a_1= 2.271130302334688E-4
a_2= -8.751881935778815E-7
a_3= 5.949000027846784E-10
e = 1.904780834666348E-7

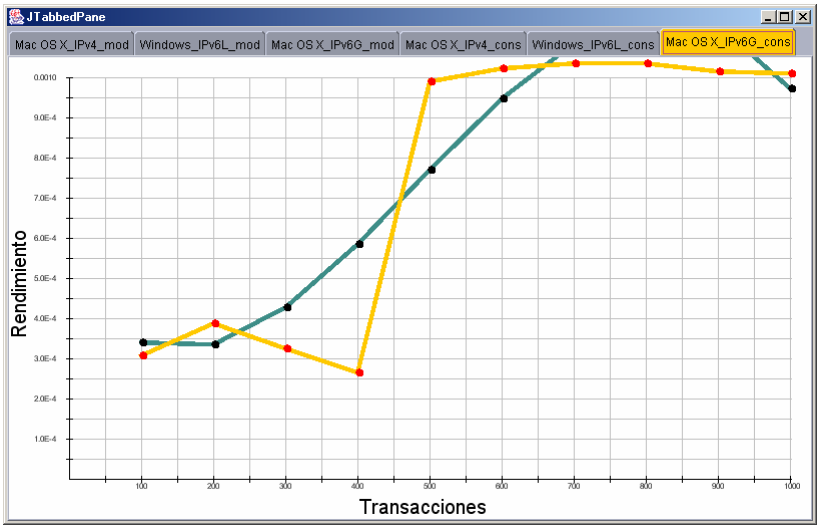


Figura 4.25 Rendimiento de IPv6 plataforma Mac OS X realizando operaciones de consulta mediante el mecanismo tunnel-broker

Capítulo 5

IP Security (IPSec)

5.1 Introducción

Los paquetes IP por si mismos no cuentan con seguridad y por tanto es relativamente fácil inspeccionar su contenido, modificarlo y reenviarlos cuando transitan por la red, por lo que al recibir un paquete IP no hay garantía de que:

a) El emisor del paquete sea quien dice ser.
b) Contenga los datos originales que el emisor colocó en él
c) El paquete no fue inspeccionado en el camino desde el emisor hasta el destino.

IPv6 por su parte basa su seguridad en el protocolo IPSec tal como se describe en el RFC 2460[28] mediante dos encabezados de extensión que pueden ser utilizadas juntas o independientemente para proveer distintos servicios de seguridad en un paquete IPv6.

IPSec brinda los siguientes servicios de seguridad sobre la capa de red:

- Confidencialidad de Datos: El nodo emisor puede cifrar los paquetes antes de transmitirlos a través de la red.
- Integridad de Datos: El nodo receptor puede autenticar los paquetes enviados por el nodo emisor para asegurar que los datos no han sido alterados durante la transmisión.
- Autenticación del origen de los Datos: El nodo receptor puede autenticar la fuente de los paquetes enviados para asegurar que dichos paquetes fueron enviados por quien dice ser el nodo emisor.
- Anti-reenvío: El nodo receptor puede detectar y rechazar paquetes reenviados.

Para poder brindar dichos servicios es necesario establecer algunos parámetros de seguridad entre las partes que se van a comunicar de manera segura, estos parámetros son definidos dentro de una Asociación de Seguridad.

5.2 Asociaciones de Seguridad (SA)

Cuando dos *hosts* desean comunicarse entre si utilizando los mecanismos de seguridad que IPv6 proporciona es necesario que estos establezcan primero algunos parámetros como son: una llave, el algoritmo de autenticación o encriptación que utilizarán, el tiempo de vida de las llaves, además de algunos parámetros adicionales específicos a dichos algoritmos utilizados. A todos estos parámetros en conjunto se les conoce como una Asociación de Seguridad (SA). Las SA's son unidireccionales y es necesaria una SA por cada servicio de seguridad a utilizar (autenticación / encriptación). Así por ejemplo, si dos hosts, Host-A y Host-B se comunican seguramente utilizando ESP, Host-A tendrá una SA (SA_{salida}) para procesar los paquetes salientes hacia Host-B, así como otra SA ($SA_{entrada}$) para procesar los paquetes entrantes. Host-B también creará dos SA's para procesar sus paquetes. La SA_{salida} de Host-A y la $SA_{entrada}$

de Host-B compartirán los mismos parámetros criptográficos, tales como las llaves. Del mismo modo, $SA_{entrada}$ de Host-A y SA_{salida} de Host-B compartirán los mismos parámetros criptográficos.

Existen dos tipos de SA:

- Modo transporte: La SA es definida entre dos sistemas terminales y se encarga de proveer ya sea autenticación y/o encriptación para la carga contenida en todos los paquetes relacionados a esa conexión.

- Modo túnel: La SA es definida entre dos *gateways* de seguridad y funciona de la siguiente manera: Los paquetes IP y la carga son encapsulados dentro de un nuevo paquete IP de salida "*wrapper*" y así poder proporcionar ya sea autenticación y/o encriptación al paquete IP completo incluyendo al encabezado IP.

Cualquier implementación de IPSec tiene una base de datos de Asociaciones de Seguridad (SADB) que mantiene las Asociaciones de Seguridad para cada túnel de comunicación establecido mediante IPSec y una base de datos de Políticas de Seguridad (SPD) que trabaja en conjunto con la SADB para procesar paquetes IPSec. En SPD se define que protocolos usar (AH/ESP), en que modos usarlos (Túnel/Transporte) y las transformaciones a usar (AH con MD5, ESP con SHA y DES, etc.). También existe otro componente importante en IPSec llamado Índice de Parámetro de Seguridad (SPI) que es un valor de 32 bits utilizado por IPSec para identificar una Asociación de Seguridad determinada entre dos hosts que se comunican de manera segura. Cuando un host envía un paquete utilizando servicios de seguridad, éste además envía dentro del paquete el SPI para que el nodo receptor pueda buscar dentro de su SADB las SA's utilizadas por el nodo emisor y validarlas. Después que una SA ha expirado el SPI puede ser nuevamente utilizado en otra SA. Para garantizar que cada SA sea única, esta se valida mediante la tripleta <SPI, dirección IP destino, protocolo> donde protocolo indica el protocolo de transporte utilizado, solo se utiliza ESP el protocolo de transporte no es accesible y se utilizan comodines para llenar este campo.

La administración de las SA's se basa principalmente en crear y borrar asociaciones de seguridad y esta puede ser llevada a cabo de dos maneras: de forma manual o mediante el uso de un protocolo de administración de llaves de Internet como es el caso de IKE.

5.3 Procesamiento de IPSec

El procesamiento de paquetes que realiza IPSec comienza a partir de que los paquetes de datos a ser enviados llegan a la capa IP. Este procesamiento se clasifica en Procesamiento Entrante y Procesamiento Saliente y a continuación se mencionan ambos.

5.3.1 Procesamiento entrante

Una vez que la capa IP es invocada por la capa de Enlace de Datos para procesar un paquete recibido por la interfaz de red, esta quita el encabezado IP de dicho paquete e invoca a la capa IPSec para que procese el paquete cuyo siguiente encabezado es ya sea AH o ESP. El procesamiento del paquete se lleva de la siguiente manera:

- La capa IPSec extrae el SPI del encabezado ya sea AH o ESP, las direcciones IP fuente, destino y protocolo del encabezado IP ya que estos tres valores son utilizados para identificar la SA utilizada para este paquete.
- IPSec busca en la SADB tratando de encontrar la SA asociada a este paquete. Si no encuentra dicha SA el paquete es borrado.

- Si la SA es encontrada, esta es utilizada por IPSec para procesar el paquete
- Las políticas de seguridad a evaluar para este paquete son obtenidas de la SPD mediante una consulta utilizando los selectores (SPI, direcciones IP fuente y destino, protocolo).
- En este punto puede surgir alguno de los siguientes errores, en cuyo caso el paquete será borrado:
  - La autenticación falla
  - La opción de protección anti-reenvío está habilitada, pero el paquete falla la prueba
  - El tiempo de vida de la SA ha expirado
  - En caso de que se haya utilizado ESP como protocolo y el paquete esté cifrado. Puede que al decriptarlo el campo de longitud de relleno tenga una longitud errónea
- En caso de que se pasen todas las pruebas el paquete es aceptado

5.3.2 Procesamiento Saliente

El procesamiento saliente se realiza en el dispositivo que funge como emisor de paquetes con servicios IPSec. Cuando un paquete va a ser enviado utilizando los servicios de Seguridad AH o ESP la Capa de Transporte envía dichos paquetes a la Capa de Red junto con las direcciones Fuente, Destino y el protocolo que se pondrá en el campo "siguiente encabezado" del paquete IP. Después es consultada la SPD para ver que servicios de seguridad serán aplicados a este paquete (AH, ESP). La consulta se realiza utilizando las direcciones IP Fuente y Destino, así como los selectores tomados del encabezado de la capa de transporte (protocolo, puerto). Esta consulta puede arrojar uno de los siguientes resultados:

- Si la política aplicada a este paquete indica que puede ser transmitido sin ningún procesamiento de seguridad, entonces el paquete IP es transmitido en claro, es decir, sin protección.
- Si la política indica que el paquete necesita seguridad el motor de políticas revisa si las SA's entre emisor y receptor ya fueron establecidas. En caso de que las SA's aún no hayan sido establecidas se notifica a IKE para establecerlas.

Una vez establecidas las SA's estas son procesadas de la siguiente manera:

- Si la SA utiliza el número de bytes como su tiempo de vida entonces el número de bytes es incrementado dependiendo del tamaño de la carga. En el caso de ESP se incrementa el número de bytes cifrados.
- Si después de hacer esto el tiempo de vida de la SA ha expirado, esta es borrada y se invoca a IKE para establecer una nueva.

Después es incrementado el campo número de secuencia con la finalidad de prevenir ataques de reenvío de paquetes. Una vez hecho esto, se agregan los componentes específicos a cada protocolo (AH, ESP) y el paquete es enviado. En el caso de utilizar una configuración IPSec en modo túnel es necesario construir un nuevo encabezado IP con las direcciones Fuente y Destino de los nodos en los que se tienen habilitados los servicios de seguridad IPSec entre los nodos Fuente y Destino de los paquetes.

5.4 Encabezado de autenticación (AH)

AH provee autenticación del origen de los datos e integridad de datos mediante un algoritmo de Códigos de Autenticación de Mensajes mediante funciones Hash (HMAC). Ya que AH no provee confidencialidad tampoco requiere algoritmos de cifrado. Puede ser utilizado para proteger protocolos de las capas superiores si reconfigura en modo transporte o puede proteger un paquete IP completo si se configura en modo túnel. Cuando AH es utilizado para proteger un paquete IP, el encabezado AH es insertado después de los encabezados de extensión cuyo valor puede cambiar durante el camino hasta llegar al destino final del paquete. Esto incluye los encabezados hop-by-hop, enrutamiento y fragmentación. El encabezado que antecede a AH debe contener el valor 51 en el campo "Siguiente encabezado". Cuando Hay ESP son

utilizados en el mismo paquete el encabezado ESP siempre deberá ir antes que el de AH. En la figura 5.1 se muestra el encabezado de AH. Como se puede apreciar su encabezado es más sencillo que el de ESP ya que no necesita los campos de relleno, ni longitud de relleno porque este no provee confidencialidad de datos.

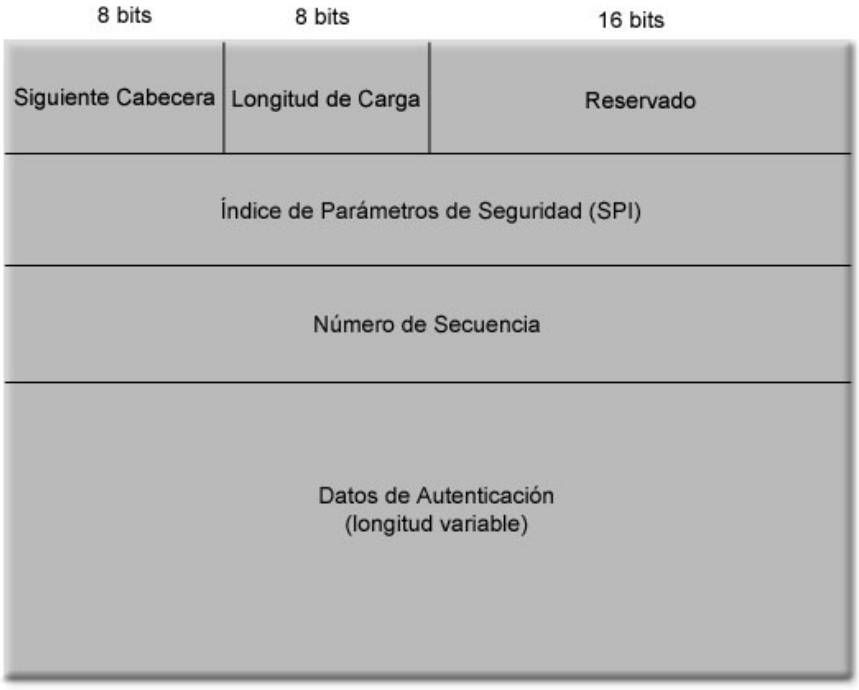


Figura 5.1. Formato del Encabezado de Autenticación

El campo "Siguiente Encabezado" tiene un tamaño de 8 bits e indica el encabezado que sigue inmediatamente después de AH. El siguiente campo es "Longitud de Carga", tiene 8 bits de longitud y provee la longitud del campo de autenticación en palabras de 32 bits, menos dos (los primeros 64 bits del encabezado de Autenticación no son contados). El valor mínimo que puede tomar este campo es 1 el cual equivale a 3 palabras de 32 bits y es solamente utilizado para propósitos de depuración. El siguiente campo está reservado para uso futuro, tiene un tamaño de16 bits y se inicializaron un valor de cero. El siguiente campo es "SPI"el cual tiene 32 bits de longitud y es utilizado junto con la dirección IPv6 destino y el protocolo para identificar la SA a utilizar para procesar dicho paquete.

El siguiente campo es "Número de Secuencia" de 32 bits de longitud, éste contiene un número que es único y monotónicamente auto incrementado para prevenir ataques de reenvío de paquetes. Cuando una SA acaba de ser establecida entre el emisor y el receptor de un paquete este valor es puesto a cero por ambas partes. Por último, el campo "Datos de autenticación" tiene una longitud variable y contiene el resultado del algoritmo de chequeo de integridad (ICV) por lo general es el resultado del cálculo de una función *hash*. Su tamaño depende del algoritmo de chequeo utilizado.

AH puede trabajar en dos modos distintos de operación: modo túnel y modo transporte y son explicados a continuación.

5.4.1 AH modo Túnel

Cuando AH es utilizado en modo Túnel, este protege a todo el paquete IPv6 colocando un nuevo encabezado IPv6 con las direcciones de los extremos del túnel IPSec. Después son insertados los encabezados de extensión e inmediatamente después de éstos AH es insertado de modo que pueda autenticar a todo el paquete IPv6 original tal y como se puede ver en la figura 5.2. Utilizar AH en modo túnel tiene un inconveniente, no brinda protección contra análisis de tráfico, solo garantiza que el paquete recibido no ha sido modificado en el camino hacia el destino.

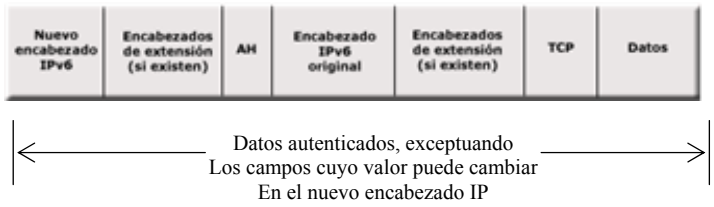


Figura 5.2. Paquete IPv6 con AH en modo Túnel

5.4.2 AH modo Transporte

Cuando AH es utilizado en modo transporte, este protege la comunicación extremo a extremo de los datos del paquete IPv6. El encabezado AH es insertado inmediatamente después del encabezado IPv6 y de los campos que no varían tales como *hop-by-hop*, enrutamiento y fragmentación, pero antes de los encabezados de los protocolos de capas superiores tal como se muestra en la figura 5.3.

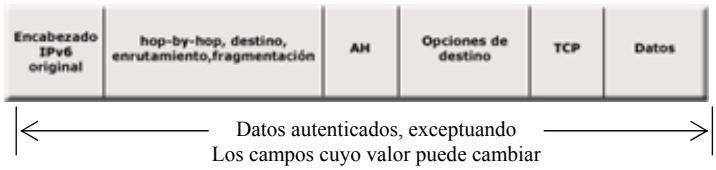


Figura 5.3. Paquete IPv6 con AH en modo Transporte

5.5 Encapsulado de Seguridad de la Carga (ESP)

ESP provee confidencialidad de información mediante un algoritmo de cifrado, autenticación del origen de los datos e integridad de datos mediante un algoritmo de Códigos de Autenticación de Mensajes mediante funciones Hash (HMAC) y protección contra ataques de reenvío de paquetes mediante un número de secuencia. Cuando ESP es utilizado para proteger un paquete IP, el encabezado ESP es insertado después de los encabezados de extensión cuyo valor puede cambiar durante el camino hasta llegar al destino final del paquete. Esto incluye los encabezados hop-by-hop, enrutamiento y fragmentación. El encabezado que antecede a ESP debe contener el valor 50 en el campo "Siguiente encabezado". En la figura 5.4 se muestra el encabezado ESP. El primer campo de éste es el "SPI" de 32 bits de longitud, es utilizado junto con la dirección IPv6 destino y el protocolo para identificar la SA a utilizar para procesar dicho paquete.

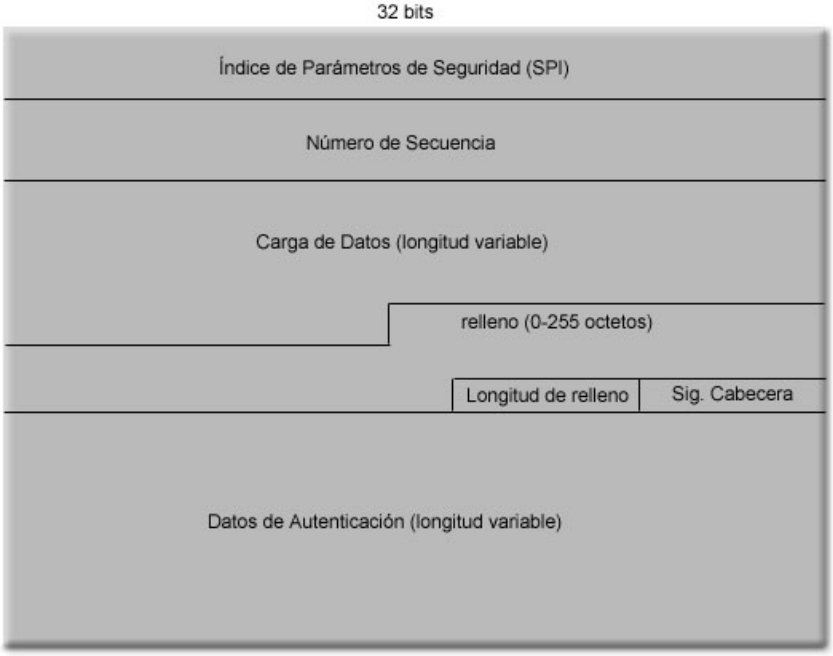


Figura 5.4. Encabezado de ESP

El siguiente campo es "Número de Secuencia" de 32 bits de longitud, éste contiene un número que es único y monotónicamente auto incrementado para prevenir ataques de reenvío de paquetes. Este campo puede ser autenticado, pero no cifrado, ya que para prevenir dichos ataques el campo debe ser analizado sin consumir recursos extra decriptándolo. Cuando una SA acaba de ser establecida entre el emisor y el receptor de un paquete este valor es puesto a cero por ambas partes. En seguida sigue el campo "Carga de Datos" con una longitud variable el cual contiene los datos a ser protegidos por ESP. El siguiente campo es "relleno", tiene una longitud que varia entre 0 y 255 octetos y contiene información sin importancia que es utilizada para realizar el proceso de cifrado, ya que hay algunos algoritmos de cifrado que requieren que la entrada (datos a ser cifrados) sea de un tamaño múltiplo de su tamaño de bloque.

El siguiente campo es "Longitud de relleno", tiene una longitud de 8 bits e indica el número de octetos de relleno que han sido agregados. Después sigue el campo "Siguiente encabezado" con 8 bits de longitud, este campo identifica al encabezado que sigue inmediatamente después del encabezado ESP. Por último, el campo "Datos de autenticación" tiene una longitud variable y contiene el resultado del algoritmo de chequeo de integridad (ICV) por lo general es el resultado del cálculo de una función hash. Su tamaño depende del algoritmo de chequeo utilizado.

ESP puede trabajar en dos modos distintos de operación: modo túnel y modo transporte y son explicados a continuación.

5.5.1 ESP modo Túnel

ESP en modo túnel es capaz de proteger un paquete IPv6 completo, ya que el encabezado IPv6, la carga de datos y el ESP Trailer son cifrados y además el encabezado ESP, el encabezado IPv6, la carga de datos y el ESP Trailer pueden ser autenticados. Para conseguir esto un nuevo encabezado IPv6 es agregado al paquete IPv6 cifrado con las direcciones de los extremos del túnel IPSec tal y como lo muestra la figura 5.5.

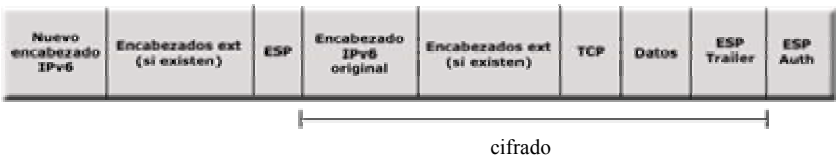


Figura 5.5. Paquete IPv6 con ESP en modo Túnel

5.5.2 ESP modo Transporte

Cuando ESP es utilizado en modo transporte este protegerá la carga de datos del paquete IPv6, así como el ESP Trailer por medio de un algoritmo de cifrado y/o autenticación. El encabezado ESP es insertado justo después del encabezado IPv6 y de los encabezados de extensión que pueden variar en el transcurso del viaje del paquete hasta su destino tal y como lo muestra la figura 5.6.

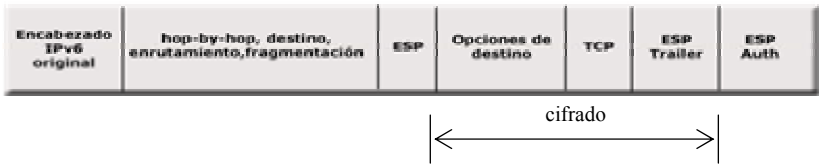


Figura 5.6. Paquete IPv6 con ESP en modo Transporte

5.6 Caso de estudio

Esta prueba se realizó con el fin de evaluar la interoperabilidad de IPSec en sus dos implementaciones IPv4 e IPv6 para los enrutadores CISCO 2620 XM, así como la viabilidad de generar túneles IPSec (IPv4 e IPv6) para brindar conectividad segura dominio a dominio. Para esto se establecieron los servicios de seguridad de IPSec en una configuración ESP modo túnel entre los enrutadores que unen los dominios de las subredes de prueba. Los detalles de la configuración utilizada se mencionan en la sección 5.6.2. La elección de utilizar una configuración ESP modo túnel se debió a que es ésta la configuración más utilizada para cifrar tráfico entre *gateways* de seguridad en un ambiente estilo VPN.

La información asegurada por el túnel IPSec constó de flujos de video producidos por una aplicación llamada VideoLAN, la cual es capaz de enviar flujos de video de diferentes formatos (MPEG1, MPEG2, MPEG4, VCD, DVD) sobre IPv4 e IPv6 utilizando una arquitectura cliente-servidor. De esta manera pudimos además constatar deficiencias en la implementación de IPv6 del IOS 12.2(11)T3 para el manejo de QoS.

5.6.1. Integración de la prueba

Debido a que sería muy extenso probar la interoperabilidad de todas las características y capacidades de IPSec con IPv6 en nuestras pruebas nos enfocamos a un pequeño subconjunto de ellas, dicho subconjunto está determinado por los siguientes factores:

Modo de IPSec: Tanto AH como ESP operan en modo transporte y en modo túnel, el modo transporte es principalmente utilizado entre dos *hosts* o entre un *host* y un enrutador. El modo túnel es

utilizado entre dos enrutadores para cifrar la información que viajará entre ellos. De esta manera decidimos utilizar modo túnel ya que nuestro principal objetivo era enviar tráfico desde una subred (IPv4/IPv6) hacia otra subred (IPv4/IPv6) de manera segura.

Intercambio de llaves: El intercambio de llaves entre ambos enrutadores puede realizarse de manera manual o de manera automática mediante el protocolo IKE. IKE tiene la ventaja de ser más escalable que el intercambio manual de llaves, además es muy flexible, ya que permite utilizar diferentes métodos de autenticación. Los tres principales métodos soportados por IKE son Llaves precompartidas, Anuncios cifrados mediante llaves RSA (RSA encr) y firmas RSA (Rivest Shamir, Adleman). Nos decidimos por utilizar el método RSA (encr) debido a que es más seguro que utilizar el método de llaves precompartidas y no requiere establecer una autoridad certificadora en una plataforma Windows para administrar los certificados de autenticación.

Algoritmos para Intercambio de llaves de sesión: IKE utiliza un protocolo criptográfico de llave pública conocido como Diffie-Hellman (DF) para establecer un secreto compartido entre ambos enrutadores, mismo que será utilizado como llave de sesión. DH utiliza principalmente dos tamaños de llave para generar los secretos compartidos: Grupo1 que consta de llaves públicas de 768 bits y Grupo 2 que utiliza llaves públicas de 1024 bits. Para nuestras pruebas utilizamos el Grupo 2 por obvias razones.

Integridad de datos: La integridad de los datos enviados es mantenida mediante el cálculo de un mensaje abreviado de los datos. IPSec implementa dos algoritmos para realizar estos mensajes resumidos mediante funciones hash. Estos dos algoritmos son MD5, el cual produce un mensaje abreviado de 128 bits de longitud y SHA que produce un mensaje abreviado de 160 bits. En nuestras pruebas decidimos utilizar SHA ya que proporciona un nivel de seguridad más alto que MD5.

Algoritmo de cifrado: La confidencialidad de los datos se logra mediante el cifrado de los mismos utilizando algoritmos de cifrado simétricos. IPSec implementa principalmente los algoritmos DES (Data Encryption Standard), el cual utiliza un tamaño de llave de 56 bits para cifrar la información,3DES (Triple Data Encryption Standard), que utiliza un tamaño de llaves de 168 bits y AES (Advanced Encryption Standard) que utiliza un tamaño variable de llave (128, 192 o 256 bits) para cifrar la información. Debido a que la versión de IOS utilizada al momento de las pruebas no tenía implementado el algoritmo AES, elegimos 3DES como algoritmo de cifrado por ser el siguiente algoritmo más seguro.

Con respecto a la aplicación que utilizamos para generar el flujo de video, esta tiene las siguientes características:

- Servidor VideoLAN:
  - Versión: 0.5.6
  - Plataforma Linux Red Hat 9
  - Formato de video: MPEG-4
  - Modos de envío: Unicast, Broadcast, Multicast
  - Tamaño del video: 304 MBytes.

- Cliente VideoLAN:

  - Versión: 0.7.1
  - Plataformas: Windows XP, Mac OS X 10.2, Linux Red Hat 9

5.6.2. Arquitectura de la prueba

En la figura 5.7 presentamos la arquitectura de la red utilizada para llevar a cabo las pruebas concernientes a IPSec. Cada una de las máquinas conectadas a la red (clientes y servidores VideoLAN) cuenta con una dirección IPv4, y una dirección IPv6-global (*6to4*). Para poder hacer uso de direcciones IPv6 globales fue necesario configurar un túnel 6to4 entre los enrutadores que unen a estas dos islas IPv6.

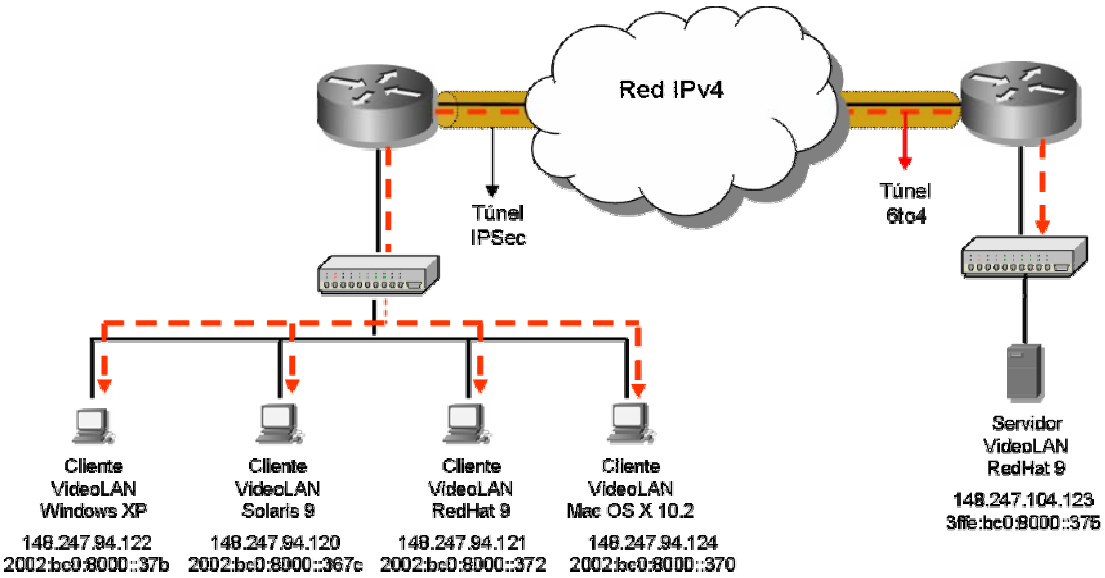


Figura 5.7. Arquitectura de red para las pruebas de IPSec

Las tablas 5.1 y 5.2 muestran las configuraciones de ambos enrutadores para generar el túnel IPSec entre ambos y así asegurar la comunicación IPv4/IPv6 que fluye entre ambas subredes.

Router_test

```
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname router_test
!
enable secret 5 $1$oN36$AhkZVWzpBpdswhKeGrmC71
enable password ipv6_proyecto
!
ip subnet-zero
!
ip domain name test.cinvestav.mx
ip host router_test2 148.247.91.7
ip name-server 148.247.1.2
ip name-server 148.247.94.121
!
ip multicast-routing
ipv6 unicast-routing
!
crypto isakmp policy 1
```

```
 authentication rsa-encr
 group 2
 lifetime 240
!
crypto ipsec transform-set ts1 ah-sha-hmac esp-3des esp-sha-hmac
!
crypto key pubkey-chain rsa
 addressed-key 148.247.91.7
  address 148.247.91.7
  key-string
   305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00D968CB 579F30F0
   BA2DC863 6B7A0404 F633FD5C D86B849F 3DF724F8 F0096EE5 E60BF0BA BA73736A
   93CF0B40 5C96B2F7 0D7C7272 DBB23DB2 AE4E57DC D969A4EA 9D020301 0001
  quit
 !
 crypto map test 10 ipsec-isakmp
 set peer 148.247.91.7
 set transform-set ts1
 match address 133
!
voice call carrier capacity active
!
mta receive maximum-recipients 0
!
interface Loopback0
 no ip address
!
interface Tunnel1
 description LINE
 no ip address
 no ip redirects
 ipv6 unnumbered FastEthernet0/0
 tunnel source Loopback0
 tunnel mode ipv6ip 6to4
!
interface FastEthernet0/0
 ip address 148.247.91.5 255.255.255.0
 ip mtu 1480
 ip pim sparse-dense-mode
 no ip mroute-cache
 speed auto
 half-duplex
 ipv6 address 2002:94F7:5B05:1::1/64
 ipv6 enable
 ipv6 rip p1 enable
 ipv6 rip default-information enable
 crypto map test
!
interface FastEthernet0/1
 ip address 148.247.94.1 255.255.255.0
 ip pim sparse-dense-mode
 no ip mroute-cache
 speed auto
 half-duplex
 ipv6 address 2002:94F7:5E01:1::1/64
 ipv6 enable
 ipv6 rip p1 enable
 ipv6 rip default-information enable
!
```

```
router rip
 version 2
 network 148.247.0.0
 default-metric 3
!
ip classless
ip route 0.0.0.0 0.0.0.0 148.247.91.1
ip http server
!
access-list 133 permit ip 148.247.94.0 0.0.0.255 148.247.104.0 0.0.0.255
ipv6 route 2002::/16 Tunnel1
ipv6 route ::/0 2002:C058:6301::
ipv6 router rip p1
!
ipv6 router rip default-information
!
ipv6 router rip WORD
¡
call rsvp-sync
!
mgcp profile default
!
dial-peer cor custom
!
line con 0
 speed 115200
line aux 0
line vty 0 4
 password snoopy
 login
!
end
```

Tabla 5.1. Configuración del enrutador ***router_test***

Router_test2

```
INVEST 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
¡
hostname router_test2
¡
enable secret 5 $1$QxvV$b88yX8bI306Yh4MqwzvHL0
enable password ipv6_proyecto
¡
ip INVES-zero
¡
ip domain name INVE.cinvestav.mx
ip host router_test 148.247.91.5
ip name-server 148.247.104.123
ip name-server 148.247.1.2
!
ipv6 unicast-routing
!
```

```
crypto isakmp policy 1
 authentication rsa-encr
 group 2
 lifetime 240
!
crypto ipsec transform-set ts1 ah-sha-hmac esp-3des esp-sha-hmac
!
crypto key pubkey-chain rsa
 named-key router_test.test.cinvestav.mx.
  key-string
  quit
 addressed-key 148.247.91.5
  address 148.247.91.5
  key-string
   305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00BF8A95 94385466
   4AA3CC81 E86A10EF 272037C9 89C3594E 55ª961FC B66106C3 62F34193 2F1B3CFF
   ADD618B9 61C891B1 3AB9C371 FBD5B2F5 52436BC8 AD566642 4F020301 0001
  quit
 !
 crypto map test 10 ipsec-isakmp
 set peer 148.247.91.5
 set transform-set ts1
 match address 144
!
voice call carrier capacity active
!
mta receive maximum-recipients 0
!
interface FastEthernet0/0
 ip address 148.247.91.7 255.255.255.0
 speed auto
 half-duplex
 ipv6 address 2002:94F7:5B07:1::1/64
 ipv6 enable
 ipv6 rip p1 enable
 crypto map test
!
interface FastEthernet0/1
 ip address 148.247.104.1 255.255.255.0
 speed auto
 half-duplex
 ipv6 address 2002:94F7:6801:1::1/64
 ipv6 enable
 ipv6 rip p1 enable
!
router rip
 version 2
 network 148.247.0.0
!
ip default-gateway 148.247.91.1
no ip classless
ip route 0.0.0.0 0.0.0.0 148.247.91.1
no ip http server
!
access-list 144 permit ip 148.247.104.0 0.0.0.255 148.247.94.0 0.0.0.255
ipv6 router rip p1
!
no call rsvp-sync
!
```

```
mgcp profile default
!
dial-peer cor custom
!
line con 0
 speed 115200
line aux 0
line vty 0 4
 password snoopy
 login
!
end
```

Tabla 5.2. Configuración del enrutador ***router_test2***

5.6.3. Resultados obtenidos

Como podemos ver en las tablas 5.1 y 5.2, CISCO aún no cuenta con una implementación de IPSec que trabaje de manera nativa sobre IPv6 (por lo menos no en la versión de IOS 12.2(11)T3), en su lugar aplica los servicios de seguridad IPSec provistos por IPv4 mediante túneles 6to4, de esta manera IPSec es aplicado a los paquetes IPv6 que han sido encapsulados dentro de paquetes IPv4. Esto ocasiona que no se aproveche el potencial real de los servicios de seguridad ofrecidos por IPv6 y genera un retardo adicional al tener que aplicar los servicios de IPSec un nivel arriba del paquete IPv6.

Desafortunadamente la versión del IOS usado en este caso de estudio no cuenta con manejo de QoS implementado sobre IPv6 provocando que no se pueda dar el manejo requerido a los paquetes de flujo de video. Por si esto fuera poco no fue posible enviar video *multicast* sobre IPv6 ya que esta versión de IOS no tiene implementados los servicios de tráfico *multicast* en IPv6.

A pesar de esto, si fue posible enviar video *unicast* sobre IPv4 e IPv6 y video *broadcast* sobre IPv4 utilizando los servicios de seguridad de IPSec. En las figuras 5.8 – 5.10 podemos ver algunas pantallas capturadas sobre los diferentes clientes de las pruebas de envío de video utilizando servicios de seguridad IPSec sobre IPv4 e IPv6. La calidad de reproducción de video varía en cada una de las plataformas debido factores de hardware y software. Entre los factores relacionados al hardware se encuentran la calidad de las tarjetas de video y de red utilizadas. Entre los factores relacionados al software se encuentran la manera en que están implementados los *sockets* (IPv4/IPv6) en cada plataforma. A pesar de estas variaciones la calidad de reproducción es aceptable.

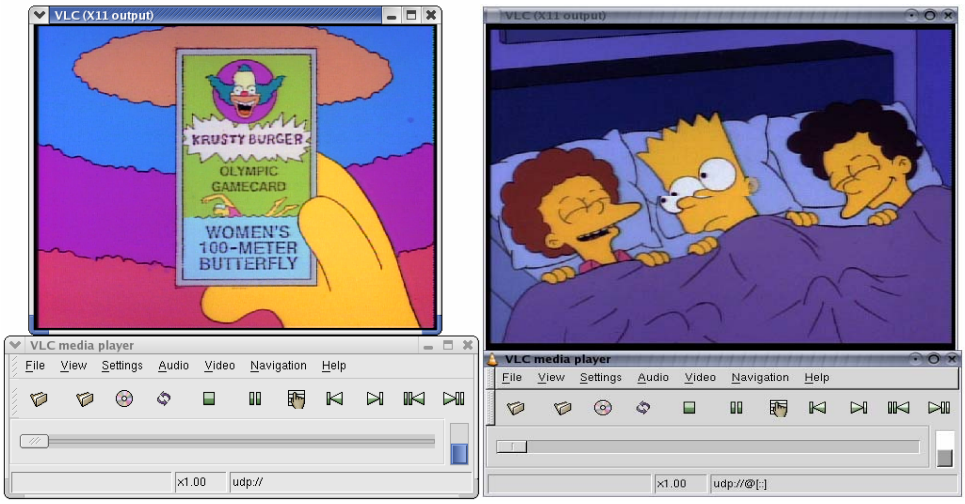


Figura 5.8 VLC sobre plataforma Linux utilizando como medio de transporte IPv4 e IPv6 respectivamente

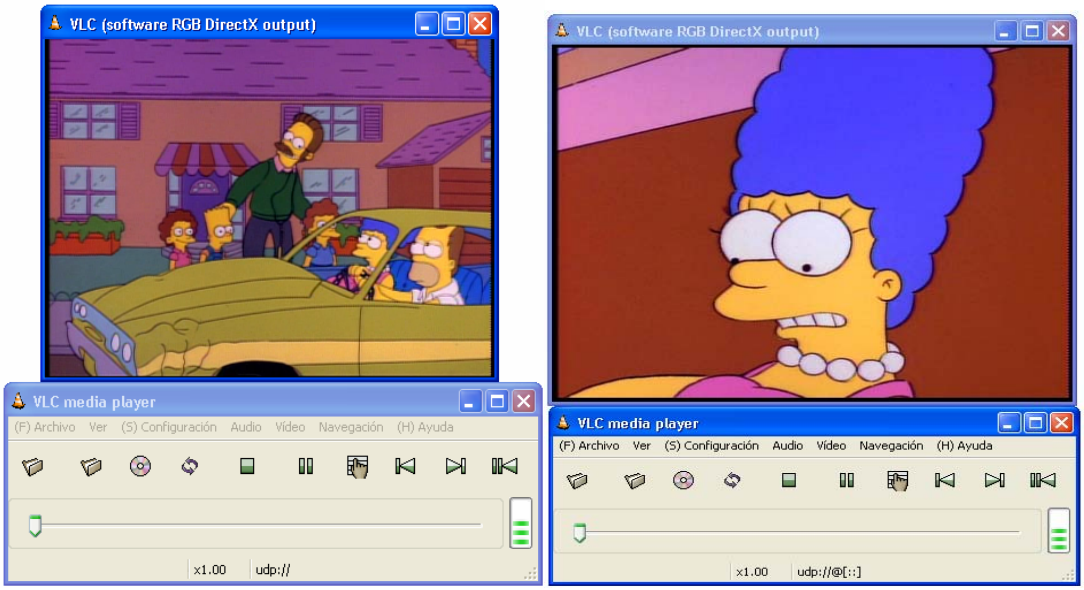


Figura 5.9 VLC sobre plataforma Windows utilizando como medio de transporte IPv4 e IPv6 respectivamente

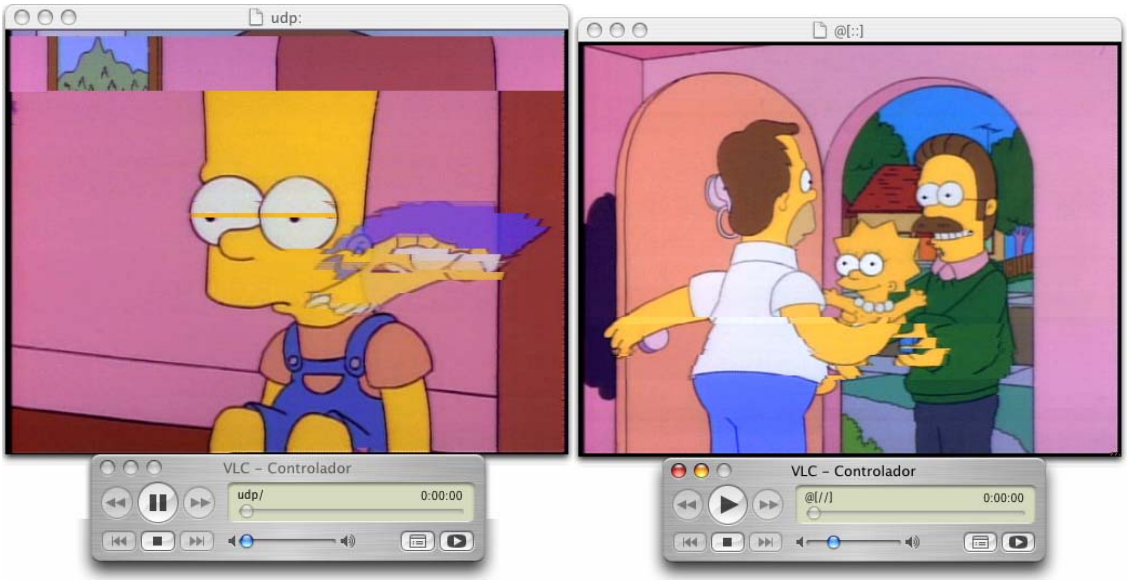


Figura 5.10. VLC sobre plataforma Mac OS X 10.2 utilizando como medio de transporte IPv4 e IPv6 respectivamente

Después de los resultados obtenidos en nuestra experiencia, pensamos que utilizando otro tipo de enrutadores los cuales brinden soporte de IPSec y QoS nativo sobre IPv6 tales como enrutadores Linux [28], o Solaris [29] se obtendrían mejores resultados que los presentados por el tipo de enrutadores utilizados en nuestro caso de estudio.

Capítulo 6

Conclusiones y Trabajo Futuro

En esta tesis reportamos algunos problemas de interoperabilidad entre plataformas encontrados durante el desarrollo de las pruebas aquí mostradas. Dentro de esos problemas de interoperabilidad están los siguientes:

- Los navegadores sobre la plataforma Windows no soportan el uso de direcciones IPv6 en su formato hexadecimal, en su lugar es necesario registrar dichas direcciones en un servidor DNS y acceder a estas direcciones mediante su nombre completamente calificado.

- Algunas de las implementaciones actuales de IPv6, tales como las que hay en el sistema operativo IOS de CISCO son muy limitadas, ya que no cuentan con una implementación de IPSec nativa IPv6, en su lugar utilizan la implementación de IPSec sobre IPv4 y la aplican sobre túneles 6to4, debido a esto se pierde el potencial real de los servicios de seguridad ofrecidos por IPv6.

- La versión 12.2(11)T3 de IOS además no cuenta con soporte para manejo de QoS, ni soporte para tráfico *multicast* sobre IPv6, por lo que no nos fue posible realizar una comparación de esta característica entre IPv4 e IPv6.

Una vez concluido este trabajo de investigación decidimos probar algunas aplicaciones como wusftp, vsftp, VPNTracker, IPSecuritas y Flash Communication Server MX sobre nuestra red IPv6 de pruebas encontrando que algunas de ellas aún no son compatibles con IPv6, tales como:

- VSFTP: servidor de correo; en su lugar se utilizó WSFTP

- VPNTracker: Aplicación de MAC OS X para asegurar comunicaciones mediante IPSec; en su lugar se utilizó la implementación nativa Racoon.

- IPSecuritas: Aplicación de MAC OS X para asegurar comunicaciones mediante IPSec; en su lugar se utilizó la implementación nativa Racoon.

- Flash Communication Server MX: Servidor de aplicaciones multimedia desarrolladas en Flash MX

Asimismo, las siguientes aplicaciones mostraron buen comportamiento sobre IPv6:

- VideoLAN: Aplicación para envío de video *unicast, multicast* y *broadcast.*

- OpenLDAP: Aplicación que brinda servicios de directorio.

- *Sendmail*: Servidor de correo

- BIND9: servidor DNS

- Apache: Servidor HTTP

De las pruebas mostradas en la sección 5.6 cabe aclarar que debido a la manera en que está construida y opera la aplicación VideoLAN no fue posible tomar como métrica de evaluación tiempos ya que el servidor de VideoLAN (VLS) en ningún momento establece una comunicación directa con los clientes (VLC) de forma que sea posible realizar dicha medición.

Las principales contribuciones de esta investigación fueron las siguientes: El desarrollo de un *benchmark* para medir el desempeño de un servidor LDAP, tomando como métricas la latencia y el rendimiento del mismo al utilizar como protocolos de comunicación IPv4 e IPv6. También desarrollamos un modelo matemático capaz de reproducir el comportamiento de éste mismo servidor LDAP basándonos en los resultados obtenidos en el *benchmark* desarrollado anteriormente, además realizamos pruebas de interoperabilidad de IPv6 con aplicaciones tales como VPN-Tracker, IPSecuritas, Freeswan, Racoon, VideoLAN y Flash Communication Server MX, así como con la implementación de IPv6 en el sistema operativo IOS 12.2(11)T3 para los enrutadores CISCO 2620 XM.

Por último podemos mencionar que IPv6 no es una moda pasajera que pueda desaparecer rápidamente. IPv6 ya es una realidad. Organizaciones como el Pentágono ya iniciaron la migración de sus redes a IPv6 por cuestiones de seguridad. Sin embargo, aún falta madurar más las implementaciones actuales de este protocolo, de manera que en los siguientes dos años sea posible desarrollar aplicaciones que puedan explotar por completo el potencial de este protocolo de red.

Sugerencias para trabajos futuros

Son muchas las cosas que pueden hacerse a partir de esta tesis, ya que uno de los objetivos alcanzados de este trabajo fue el construir una red de pruebas que trabaje sobre IPv6. Sobre esta red pueden desarrollarse toda clase de aplicaciones como se necesite, el único limitante es que se puedan rebasar las capacidades del Enrutador utilizado dado que en la versión del IOS utilizada no se puede configurar QoS ni envío multicast sobre IPv6, en cuyo caso deberá actualizarse su sistema operativo.

Con respecto a las pruebas desarrolladas en esta tesis pueden realizarse los siguientes trabajos futuros:

- Construir un PKI para asegurar el acceso a contenido de información sobre la red IPv6.

- Extender el modelo propuesto en el capítulo 4 de tal manera que éste logre ser independiente de protocolo (LDAP).

- Migrar la red interna del CINVESTAV a IPv6. Esto se puede hacer de manera inmediata utilizando direcciones 6to4 y configurando la salida a I2 mediante la configuración de un *relay router*.

- Construir un sistema que provea servicio de videoconferencia sobre IPv6/IPv4 a partir de la implementación VideoLAN, pero que a diferencia de ésta, las transmisiones puedan iniciarse a partir de peticiones de los clientes.

Glosario

Asociación de Seguridad (SA): Establece los parámetros de seguridad necesarios para comunicar dos nodos mediante IPSec, algunos de ellos son: una llave, el algoritmo de autenticación o encriptación tiempo de vida de la llave, etc.

Benchmark: Es una prueba o conjunto de pruebas diseñadas para comparar el rendimiento de un sistema contra el rendimiento de otros.

Datagrama: un datagrama es un paquete de datos transmitido sobre una red sin que se establezca una conexión a prior que los datos sean transmitidos. (*connectionless*).

Enrutador: Es un nodo que provee conexión entre redes.

Host: cualquier nodo que no es un enrutador.

Internet Engineering Task Frce (IETF): Comunidad de diseñadores de redes, operadores, vendedores e investigadores relacionados con la evolución y operación de Internet.

Internet Protocol (IP): El protocolo de Internet provee un servicio de entrega de paquetes no confiable sin conexión entre redes. Este servicio es llamado sin conexión porque se asemeja mucho al servicio postal. Cada paquete IP es estampado con una dirección del nodo destinatario y del nodo remitente.

IPSecurity (IPSec): Conjunto de protocolos definidos por el IETF para soportar el intercambio seguro de paquetes sobre la capa IP. Soporta dos modos de encriptación: Modo Transporte y Modo Túnel. El Modo Transporte encripta solo la porción de datos de cada paquete dejando el encabezado del mismo al descubierto. El Modo Túnel encripta ambos encabezado y datos del paquete.

IPv4: Es la versión actual del Protocolo de Internet

IPv6: Es la nueva versión del Protocolo de Internet diseñada para resolver los problemas del protocolo IP actual, algunos de estos problemas son el espacio de direccionamiento disponible, la Calidad de Servicio y la seguridad.

LDAP: El Protocolo Ligero de Acceso a Directorio (LDAP) es un conjunto de protocolos diseñados para brindar servicio de directorio. Desciende de otro protocolo de servicio de directorio llamado X500, pero a diferencia de éste, LDAP corre sobre TCP por lo que es muy utilizado como un substituto de base de datos.

MTU: Unidad Máxima de Transmisión es el tamaño más grande de paquete que puede ser transferido sobre una interfaz. El MTU para Ethernet es de 1500 bytes.

Nodo: dispositivo que tiene implementado IPv4 o IPv6.

Paquete: Un paquete es la unidad de datos la cual es enrutada entre un origen y un destino sobre Internet o cualquier otra red de paquetes switcheados. Cuando cualquier archivo (mensaje de e-mail, archivo HTML, archivo GIF, petición URL, etc.) es enviado de un lugar a otro sobre Internet, TCP divide el archivo en pedazos (paquetes) lo suficientemente pequeños para que estos sean enrutados de manera eficiente hasta llegar a su destino donde posteriormente serán reensamblados.

Túnel configurado: Es un método de transición estandarizado por el IETF para utilizar IPv6 en coexistencia con IPv4, encapsulando los paquetes IPv6 en datagramas IPv4.

A

Configuración de IPv6 en plataformas de prueba

En este apéndice damos las guías para configurar IPv6 en las siguientes plataformas:

- Windows XP
- Linux Red Hat 8/9
- Solaris 8/9
- Mac OS X 10.2 *Jaguar*

También damos una guía para configurar IPv6 en enrutadores Cisco que utilicen versiones del sistema operativo IOS las cuales incluyan soporte para IPv6. Por último mostramos como configurar un servidor DNS con IPv6, y damos un ejemplo de cómo agregar direcciones IPv6 en los archivos de zona del servidor DNS.

A.1 Habilitación de IPv6 sobre plataforma Windows XP

Microsoft comenzó a contribuir en el esfuerzo de estandarizar IPv6 a partir de 1996 [25] por medio del proyecto MSR-IPv6. En este proyecto se escribió una implementación del protocolo IPv6 para la plataforma Windows NT/2000 y posteriormente se migró este trabajo a la línea de productos Windows XP y Windows .NET Server. (Los objetivos principales de MSR-IPv6 son: brindar simplicidad, seguridad, API de desarrollo y movilidad para IPv6.)

Windows XP ya incluye una implementación de IPv6 lista para instalar, además, el *service pack 1* contiene una actualización para esa implementación. Además, Microsoft dio a conocer el *Windows XP Advanced Networking Pack* que incluye una nueva implementación de IPv6, así como nuevas características para utilizar túneles TEREDO y un API de programación para Firewalls IPv6.

¿Cómo habilitar IPv6 en Windows?

Ya sea que utilicemos la implementación que trae consigo Windows XP, el SP1 ó el *Windows XP Advanced Networking Pack,* los pasos para instalarlo son los mismos:

- Abrir consola de comandos
- En el prompt del sistema teclear: ipv6 install

En nuestro caso realizamos la configuración con cada uno de estos métodos y

Una vez hecho esto, debemos verificar que las interfaces de red ya cuentan con el protocolo IPv6 habilitado. Esto se logra de dos formas distintas:

- Tecleando: ipv6 if
- Tecleando: ipconfig /all

En ambos casos debemos observar que todas las interfaces ya cuenten con IPv6 habilitado.

¿Cómo configurar direcciones IPv6 en una interfaz sobre Windows?

Una vez que IPv6 ha sido habilitado en el sistema, una dirección de tipo IPv6-Local es asignada automáticamente a cada una de las interfaces de red por medio del protocolo de descubrimiento de vecinos. Estas direcciones, como ya se mencionó en el capítulo 1, solo tienen significado entre las interfaces que se encuentran en el mismo segmento de red, así que para poder acceder a servicios IPv6 fuera de nuestra LAN fue necesario configurar direcciones IPv6 globales.

Para configurar una dirección IPv6-global de manera manual hacemos uso de la herramienta *netsh* que viene precargada en Windows. *Netsh* nos brinda un intérprete de comandos para la configuración de las interfaces de red. En este caso los comandos a seguir son:

1. C:\netsh
2. netsh> interface
3. netsh interface> ipv6
4. netsh interface ipv6> set address [interface=] <nombre interface> [address=]<dirección IPv6>[[type=] unicast | anycast]

Ejemplo:

1. C:\netsh
2. netsh> interface
3. netsh interface> ipv6
4. netsh interface ipv6> set address interface= “Local Area Connection” address= 3ffe:bc0:8000::374 type=unicast

A.2 Habilitación de IPv6 sobre plataforma Red Hat Linux 8.0 y 9.0

La primera implementación de IPv6 en Linux fue hecha en Noviembre de 1996 para el *kernel* 2.1.8 y estaba basada en el API de BSD. Sin embargo, esta implementación no fue capaz de ajustarse a los cambios y nuevos RFC’s que se publicaron con el paso del tiempo (ver Tabla 1).

RFC	Título	RFC obsoleto	Principales cambios
2460	Internet Protocol, Version 6 (IPv6) Specification	1883	El RFC 2460 realiza cambios en la especificación de IPv6, tales como el tamaño de MTU mínimo de IPv6, cambio de nombre en algunos campos del encabezado IPv6, etc.
2461	Neighbor Discovery for IP Version 6 (IPv6)	1970	Algunos de los cambios que este RFC realiza sobre el RFC 1970 son la eliminación de referencias al campo de prioridad IPv6, cambia la constante "576" por "1280" en lugares donde este valor especificaba el tamaño mínimo de los paquetes IPv6 que todos los enlaces debían soportar, etc.
2462	IPv6 Stateless Address Autoconfiguration	1971	El principal cambio es la adición de reglas al proceso de envío de avisos de enrutamiento para prevenir ataques de denegación de servicio (DOS)
2464	Transmission of IPv6 Packets over Ethernet Networks	1972	Se modificaron las longitudes de prefijos para direcciones locales de a 64 bits
2472	IP Version 6 over PPP	2023	Se incrementa el tamaño del identificador de interfaz a 46 bits, además se agregan métodos para seleccionar un identificador de interfaz a partir de un identificador global IEEE único cuando este está disponible en cualquier parte del nodo.
2675	IPv6 Jumbograms	2147	Separa la especificación de la opción Jumbo Payload (contenida en RFC 1883) en un solo RFC
2893	Transition Mechanisms for IPv6 Hosts and Routers	1933	Se agregaron definiciones para modos de operación para nodos IPv6/IPv4, se hicieron cambios en la terminología utilizada. Se actualizó e algoritmo que determina el MTU de un túnel para reflejar el cambio en el MTU mínimo en IPv6 que pasó de 576 a 1280 bytes
3493	Basic Socket Interface Extensions for IPv6	2553	Se agregó una nueva opción en la capa de sockets para permitir a los nodos no procesar paquetes IPv4 como direcciones IPv4 mapeadas en las implementaciones
3513	Internet Protocol Version 6 (IPv6) Addressing Architecture	2373	Se hicieron cambios para permitir el uso de "::" para representar uno o más grupos de 16 bits de ceros. Se modificó el formato de Direcciones Locales de Sitio para hacer el tamaño del campo "Subset ID" de 54 bits de longitud. También se removió el campo de 38 bits "zero"
3587	IPv6 Global Unicast Address Format	2374	Cambia el formato de direcciones IPv6

Tabla A.1. Cambios realizados en algunos de los RFC's para IPv6

Para octubre del 2000 surgió un nuevo proyecto en Japón denominado USAGI (*Universal Playground for IPv6*) cuyo objetivo era implementar todos estos cambios y cosas faltantes de la implementación de IPv6 en Linux existente. Esta nueva implementación tenía la desventaja de ser muy grande en tamaño, así que no se incluyó completamente en el código del kernel 2.4 ocasionando con esto algunos problemas de interoperabilidad con otros sistemas operativos.

¿Cómo habilitar IPv6 en Linux?

Hay dos formas de habilitar IPv6 en Linux, en ambos casos es necesario tener privilegios de administrador:

- Primer forma:
    - Estando como súper-usuario, hacemos uso de la herramienta de configuración *xconfig***:**
      /etc/usr/src/linux/make xconfig
      Inmediatamente abrirá una ventana como la que se muestra en la figura 1

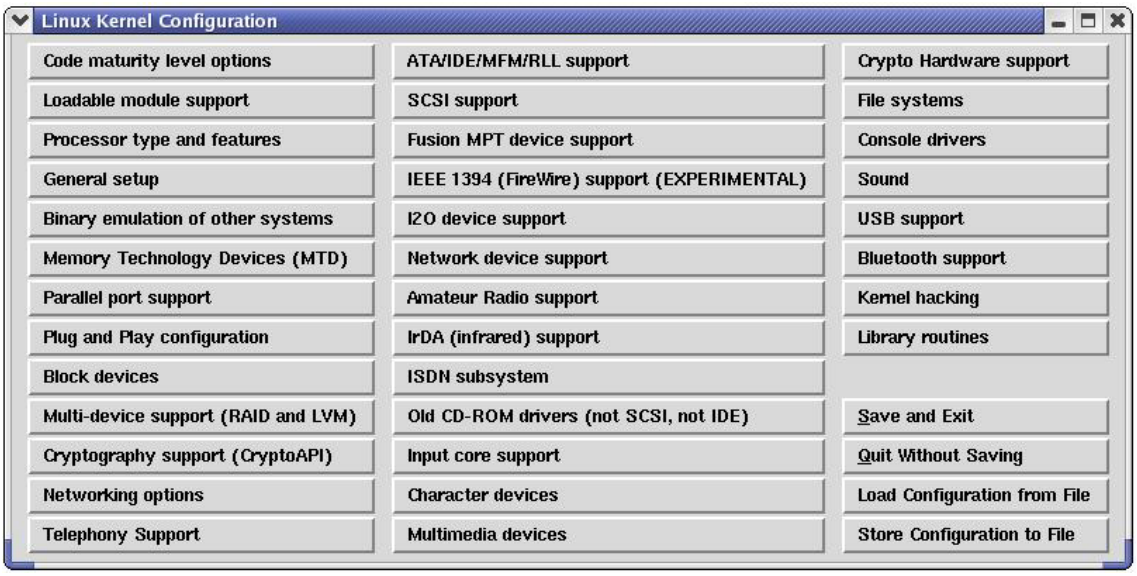


Figura A.1. Herramienta de configuración del kernel **xconfig**

- Elegimos la opción “Code maturity level options”
- Estando en “Code maturity level options” habilitamos la opción “Prompt for development and/or incomplete code/drivers” tal como lo muestra la figura 2 y después regresamos al menú principal (“Main Menu”)

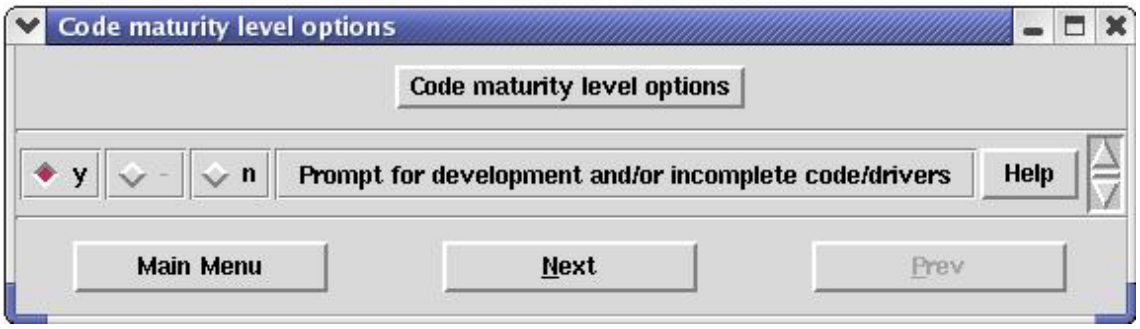


Figura A.2. Opción *Code maturity level options*

- Ahora agregamos el modulo de IPv6 en las opciones de red del menú principal

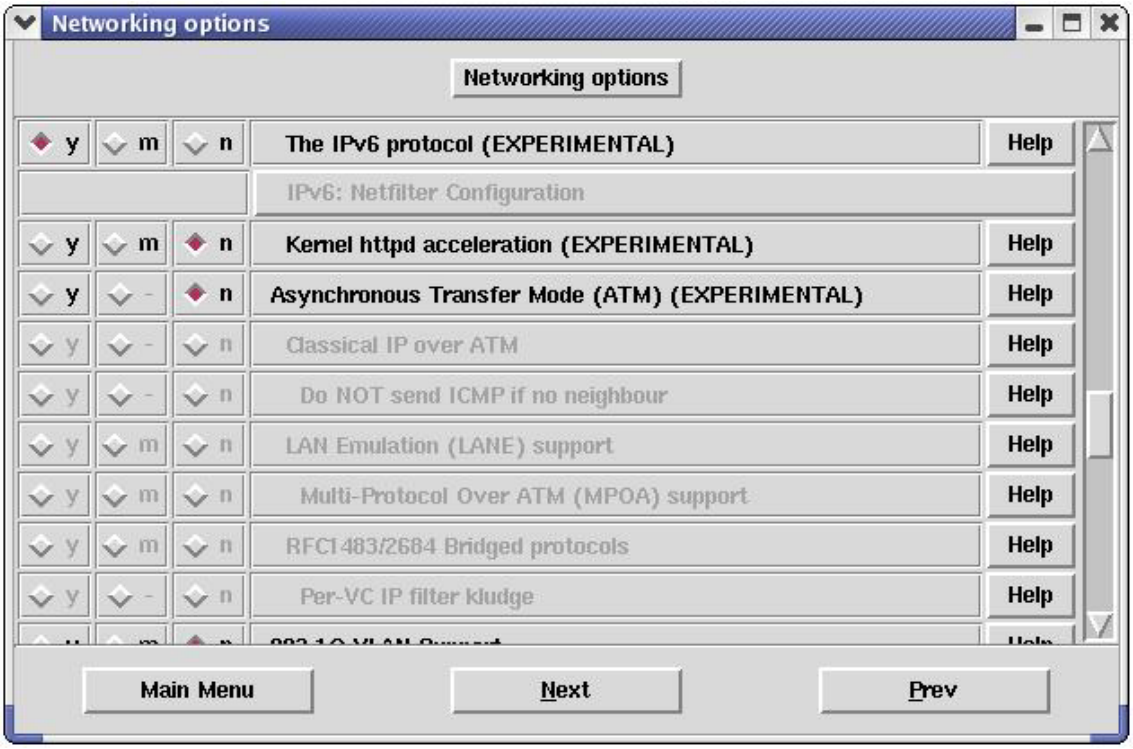


Figura A.3. Carga del modulo IPv6 en el kernel de Linux

- Regresamos al menú principal y guardamos los cambios (elegir la opción “Save and Exit” en el menú principal):

Figura A.4. Instrucciones para compilar el kernel de Linux

- Una vez guardada la configuración compilamos el kernel;

/usr/src/linux/make dep

- Ahora ya se tiene habilitado el protocolo IPv6 en el kernel de Linux

- Segunda forma:

  - Esta forma de habilitar IPv6 en el kernel de Linux es más rápida que la antes mencionada, pero tiene la desventaja que se tiene que realizar cada vez que deseemos

utilizar IPv6, a menos que escribamos un script para cargar el módulo al momento de iniciar el sistema. Ya estando como súper-usuario, instalamos el módulo IPv6 al kernel:

```
/sbin/insmod ipv6
```

- Una vez instalado es necesario cargarlo al sistema:

```
/sbin/modprobe ipv6
```

Ambas opciones nos permiten habilitar IPv6 en el kernel de Linux, para cerciorarnos que se cargó correctamente el módulo de IPv6 en el kernel podemos hacer uso de la herramienta *lsmod*: `/home/lsmod`

Esta herramienta nos muestra todos los módulos que se encuentran actualmente cargados en el kernel del sistema y entre ellos debe aparecer IPv6.

¿Cómo configurar direcciones IPv6 en una interfaz sobre Linux?

Una vez que ya se cuenta con el protocolo IPv6 habilitado en el kernel de Linux solo es necesario seguir los siguientes pasos para configurar direcciones IPv6 en las interfaces:

- Obtener privilegios de administrador
- Abrir una consola de comandos
- Teclear los siguientes comandos:

```
# ifconfig <interfaz> add  <direcciónipv6>/<prefijo de
red>
# ifconfig <interfaz> up
```

Ejemplo:

```
# ifconfig eth0 add  3ffe:bc0:8000::370/128
# ifconfig eth0 up
```

Podemos que la dirección ha sido agregada a la interfaz utilizando el comando *ifconfig* para mostrar el estatus de las interfaces activas:

Ejemplo:

```
# /sbin/ifconfig -a
```

A.3 Habilitación de IPv6 sobre plataforma Solaris 8.0 y 9.0

SUN desarrolló un primer prototipo de implementación de IPv6 para Solaris 7 en 1995, el cual podía ser instalado adicionalmente, pero no fue sino hasta la liberación de Solaris 8 en febrero del año 2000 que se ofreció un soporte completo de IPv6 para la plataforma Solaris.

¿Cómo habilitar Ipv6 en Solaris 8/9?

Hay dos formas de habilitar IPv6 en Solaris:

- Primera forma:

Esta es la forma más sencilla de habilitar IPv6 en Solaris, ya que se realiza durante el proceso de instalación de la plataforma. En el transcurso de la instalación se nos solicita responder si deseamos habilitar IPv6 en la máquina donde se está instalando el sistema, tal como se puede apreciar en la tabla 2, a lo cual debemos escribir "SI"

Información necesaria para instalar	Descripción/Ejemplo	Introduce tu respuesta aquí
Red	¿El sistema está conectado a la red?	Si / No
DHCP	¿El sistema puede utilizar el Protocolo de Configuración de Host Dinámico (DHCP) para configurar sus interfaces de red?	Si / No
Nombre del Host	Nombre de Host que elijas para el sistema	
Dirección IP	Si no estás utilizando DHCP, proporciona la dirección IP para el sistema Ejemplo: 129.200.9.1	
Subred	Si no estás utilizando DHCP, el sistema es parte de una subred? De ser así cual es la máscara de red de la subred? Ejemplo: 255.255.0.0	Si / No
IPv6	**Deseas habilitar IPv6 en ésta máquina?**	**Si / No**

Tabla A.2. Información solicitada cuando se instala Solaris

- Segunda forma:

Si ya se tiene instalado Solaris y deseamos habilitar IPv6 de manera manual, tenemos que realizar los siguientes pasos:

- Convertirse en súper-usuario
- En la línea de comando escribir lo siguiente para cada interfaz de red:

```
# touch /etc/hostname6.interface

"interface" es el nombre de la interfaz de red, por ejemplo:
le0, le1
```

- Ahora debemos reiniciar la máquina

```
# reboot
```

- Para ver si se habilitó IPv6 en las interfaces de red podemos hacer uso de **ifconfig**:

```
# /sbin/ifconfig -a

-a muestra información acerca de todas las interfaces de red
```

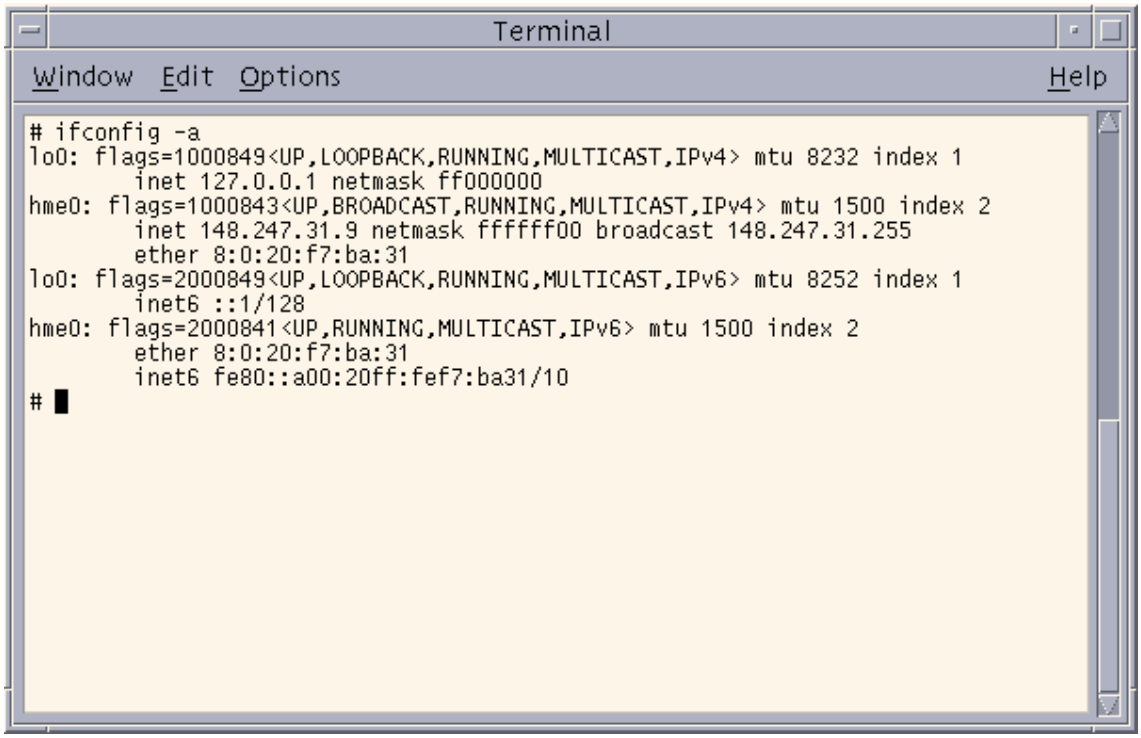


Figura A.5. Uso de *ifconfig* para ver información de interfaces de red en Solaris

¿Cómo configurar direcciones IPv6 en una interfaz sobre Solaris?

Una vez que ya se cuenta con el protocolo IPv6 habilitado en el *kernel* de solaris solo es necesario seguir los siguientes pasos para configurar direcciones IPv6 en las interfaces:

- Es necesario agregar esta dirección estática en el archivo que se creó al habilitar IPv6 en el kernel, esto es, el archivo **/etc/hostname6.***interface* donde *interface* es la interfaz en la cual se desee configurar la dirección IPv6.

Ejemplo:

Suponiendo que deseamos configurar la dirección IPv6 3ffe:bc0:8000::770 en la interfaz hme0 de la máquina mimaquinaipv6.midominio.com, tendríamos que editar el archivo /etc/hostname6.hme0 y agregar lo siguiente:

3ffe:bc0:8000::375 mimaquinaipv6.midominio.com mimaquinaipv6

Ahora solo es necesario reiniciar la máquina para notar los cambios.

A.4 Habilitación de IPv6 sobre plataforma Mac OS X 10.2 “Jaguar”

Apple ofreció un soporte completo de IPv6 a partir del lanzamiento del Mac OS X 10.2 en agosto del 2002. Antes de esta versión no se contaba con soporte para IPv6, salvo por unas pocas implementaciones por parte del grupo de desarrollo KAME.

¿Cómo habilitar Ipv6 en Mac OS X 10.2?

Los pasos a seguir para implementar ipv6 en el kernel de la versión jaguar de Mac OS X son los siguientes:

- Obtener privilegios de administrador
- Abrir una consola de comandos
- Ejecutar los siguientes comandos:

```
ip6config start-v6 [all] [interfaz]
```

ejemplo 1: `ip6config start-v6 all` para habilitar IPv6 en todas las interfaces

ejemplo 2: `ip6config start-v6 gif0` para habilitar IPv6 en la interfaz gif0

- Para verificar que efectivamente se ha habilitado IPv6 en el kernel del sistema hacemos uso de la utilería *ifconfig* tal como se muestra en la figura 6.

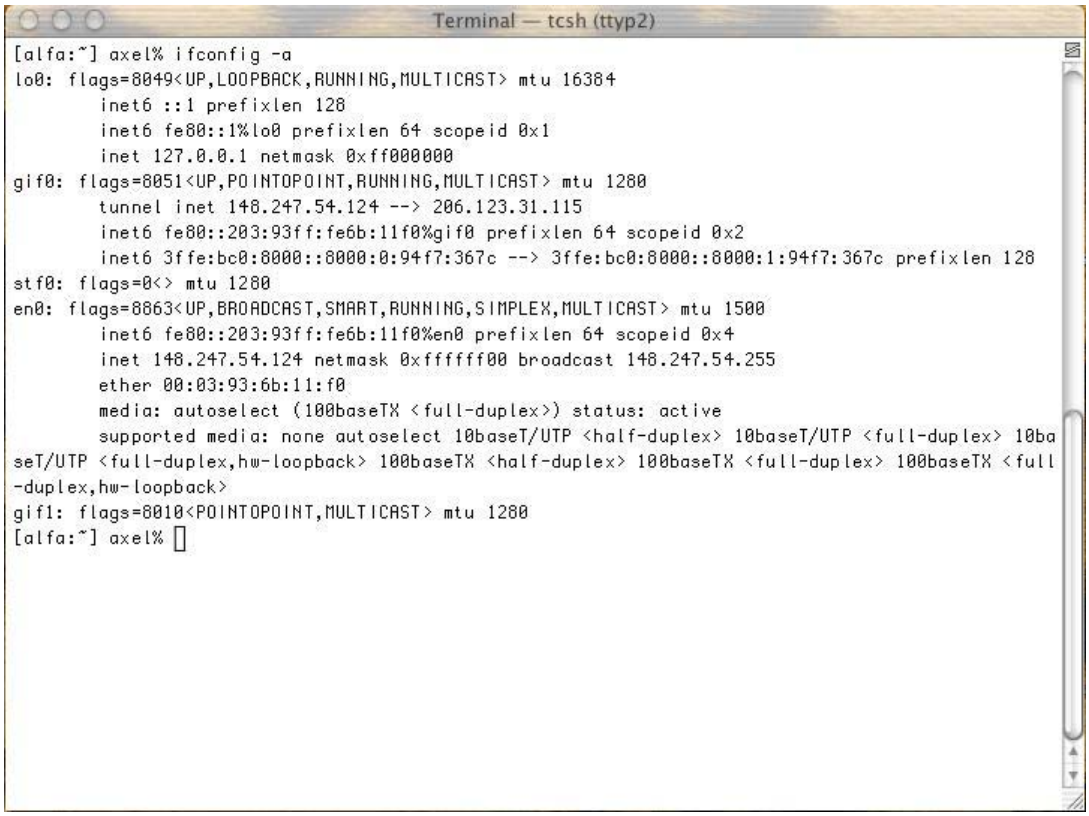


Figura A.6. Uso de *ifconfig* para verificar que IPv6 está habilitado en todas las interfaces de red en Mac OS X

¿Cómo configurar direcciones IPv6 en una interfaz sobre Mac OS X?

Una vez que ya se cuenta con el protocolo IPv6 habilitado en el *kernel* de Mac OS X solo es necesario seguir los siguientes pasos para configurar direcciones IPv6 en las interfaces:

Primer Caso: ya se cuenta con una dirección IPv6 estática

- Obtener privilegios de administrador
- Abrir una consola de comandos
- Teclear los siguientes comandos:

```
# ifconfig <interfaz> <tipo de dirección> <direcciónipv6>/<prefijo de red>
# ifconfig <interfaz> up
<tipo de dirección> puede ser inet o inet6
```

Ejemplo:

```
# ifconfig gif0 inet6  3ffe:bc0:8000::370/128
# ifconfig gif0 up
```

Para verificar que la dirección ha sido agregada a la interfaz podemos usar el comando *ifconfig* para mostrar el estatus de las interfaces activas:

Ejemplo:

```
# ifconfig -a
```

A.5 Habilitación de IPv6 sobre plataforma IOS

Cisco Internetwork Operating System (IOS)

Cisco ha estado inmerso en el desarrollo de IPv6 durante varios años mediante el 6BONE [31] que es una red de pruebas la cual corre sobre IPv6. Además ha incluido implementaciones de IPv6 en el sistema operativo de sus enrutadores a partir de la versión 12.2(2)T del IOS.

¿Cómo habilitar Ipv6 en IOS 12.2(11)T10?

Ya teniendo instalada una versión del IOS posterior a la 12.2(2)T (versiones anteriores no cuentan con soporte para IPv6) tenemos que seguir los siguientes pasos.

- Abrir una conexión con el enrutador
- Tener privilegios de administrador (enable)
- Introducir los siguientes comandos:

```
# conf t
# ipv6 unicast-routing
# ctrl. z
```

Estos comandos son suficientes para habilitar IPv6 en el IOS, sin embargo, falta habilitar IPv6 en cada una de las interfaces del enrutador y eso se logra con los siguientes comandos:

```
# conf t
# int FastEthernet0/0
# ipv6 enable
# ctrl. z

# conf t
# int FastEthernet0/1
# ipv6 enable
# ctrl. z
```

- Guardamos los cambios en la memoria no volátil

```
# write mem
```

Ahora que ya está habilitado IPv6 en el enrutador hacemos uso del comando show ipv6 brief (figura 7) para verificar que ya se cuenta con direcciones IPv6-locales en ambas interfaces, las cuales son asignadas mediante el protocolo de descubrimiento de vecinos de IPv6.

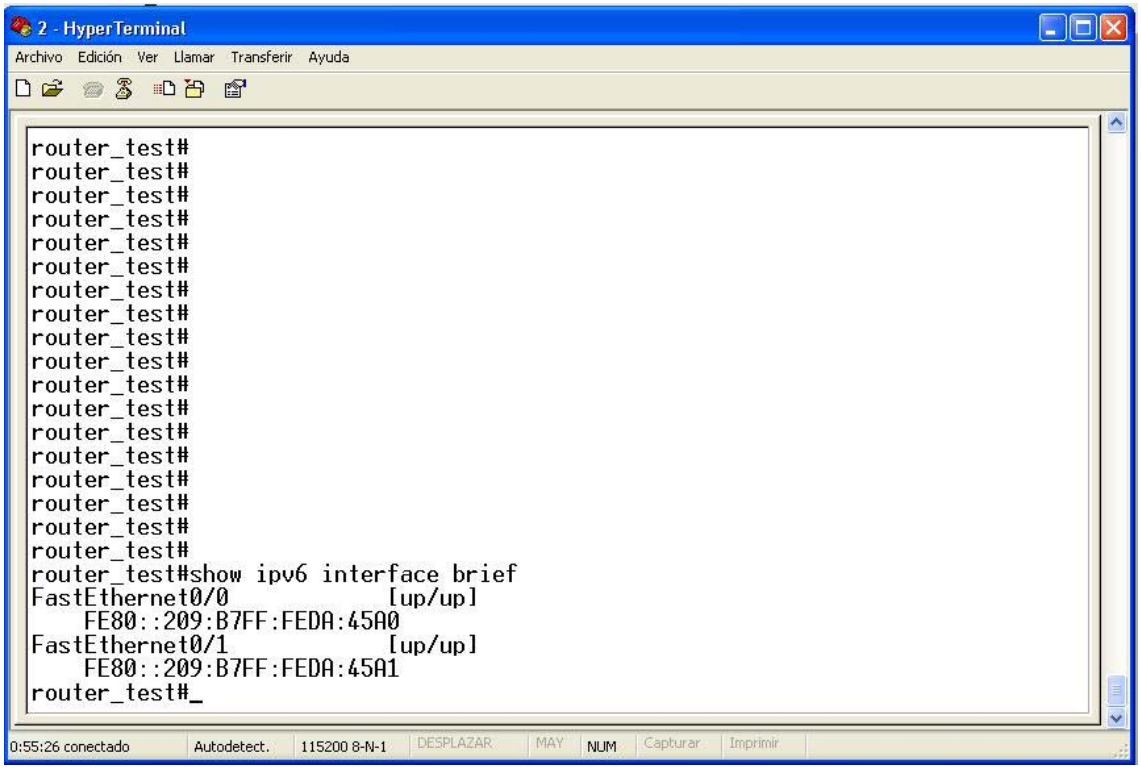


Figura A.7. Uso de comando *show ipv6 brief* para ver las direcciones IPv6 en las interfaces del enrutador Cisco

A.6 Configuración de direcciones IPv6 en Servidor DNS

Antes de configurar direcciones IPv6 en el servidor DNS (*Domain Name System*) que para nuestro caso de estudio es un servidor Linux, es necesario habilitar la resolución de direcciones IPv6 en dicho servidor DNS. Basándonos en la información del libro "DNS and BIND CookBook" [32] esto se logra mediante los siguientes pasos:

- Obtener privilegios de administrador (*root*)
- Editar el archivo /etc/named.conf
- Agregar las siguientes líneas:

```
options {
        directory "/var/named";
        listen-on-v6 { any; };
};
```

Con estas líneas conseguimos que el servidor de nombres pueda escuchar y resolver direcciones IPv6 desde cualquier puerto de comunicación

- Reiniciar el servicio de DNS

```
/etc/rc.d/init.d/named restart
```

Una vez hecho esto, procedemos a configurar las direcciones IPv6 al servidor DNS. Una de las razones por las cuales es necesario agregar las direcciones IPv6 al servidor es que la plataforma Windows solo reconoce nombres de dominio IPv6 completamente calificados y no en formato hexadecimal. La tabla 3 muestra un ejemplo de tales formatos.

Dirección IPv6 válida desde Windows (nombre de dominio completamente calificado)	Dirección IPv6 NO válida desde Windows (formato hexadecimal)
zeta6.test.cinvestav.mx	3ffe:bc0:8000::370

Tabla A.3. Formato de direcciones IPv6 válido e inválido en Windows XP

Los pasos a seguir para agregar las direcciones en el servidor DNS son los siguientes

- Obtener privilegios de administrador
- Editar archivos de zona e inverso de zona situados en "/var/named"
- En el archivo de zona agregar el nombre de dominio completamente calificado, seguido del tipo de registro "IN AAAA" y la dirección IPv6. En la figura 8 se muestra un ejemplo de configuración para el archivo de zona "test.cinvestav.mx.zone"

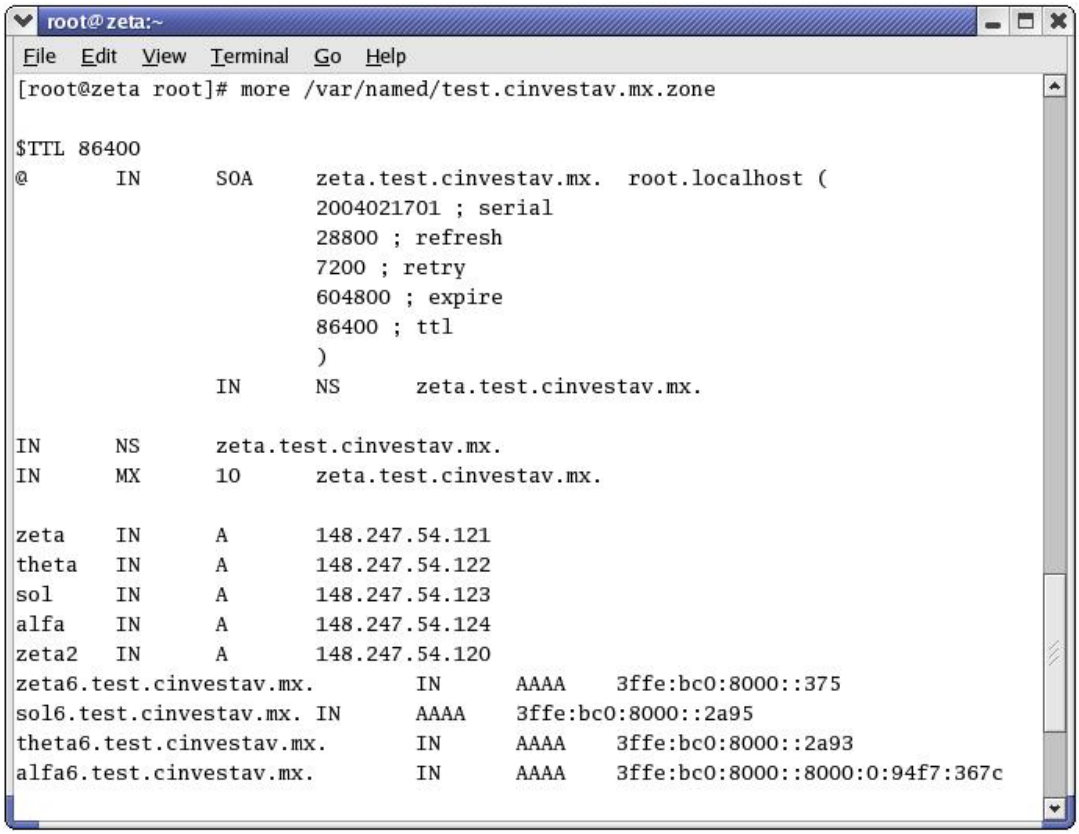


Figura A.8. Archivo de configuración para la zona test.cinvestav.mx: El tipo de registro AAAA mapea un nombre de dominio en una dirección IPv6, tal como un registro de tipo A mapea un nombre de dominio en una dirección IPv4

- En el archivo inverso de zona agregar la dirección IPv6, delimitándola en grupos de 4 bits separados por puntos "." en lugar de usar grupos de 16 bits separados por dos puntos ":". Agregar un registro "IN PTR" y el nombre de dominio completamente calificado. En la figura 9 se muestra un ejemplo de configuración para el archivo de zona inversa "54.247.148.in-addr.arpa.zone"

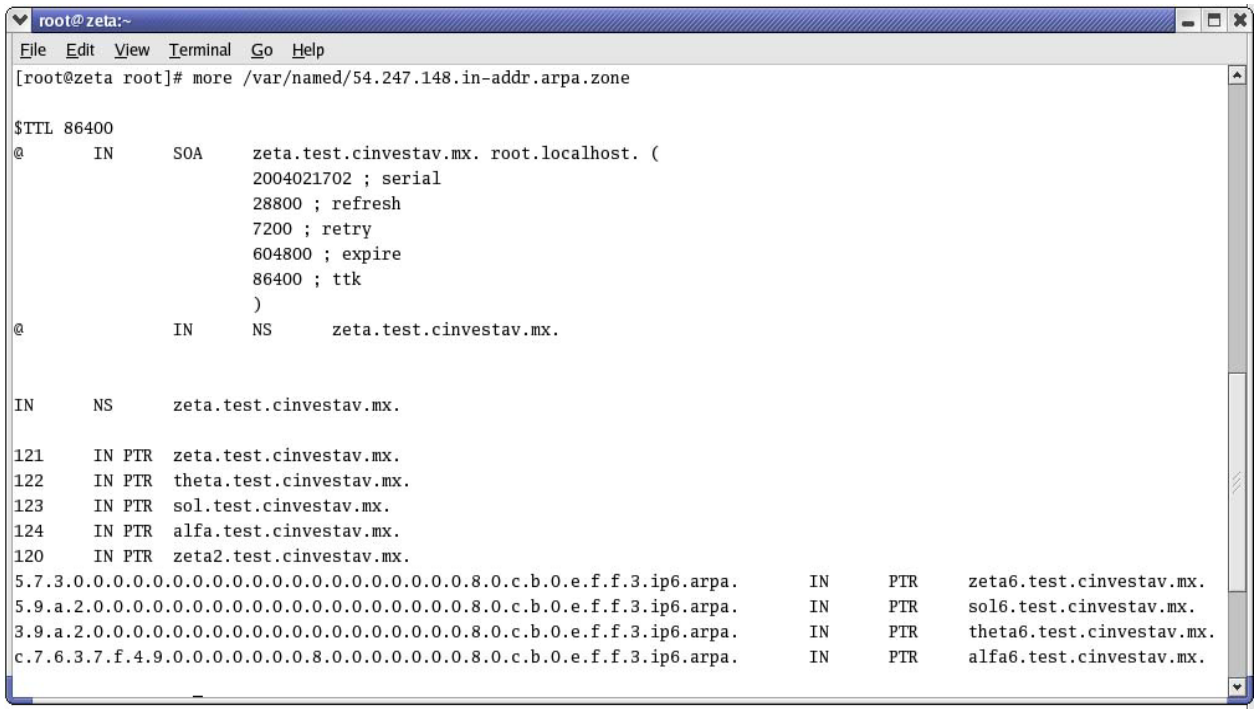


Figura A.9. Archivo de configuración para la zona inversa 54.247.148: El tipo de registro PTR mapea una dirección (IPv6/IPv4) a un nombre de dominio.

Para asegurarnos que nuestro servidor DNS es capaz de resolver las direcciones IPv6 que acabamos de agregar hacemos uso de la utilidad "dig" realizando consultas al servidor, tal como se muestra en la figura A.10.

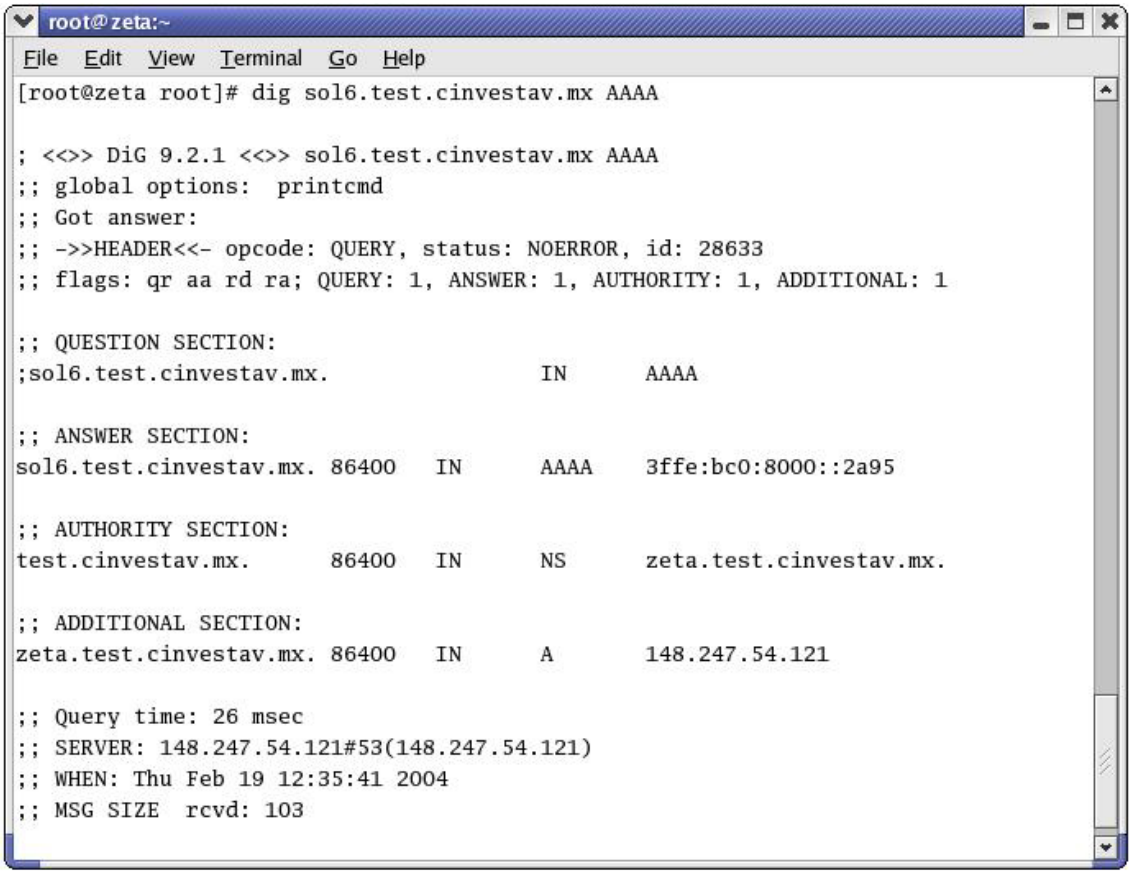


Figura A.10. Uso de la herramienta ***dig*** para realizar consultas IPv6 al servidor DNS

En la figura A.10 podemos ver que para realizar consultas de tipo IPv6 al servidor DNS es necesario especificar el tipo de registro de direcciones que usa IPv6: “AAAA” en lugar de “A” utilizado por IPv4. Una vez que se realiza la consulta por una dirección IPv6, el servidor DNS mostrará información acerca de dicha dirección, tal como su dirección, nombre de dominio completamente calificado, estado de autoridad (servidor DNS), etc.

Ahora que el servidor DNS es capaz de resolver direcciones IPv6 podemos realizar pruebas básicas desde nuestras plataformas, tales como las presentadas en la sección 3.3

B

Obtención de direcciones IPv6 globales a través de Freenet6

En esta tesis se utilizaron los servicios de Freenet6 para obtener direcciones IPv6 Globales. En este apéndice mostraremos como configurar direcciones IPv6 Globales por medio de clientes Freenet6 en las 4 plataformas de prueba.

B.1. Antecedentes de Freenet6

Freenet6 es un proyecto 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. Los túneles configurados utilizan una arquitectura cliente - servidor por medio de un modelo de broker IPv6 como se muestra en la figura 1.

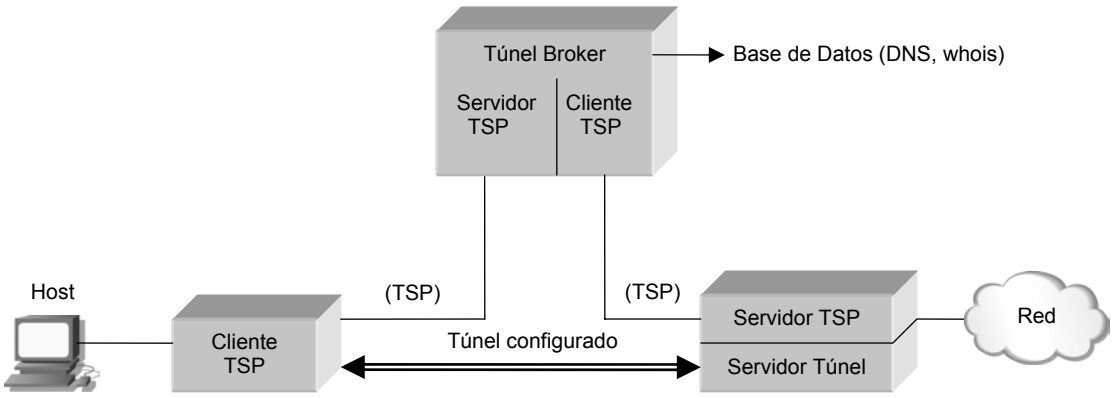


Figura B.1. Arquitectura de TSP

En un modelo de Túnel Broker el broker es el responsable de toda la comunicación entre los servidores de túnel y los clientes de túnel. Los clientes solicitan túneles al broker y este 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.

B.2. Configuración de cliente Freenet6 sobre Windows XP

Para configurar el cliente de Freenet6 sobre la plataforma Windows XP y así obtener una dirección IPv6 Global de manera instantánea solo es necesario seguir los siguientes pasos:

- Descargar cliente de Freenet6: Usualmente este puede ser descargado de la página http://www.freenet6.net

- Editar el archivo tspc.conf y configurar las siguientes variables:

  - tsp_dir = [ruta_directorio] :aquí indicamos la ruta donde se encuentra instalado el cliente tsp
  - client_v4 = [[nnn].[nnn].[nnn].[nnn] | auto]: en esta variable debemos indicar la dirección IPv4 de nuestra interfaz que servirá como entrada del túnel IPv6 sobre IPv4 con freenet6; puede tomar dos valores distintos, ya sea el número IP en el formato **[nnn].[nnn].[nnn].[nnn]** ó el valor ***auto*** en cuyo caso se utiliza el valor de la interfaz activa.
  - userid = [nombre usuario | anonymous]
  - passwd = [| password]
  - template = [S.O.]
  - server = [dirección del broker de túnel]
  - retry_delay = [tiempo (seg)]
  - if_tunnel = [interfaz]

  Ejemplo de configuración:

  tsp_dir = **.**
  client_v4 = ***auto***
  userid = **axel123**
  passwd = **BEYc1ZFz2f**
  template = **WindowsNT-2K**
  server = **tsps2.freenet6.net**
  retry_delay = **0**
  if_tunnel = **eth0**

- Iniciar comunicación con el Servidor: para ejecutar el cliente solo necesitamos hacer lo siguiente: Nos situamos en el directorio donde instalamos el cliente freenet6 y ejecutamos:

  ..\freenet6-0.9.8**tspc –vvf tspc.conf**

  Con la opción –v solicitamos que se desplieguen en pantalla los mensajes que surjan como resultado del establecimiento del túnel (verbose) y con la opción –f solicitamos que se tome como archivo de configuración tspc.conf.

  Si el cliente logra establecer un túnel IPv6 sobre IPv4 con Freenet6 devolverá un valor de 0, en otro caso devolverá un valor de 1. En la figura 2 mostramos un ejemplo de la salida que devuelve el cliente TSP al levantar un túnel 6to4 con Freenet6 sobre la plataforma Windows XP. En este caso se devuelve un valor de 0 el cual indica que no hubo ningún error en el establecimiento del túnel.

Figura B.2. Salida retornada por cliente TSP sobre Windows XP

B.3. Configuración de cliente Freenet6 sobre Red Hat Linux 8.0 y 9.0

Para configurar el cliente de Freenet6 sobre la plataforma Red Hat Linux y así obtener una dirección IPv6 Global de manera instantánea solo es necesario seguir los siguientes pasos:

- Descargar cliente de Freenet6: Usualmente este puede ser descargado de la página http://www.freenet6.net y puede instalarse manualmente ó como un rpm.

- Editar el archivo tspc.conf, el cual por lo general se encuentra en /usr/local/freenet6-client-versión/bin y configurar las siguientes variables:

  - tsp_dir = [ruta_directorio] :aquí indicamos la ruta donde se encuentra instalado el cliente tsp
  - client_v4 = [[nnn].[nnn].[nnn].[nnn] | auto]: en esta variable debemos indicar la dirección IPv4 de nuestra interfaz que servirá como entrada del túnel IPv6 sobre IPv4 con freenet6; puede tomar dos valores distintos, ya sea el número IP en el formato **[nnn].[nnn].[nnn].[nnn]** ó el valor ***auto*** en cuyo caso se utiliza el valor de la interfaz activa.
  - userid = [nombre usuario | anonymous]
  - passwd = [| password]
  - template = [S.O.]
  - server = [dirección del broker de túnel]

- retry_delay = [tiempo (seg)]
- if_tunnel = [interfaz]

Ejemplo de configuración:

tsp_dir = **.**
client_v4 = ***auto***
userid = **axel123**
passwd = **BEYc1ZFz2f**
template = **linux**
server = **tsps2.freenet6.net**
retry_delay = **0**
if_tunnel = **eth0**

- Iniciar comunicación con el Servidor: para ejecutar el cliente solo necesitamos hacer lo siguiente: Nos situamos en el directorio /usr/local/freenet6-client-verión/bin y ejecutamos:

../freenet6-client-1.0/bin/**tspc –vvf tspc.conf**

Con la opción –v solicitamos que se desplieguen en pantalla los mensajes que surjan como resultado del establecimiento del túnel (verbose) y con la opción –f solicitamos que se tome como archivo de configuración tspc.conf.. Si el cliente logra establecer un túnel IPv6 sobre IPv4 con Freenet6 devolverá un valor de 0, en otro caso devolverá un valor de 1.

Podemos verificar que la dirección ha sido agregada a la interfaz usando el comando *ifconfig* para mostrar el estatus de las interfaces activas:

Ejemplo:

```
# ifconfig -a
```

B.4. Configuración de cliente Freenet6 sobre Solaris 8.0 y 9.0

Para configurar el cliente de Freenet6 sobre la plataforma Solaris solo es necesario seguir los siguientes pasos:

- Descargar cliente de Freenet6: Usualmente este puede ser descargado de la página http://www.freenet6.net

- Compilar el paquete. Esto se hace mediante el siguiente comando:
  make all target=solaris8
  (solaris 9 también usa target=solaris8)
- Instalar el paquete. Utilizar el siguiente comando:
  make install target=solaris8 installdir= =/usr/local/tsp

- Editar el archivo tspc.conf, el cual por lo general se encuentra en /usr/local/tsp/bin y configurar las siguientes variables:

  - tsp_dir = [ruta_directorio] :aquí indicamos la ruta donde se encuentra instalado el cliente tsp
  - client_v4 = [[nnn].[nnn].[nnn].[nnn] | auto]: en esta variable debemos indicar la dirección IPv4 de nuestra interfaz que servirá como entrada del túnel IPv6 sobre IPv4 con freenet6; puede tomar dos

valores distintos, ya sea el número IP en el formato **[nnn].[nnn].[nnn].[nnn]** ó el valor ***auto*** en cuyo caso se utiliza el valor de la interfaz activa.

- userid = [nombre usuario | anonymous]
- passwd = [| password]
- template = [S.O.]
- server = [dirección del broker de túnel]
- retry_delay = [tiempo (seg)]
- if_tunnel = [interfaz]

Ejemplo de configuración:

tsp_dir = **.**
client_v4 = ***auto***
userid = **axel123**
passwd = **BEYc1ZFz2f**
template = **solaris8**
server = **tsps2.freenet6.net**
retry_delay = **0**
if_tunnel = **ip.tun0**

- Iniciar comunicación con el Servidor: para ejecutar el cliente solo necesitamos hacer lo siguiente: Nos situamos en el directorio /usr/local/freenet6-client-verión/bin y ejecutamos:

../tsp/bin/**tspc –vvf tspc.conf**

Con la opción –v solicitamos que se desplieguen en pantalla los mensajes que surjan como resultado del establecimiento del túnel (verbose) y con la opción –f solicitamos que se tome como archivo de configuración tspc.conf.. Si el cliente logra establecer un túnel IPv6 sobre IPv4 con Freenet6 devolverá un valor de 0, en otro caso devolverá un valor de 1.

Podemos verificar que la dirección ha sido agregada a la interfaz usando el comando *ifconfig* para mostrar el estatus de las interfaces activas:

Ejemplo:

```
# ifconfig -a
```

B.5. Configuración de cliente Freenet6 sobre Mac OS X 10.2

Para configurar el cliente de Freenet6 sobre la plataforma Mac OS X solo es necesario seguir los siguientes pasos:

- Descargar cliente de Freenet6: Usualmente este puede ser descargado de la página http://www.freenet6.net

- Compilar el paquete. Esto se hace mediante el siguiente comando:
  make all target=darwin
  (solaris 9 también usa target=solaris8)
- Instalar el paquete. Utilizar el siguiente comando:
  make install target=darwin installdir= =/usr/local/tsp

- Editar el archivo tspc.conf, el cual por lo general se encuentra en /usr/local/tsp/bin y configurar las siguientes variables:

  - tsp_dir = [ruta_directorio] :aquí indicamos la ruta donde se encuentra instalado el cliente tsp
  - client_v4 = [[nnn].[nnn].[nnn].[nnn] | auto]: en esta variable debemos indicar la dirección IPv4 de nuestra interfaz que servirá como entrada del túnel IPv6 sobre IPv4 con freenet6; puede tomar dos valores distintos, ya sea el número IP en el formato **[nnn].[nnn].[nnn].[nnn]** ó el valor ***auto*** en cuyo caso se utiliza el valor de la interfaz activa.
  - userid = [nombre usuario | anonymous]
  - passwd = [| password]
  - template = [S.O.]
  - server = [dirección del broker de túnel]
  - retry_delay = [tiempo (seg)]
  - if_tunnel = [interfaz]

  Ejemplo de configuración:

```
tsp_dir = .
client_v4 = auto
userid = axel123
passwd = BEYc1ZFz2f
template = darwin
server = tsps2.freenet6.net
retry_delay = 0
if_tunnel = gif0
```

- Iniciar comunicación con el Servidor: para ejecutar el cliente solo necesitamos hacer lo siguiente: Nos situamos en el directorio /usr/local/freenet6-client-verión/bin y ejecutamos:

```
../tsp/bin/tspc –vvf tspc.conf
```

  Con la opción –v solicitamos que se desplieguen en pantalla los mensajes que surjan como resultado del establecimiento del túnel (*verbose*) y con la opción –f solicitamos que se tome como archivo de configuración tspc.conf. Si el cliente logra establecer un túnel IPv6 sobre IPv4 con Freenet6 devolverá un valor de 0, en otro caso devolverá un valor de 1.

Podemos verificar que la dirección ha sido agregada a la interfaz usando el comando *ifconfig* para mostrar el estatus de las interfaces activas:

Ejemplo:

```
# ifconfig –a
```

C

Configuración de *Hosts* virtuales IPv6/IPv4 en Apache

El termino *Host* Virtual se refriere a tener más de un servidor web sobre una máquina diferenciando ambos servidores ya sea solo por su nombre de host aparente o por su dirección IP aparente.

Host Virtual Basado en Nombre: Cuando se tienen diferentes servidores en una sola maquina diferenciados por su nombre de host se le conoce como Hosts Virtuales basados en Nombre.
Ejemplo1, si dos compañías comparten un servidor para tener sus propios dominios, tales como http://www.compañia1.com y http://www.compañia2.com dichos dominios serán independientes el uno del otro a pesar de tener ambos la misma dirección IP de la maquina donde se encuentran alojados dichos servidores.

Host Virtual Basado en Dirección IP: Cuando se tienen diferentes servidores en una sola maquina diferenciados por una distinta dirección IP se le conoce como Hosts Virtuales basados en Dirección IP.

Ejemplo2, si dos compañías comparten un servidor para tener sus propios dominios, tales como http://www.compañia1.com y http://www.compañia2.com y además desean tener direcciones IP distintas por cuestiones de seguridad, dichos dominios serán independientes el uno del otro a pesar de compartir la misma maquina donde se encuentran alojados dichos servidores, ya que cada uno de estos dominios cuenta con su propia dirección IP. Esta característica también puede ser utilizada para mantener dos servidores (IPv4 e IPv6) en una sola máquina.

C.1. Configuración de *Host* virtual

Para configurar tanto el host virtual en IPv4 como en IPv6 en el servidor Apache hicimos uso de la directiva de configuración ***<VirtualHost>***; esta directiva nos permite diferenciar ambos dominios, además de poder asignar ya sea el mismo o diferentes directorios de publicación HTML mediante la variable ***DocumentRoot.***

Por ejemplo, para configurar nuestro servidor web y que este pueda escuchar peticiones tanto por IPv4 como por IPv6, en el archivo de configuración /etc/httpd/conf/httpd.conf definimos dos hosts virtuales tal como se muestra abajo:

```
<VirtualHost 148.247.54.121>
  DocumentRoot /var/www/sitipv4
  ServerName zeta.test.cinvestav.mx
  ServerAlias zeta
</VirtualHost>
```

```
<VirtualHost [3ffe:bc0:8000::375]>
  DocumentRoot /var/www/sitipv6
  ServerName zeta6.test.cinvestav.mx
  ServerAlias zeta2
</VirtualHost>
```

Donde VirtualHost define la dirección IPv4/IPv6 por donde el servidor va a atender peticiones HTTP. DocumentRoot define el directorio de publicación web, este puede ser el mismo para ambos servidores (IPv4/IPv6) o distinto si se desea tener diferentes contenidos en ellos. Server Name define el nombre que va a identificar al dominio generado por nuestro servidor y Server Alias es un nombre corto para identificar el dominio definido por ServerName.

C.2. Uso de SSL

Secure Sockets Layer (SSL) es un protocolo que provee un canal de comunicación seguro entre dos máquinas con la finalidad de autenticar ambas máquinas y proteger los datos que transitan entre ellas. El canal creado por SSL es transparente para cualquier protocolo que pueda correr sobre TCP. En nuestro caso configuramos SSL para ver si este también resultaba ser transparente a IPv6, es decir que pudiéramos realizar la comunicación con el servidor web vía IPv6 mediante una conexión segura. En la figura de abajo mostramos los ajustes necesarios para habilitar conexiones seguras sobre IPv6 en el servidor Apache:

```
<VirtualHost 148.247.54.121>
  DocumentRoot /var/www/sitipv4
  ServerName zeta.test.cinvestav.mx
  ServerAlias zeta
  SSLEngine on
  SSLCertificateFile /etc/httpd/conf/ssl.crt/server.crt
  SSLCertificateKeyFile /etc/httpd/conf/ssl.key/server.key
</VirtualHost>

<VirtualHost [3ffe:bc0:8000::375]>
  DocumentRoot /var/www/sitipv6
  ServerName zeta6.test.cinvestav.mx
  ServerAlias zeta2
  SSLEngine on
  SSLCertificateFile /etc/httpd/conf/ssl.crt/server.crt
  SSLCertificateKeyFile /etc/httpd/conf/ssl.key/server.key
</VirtualHost>
```

SSLCertificateFile y SSLCertificateKeyFile indican el directorio donde se encuentran el certificado y llave que autentican al servidor.

D

Configuración de Servidor y clientes LDAP

D.1. Configuración del servidor LDAP

En nuestro caso elegimos como servidor LDAP la plataforma RedHat Linux 8.0 y la versión 2.0.27 de OpenLDAP debido a la gran disponibilidad, tanto de información, como de los paquetes necesarios para su configuración. Los rpm's los obtuvimos de la página http://rpm.pbone.net/ y son los siguientes:

- openldap-2.0.27-2.8.0.i386.rpm
- openldap-clients-2.0.27-2.8.0.i386.rpm
- openldap-devel-2.0.27-2.8.0.i386.rpm
- openldap-servers-2.0.27-2.8.0.i386.rpm

Una vez instalados los paquetes necesarios, lo siguiente fue editar los archivos slapd.conf y ldap.conf. En el archivo ldap.conf es donde especificamos la dirección de nuestro servidor LDAP, así como también el directorio base donde nos situamos al realizar una consulta.

BASE `dc=zeta, dc=test, dc=cinvestav, dc=mx`
URI `ldap://zeta.test.cinvestav.mx ldap://zeta.test.cinvestav.mx:389`
HOST `3ffe:bc0:8000::375`

La variable HOST contiene la dirección que va a escuchar el directorio LDAP, ya sea IPv4 o IPv6.

En el archivo slapd.conf definimos el tipo de base de datos back-end a utilizar para guardar el directorio, así como también el administrador del directorio, su *password* de acceso, el dominio de nuestro servidor, además de otros parámetros.

```
database    ldbm
suffix          "dc=zeta,dc=test,dc=cinvestav,dc=mx"
rootdn          "cn=Manager,dc=zeta,dc=test,dc=cinvestav,dc=mx"
# Cleartext passwords, especially for the rootdn, should
# be avoided.  See slappasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw            {crypt}ijFYNcSNctBYg
```

Para generar un password cifrado hicimos uso de la utilería slappasswd situado en /usr/sbin/

Ya modificados estos dos archivos solo falta levantar el servidor, esto lo hacemos por medio del demonio ldap situado en /etc/rc.d/init.d/

```
# /etc/rc.d/init.d/ldap start
```

Para mayor información el CD adjunto contiene los archivos de configuración del servidor LDAP.

D.2. Probando el servidor

Para probar si el servidor estaba funcionando correctamente hicimos uso de un comando de búsqueda *ldapsearch* para hacer una consulta al servidor:

/usr/bin/ldapsearch –x –LL –b 'dc=zeta,dc=test,dc=cinvestav,dc=mx' '(cn=*)'

Describamos las opciones:

-x utiliza autenticación simple, en lugar de SASL
-L muestra los resultados de la consulta en formato LDIF, el uso de una doble L elimina los comentarios
-b este comando sirve para indicar el punto de inico donde comenzaremos la búsqueda en el directorio
Si el servidor funciona correctamente obtendremos algún tipo de respuesta.

En seguida, cargamos nuestro directorio en formato LDIF al servidor LDAP. Esto se hace utilizando el comando *ldapadd*

/usr/bin/ldapadd –x –D "cn=Manager, dc=zeta, dc=test, dc=cinvestav, dc=mx" –W –f arch1.ldif

-D usa un nombre distinguido para ligarse al directorio
-W pregunta por autenticación simple, es mejor que mandar el *password* escrito en el comando
-f lee el directorio en formato LDIF que está en el archivo

Por último, si volvemos a hacer la consulta, ahora obtendremos información sobre el contenido del directorio que acabamos de introducir.

D.3. Configuración de los clientes LDAP

La configuración de las máquinas cliente se hizo de la siguiente manera:

- Solaris 9: solo fue necesario añadir el paquete para el cliente OpenLDAP, así como el cliente SSL y el back-end para la base de datos utilizado por OpenLDAP. Estos paquetes uno los puede descargar del sitio de SUN microsystems. También se instaló el J2SDK 1.4

- Windows XP: se instaló el Service Pack1 y se instaló el J2SDK 1.5beta ya que las versiones anteriores del J2SDK muestran problemas de interoperabilidad con el Stack IPv6 de Windows XP

- MacOS X 10.2: Esta plataforma ya soporta LDAP, solo fue necesario instalar el J2SDK 1.4

- Linux RedHat 9: se instalaron los siguientes paquetes:

    - ✓ OpenLDAP-clients
    - ✓ OpenLDAP-devel
    - ✓ OpenLDAP
    - ✓ J2SDK 1.4

E

Configuración de VideoLAN

E.1. Instalación del servidor VideoLAN

Para Instalar el servidor VLS (VideoLAN Server) sobre la plataforma Linux fue necesario instalar antes los siguientes paquetes:

- libvcd-devel
- libvcd
- libcdio.so
- libiso9660.so
- libcddb
- ffmpeg
- faac
- faad2
- imlib2
- lame
- libfaad.so
- libp3lame.so
- ffmpeg-devel
- libpostproc
- libmpeg
- pmeg2dec
- mpeg2dec-devel

Una vez que ya se han instalado es necesario compilar e instalar la aplicación VLS

```
# ./configure
# make
# make install
```

Por ultimo se inicia el servicio vls mediante el comando: `# vls`

E.2. Instalación del cliente VideoLAN

Instalación sobre Windows

Para instalar el cliente VLC (VideoLAN Client) sobre Windows solo es necesario descargar la aplicación desde http://www.videolan.org/vlc/download-windows.html y ejecutar el archivo de instalación vlc-versión-win32.exe

Instalación sobre Linux

Para instalar el cliente VLC (VideoLAN Client) sobre Linux fue necesario instalar antes los siguientes paquetes:

- libdvdcss
- libmad
- libvcdinfo
- libvcd-devel
- libvcd
- libcdio.so
- libiso9660.so
- libcddb

Después de instalar estos paquetes el siguiente paso es descargar el cliente desde el sitio http://www.videolan.org/vlc/download-redhat.html y descomprimirlo. Una vez hecho esto el siguiente paso es compilarlo e instalarlo

```
# ./configure
# make
# make install
```

Por ultimo se inicia el servicio vlc mediante el comando: `# vlc`

Instalación sobre Mac OS X

Para instalar el cliente VLC (VideoLAN Client) sobre Mac OS X el primer paso es descargar la aplicación desde el sitio http://www.videolan.org/vlc/download-macosx.html y copiar el contenido de este archivo a la carpeta "Aplicaciones"
Para ejecutarlo solo es necesario dar doble *click* sobre la aplicación.

E.3. Configuración del servidor VideoLAN

Para configurar el servidor VLS (VideoLAN Server) es necesario modificar el archivo vls.cfg. En este archivo se definen los canales de comunicación y las direcciones IPv4/IPv6 de los clientes que escucharán las transmisiones de VLS. En nuestro caso de estudio solamente fue necesario modificar 3 secciones del archivo vls.cfg: La sección "*Inputs*" en la cual definimos el archivo a ser transmitido sobre IP, la sección "*channels*" en la que se define el medio de transmisión de video y la sección "*channels configuration*" en la cual definimos las direcciones IPv4/IPv6 que escucharán estas transmisiones. En la tabla e.1 se muestra un ejemplo de esta configuración.

Archivo vls.cfg

```
BEGIN "1"     # MPEG2 stream stored in /home/videolan/streams/Dolby.vob
  Name     = "dolby"
  FileName = "/home/videolan/streams/VTS_01_5.VOB"
  Type     = "Mpeg2-PS"
END

#BEGIN "2"     # another file
#  Name     = "canyon"
#  FileName = "Dolby_Canyon.vob"
#  Type     = "Mpeg2-PS"
#END

#BEGIN "3"     # DVD
#  Name       = "film"
#  Device     = "/dev/cdrom"
#  Type       = "dvd"
#  DvdTitle   = "1"    # Start title
#  DvdChapter = "1"    # Start chapter
#END

# ChannelName   = "Type"
# --- Example:
  localhost     = "network"
  client1       = "network"     # windows_ipv4
  client2       = "network"     # mac_ipv4
  client3       = "network"     # mac_ipv6
  client4       = "network"     # windows_ipv6
  multicast     = "network"
  bcastipv4     = "network"     # bcast_ipv4
  ipv6_win      = "network"     #ipv6_en_mac
#  localfile     = "file"
END

# Channels configuration
BEGIN "localhost"         # The client is on the same host as the server
  DstHost = "127.0.0.1"
  DstPort = "1234"
END

BEGIN "client1"                          # unicast example
  DstHost = "148.247.94.123"             # destination host
  DstPort = "1234"                       # destination port
END

BEGIN "client2"                          # unicast example
  DstHost = "148.247.94.124"             # destination host
  DstPort = "1234"                       # destination port
END

BEGIN "bcastipv4"                        # broadcast example
  Domain  = "inet4"                      #tipo de direcciones
  Type    = "broadcast"
  DstHost = "148.247.94.255"             # destination host
  DstPort = "1234"                       # destination port
END

BEGIN "ipv6_win"                          # unicast with IPv6 example
  Domain  = "inet6"
  DstHost = "2002:94f7:5e01:1:230:84ff:fe88:d7d6"         # destination
domain
  DstPort = "1234"                       # destination port
END
```

Tabla E.1 Ejemplo de configuración servidor VLS

E.4. Configuración de los clientes VideoLAN

No es necesaria ninguna configuración adicional para los clientes VLC

Bibliografía

[1] Ettikan Kandasamy Karuppiah, Gopi Kurup, Takefumi Yamazaki,"Application Performance Analysis In Transition Mechanism From Ipv4 to IPv6", *Proceedings APAN Conf. 2000*, Tsukuba, Japan, Feb. 14-18, 2000. http://www.my.apan.net/ipv6/Papers/ettikan.PDF

[2] Ettikan Kandasamy, Tong Hui Tee, Seow Chen Yong, "Transition Mechanism Between IPv4 & IPv6 and deciding your choice", *APAN Conf. 2002*, Jan. 22-25, Paper, http://www.my.apan.net/ipv6/Papers/Transition-Mechanism-IPv4_IPv6.pdf

[3] Ettikan Kandasamy ,"IPv6 Dual Stack Transition Technique Performance Analysis : KAME on FreeBSD as the Case", *Proceedings MMU International Symposium on Information and Communication Technologies*, Malaysia, 5th - 6th Oct., 2000. http://www.my.apan.net/ipv6/Papers/M2USIC_Perf.PDF

[4] Reporte de espacio de direcciones IPv4, http://bgp.potaroo.net/ipv4/

[5] Network Wizards, http://www.nw.com

[6] F. Solensky, "IPv4 Address Lifetime Expectations," in IPng: Internet Protocol Next Generation (S. Bradner, A. Mankin, ed), Adddison Wesley, 1996

[7] "*Global IPv6 allocations made by the Regional Internet Registries*", http://www.ripe.net/cgi-bin/ipv6allocs

[8] S.Bradner, A.Mankin, "*IP: Next Generation (IPng) White Paper Solicitation*", RFC 1550, diciembre 1993, http://www.ietf.org/rfc/rfc1550.txt?number=1550

[9] C. Partridge, F. Kastenholz, "*Technical Criteria for Choosing IP The Next Generation (IPng)*", RFC 1726, diciembre 1994, http://www.ietf.org/rfc/rfc1726.txt?number=1726

[10] S. Bradner, A. Mankin, "The Recommendation for the IP Next Generation Protocol", RFC 1752, enero 1995, http://www.ietf.org/rfc/rfc1752.txt

[11] Hinden, R. and S. Deering, "*IP Versión 6 Addressing Architecture*", RFC 1884, diciembre 1995, http://www.ietf.org/rfc/rfc1884.txt

[12] Rekhter, Y., and T. Li. "*An Architecture for IPv6 Unicast Address Allocation*", RFC 1887, diciembre 1995, http://www.ietf.org/rfc/rfc1887.txt

[13] Peter Todd, "Get IPv6 Now with Freenet6 ", *Linux Journal* , enero 2003

[14] R. Atkinson, "Security Architecture for the Internet Protocol", RFC 1825, agosto 1995, http://www.ietf.org/rfc/rfc1825.txt

[15] R. Atkinson, "IP Authentication Header", RFC 1826, agosto 1995, http://www.ietf.org/rfc/rfc1826.txt

[16] R. Atkinson, "IP Encapsulating Security Payload", RFC 1827, agosto 1995, http://www.ietf.org/rfc/rfc1827.txt

[17] R. Rivest, "The MD5 Message-Digest Algorithm", RFC 1321, abril 1992, http://www.ietf.org/rfc/rfc1321.txt

[18] MD-5, http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html , http://www.rsasecurity.com/rsalabs/faq/3-6-6.html

[19] SHA-1, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf , http://www.rsasecurity.com/rsalabs/faq/3-6-5.html

[20] R. Hinden, B. Carpenter, L.Masinter, "Format for Literal IPv6 Addresses in URL's", RFC 2732, diciembre 1999, http://www.ietf.org/rfc/rfc2732.txt

[21] R. Gilligan, E.Nordmark, "Transition Mechanisms for Ipv6 Hosts and Routers ", RFC 2893, agosto 2000, http://www.ietf.org/rfc/rfc2893.txt

[22] G. Tsirtsis, P. Srisuresh, "Network Address Translation – Protocol Translation (NAT-PT)", RFC 2766, febrero 2000, http://www.ietf.org/rfc/rfc2766.txt

[23] A. Durand, P. Fasano, I. Guardini, D. Lento, "IPv6 Tunnel Broker", RFC 3053, enero 2001, http://www.ietf.org/rfc/rfc3053.txt

[24] Ibrahim Haddad, David Gordon, "Apache Talkng Ipv6", *Linux Journal*, enero 2003

[25] A. Conta, S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (Ipv6) Specification", RFC 2463, diciembre 1998, http://www.ietf.org/rfc/rfc2463.txt

[26] M. Wahl, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, diciembre 1997, http://www.faqs.org/rfcs/rfc2251.html

[27] John Nagle,"The Nagle Algorithm",RFC 896, enero 1984, http://www.ietf.org/rfc/rfc0896.txt?number=896

[28] S. Deering, R. Hinden, "Internet Protocol Version 6 (Ipv6) Specification"RFC 2460, diciembre 1998, http://www.ietf.org/rfc/rfc2460.txt

[29] Linux IPv6 Router, http://staff.csc.fi/~psavola/ipv6/

[30] Solaris IPv6 Router, http://www.optix.org/~dxy/solaris/ipv6/

[31] 6BONE, http://www.6BONE.net

[32] Cricket Liu, *DNS & BIND Cookbook*, 4th ed., 2002, O'Reilly