Universidad Central de Venezuela

Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

Evaluación del Desempeño de Diferentes Tecnologías de Transición IPv4-IPv6

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
por el Bachiller:

Quintero S. Adira Y.
V-18.720.739
adira0411@gmail.com

para optar al título de Licenciado en Computación

Tutores**:**
Prof. Eric Gamess y Prof. Francisco Sans

Caracas, Octubre 2015

Universidad Central de Venezuela

Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

ACTA DE VEREDICTO

Quienes suscriben, miembros del jurado designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por la Bachiller Adira Quintero, identificada por la cédula de identidad V-18.720.739, con el título "**Evaluación del Desempeño de Diferentes Tecnologías de Transición IPv4-IPv6**", a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el nombrado trabajo por cada uno de los miembros del jurado, éste fijó el día 14 de octubre de 2015, para que su autora lo defendiera en forma pública, en el laboratorio de Internet II de la Escuela de Computación, mediante una exposición oral de su contenido, luego de la cual respondió satisfactoriamente a las preguntas que le fueron formuladas por el jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela.

Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente acta en Caracas a los catorce días del mes de octubre del año dos mil quince, dejando constancia que el Profesor Eric Gamess actuó como coordinador del jurado.

Prof. Eric Gamess
Tutor

Prof. Francisco Sans
Tutor

Prof. Roger Bello
Jurado Principal

Prof. Miguel Astor
Jurado Principal

Agradecimientos

Primero que todo le doy gracias a Dios por permitirme culminar mi tan anhelada meta de egresar como Lic. En Computación de la UCV a pesar de las múltiples adversidades presentadas. Gracias por dotarme con una familia tan maravillosa que día tras día me apoya y brinda el cariño necesario que me impulsa a seguir. Dedico este trabajo a ellos, especialmente a padres, hermanos, abuelo y todos aquellos que ya no están con nosotros físicamente. ¡Mi triunfo es para ustedes!

Una vez más, gracias a mis padres, Elcida Sánchez e Ysauro Quintero por hacer de mí la persona que hoy día soy, por brindarme su cariño, amor, fortaleza y confianza. Gracias por estar siempre conmigo en las buenas y en las no tan buenas, sin duda, mi triunfo les pertenece. ¡Los adoro!

Gracias a mis hermanos, Dulce y José Ángel Quintero por apoyarme siempre hasta en el mas desatinado de mis inventos y estar a mi lado para hacer que ponga los pies sobre la tierra cada vez que es necesario. Gracias por acompañarme en esta travesía tan maravillosa de estudiar en la casa que vence la sombras y entenderlo a la perfección porque ustedes también transitan uno de los caminos que ella brinda. Gracias por estudiar conmigo las 24 horas del día sobre todo en los momentos cruciales de la carrera y regañarme cuando hacia lo contrario. Gracias por siempre estar conmigo.

Gracias a mi sobrino Ysauro Sebastian Enrique por ser ese rayito de luz en mi vida y en la de mi familia, por llenar de alegría mi hogar, por hacer amenos mis días y mis noches cantando, bailando, gritando y llenándome de besitos, ternura y amor. Gracias a mi abuelo Ángel de Jesús Sánchez por quererme tanto y aguantar todas las travesuras de su querida nieta ¡Gracias por existir!

Gracias a mi Tutor, mi profesor, mi amigo, Eric Gamess, por su constante disposición e invaluables enseñanzas, por compartir conmigo sus conocimientos y esperarme un poquito cuando realmente lo necesite. Gracias por aguantarme tanto tiempo en LACORE y sobre todo gracias por creer en mí. Sin duda fue un acierto pedirle que fuera mi Tutor, usted es un excelente Profesor. Gracias a Francisco Sans por su apoyo, por su disposición y colaboración con nosotros y con el proyecto, Gracias por ser mi Tutor.

Gracias a la UCV y a aquellos profesores que ayudaron en mi formación y crecimiento personal y profesional. Gracias a Luis, Julio, Richard y Roger por acompañarme todos los días en el Laboratorio y apoyarme, ayudarme e impulsarme a seguir cada vez que encontraba un obstáculo en la investigación.

Gracias a mis tías, primos, primas, amigos, amigas, y a todas las personas que han puesto su granito de arena para poder alcanzar mi meta. Gracias a los que ya no están con nosotros, que sin duda, desde el cielo celebran conmigo y con todos mis afectos. Gracias a todos los que no nombré pero que fueron cruciales angelitos en mi camino.

¡Mil Gracias a todos!

Quintero S. Adira Y.

Resumen

Título:
Evaluación del Desempeño de Diferentes Tecnologías de Transición IPv4-IPv6

Autor:
Quintero S. Adira Y.

Tutores:
Prof. Eric Gamess y Prof. Francisco Sans

El Internet se basa actualmente en el protocolo IPv4. Debido al crecimiento rápido e inesperado del Internet, el pool central de direcciones IPv4 administrado por el IANA se agotó en febrero de 2011. Una situación similar pasó poco tiempo después con APNIC y RIPE NCC, los RIRs de Asia y Europa, respectivamente. Pronto, LACNIC, el RIR para América Latina, también agotará sus bloques de direcciones IPv4. IPv6, la nueva versión del protocolo IP, es la respuesta a esta problemática por parte de la comunidad de Internet. IPv6 tiene un enorme número de direcciones, y ha mejorado ciertos aspectos de IPv4. Como no es posible llevar a cabo la transición de IPv4 a IPv6 en un corto período de tiempo; IPv6 debe implementarse gradualmente, y durante algunos años se vivirá la coexistencia de los dos protocolos. Para ello, se han creado una serie de mecanismos de transición con diferentes características, teorías operativas y disponibilidad. Por esta razón, es importante evaluar el desempeño que presentan estos métodos de transición para ayudar a los investigadores y los administradores de red, en la selección de la tecnología más adecuada según sus necesidades y según el desempeño que cada una muestra.

En este Trabajo Especial de Grado se realiza la evaluación del desempeño de IPv4 e IPv6 nativo, de dos mecanismos de túnel (ISATAP y 6to4) y de una tecnología de traducción (NAT64), en escenarios de prueba reales. Las métricas reportadas incluyen el OWD y el throughput que son reportados para Ethernet y Fast Ethernet, cuando se utilizan diferentes sistemas operativos (Windows 7, Windows 8.1, Windows 10 y Debian 7). A partir de esta investigación, se puede concluir que IPv4 nativo tiene el mejor desempeño. Sin embargo, debido a la escasez de direcciones IPv4, es necesario ir abandonándolo. IPv6 nativo muestra el segundo mejor desempeño, y es la solución a largo plazo. Para el periodo de transición, ISATAP y 6to4 son dos buenas posibilidades para túneles con un desempeño muy similar, y la elección de uno u el otro dependerán de las necesidades de red. En caso de requerir un traductor, NAT64 parece ser el candidato, ya que tiene un buen desempeño como se muestra en esta investigación, es un estándar del IETF, y que su mayor competidor (NAT-PT) ha sido llevado al estado obsoleto hace pocos años.

Palabras Claves: Evaluación del Desempeño, Tecnologías de Transición, IPv4, IPv6, ISATAP, 6to4, NAT64, Tayga, Jool, Túneles, Traductores.

Tabla de contenido

Índice de Figuras

Índice de Tablas

1. Introducción

Internet Protocol version 4 (IPv4) es el protocolo que ha soportado el Internet desde hace años. IPv4 tiene un espacio de direcciones de 32 bits (2^{32}), es decir, 4.294.967.296 direcciones IP.

El reducido espacio de direcciones de IPv4 y la falta de coordinación para su asignación durante la década de los 80s, sin ningún tipo de optimización, dejando incluso espacios de direcciones discontinuos, generan en la actualidad dificultades no previstas en aquel momento (por ejemplo, el problema de la sumarización). Esto, sumado a otras limitaciones de enrutamiento, seguridad, configuración, calidad de servicio (QoS) y al crecimiento exponencial del Internet aumentó la necesidad de la creación de un nuevo protocolo.

Internet Protocol versión 6 (IPv6) aparece para solventar las limitaciones de IPv4 y aumentar funcionalidades, sin embargo, las transiciones no son fáciles y la transición de IPv4 a IPv6 no es la excepción. Durante el proceso de transición la red IPv6 debe interactuar con la red IPv4, y para lograr la interoperabilidad entre ellos se han creado diferentes técnicas o tecnologías denominadas mecanismos de transición IPv4-IPv6.

Dado que existen diferentes mecanismos de transición (ISATAP, 6to4, Teredo, NAT64, etc) y que de dichos mecanismos se han desarrollado diferentes implementaciones bajo los entornos Linux y Windows, nace el objetivo que motiva a realizar la presente investigación, que comprende principalmente la evaluación del desempeño de diferentes tecnologías de Transición de IPv4-IPv6 con las implementaciones desarrolladas en ambos sistemas operativos.

De acuerdo a las necesidades planteadas, el resto del trabajo especial de grado se encuentra estructurado de la siguiente forma:

2. **El Problema:** En este capítulo se realiza un planteamiento formal del problema que será estudiado y los objetivos que se desean alcanzar.
3. **Internet Protocol version 6 (IPv6):** En el capítulo se describen las características más importantes de este protocolo, última versión existente del protocolo de Internet, cuya intención es mejorar algunos aspectos de su predecesor IPv4.
4. **Mecanismos de Transición:** En este capítulo se describen los mecanismos de transición disponibles para las categorías dual-stack (doble pila), túneles y traductores.
5. **Trabajos Relacionados:** Este capítulo presenta los trabajos relacionados recopilados durante la investigación.
6. **Benchmarking:** Este capítulo describe las técnicas, métricas y herramientas de evaluación de desempeño.
7. **Experimentos y Análisis de Resultados:** En este capítulo se describen los escenarios y experimentos diseñados para la evaluación del rendimiento de los mecanismos de transición IPv4-IPv6 y se realiza un estudio analítico de los resultados obtenidos.
8. **Conclusiones.**

2. El Problema

En este capítulo se realiza un planteamiento formal del problema que será estudiado y los objetivos que se desean alcanzar.

2.1 Planteamiento del Problema

En febrero de 2011, el pool central de direcciones IPv4 del IANA se agotó, dos meses después lo hizo el pool del RIR (Regional Internet Registery) de Asia/Pacífico (APNIC) y a finales de 2012 ocurrió en Europa (RIPE NCC)[1]. Una situación similar ocurrirá pronto con Norte América (ARIN) y Latinoamérica (LACNIC).

Debido a la evidente escasez de direcciones IPv4, recurso vital para el correcto funcionamiento del Internet, el IETF ha desarrollado una nueva versión del protocolo de Internet llamado IPv6, que solventa las limitaciones de IPv4 y aumenta considerablemente el número de direcciones y funcionalidades.

Una de las premisas del diseño de IPv6, fue que pudiera realizarse una transición suave hacia la nueva versión del protocolo IP, sin que fuera necesario pasar de una versión a otra en forma abrupta[2]. Durante este proceso de transición la red IPv6 debe interactuar con la red IPv4, y para lograr la interoperabilidad entre ellos se han creado diferentes técnicas o tecnologías denominadas mecanismos de transición IPv4-IPv6. Estos mecanismos han sido diseñados con diferentes teorías operativas (dual stack, túnel y traducción) y su disponibilidad depende del entorno de red y de la existencia de su implementación en un determinado sistema operativo.

En consecuencia, existe una gran variedad de tecnologías para la transición y es importante evaluar su rendimiento para ayudar a los usuarios, investigadores y administradores de red en la escogencia de la tecnología que más se adapte a sus requerimientos, mediante la comparación del desempeño de los diferentes mecanismos de transición en diferentes sistemas operativos.

2.2 Justificación

Inicialmente se pensó que la adopción gradual de IPv6 tendría la rapidez necesaria como para ir desplazando a IPv4 antes del agotamiento del pool de direcciones de IPv4. Sin embargo, esto no sucedió así, dando a los mecanismos de transición una importancia mayor en el proceso de implantación y adopción total de IPv6.

Se han diseñado muchos mecanismos para lograr la convivencia entre ambas versiones de IP, y dependiendo de la técnica operativa, la implementación o el entorno de red en el que se está realizando el proceso de transición, puede presentar mejor desempeño un mecanismo u otro, surgiendo la necesidad de evaluar el rendimiento de IPv4 e IPv6 nativo y de las diferentes técnicas para realizar una comparación analítica entre ellos y ayudar en la selección del mecanismo a utilizar.

[1] http://portalipv6.lacnic.net/como-es-la-transicion
[2] http://portalipv6.lacnic.net/mecanismos-de-transicion

2.3 Objetivos

En esta sección se definen los objetivos que se quieren conseguir.

2.3.1 Objetivo General

Evaluación del desempeño de diferentes tecnologías de transición de IPv4-IPv6 en diferentes sistemas operativos.

2.3.2 Objetivos Específicos

- Seleccionar los mecanismos de transición IPv4-IPv6 a evaluar, con implementaciones en los sistemas operativos Linux y Windows.
- Obtener un amplio conocimiento de la arquitectura y procedimientos operativos de los mecanismos de transición IPv4-IPv6 seleccionados.
- Proponer un conjunto de escenarios, métricas y herramientas a utilizar durante la evaluación.
- Probar en cada escenario y sistema operativo los mecanismos de transición y hacer mediciones de desempeño.
- Realizar el análisis de los resultados obtenidos.

2.4 Alcance

Se trata de medir el desempeño de IPv4 nativo, IPv6 nativo, y tecnologías de transición en un ambiente controlado, para Windows y Linux.

3. Internet Protocol version 6 (IPv6)

IPv6 es el protocolo diseñado por el IETF (Internet Engineering Task Force) para reemplazar a su predecesor IPv4.

IPv4 es un protocolo que tiene más de veinte años y aun cuando es bastante funcional, actualmente empieza a tener algunas deficiencias, como la escasez de direcciones en algunos países[3].

3.1 Características de IPv6

IPv6 es un protocolo con nuevas características y algunas mejoras sobre IPv4. Las características más relevantes se mencionan a continuación:

- **Direccionamiento extendido:** IPv6 ataca el problema de escasez de direcciones IPv4, incrementando el tamaño de las direcciones de 32 bits (IPv4) a 128 bits (IPv6).
- **Simplificación de la cabecera:** A pesar de que el encabezado aumenta de tamaño, se simplifica y pasa de tener 13 campos (IPv4) a tener 8 campos (IPv6).
- **Soporte para opciones mejorado:** En IPv4 el espacio para opciones va de 0 a 40 bytes, mientras que en IPv6 se maneja con encabezados de extensión lo cual permite encadenar las opciones y no tener un límite de espacio.
- **Direccionamiento jerárquico:** Mejoramiento de las capacidades de enrutamiento dando mayores facilidades en el proceso de agregación.
- **Direcciones multicast mejoradas:** Se eliminan las direcciones de broadcast. Si es necesario enviar un mensaje a todos los nodos de una misma subred, se hace a través de direcciones multicast. Este mecanismo ya existía en IPv4 pero ahora toma fuerza y las direcciones multicast de IPv6 tienen un alcance.
- **Direcciones anycast:** Además de hacer uso intensivo de las direcciones multicast, IPv6 introduce el concepto de direcciones anycast. Estas nuevas direcciones sirven para enviar un paquete al nodo más cercano de un grupo determinado.
- **Mejor soporte de QoS:** En IPv4 existe un campo de 8 bits (Type of Service) para soportar QoS. En IPv6 se tienen dos campos para este propósito (Traffic Class y Flow Label).
- **Fragmentación en el emisor:** La fragmentación de paquetes se utiliza para enviar un paquete cuyo tamaño es mayor al MTU (Maximum Tranfers Unit). En IPv4 la fragmentación se puede dar tanto en el emisor como en cualquier dispositivo intermedio de capa de red. En IPv6, sólo ocurre en el emisor lo cual disminuye la carga de los routers.
- **Soporte para movilidad:** Se refiere a la posibilidad que tiene un equipo de mantener la misma dirección IP aun después de cambiar de red. En IPv4 existe un proceso que realiza esta función, sin embargo en IPv6 se crea un protocolo especial para manejar esta situación.

[3] http://www.ipv6.org

- **Autoconfiguración:** Otro aspecto muy importante de IPv6 es que está hecho para permitir la autoconfiguración, es decir, que cada dispositivo final de una red podrá obtener automáticamente sus direcciones IPv6.

3.2 Cabecera IPv6

La cabecera IPv6 consta de 40 bytes divididos en ocho campos. Los campos de la cabecera IPv6 se muestran en la Figura 3.1(tomada de [02]):

- **Version:** El campo de 4 bits “Version” indica la versión del protocolo de Internet que se utiliza, en este caso el valor del campo siempre debe ser 6.
- **Traffic Class:** El campo de 8 bits “Traffic Class” facilita el manejo de datos en tiempo real o que requieran cualquier tipo de trato especial. Los nodos emisores y routers intermedios pueden utilizarlo para identificar y distinguir entre diferentes clases o prioridades de paquetes IPv6. Este campo es equivalente al campo “Tipo de Servicio” de IPv4 [04].

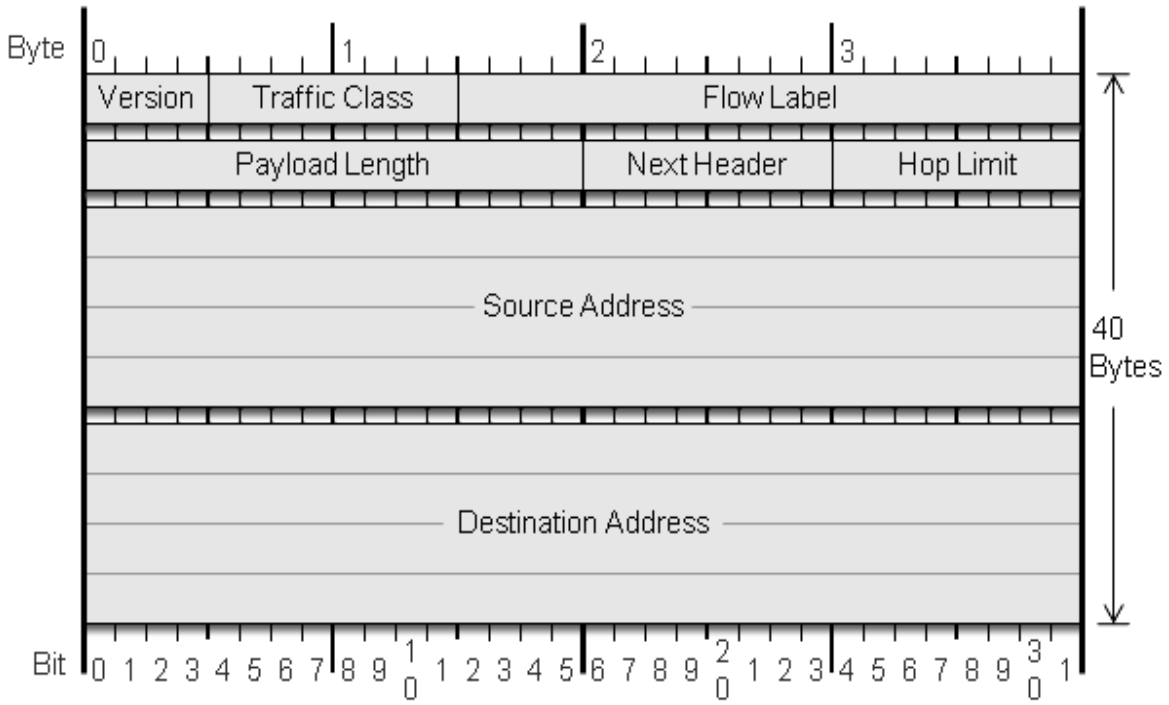


Figura 3.1: Cabecera IPv6

- **Flow Label:** El campo de 20 bits “Flow Label” distingue paquetes que poseen el mismo procesamiento con el fin de mejorar el tráfico en tiempo real. Un emisor etiqueta secuencias de paquetes con diferentes opciones. Los routers pueden procesar paquetes de la misma secuencia de forma más eficiente dado que no deben reprocesar su cabecera [04].
- **Payload Length:** El campo de 16 bits “Payload Length” indica cual es la longitud en bytes de la carga útil, incluyendo las cabeceras de extensión. El valor debe ser un entero sin signo [02].

- **Next Header:** El campo de 8 bits “Next Header” contiene el número que identifica el protocolo o cabecera de extensión que sigue inmediatamente después de la cabecera IPv6.
- **Hop Limit:** El campo de 8 bits “Hop Limit” especifica el número máximo de routers por los cuales puede ser transmitido un paquete antes de ser descartado. Es un contador que va en decremento a medida que el paquete es reenviado [03].
- **Source Address:** El campo de 128 bits “Source Address” indica la dirección IPv6 del emisor del mensaje.
- **Destination Address:** El campo de 128 bits “Destination Address” representa la dirección IPv6 del receptor del mensaje. Si la cabecera de extensión Routing Header está presente, es posible que la dirección de destino no sea el nodo final sino algún nodo intermedio.

3.2.1 Cabeceras de Extensión

En IPv4 existe un campo de opciones que comienza en el byte 20, que puede medir entre 0 y 40 bytes. Este campo, cuando existe, es verificado en todos los nodos desde el origen hasta el destino, lo que produce mayor latencia (ya que habrá más procesamiento en cada nodo). Para mejorar el manejo de opciones, IPv6 utiliza cabeceras de extensión, las cuales no son verificadas en los nodos intermedios (a excepción de la opción “Hop-by-Hop”), reduciendo la latencia.

Existen seis cabeceras de extensión distintas:

- Hop-by-Hop Options
- Routing
- Fragment
- Destination Options
- Authentication
- Encapsulating Security Payload

Un paquete IPv6 puede contener cero o más cabeceras de extensión localizadas entre la cabecera IPv6 y la cabecera del protocolo de capa superior. Cada cabecera de extensión es identificada por el campo “Next Header” de la cabecera previa. Véase la Figura 3.2.

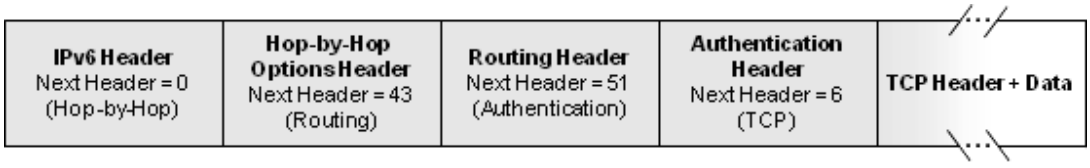


Figura 3.2: Cabeceras de Extensión

La longitud de las cabeceras de extensión debe ser un número entero múltiplo de 8 bytes. En caso de que la opción no cumpla con dicho requerimiento se utilizarán bits de relleno para lograrlo [03].

Las cabeceras de extensión son procesadas en el orden en que se presentan. Los nodos intermedios sólo toman en cuenta la opción “Hop-by-Hop” por lo tanto ésta debe ser la primera cabecera de extensión. Adicionalmente el RFC 2460 [02], recomienda utilizar el siguiente orden para concatenar las cabeceras:

1. IPv6 header
2. Hop-by-Hop Options header
3. Destination Options header (para opciones que serán procesadas en el primer destino que aparece en el campo Destination Address, más los destinos subsecuentes listados en el “Routing Header” [02])
4. Routing header
5. Fragment header
6. Authentication header
7. Encapsulating Security Payload header
8. Destination Options header (para opciones que serán procesadas sólo por el destinatario final del paquete)
9. Upper Layer header (Cabecera de capa superior)

A continuación se explicarán las seis cabeceras de extensión:

- **Hop-by-Hop Options Header:** Esta cabecera posee información opcional que debe ser examinada por cada uno de los nodos a lo largo del camino del paquete. La cabecera Hop-by-Hop debe venir inmediatamente después de la cabecera IPv6, la cual utiliza el valor del campo Next Header en cero (0) para indicarlo [04].

  La cabecera Hop-by-Hop posee tres campos que son: Next Header, Header Extension Length y Options (que puede ser de tamaño variable).

  El campo de 8 bits “Next Header” indica cual será la cabecera que sigue después de procesadas las opciones.

  El campo de 8 bits “Header Extension Length” indica cual será el tamaño del siguiente campo en unidades de 8 bytes. Si la siguiente opción utiliza menos de 64 bits, entonces el valor del campo “Header Extension Length” será cero. Puede haber una o más opciones en el campo “Options”. El tamaño en el campo Options es variable y es determinado según el campo “Header Extension Length”.

  Dos de las opciones más destacadas son:

  1. *Type Jumbogram*. El campo “Payload Length” de la cabecera IPv6 ocupa 16 bits, esto significa que la carga útil máxima de un paquete es 2^{16} bytes. Cuando se desea usar la opción Jumbo Payload (Hop-by-Hop Options Type = 194) el valor del campo “Payload Length” se coloca en cero (0), y se otorga un nuevo campo para colocar el tamaño de la carga útil. Este nuevo campo es de 32 bits, permitiendo así una carga útil máxima de 2^{32}-1 bytes.
  2. *Router Alert*. Le indica a los nodos intermedios que el paquete contiene información importante para el momento de reenviar. Se utiliza mayormente por los protocolos Multicast Listener Discovery y Resource Reservation Protocol [04].

- **Routing Header:** Esta cabecera es usada por un emisor IPv6 para listar uno o más nodos que deben ser "visitados" en el camino hacia el destino. El "Routing Header" se identifica en el campo "Next Header" con el valor 43 [04].

  El "Routing Header" posee cinco campos: Next Header, Header Extension Length, Routing Type, Segments Left y type-specific data. Next Header y Header Extension Length cumplen la misma función que en el Hop-by-Hop Options Header. El campo type-specific data es de tamaño variable, y múltiplo de 8 bytes. Este campo depende del valor que indique el campo Routing Type. En caso de que el valor del Routing Type sea desconocido por el nodo que lo recibe, deberá actuar dependiendo del valor que se indica en el campo Segments Left. Si este campo no contiene los nodos que deben ser visitados se debe ignorar el Rounting Header y procesar la siguiente cabecera, si el campo Segments Left no es cero, el nodo debe descartar el paquete y enviar un mensaje ICMP de tipo Parameter Problem con el campo Code en cero ("0") [4].
- **Fragment Header:** Si un emisor IPv6 desea enviar un paquete a su destinatario, debe utilizar el protocolo Path MTU discovery para determinar el tamaño máximo que puede tener el paquete a lo largo del camino hacia el destino. Si el paquete es más grande que el Path MTU soportado, entonces se fragmenta. En IPv6 a diferencia de IPv4 la fragmentación se lleva a cabo sólo en el origen, y no en los nodos intermedios. El receptor sabe cómo manejar los fragmentos para reensamblarlos. El valor del campo "Next Header" correspondiente a esta opción es 44.
- **Destination Options Header:** La cabecera "Destination Options" es usada para llevar información opcional que debe ser examinada en los destinatarios. Esta cabecera se identifica con el valor 60 en el campo "Next Header" de la cabecera previa.

  La cabecera "Destination Options" puede aparecer dos veces en un paquete IPv6, cuando se inserta antes del "Routing Header" contiene información que debe ser procesada en la lista de routers que ofrece. Cuando la cabecera "Destination Options" se inserta antes del protocolo de capa superior, la información debe ser procesada en el nodo destinatario.
- **Encapsulating Security Payload (ESP):** El propósito de esta opción es encriptar la información que se envía en el paquete IP. El encriptado se obtiene aplicando transformaciones especificas en los datos que se quieren proteger. Los datos se pueden empezar a encriptar en cualquier parte después de la cabecera IP y antes de la cabecera de capa de transporte [07]. El protocolo ESP forma parte de la arquitectura de seguridad del protocolo de Internet (IPSec). ESP proporciona una comprobación de integridad, autenticación y cifrado para los datagramas del protocolo de Internet (IP)[4].
- **Authentication Header (AH):** Esta cabecera es utilizada para proveer integridad y autenticación a los datos de origen para el datagrama IP. La información de autenticación es calculada usando todos los campos del datagrama que no varían durante la transmisión. Esta cabecera debe ser colocada después del "Fragment Header" y antes del protocolo de capa de transporte. Si se implementa ESP, el Authentication Header debe ir antes del ESP.

[4] http://publib.boulder.ibm.com/html/as400/v4r5/ic2931/info/RZAFM231ESPDEFANDCO.HTM

3.3 Direccionamiento en IPv6

A continuación se profundizará en los aspectos relacionados con las direcciones IPv6, tales como su representación y sus diferentes tipos. El RFC 4291 [05] define estos aspectos.

3.3.1 Notación de las Direcciones IPv6

El tamaño de una dirección IPv6 es de 128 bits, lo que quiere decir que se tiene un máximo de 2^{128} direcciones disponibles. Para su representación, los 128 bits se dividen en bloques de 16 bits, y cada uno de estos es convertido a cuatro dígitos hexadecimales separados por el símbolo dos puntos (":") [03].

Una dirección IPv6 puede ser simplificada eliminando los ceros no significativos de un bloque, siempre dejando al menos un dígito.

Finalmente es posible realizar una compresión de ceros, reemplazando bloques de ceros consecutivos con doble dos puntos ("::"). En la Figura 3.3 se puede apreciar el proceso completo.

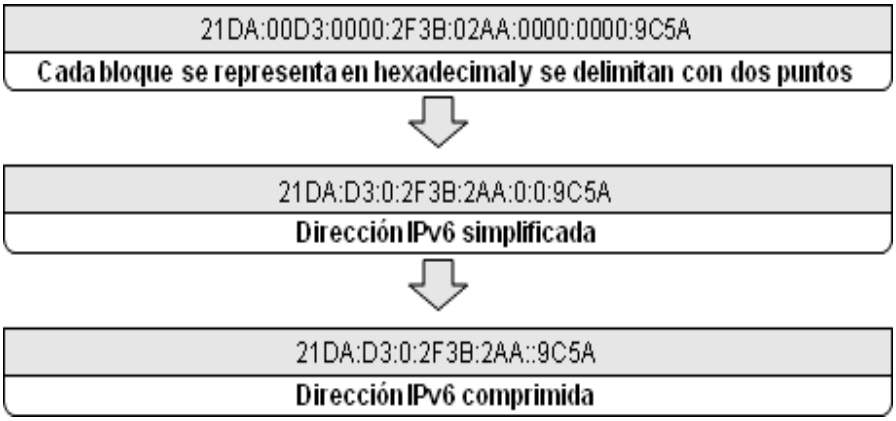


Figura 3.3: Representación de Dirección IPv6

3.3.2 Notación de los Prefijos en IPv6

Un prefijo son los bits más significativos de una dirección IPv6, los cuales son usados para identificar una subred o un tipo de dirección en específico. La notación es la siguiente: <dirección IPv6>/<longitud del prefijo>.

En la práctica, todas las subredes poseen un prefijo de 64 bits, es por ello que cuando se habla de una dirección unicast no es necesario indicar la longitud.

3.3.3 Direcciones Unicast

Es un identificador para una interfaz única. Los paquetes dirigidos a una dirección de este tipo son entregados solamente a la interfaz identificada con esta dirección.

- **Global Unicast:** Son direcciones globales identificadas con un formato de prefijo iniciado por los bits 001. Las direcciones globales de IPv6 son equivalentes a las direcciones públicas de IPv4, son enrutables mundialmente y alcanzables en Internet.

Su estructura está definida en el RFC 4291 [05] y se puede apreciar en la Figura 3.4 (tomada de [05]).

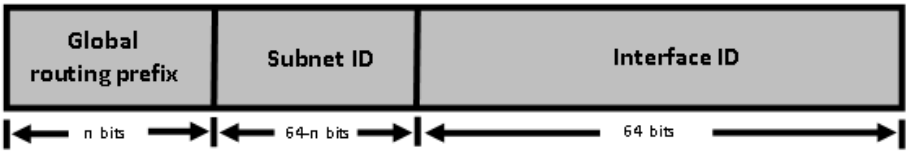


Figura 3.4: Estructura de la Dirección Global Unicast

El campo "global routing prefix" identifica el rango de direcciones asignado por los RIR[5] y el proveedor de Internet (ISP). El campo "subnet ID" identifica el enlace o subred. Por último, el campo "interface ID" identifica una interfaz en la subred y debe ser única para esa subred.

El prefijo 2001:DB8::/64 se encuentra reservado para ser utilizado con fines documentales.

- **Direcciones especiales:** En IPv6 se definen las siguientes direcciones especiales:
  - Unspecified (no especificada): La dirección no especificada es usada solamente para indicar la ausencia de dirección válida. Es comúnmente usada como dirección fuente cuando una dirección no ha sido asignada aún. Esta dirección posee el valor 0:0:0:0:0:0:0:0, por lo cual aplicando el método de compresión de ceros se puede abreviar como ::.
  - Loopback: La dirección loopback 0:0:0:0:0:0:0:1 es usada para identificar una interfaz loopback, permitiendo enviar paquetes a sí mismo sin salir a la red, lo cual es conveniente para comprobar el correcto funcionamiento de la pila IPv6. Se puede abreviar como ::1.
  - Direcciones de transición: Creadas para ayudar al proceso de transición de IPv4 a IPv6 y permitir la coexistencia de ambos tipos de hosts (Capítulo 1).
- **Link-Local:** Se identifican por el prefijo FE80::/64. Son usadas por los nodos al comunicarse con sus vecinos en el mismo enlace, su alcance es la subred y nunca deben ser enrutadas. Pueden ser utilizadas en mecanismos de autoconfiguración, en Neighbor Discovery y en redes sin routers, por lo tanto son bastante útiles en la creación de redes temporales. En la Figura 3.5 (tomada de [05]) se muestra la estructura.

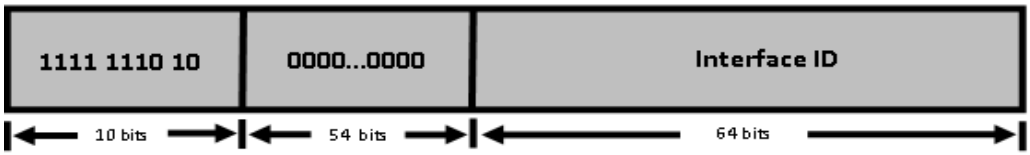


Figura 3.5: Estructura de la Dirección Link-local

Generalmente son autoconfiguradas y derivadas de la dirección MAC, aun cuando existe la posibilidad de configurarlas en forma manual.

[5] Regional Internet Registry: es una organización que supervisa la asignación y el registro de recursos de direcciones IPv4 e IPv6 dentro de una región particular del mundo.

El proceso para obtener una dirección link-local es (véase la Figura 3.6):

1. Obtener el EUI-64 a partir de la dirección MAC, insertando los bytes 0xFF y 0xFE entre el tercer y cuarto byte.
2. Se invierte el valor del bit "u", el cual está situado en la séptima posición más significativa. Obteniendo así la dirección EUI-64 modificada (identificador de interfaz) [05].
3. Se antepone el prefijo link-local FE80::/64 a la dirección obtenida en el paso previo.

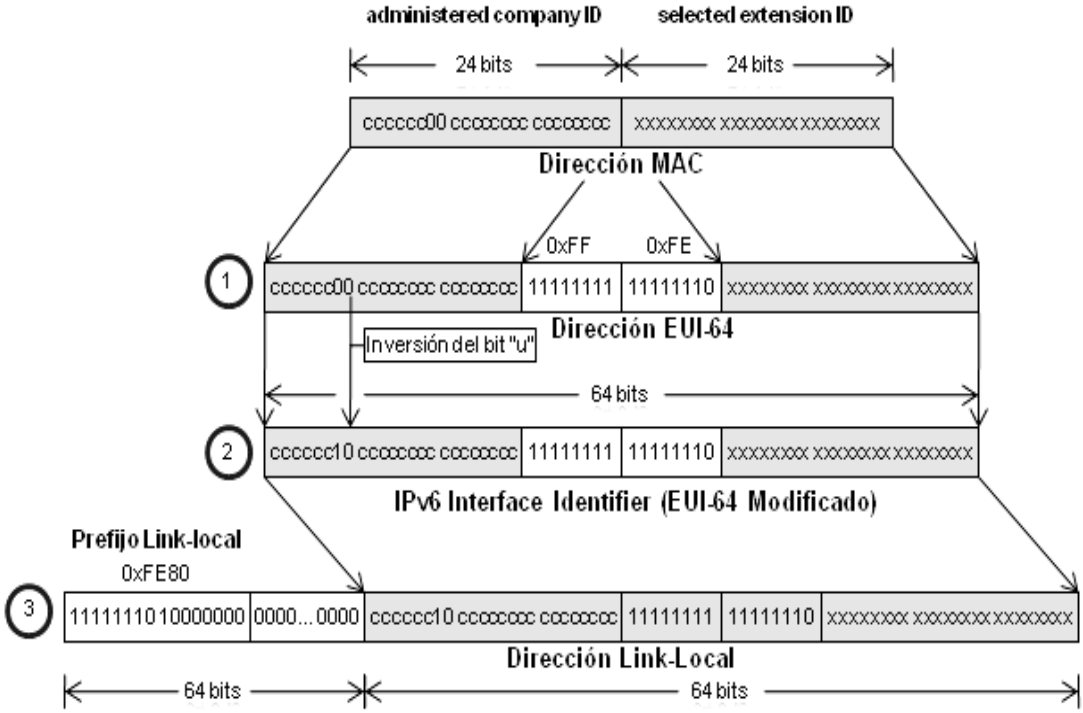


Figura 3.6: Obtención de Dirección Link-local

- **Site-local:** Las direcciones site-local son las equivalentes al espacio de direcciones privadas de IPv4. Estas direcciones no son alcanzables desde otras redes, y los routers no deben retransmitir este tipo de tráfico fuera de la red. El prefijo de una dirección site-local es FEC0::/10, su estructura se muestra en la Figura 3.7 (tomada de [05]):

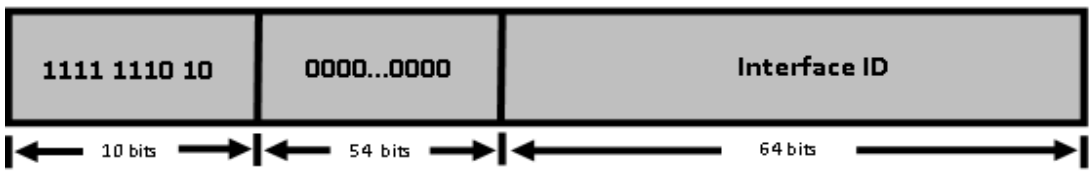


Figura 3.7: Estructura de la Dirección Site-local

Este tipo de direcciones se consideran obsoletas (RFC 3879 [45]). Fueron reemplazadas por las direcciones Unique-local.

- **Unique-local:** Estas direcciones son únicas globalmente pero no deben ser enrutadas hacia Internet. Comúnmente llamadas local IPv6 address (dirección IPv6 local), son especificadas en el RFC 4193 [09]. Su estructura se puede apreciar en la Figura 3.8 (tomada de[09]).

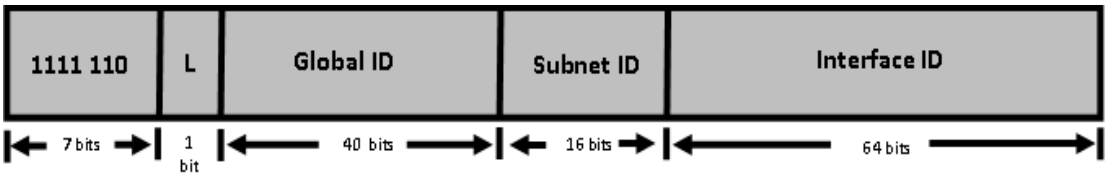


Figura 3.8: Estructura de la Dirección Unique-local

Utilizan el prefijo FC00::/7. El bit "L" con valor 1 indica que el prefijo es asignado localmente, el uso del bit en 0 será definido en el futuro. Los siguientes 40 bits correspondientes al "Global ID" son usados para crear un prefijo único globalmente, el cual es generado de forma aleatoria. Finalmente, los últimos 80 bits correspondientes a los campos "Subnet ID" e "Interface ID" identifican la subred y la interfaz respectivamente.

En consecuencia de que el bit 'L' por ahora siempre posee valor 1, el prefijo toma el valor FD00::/8.

3.3.4 Direcciones Multicast

Una dirección multicast es un identificador para un grupo de nodos. Un nodo IPv6 puede estar a la escucha de múltiples direcciones multicast. Los nodos pueden unirse o dejar un grupo multicast en cualquier momento. Las direcciones multicast no pueden ser utilizadas como direcciones origen ni como un destino intermedio mediante el uso de la cabecera de extensión "Routing header".

Las direcciones multicast poseen el prefijo 0xFF, seguido de tres campos. La Figura 3.9 (tomada de [05]) muestra la estructura de una dirección multicast.

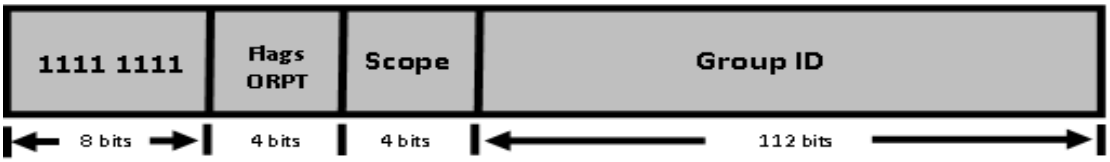


Figura 3.9: Estructura de la Dirección Multicast

El valor de los primeros 8 bits es 1. Los siguientes 4 bits son banderas utilizadas de la siguiente forma: el primer bit está reservado para uso futuro y debe ser cero (0). El segundo bit ("R") indica si la dirección multicast embebe un Rendezvous Point[6]. El tercer bit ("P") en 1 indica que la dirección multicast posee información de prefijo. Por último el bit "T" en 1 indica si la dirección es temporal y en cero (0) indica si es una dirección asignada por la IANA.

El campo "Scope" es usado para determinar el alcance de la dirección multicast. Los posibles valores se encuentran en la Tabla 3.1.

[6]Rendezvous Point: un punto de distribución para un flujo multicast en específico.

Valor	Alcance (Scope)	Descripción
1	Interface-local	Extiende su alcance a la interfaz de un nodo.
2	Link-local	Su alcance es dentro del enlace.
4	Admin.-local	El alcance más corto que debe ser administrativamente configurado.
5	Site-local	Su alcance permanece dentro del sitio.
8	Organization-local	Su alcance son múltiples sitios pertenecientes a una misma organización.
6, 7, 9 A, B, C, D	Unassigned	No han sido asignados.
E	Global	Poseen alcance global.
0, 3, F	Reserved	Están reservados, no se permite su uso.

Tabla 3.1: Valores del Campo "Scope"

- **Direcciones multicast conocidas:** El RFC 2375 [46] define direcciones multicast que han sido asignadas permanentemente, la Tabla 3.2 muestra algunas de estas direcciones.

Dirección	Descripción
FF01::1	Todos los nodos en la interfaz.
FF01::2	Todos los routers en la interfaz.
FF02::1	Todos los nodos en el enlace.
FF02::2	Todos los routers en el enlace.
FF05::2	Todos los routers en el sitio.
FF02::1:2	Todos los agentes DHCPv6 en el enlace.
FF05::1:3	Todos los servidores DHCPv6 en el sitio

Tabla 3.2: Algunas Direcciones Multicast Conocidas

- **Solicited-Node Multicast Address:** Es una dirección multicast que cada nodo debe tener por cada dirección unicast y anycast que posee. Es usada en el protocolo Neighbor Discovery. El RFC 4291 [05] define este tipo de direcciones.

  Las direcciones solicited-node se obtienen tomando los 24 bits menos significativos de la dirección IPv6 y concatenándolos al prefijo FF02::1:FF00:0/104. Un ejemplo se puede apreciar en la Figura 3.10.

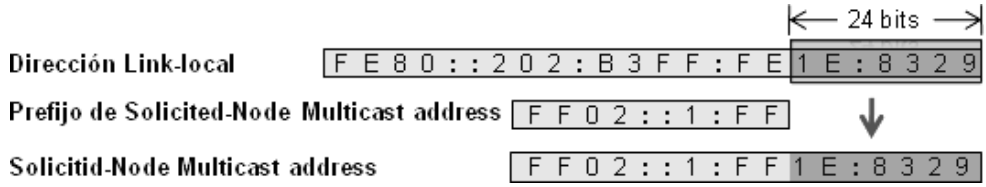


Figura 3.10: Ejemplo de Dirección multicast Solicited-Node

- **Dirección link-layer asociada a un grupo multicast:** Una dirección link-layer se refiere a la dirección de capa de enlace, por ejemplo una dirección MAC. El prefijo utilizado para las direcciones link-layer multicast es 33:33. El resto de la dirección son los últimos 4 bytes de la dirección IPv6 multicast. Por ejemplo, la dirección IPv6 multicast FF02::1:FF1E:8329 tiene asociada la dirección link-layer multicast 33:33:FF:1E:83:29.

3.3.5 Direcciones Anycast

Son un intermedio entre las direcciones unicast y multicast. Usan el espacio de direcciones unicast y los paquetes siempre son enviados a una sola interfaz. Pero hay varias interfaces escuchando con esa dirección, como es el caso de multicast[08].

Las direcciones anycast se encuentran aún en desarrollo. Hasta ahora sólo es soportada la dirección anycast Subnet Router. Esta direcciona a todos los routers de una subred. La dirección se representa con el prefijo de subred seguido de ceros. Por ejemplo, la dirección anycast 2001:db8:1234:5678:: corresponde a los routers de la subred 2001:db8:1234:5678::/64.

4. Mecanismos de Transición

IPv6 e IPv4 coexistirán durante muchos años, y hay una amplia gama de técnicas que hacen posible su convivencia para proporcionar una transición más fácil [04], denominados mecanismos de transición.

Estas técnicas son separadas en tres categorías principales:

- **Dual-stack (Doble Pila):** Permite a IPv4 e IPv6 coexistir en los mismos dispositivos y redes.
- **Túneles (Tunneling):** Permite el transporte de tráfico IPv6 sobre la infraestructura existente de IPv4.
- **Traducción (Translation):** Permite a los nodos IPv6-only comunicarse con nodos IPv4-only [04].

En este capítulo se describen los mecanismos disponibles para cada una de estas categorías y sus implementaciones para el entorno Windows y Linux desarrolladas en la actualidad (2014). El RFC 4213 [10], describe las categorías Dual-Stack y Túneles.

4.1 Direcciones de Transición

Creadas para ayudar al proceso de transición de IPv4 a IPv6 y permitir la coexistencia de ambos tipos de hosts. En [05][06][12][13], se describen las siguientes direcciones:

- **IPv4-Compatible:** Son usadas para establecer una comunicación con IPv6 a través de una infraestructura de IPv4 que utiliza direcciones públicas, tal como Internet. Este tipo de direcciones es muy raramente usado y se considera obsoleto.

  IPv4-compatible no están respaldas por el protocolo IPv6 para Windows Server 2012, Windows Server 2008, Windows 8, Windows 7 y Windows Vista. El protocolo IPv6 para Windows XP y Windows Server 2003 es compatible, pero se encuentra desactivado por defecto [03]. La Figura 4.1 (tomada de [05]) muestra la estructura de este tipo de direcciones.

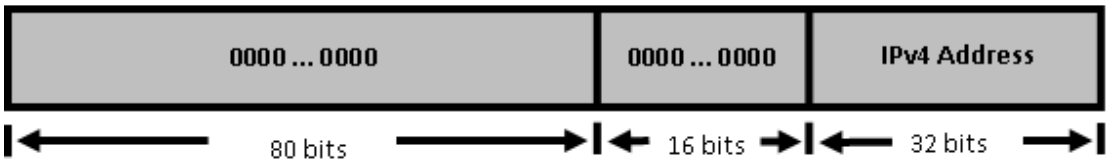


Figura 4.1: Estructura de la Dirección IPv4-Compatible IPv6

- **IPv4-Mapped:** Es un tipo de dirección usado para representar las direcciones de nodos IPv4 como direcciones IPv6. Un nodo IPv6 puede usar esta dirección para enviar un paquete a un nodo IPv4.

  IPv4-mapped es compatible con el protocolo IPv6 para Windows Server 2012, Windows Server 2008, Windows 8, Windows 7, y Windows Vista, mientras que para el protocolo IPv6 para Windows XP y Windows Server 2003 no las admite. La Figura 4.2 (tomada de [05]) muestra la estructura de este tipo de direcciones.

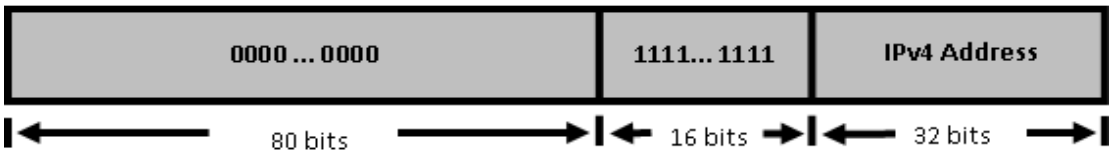


Figura 4.2: Estructura de la Dirección IPv4-Mapped IPv6

- **Direcciones 6to4:** Las direcciones 6to4 fueron diseñadas para permitir a hosts o redes IPv6 comunicarse a través de una infraestructura que sólo soporta IPv4. En la Figura 4.3 (tomada de [06]) se puede observar la estructura de este tipo de direcciones.

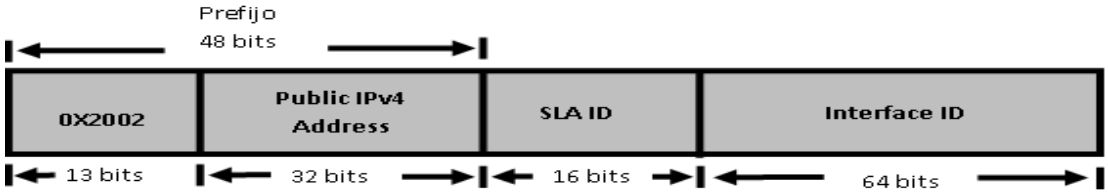


Figura 4.3: Estructura de la Dirección 6to4

El prefijo comienza con 0x2002, seguido de la dirección pública IPv4, formando así un prefijo que se puede abreviar como 2002:DirIPv4::/48 [06]. Posteriormente se encuentra el campo SLA de 16 bits, el cual identifica la subred y por último el campo de 64 bits que indica el identificador de interfaz.

- **Direcciones ISATAP (Intra-Site Automatic Tunnel Addressing Protocol):** Se compone de un prefijo unicast de 64 bits, seguido de un campo que indica el identificador de interfaz de 32 bits. Según el identificador de interfaz que se utilice, las direcciones ISATAP pueden tomar los siguientes formatos.

1. 64bitsPrefUnicast:0:5EFE:DirIPv4, donde DirIPv4 es una dirección IPv4 privada asignada a un nodo.
2. 64bitsPrefUnicast:200:5EFE:DirIPv4, donde DirIPv4 es una dirección pública, por ende, son los identificadores de interfaz administrados localmente.
3. 64bistsPrefUnicast, es cualquier prefijo de 64 bits unicast, incluyendo prefijos Link-local, global y unique.

Las direcciones ISATAP son soportadas por el protocolo IPv6 para Windows XP, Windows Vista, Windows 7, Windows Mobile, Linux y algunas versiones de Cisco IOS. En la Figura 4.4 (tomada de [04]), se puede observar la estructura de este tipo de direcciones.

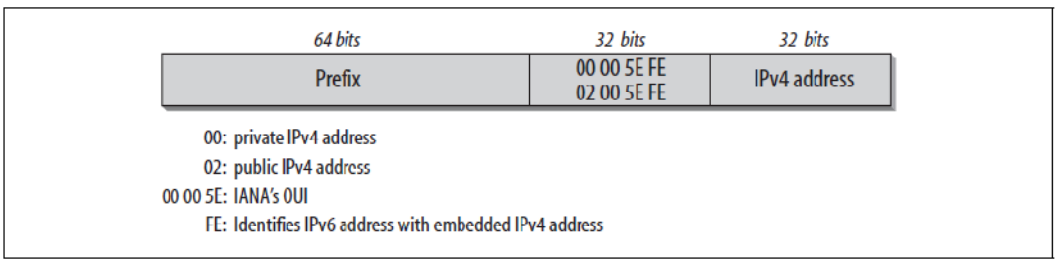


Figura 4.4: Estructura de la Dirección ISATAP

- **Direcciones Teredo:** Se basan en el prefijo de 32 bits de la forma 2001::/32. Se utilizan para crear direcciones IPv6 globales para los nodos IPv6/IPv4 que están conectados al Internet IPv4. El campo ServerIPv4address, es de 32 bits y contiene la dirección IPv4 del servidor Teredo, seguido de un campo de 16 bits (Flags), donde se define la dirección y el tipo de NAT que se utiliza. Posteriormente, se encuentra el campo Port de 16 bits donde se encuentra el puerto UDP asignado al servicio del cliente Teredo. Por último, el campo ClientIPv4address de 32 bits, contiene la dirección IPv4 asignada al cliente.

  Las direcciones Teredo son soportadas por el protocolo IPv6 para Windows. En la Figura 4.5 (tomada de [04]), se puede observar la estructura de este tipo de direcciones.

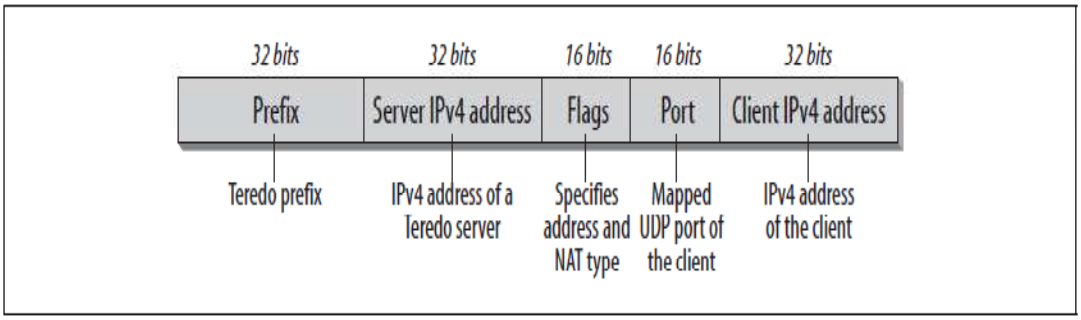


Figura 4.5: Estructura de la Dirección Teredo

4.2 Dual-Stack (Doble Pila o Dual IP Layer)

Un nodo Dual-Stack provee soporte completo para ambas versiones del protocolo de Internet. Este tipo de nodo también es conocido como nodo IPv6/IPv4 [10] y tiene la capacidad de enviar y recibir paquetes de ambos protocolos.

Esta técnica permite que cuando se produce la comunicación IPv6, el nodo se comporte como un nodo IPv6-only y cuando la comunicación se produce con IPv4, se comporte como un nodo IPv4-only.

Cada nodo IPv6/IPv4 debe tener por lo menos una dirección para cada versión del Protocolo de Internet, para ello se utilizan los mecanismos de asignación de direcciones dependiendo del protocolo. En IPv4, configuración estática o Dynamic Host Configuration Protocol (DHCP). En IPv6, configuración estática o autoconfiguración.

4.2.1 Arquitectura Dual-Stack

Una arquitectura Dual-Stack contiene los protocolos IPv4 e IPv6 en la capa de Internet, con una implementación de protocolos de capa de transporte tales como Transmission Control Protocol (TCP) y User Datagram Protocol (UDP). La Figura 4.6 (tomada de [03]) muestra una arquitectura Dual-Stack.

El protocolo TCP/IP en Windows Server 2012, Windows Server 2008, Windows 8, Windows 7, and Windows Vista incluyen tanto IPv4 como IPv6 en una arquitectura Dual-Stack [03].

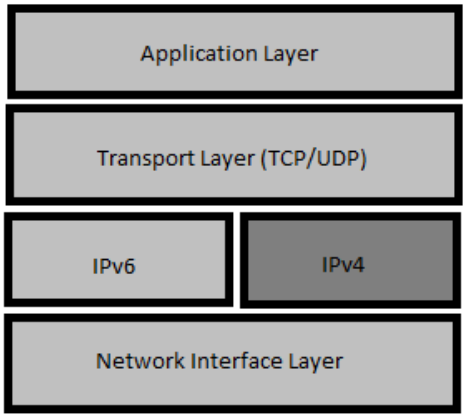


Figura 4.6: Arquitectura Dual-Stack

4.2.2 DS-Lite: Dual Stack Lite

Diseñado para permitir a un proveedor de servicios de Internet (ISP) omitir la asignación de una dirección IPv4 al equipo local de cliente (CPE). En su lugar, asignan únicamente direcciones IPv6 globales (Un entorno Dual Stack normal requiere la asignación de direcciones públicas IPv4 e IPv6)[7].

DS-Lite permite a los ISPs asignar direcciones IPv6 de forma nativa a los nuevos clientes sin dejar de apoyar a los clientes IPv4. Sus principales componentes son B4 (Basic Bridging Broad Band) y AFTR (Address Family Translation Router). Definido en el RFC 6333 [49], en el RFC 7335 [51] y en el RFC 6334 [50] dedicado a la configuración automática.

4.3 Túneles

Es el proceso en donde la información de un protocolo se encapsula dentro del paquete de otro protocolo, permitiendo llevar así los datos originales encima del segundo protocolo para que se produzca la comunicación. Por tanto, es una técnica para el establecimiento de una "relación virtual", entre dos nodos para la transmisión de paquetes de datos como carga útil.

El proceso de túnel involucra tres pasos: encapsulación, desencapsulación y administración del túnel. Se requiere dos extremos del túnel, los cuales son generalmente nodos Dual-Stack IPv4/IPv6, que se ocupan de la encapsulación y la desencapsulación [11] de paquetes.

Un nodo encapsula los paquetes originales recibidos desde otros nodos o desde sí mismo y reenvía el paquete resultante a través del túnel. El otro nodo desencapsula los paquetes recibidos y reenvía los paquetes originales a su destino.

El nodo encapsulador se denomina punto de entrada del túnel, y es la fuente de los paquetes. El nodo desencapsulador se llama punto de salida del túnel, y es el destino de los paquetes como se muestra en la Figura 4.7 (Tomada de [14]).

[7] http://es.wikipedia.org/wiki/Mecanismos_de_transici%C3%B3n_IPv6

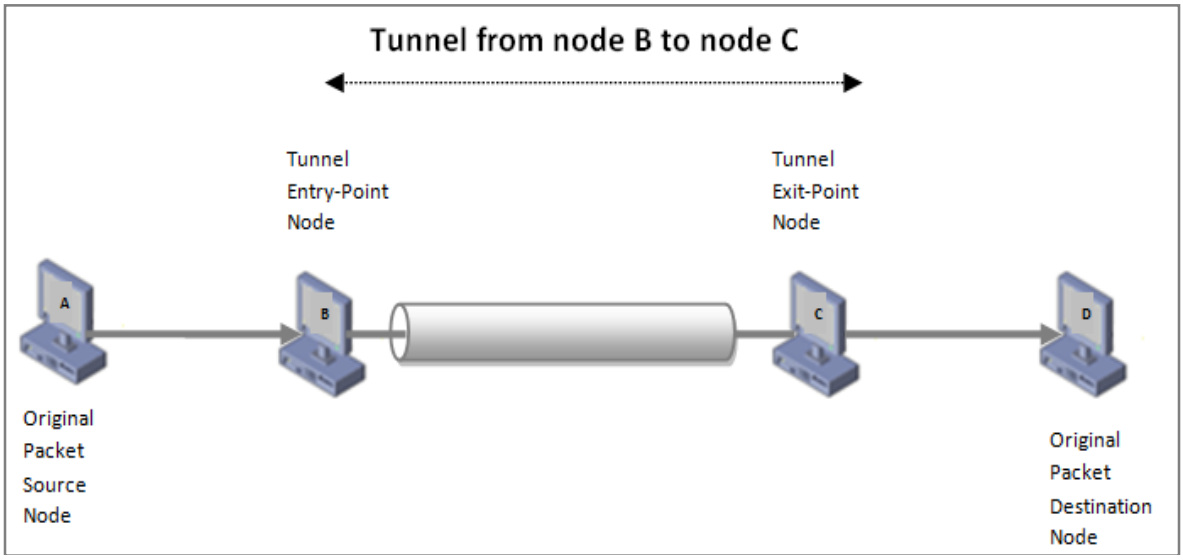


Figura 4.7: Túnel

Están definidos en [06][14]. Se definen dos tipos de túneles:

- **Túneles configurados manualmente de IPv6 sobre IPv4:** Túneles point-to-point configurados manualmente hechos para encapsular paquetes IPv6 dentro del protocolo IPv4 y ser enviados a través de una infraestructura IPv4. Ambos extremos debe soportar nodos Dual Stack IPv4/IPv6 [10].

  El trabajo administrativo para gestionar túneles configurados es mayor que con los túneles automáticos, pero por razones de seguridad, puede ser deseable, ya que ofrece más posibilidades para controlar la ruta de acceso de reenvío de paquetes IPv6 [04].
- **Túnel Automático:** Permiten a los nodos IPv6/IPv4 comunicarse a través de una infraestructura IPv4 sin la necesidad de tener un túnel preconfigurado en el destino. Un túnel automático no necesita configuración manual, los extremos se determinan automáticamente usando direcciones IPv6/IPv4 compatibles [04].

4.3.1 Configuración de Túneles

Los túneles permiten utilizar las infraestructuras IPv4 mientras se implanta totalmente la red IPv6. Consisten en enviar datagramas IPv6 encapsulados en paquetes IPv4. Los extremos finales del túnel son los encargados de realizar el proceso de encapsulamiento y desencapsulamiento de paquetes.

Estos túneles pueden ser configurados de diferentes formas:

- **Router-to-Router:** Routers IPv6/IPv4 interconectados con una infraestructura IPv4 usan túneles para transportar paquetes IPv6. En este caso, el túnel comprende un segmento end-to-end que incluye la ruta completa que siguen los paquetes IPv6.
- **Host-to-Router:** Un nodo IPv6/IPv4 que reside dentro de una infraestructura IPv4 crea un túnel IPv6 sobre IPv4 para alcanzar un router IPv6/IPv4. El túnel comprende el primer segmento de la ruta seguida por los paquetes.

- **Host-to-Host:** Un nodo IPv6/IPv4 que reside dentro de una infraestructura IPv4 crea un túnel IPv6 sobre IPv4 para alcanzar otro nodo IPv6/IPv4 que reside dentro de la misma infraestructura IPv4. El túnel comprende la ruta completa que siguen los paquetes.
- **Router-to-Host:** Routers IPv6/IPv4 crea un túnel IPv6 sobre IPv4 a través de una infraestructura IPv4 para alcanzar un nodo IPv6/IPv4. El túnel comprende el último segmento de la ruta.

4.3.2 ISATAP: Intra-Site Automatic Tunnel Addressing Protocol

Diseñado para proporcionar conectividad IPv6 para nodos Dual-Stack (IPv6/IPv4) sobre una red basada en IPv4. Es un mecanismo de superposición automática que utiliza la red IPv4 como una capa de enlace, y los mecanismos de tunelización básicos especificados en la Sección 4.3.1.

El túnel se realiza automáticamente mediante una interfaz de túnel ISATAP que trata todo nodo IPv4-only como una capa de enlace única, de la misma manera como lo hace Ethernet. En el caso de ISATAP la encapsulación de la capa de enlace es IPv4.

Soporta un sistema automático de abstracción de túnel similar a la red de acceso múltiple de no difusión (Non-broadcast multiple-access network, NBMA), por lo que no requiere que la red IPv4 subyacente soporte multicast.

Define un método para generar direcciones IPv6 link-local a partir de una dirección IPv4, y un mecanismo para realizar el protocolo Neighbor Discovery (ND) de IPv6 a través de IPv4.

Para el intercambio de tráfico dentro del túnel se establece una interfaz de red IPv6 virtual mediante la creación de direcciones ISATAP de tipo unicast, con formato EUI-64, que incluyen un prefijo de 64 bits y un identificador de interfaz de 64 bits (detallada en la Sección 4.1).

Las interfaces ISATAP deben procesar las fallas con los protocolos ARP (Address Resolution Protocol) e ICMPv4, que dan la información necesaria para verificar si un enlace ha fallado.

El tráfico IPv6 basado en ISATAP se tuneliza o encapsula con un encabezado IPv4, también conocido como IPv6-over-IPv4, por lo que, es similar a 6over4, pero sin el requerimiento del empleo de multicast sobre la red. Aunque el mecanismo de tunelización ISATAP es similar a otros mecanismos de transición, está diseñado para el transporte de paquetes dentro de un sitio, no entre sitios.

En marzo de 2008, ISATAP fue publicado como un RFC de tipo Informativo, RFC 5214 [12], quedando obsoleto el anterior de tipo experimental (RFC 4214)[8].

En abril de 2008, ISATAP es implementado en el núcleo del sistema operativo Linux, comenzando con linux-2.6.25, además de tener implementaciones en otros sistemas operativos como: Microsoft Windows XP, Windows Vista, Windows 7, Windows 8, Cisco IOS, 6WIND 6WINDGate y Free BSD/KAME y algunas versiones para sistemas Android.

[8] http://www.isatap.org

4.3.3 6to4: Connection of IPv6 Domains via IPv4 Clouds

6to4 permite que sitios aislados IPv6 o host conectados a una red de área extensa, que no tienen IPv6 nativo puedan comunicarse a través de una red IPv4 con un mínimo de configuración manual, sin la cooperación de los proveedores de Internet (ISP).

Trata a toda la red IPv4 como un solo enlace, de la misma forma que Ethernet. Utiliza la configuración de tunelización de router-to-router, host-to-router y router-to-host (detallado en la Sección 4.3.1).

6to4 mantiene la función de encapsulamiento de IPv6 en IPv4 y cuenta con servidores especialmente diseñados que actúan como Relay para permitir la comunicación.

Los sitios IPv6 o hosts conectados a 6to4, no requieren la utilización de IPv4-compatible para las direcciones IPv6 o en los túneles automáticos. De esta forma gana independencia y puede pasar por muchas subredes IPv4.

En 6to4, solo una pequeña cantidad de routers requieren modificación. En general, se configura un router de frontera, proveyendo conectividad unicast a toda una red, o en un host en particular; En ambos casos necesita una dirección IP pública, ya que, la clave del sistema consiste en la asignación de direcciones IPv6 que contienen embebidas la dirección IPv4 pública del router. Estas direcciones tienen todas el prefijo 2002::/16 (formato detallado en la Sección 4.1). De esta manera, cuando es necesario convertir un paquete IPv6 para que atraviese la red IPv4, el router sabe la dirección a la que debe estar dirigido el paquete IPv4 generado.

La Figura 4.8 muestra los componentes de una arquitectura 6to4 y su colocación según la red IPv4 y la red IPv6. Dichos componentes se detallan a continuación:

- **6to4 host:** Es un host IPv6 que pasa a tener al menos una dirección 6to4. En todos los demás aspectos es un host IPv6 estándar [06], por lo cual, no requiere ningún tipo de apoyo de configuración manual adicional y se puede crear direcciones 6to4 utilizando mecanismos de autoconfiguración de direcciones. Eso es, un host 6to4 no tiene una interfaz de túnel 6to4 y no realiza túnel 6to4 [03].
- **6to4 router (router de frontera 6to4):** Un router IPv6/IPv4 que soporta una interfaz de túnel 6to4 para reenviar el tráfico 6to4 entre los hosts 6to4, los 6to4 relays y otros routers 6to4 [03]. Es normalmente el router de frontera entre un sitio IPv6 y una red de área amplia IPv4 [06]. Los routers 6to4 necesitan configuración manual.
- **6to4 relay:** Un 6to4 router configurado para soportar tránsito de enrutamiento entre las direcciones 6to4 y direcciones IPv6 nativas [06].

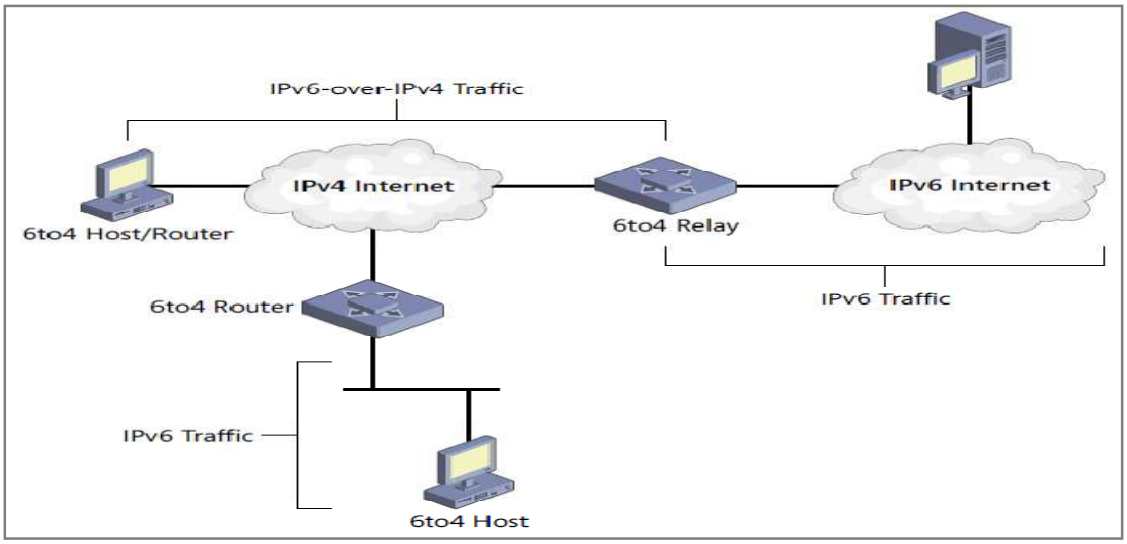


Figura 4.8: Componentes de una Infraestructura 6to4

Además, se utilizan los siguientes términos:

- **Interfaz 6to4:** Es el punto donde se produce la encapsulación 6to4 de paquetes IPv6 dentro de paquetes IPv4. Puede ser tratada como cualquier otra interfaz o como si fuera el punto final del túnel.
- **Sitio 6to4:** Un sitio que direcciones IPv6 utilizando 6to4 internamente, por lo tanto contiene al menos un 6to4 host y un 6to4 router.
- **6to4 Dominio de Enrutamiento Externo:** Es un dominio de enrutamiento de interconexión entre el conjunto de 6to4 routers y 6to4 relays.

El RFC 3068 [17] define una dirección anycast para routers de reenvío, con el fin de simplificar la configuración de routers 6to4 que necesitan una ruta predeterminada para la retransmisión de paquetes 6to4 al Internet IPv6 nativo. El IANA asigna como prefijo IPv4 6to4 Relay anycast la dirección 192.88.99.0/24 (Información complementada en el RFC 6343 [16]).

6to4 fue estandarizado en febrero de 2001, mediante la publicación del RFC 3056 [06]. 6to4 es soportada en diferentes versiones de Windows como: Windows Server 2008, Windows 7, y Windows Vista.

4.3.4 Teredo

Teredo es una tecnología que permite túneles IPv6 automáticos entre nodos que se encuentran en la red IPv4, incluso cuando estos nodos están detrás de uno o más IPv4 NATs [03]. El tráfico IPv6 de los clientes Teredo pueden fluir a través de NAT, ya que se envía como mensajes UDP IPv4, es decir, encapsula los datagramas IPv6 dentro de datagramas UDP (User Datagram Protocol) IPv4. Si el servidor NAT es compatible con la traducción de puertos UDP, soporta Teredo (con la excepción de NAT simétricos).

La Figura 4.9 (tomada de [03]) muestra el formato de un paquete Teredo. Consiste en una cabecera IPv4 de 20 bytes de longitud, que contiene las direcciones de origen y destino de

IPv4. Seguido por una cabecera UDP de 8 bytes, donde se encuentran los puertos de origen y destino UDP para el tráfico Teredo y que puede ser traducido por un NAT.

Luego se encuentra la cabecera IPv6, que contiene las direcciones IPv6 de origen y destino, donde debe existir al menos una dirección Teredo. Seguido de un campo de n bytes con las cabeceras de extensión y el protocolo de capa superior. Esta carga también puede ser nula (Teredo Bubble).

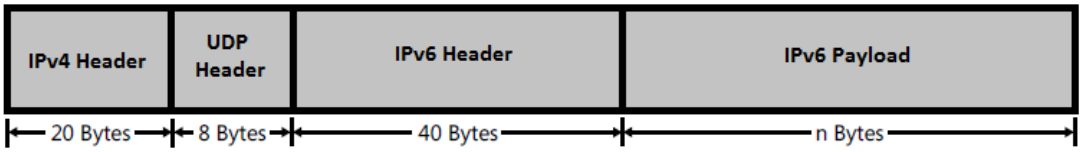


Figura 4.9: Formato del Paquete Teredo

La arquitectura Teredo está formada por los siguientes componentes:

- **Cliente Teredo:** Un nodo que tiene algún tipo de acceso al Internet IPv4 y quiere ganar el acceso a la red IPv6 [13]. Tiene una interfaz Teredo para la comunicación con otros clientes Teredos o con nodos IPv6 (a través de Teredo Relay). Cuando un cliente Teredo quiere enviar un paquete a un nodo IPv6, este envía el paquete encapsulado en UDP/IPv4l al servidor Teredo, quien lo dirige a la red IPv6 y envía la respuesta al cliente [04].
- **Servidor Teredo:** Un nodo que tiene acceso al Internet IPv4 a través de una dirección pública enrutable a nivel mundial, que se utiliza como ayudante para proporcionar conectividad a los clientes Teredo [13].

  Se conecta tanto con IPv4 como IPv6. Soporta una interfaz de túneles sobre la cual son recibidos los paquetes. La función principal de los servidores Teredo es asistir en la configuración de direcciones de los clientes Teredo, así como facilitar la comunicación inicial entre varios clientes Teredo o entre clientes Teredo y hosts IPv6 [03].
- **Teredo Relay:** Un enrutador IPv6 que puede recibir el tráfico destinado a los clientes Teredo y lo remitirá a utilizar el servicio Teredo [13]. En algunos casos interactúa con el servidor Teredo para facilitar la comunicación inicial entre los clientes Teredo y los hosts IPv6 [03]. Un Teredo Relay escucha el tráfico Teredo en el puerto UDP 3544.
- **Teredo Host-Specific Relay:** La comunicación entre clientes Teredo y hosts IPv6 que son configurados con una dirección global debe pasar necesariamente a través de un Teredo Relay. Un Teredo host-Specific Relay es un nodo IPv6/IPv4 que tiene una interfaz que proporciona conectividad tanto a nodos IPv4 como a nodos IPv6 [03].

La Figura 4.10 (tomada de [03]) muestra un escenario con los componentes de una infraestructura Teredo.

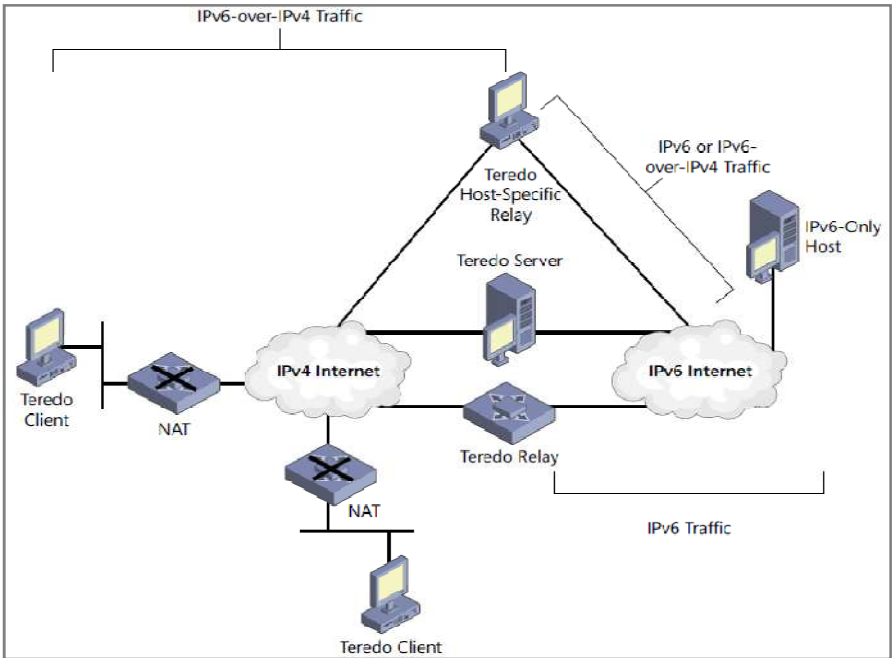


Figura 4.10: Componentes de una Infraestructura Teredo

Además, se utilizan los siguientes términos:

- **Teredo IPv6 Service Prefix:** Un prefijo de direccionamiento IPv6 que se utiliza para construir la dirección IPv6 de los clientes Teredo [13]. El prefijo global Teredo asignado por el IANA es 2001:0000 :: / 32 (información detallada en la Sección 4.1).
- **Teredo Bubble:** Es un paquete IPv6 mínimo, hecho de una cabecera IPv6 y un payload nulo. El tipo de payload típico se establece en 59, no Next-Header como lo establece el RFC 2460 [02]. Los clientes Teredo y Teredo Relay pueden enviar Bubble para crear asignaciones en un NAT. La Figura 4.11 (tomada de [03]) muestra el formato de un paquete Teredo Bubble que consiste en una IPv6 cabecera sin carga útil IPv6.

Figura 4.11: Formato de Paquete Teredo Bubble

- **Teredo intervalo de actualización:** Es el intervalo en el que se espera que una dirección Teredo IPv6 permanezca válida en ausencia de un “refresh”. Para un cliente situado detrás de un NAT, el intervalo depende de los parámetros de configuración de la NAT local o la combinación de NAT en la ruta de acceso al servidor Teredo. De forma predeterminada, los clientes asumen un valor de 30 segundos [13].
- **Teredo IPv4 Discovery Address:** Una dirección multicast utilizada para el descubrimiento de otros clientes Teredo en la red IPv4 subyacente. La dirección multicast es 224.0.0.253.

En cuanto al diseño y objetivo de Teredo, la solución propuesta transporta paquetes IPv6 como carga útil de paquetes UDP/IPv4. Esto se basa en la observación de que TCP y UDP son los únicos garantizados para cruzar la mayoría de los dispositivos NAT.

Sin embargo, UDP fue seleccionado sobre TCP debido a que la transmisión de paquetes a través de TCP sería posible pero tendría poca calidad de servicio, en cambio en el encapsulado UDP se presenta en menor medida.

Teredo está diseñado para permitir enérgicamente el tráfico IPv6 a través de NAT pero el precio de la robustez es una cantidad considerable de gastos generales, debido a la encapsulación UDP y la transmisión de Teredo bubble [13].

Fue desarrollado por Chistian Huitema en Microsoft y estandarizado por el IETF mediante la publicación del RFC 4380 [13]. Es soportado por diferentes versiones de Microsoft Windows como: Windows Server 2012, Windows 8, Windows 7, Windows Server 2008, Windows Vista, Windows XP con Service Pack 2, Windows Server 2003 con Service Pack 1, además existe una implementación denominada Miredo, distribuida bajo los términos GNU (General Public License), ejecutable en GNU/Linux, Free BSD, NetBSD y MAC OS X. Miredo fue originalmente desarrollado y es activamente mantenido por Rémi Denis-Courmont[9]. Incluye todos los componentes de la arquitectura Teredo antes mencionados.

4.3.5 TB: Tunnel Broker

Se pueden ver como proveedores de túneles virtuales que proporcionan conectividad IPv6 sobre una red IPv4.

Es un enfoque basado en la provisión de servidores, llamados túneles, dedicados a gestionar automáticamente las peticiones de túneles procedentes de los usuarios finales o redes [15]. La principal diferencia entre Tunnel Broker y 6to4 es que sirven a segmentos diferentes de la comunidad IPv6. Tunnel Broker encaja muy bien para pequeños sitios aislados de IPv6, especialmente los hosts IPv6 aislados en la red IPv4.

La Figura 4.12 (tomada de [15]), muestra el funcionamiento de un Tunnel Broker. Consiste básicamente en un nodo Dual-Stack (usuario IPv6/IPv4), un Tunnel Broker y al menos un servidor de túnel, conectados entre sí, para lograr la conectividad IPv6 sobre IPv4. Un usuario Dual-Stack es un host o router conectado a la red IPv4. Para establecer una conexión IPv6 se debe registrar en el Tunnel Broker, que es el encargado de gestionar el establecimiento, mantenimiento y la eliminación del túnel. Debe autentificarse con los procedimientos estándares (por ejemplo, RADIUS), para que el uso no autorizado de los servicios del túnel se pueda evitar. Una vez que esté autorizado proporciona su dirección IPv4 y un nombre para el registro de su dirección IPv6 en el DNS, y una indicación de si se trata de un router o un host. Si es un router, debe enviar al Tunner Broker el número de direcciones IPv6 que desea para que le sea asignado el prefijo adecuado.

Por ello, tiene que ser accesible con una dirección IPv4 pública (si se utilizan direcciones privadas, se debe utilizar otro tipo de mecanismo de transición), aunque puede tener una

[9] http://www.remlab.net/miredo

dirección IPv6. El tráfico de información entre Tunnel Broker y el servidor de túnel soporta ambos tipos de direcciones. El Tunnel Broker puede compartir el tráfico de datos con varios servidores de túneles y registrar direcciones DNS si es configurado para hacerlo. Además, envía la información de configuración a un servidor de túnel cuando quiere establecer, cambiar o eliminar un túnel. Un servidor de Túnel es un router Dual-Stack conectado a la red global.

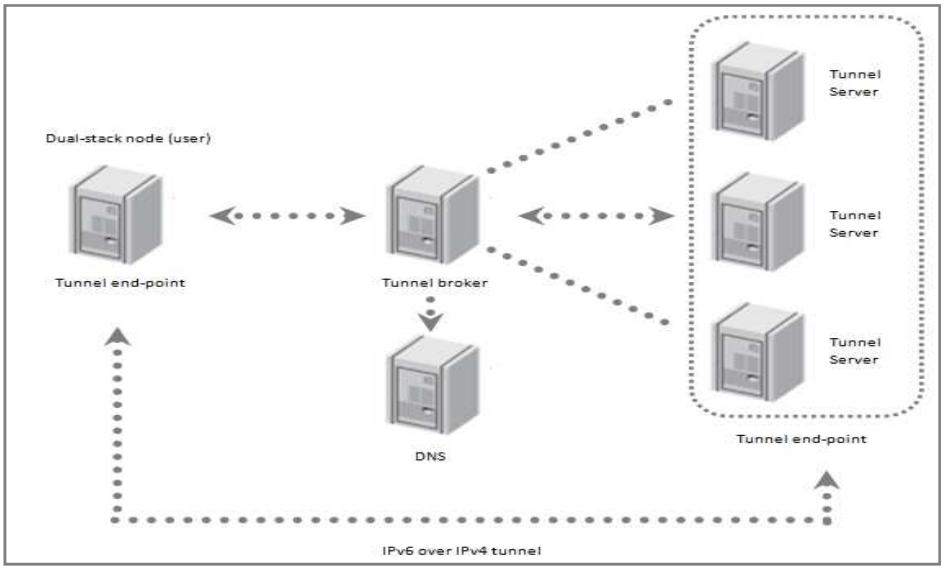


Figura 4.12: Funcionamiento de un Tunnel Broker

Un Tunnel Broker cumple con las siguientes funciones (especificadas en [15]):

- Elección de un servidor de túnel como punto de salida. Si existe más de una opción, la elección se realizará en base a los criterios de carga pre-configurados.
- Elección de un prefijo de cliente, de cualquier longitud. Los más comunes son: /48 para los prefijos de sitio, /64 para los prefijos de subred y /128 para host.
- Definición del curso de vida para el túnel.
- Registro de direcciones IPv6 globales asignadas en el DNS, si ha sido configurado para hacerlo.
- Configuración del servidor de túnel.
- Envío de la información resultante al usuario. Incluye los parámetros del túnel y los nombres DNS.

El enfoque Tunnel Broker fue publicado en enero de 2001, en el RFC 3053 [15] donde solo se provee información, y no se especifica un estándar.

Existe una serie de ISP que ofrecen servicios de Tunnel Broker, donde los usuarios pueden registrase y tener acceso a los servicios de configuración de este tipo de túnel[10].

[10] http://www.ipv6tf.org

4.3.6 Tecnología 6in4

Es un mecanismo de transición basado en el RFC 4213 [10] que permite conectividad IPv6 sobre IPv4, conocido también como Proto 41.

Consiste en la encapsulación de paquetes IPv6 dentro de paquetes IPv4, para ser transmitidos a través de túneles configurados manualmente.

Un cliente del túnel (host o router) con una dirección IPv4 privada conectado a una red IPv4-only puede utilizar un Tunnel Broker o un router IPv6 para crear el túnel 6in4.

El encapsulado 6in4 consiste en la agregación de una cabecera IPv4 (Header IPv4) de 20 bytes al paquete IPv6 a ser transmitido. En el campo Protocol de la cabecera IPv4 se asigna el número 41. Este número se designa específicamente para el encapsulado IPv6. Además de añadir un encabezado IPv4, el nodo encapsulador también debe determinar cuándo fragmentar un paquete 6in4 y cuando enviar un mensaje ICMPv6 de error por paquete demasiado grande. El MTU del túnel puede ser tanto estático como dinámico (aunque, 6in4 se caracteriza por la configuración estática). Un nodo que utiliza un MTU de túnel estático tiene un MTU fijo. Por defecto, la MTU debe estar entre 1280 y 1480 bytes (ambos inclusive), pero su valor por defecto es 1280 bytes. Si el valor por defecto no es 1280 bytes, la aplicación debe ser configurada para cambiar el valor del MTU [10].

La sobrecarga del paquete 6in4 viene dado por el tamaño de la cabecera IPv4, por ejemplo, el MTU de Ethernet es de 1500 bytes, por lo cual, solo se pueden enviar paquetes IPv6 con una longitud máxima de 1480 bytes (sin fragmentación). Los extremos del túnel deben soportar la fragmentación de paquetes para garantizar la interoperabilidad correcta entre cualquier MTU fijo que se encuentre entre 1280 y 1480 bytes.

La determinación del MTU en forma dinámica es opcional Si el nodo usa MTU dinámico, la fragmentación en el interior del túnel se puede reducir a un mínimo por tener el seguimiento del MTU a través del camino del túnel, usando el IPv4 Path MTU Discobery Protocol (RFC 1191) y la grabación de la trayectoria resultante[10]. Por lo tanto, se puede ver el MTU dinámico como un MTU igual al tamaño de la ruta de los extremos, menos el tamaño de la cabecera de túnel 6in4.

Además, cuando alguno de los extremos del túnel se encuentra detrás de un NAT, es posible en algunos casos utilizar las funcionalidades NAT del router, para retransmitir todos los paquetes 6in4.

La Figura 4.13 muestra encapsulado de un paquete IPv6 que va a ser transmitido mediante el método 6in4.

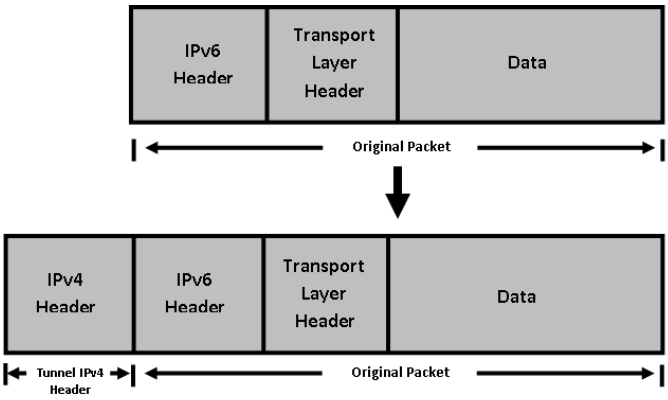


Figura 4.13: Proceso de Encapsulamiento en 6in4

4.3.7 Tecnología 4in6

Es un mecanismo de transición que permite conectividad IPv4 sobre IPv6. Consiste en la encapsulación de paquetes IPv4 dentro de paquetes IPv6, para ser transmitidos a través de túneles configurados como se define en el RFC 2473 [14].

Un túnel 4in6 es una "relación virtual" entre dos nodos IPv4 para la transmisión de paquetes de datos como caga útil de paquetes IPv6. El protocolo IPv6 funciona como un protocolo de capa de enlace.

El encapsulado 4in6 consiste en la agregación de una cabecera IPv6 (Header IPv6) de 40 bytes al paquete IPv4 a ser transmitido, opcionalmente, un conjunto de cabeceras de extensión IPv6 (llamadas cabeceras de túnel IPv6). Las cabeceras de extensión del túnel deben aparecer en el orden recomendado por las especificaciones que definen las cabeceras de extensión de IPv6 [14].

Un escenario típico de 6in4 es el caso en el que un nodo intermedio ejerce control explícito del enrutamiento de paquetes IPv6, especificando caminos de reenvío particulares para determinados paquetes seleccionados. Este control es logrado anteponiendo cabeceras IPv6 a cada uno de los paquetes originales. Estas cabeceras antepone identifican los caminos de reenvío [14].

Además, pueden existir escenarios donde se presente una encapsulación anidada. La encapsulación anidada IPv6 tiene lugar cuando un salto del túnel IPv6 es un túnel. El túnel que contiene un túnel se denomina túnel exterior. El túnel contenido en el túnel exterior se llama un túnel interior. El túnel interior y sus túneles exteriores están anidados [14].

Un túnel exterior recibe el paquete encapsulados por el túnel interior y los trata como paquetes originales para realizar una nueva encapsulación. El paquete resultante son paquetes de túnel 4in6 para el túnel interior y paquete de túnel anidado para el túnel exterior.

Un paquete de túnel 4in6 está limitado al tamaño máximo de los paquetes de IPv6. El encapsulado 4in6 tiene como sobrecarga la longitud de las cabeceras IPv6. El número de

cabeceras de túnel y el número de encapsulaciones anidadas está limitado por el tamaño máximo del paquete. El MTU del túnel se establece dinámicamente (MTU de la ruta de los extremos del túnel, menos el tamaño de la cabecera 4in6).

El encapsulado de paquetes 4in6 aumenta el tamaño del paquete resultante, en consecuencia, el paquete 4in6 resultante puede requerir fragmentación. Los extremos del túnel deben ser compatibles con la fragmentación de paquetes IPv6. La Figura 4.14 muestra el encapsulado de un paquete IPv6 que va a ser transmitido mediante el método 4in6.

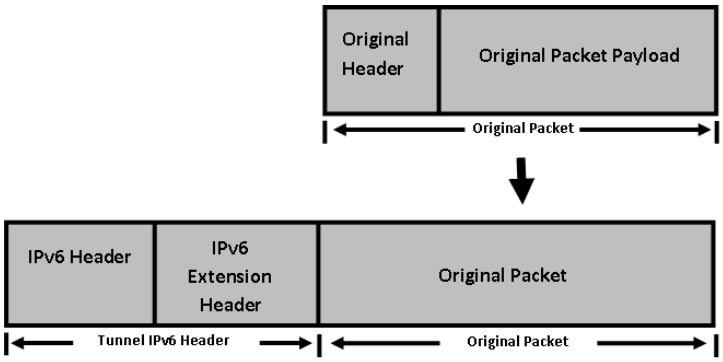


Figura 4.14: Proceso de Encapsulamiento

4.3.8 Tecnología 6over4

6over4 es un mecanismo diseñado para permitir el tráfico IPv6 entre nodos Dual-Stack sobre una infraestructura IPv4-only. Proporciona conectividad unicast y multicast a través de una red IPv4 con soporte para direcciones multicast, empleando la red IPv4 como capa de enlace virtual.

Consiste en la encapsulación de paquetes IPv6 dentro de paquetes IPv4, para ser transmitidos a través de túneles configurados manual o automáticamente, para ello, los extremos del túnel necesitan una dirección IP pública. El tamaño del MTU para los paquetes IPv4 es de 1480 bytes. Si la configuración es automática, se usa stateless.

Al igual que 6in4, el encapsulado 6over4, consiste en la agregación de una cabecera IPv4 (Header IPv4) de 20 bytes al paquete IPv6 a ser transmitido. En el campo Protocol de la cabecera IPv4 se asigna el número 41. Este número se designa específicamente para el encapsulado IPv6.

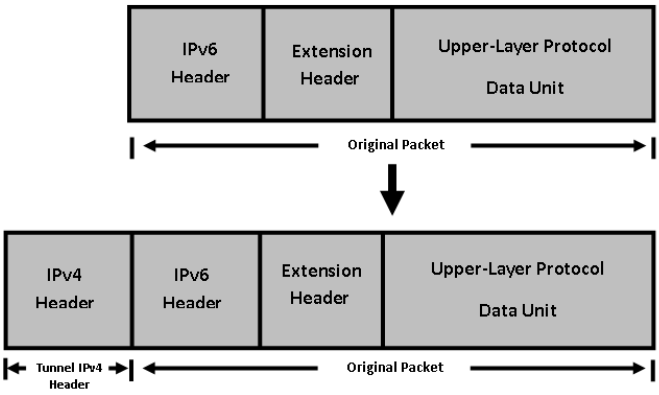


Figura 4.15: Túnel 6over4

6over4, es publicado como estándar, en marzo de 1999, mediante el RFC 2530 [39], donde se especifica el formato del paquete para la transmisión 6over4 y la generación de la dirección IPv6 link-local (entre otras cosas). La Figura 4.15 (tomada de [03]) muestra un túnel 6over4.

4.3.9 Tecnología 6rd

6rd es un mecanismo derivado de 6to4, diseñado para facilitar el rápido despliegue de IPv6 a través de infraestructuras IPv4, mediante proveedores de servicios de Internet (ISP). La diferencia fundamental entre 6to4 y 6rd radica en que el tráfico IPv6 transmitido a través de la red IPv4 usando el enfoque 6rd, opera enteramente dentro del bloque IPv6 unicast del ISP del usuario final, solucionando los problemas inherentes del diseño original de la arquitectura.

En 6rd, el ISP implementa un prefijo IPv6 propio, en lugar del prefijo especial (2001::/16) de 6to4, que garantiza a todos los nodos ser alcanzables desde todas las direcciones IPv6 nativas. La calidad de servicio es responsabilidad directa del ISP.

6rd consiste básicamente en un router 6rd Customer Edge (CE) y uno o varios 6rd Border Relays (BRs) conectados a un ISP, transmite paquetes IPv6 dentro de túneles IPv4 automáticos que siguen las mismas rutas optimizadas entre nodos como cualquier otro paquete IPv4, además utiliza traducciones asíncronas entre direcciones IPv4 e IPv6.

La Figura 4.16 muestra el formato de una IPv6 global unicast (más información en la Sección 3.3.3) con un prefijo 6rd y una dirección CE IPv4 embebidas.

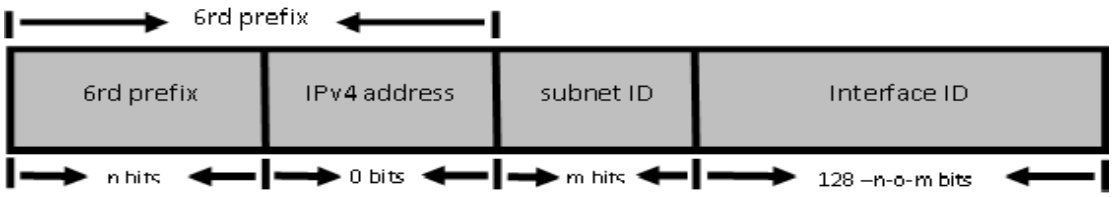


Figura 4.16: Formato del Paquete 6rd

En enero de 2010, fue publicado el RFC 5569 [21] desarrollado por Rémi Despres y posteriormente estandarizado en agosto del mismo año, mediante la publicación del RFC 5969 [22].

4.3.10 TSP - Tunnel Setup Protocol

Consiste básicamente en la implementación de un Tunnel Broker con TSP para proveer a un nodo conectividad IPv6 sobre IPv4, inclusive si se encuentra detrás de un NAT IPv4.

TSP es un protocolo de señalización flexible y extensible, usado para establecer los parámetros de configuración de los puntos finales de un túnel IPv6 y establecer un Tunnel Broker. Es similar a Tunnel Broker, pero no está basado en una interfaz web sino en una arquitectura cliente-servidor.

Un nodo implementado con TSP puede conectarse tanto a redes IPv4 e IPv6 si se trata de redes IPv4-only, IPv4 detrás de un NAT, o IPv6-only. TSP negocia parámetros de configuración entre los dos extremos del túnel, siendo un extremo un pequeño código de cliente y otro el servidor que va a configurar el túnel. Los parámetros que siempre se negocian se muestran en la Tabla 4.1.

Parámetros	Descripción
Autentificación de Usuarios	La autentificación (incluyendo la anónima) se hace a través del mecanismo Simple Authentication and Security Layer (SASL, RFC 4422)
Encapsulación del Túnel	Negocia el tipo de túnel. Este puede ser IPv6 sobre IPv4 (RFC 4213), IPv4 sobre IPv6 (RFC 2473) o IPv6 sobre UDP-IPv4 para NAT transversal.
Dirección IP	Negociar la dirección IP que tendrán los puntos finales del túnel.
Registro DNS	Negociar el registro DNS de la dirección IP de los puntos finales del túnel (AAAA).
Otros	Tunel Keep-alive, asignación de un prefijo IPv6 cuando el cliente es un router, delegación DNS (árbol inverso) en base al prefijo asignado, y los protocolos de enrutamiento.

Tabla 4.1: Parámetros de Configuración de un Tunnel Broker con TSP

El cliente del túnel puede especificar el tipo de encapsulación o se puede determinar durante el intercambio de señales TSP en el túnel. El establecimiento del tipo de encapsulado en el túnel se utiliza para detectar la presencia en la ruta de un NAT, de ser positivo, se selecciona IPv6 sobre UDP/IPv4.

Utiliza XML básico para el envío de mensajes sobre TCP o UDP. Los túneles establecidos por TSP son túneles estáticos, requieren que la dirección IP y el prefijo sean estables para poder ser desarrollados y utilizados. Las especificaciones fueron publicadas en febrero de 2010 mediante el RFC 5572 [24].

La Figura 4.17 muestra la arquitectura de un Tunnel Broker con TSP, donde intervienen los siguientes componentes: el cliente TSP, el punto final del cliente TSP, el servidor TSP y el punto final de servidor TSP. El intercambio de parámetros se realiza entre los puntos finales o extremos del túnel que se desea establecer.

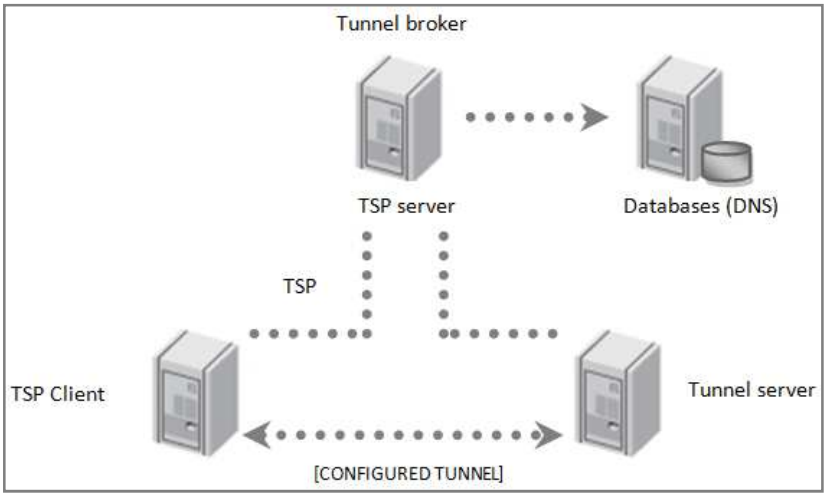


Figura 4.17: Arquitectura de Tunnel Broker con TSP

4.3.11 Softwires

Mecanismo que permite la utilización de otros mecanismos de transición existentes, para proveer conectividad IPv6 en redes IPv4-only.

Se basa en la utilización de Layer Two Tunneling Protocol Version 2 (L2TPv2). Consiste en la encapsulación de paquetes IPv6 en IPv4, IPv6 en IPv6, IPv4 en IPv6 y IPv4 en IPv4, aunque el nodo se encuentre detrás de un NAT. Además permite la autentificación de usuarios para la creación de túneles mediante la interacción con infraestructura AAA.

Softwires utiliza Point-to-Point Protocol (PPP, RFC 1661) para transportar paquetes (opcionalmente se puede encapsular los paquetes PPP en UDP en caso de que haya que atravesar un NAT). Es un enfoque basado en la arquitectura cliente-servidor, compuesto básicamente de un iniciador softwires (cliente) y un concentrador softwires (servidor).

Un iniciador softwires (SI) es el que inicia la creación del túnel software, encargado de solicitar los servicios de túnel. Un concentrador softwires (CS) es el nodo que termina el túnel sofwires y provee los servicios a la red que han sido solicitados. No hay protocolo de enrutamiento dinámico entre SI y CS, se utiliza la ruta por defecto.

El SI y CS deben seguir las operaciones descritas en L2TPv2 (RFC 2661) para realizar el establecimiento y/o desmontaje de un escenario Softwires. L2TPv2 soporta los siguientes encapsulados:

1. IPv6/PPP/L2TPv2/UDP/IPv4
2. IPv4/PPP/L2TPv2/UDP/IPv6

3. IPv4/PPP/L2TPv2/UDP/IPv4
4. IPv6/PPP/L2TPv2/UDP/IPv6

UDP bypass no es soportado por L2TPv2 puesto que no soporta “autodetect” de NAT/NAPT.

L2TPv2 es un protocolo ampliamente desplegado en los servicios de banda ancha, y su escalabilidad ha sido probada en múltiples despliegues a gran escala de redes virtuales privadas IPv4 [23]. En junio de 2009, es estandarizado mediante la publicación del RFC 5571 [23]. Se puede utilizar el modelo Softwires, con los mismos componentes y funcionamiento pero siguiendo los procedimientos y particularidades de L2TPv3 (RFC 3991 [47]).

4.3.12 IP-HTTPS

Permite la conectividad IPv6 sobre una red IPv4-only. Utiliza HyperText Transfer Protocol (HTTP) a través de Secure Sockets Layer (SSL), también conocido como Transport Layer Security (TSL) para lograr el tráfico IPv6 sobre IPv4.

IP-HTTPS encapsula los paquetes IPv6 sobre una sesión HTTPS o HTTP permitiendo la transferencia de paquetes IPv6 sobre IPv4 o IPv6. La estructura de un paquete IPv6 encapsulado sobre IP-HTTPS involucra los siguientes protocolos:

- **IPv6:** Para el envío y recepción de paquetes IPv6.
- **HTTP:** Actúa como portador de los paquetes IPv6. Puede ser protegido con HTTPS.
- **TCP:** Orienta a HTTP en la transmisión de datos fiables.
- **IPv4:** Permite la transmisión de tráfico IPv4 e IPv6 encapsulado en IPv4.

La Figura 4.18 (tomada de [03]) muestra el formato de un paquete IPv6 encapsulado en IP-HTTPS.

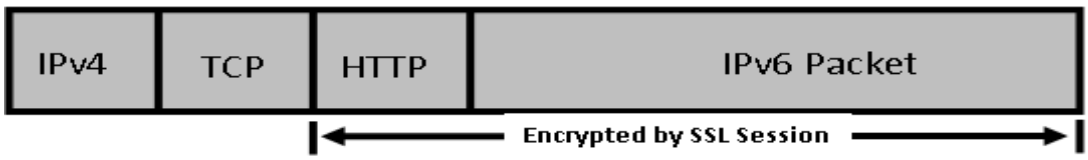


Figura 4.18: Tráfico IP-HTTPS

IP-HTTPS está basado en la arquitectura cliente-servidor, por tanto, está compuesto básicamente por un servidor IP-HTTPS y un cliente IP-HTTPS, como se muestra en la Figura 4.19 (tomada de [03]). El servidor IP-HTTPS es similar a un servidor de una red privada virtual (VPN) tradicional en el Internet IPv4. El servidor IP-HTTPS se encuentra en el borde de la red y escucha las solicitudes de IP-HTTPS entrantes [03].

Un cliente IP-HTTPS, también es similar a un cliente tradicional de VPN en el Internet IPv4 o en una red que tiene acceso al internet basado en proxy. El cliente IP-HTTPS es el encargado de iniciar las conexiones básicas con un servidor IP-HTTPS configurado.

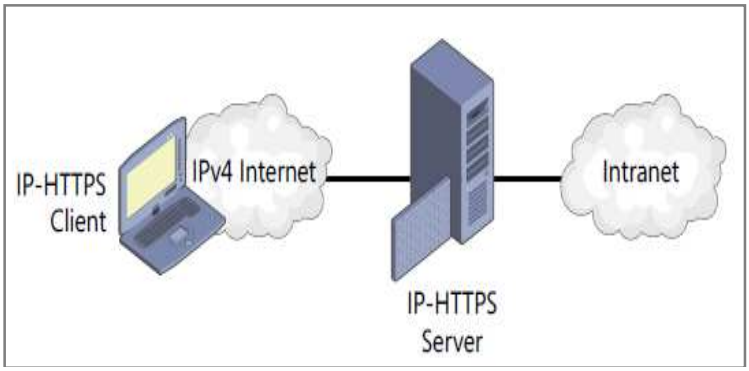


Figura 4.19: Componentes de una Arquitectura IP-HTTPS

IP-HTTPS tiene implementaciones para Windows Server 2012, Windows Server 2008 R2, Windows 8 y Windows 7, donde el servidor IP-HTTPS es un servidor de DirectAccess[11].

4.3.13 DSTM - Dual-Stack IPv6 Dominant Transition Mechanism

DSTM consiste en el uso de túneles IPv4 sobre IPv6 para transportar tráfico IPv4 en una red IPv6 y proporciona un método para asignar una dirección IPv4 temporal a nodos IPv6/IPv4 de la capa Dual Stack [25]. Está diseñado para permitir la comunicación de nodos IPv6 con nodos IPv4 sin la necesidad de utilizar traducciones NATs.

Es un mecanismo de transición que utiliza los protocolos existentes, DSTM no especifica un protocolo. Sin embargo, DSTM define cliente, servidor, y el comportamiento de un router de borde y las propiedades que tiene el mecanismo temporal de asignación.

Los siguientes términos se utilizan con DSTM:

- **Dominio DSTM:** Áreas de Red en la Intranet donde los nodos duales IPv6/IPv4 utilizan DSTM para asegurar la comunicación IPv4. Una asignación de direcciones IPv4 puede ser desplegada dentro de este dominio.
- **Cliente DSTM:** Un nodo IPv6/IPv4 de la capa Dual Stack implementado con el software de cliente DSTM.
- **Servidor DSTM:** Un nodo IPv6/IPv4 de la capa Dual Stack implementado con el software de servidor DSTM.
- **DSTM Border Router (TEP):** Un nodo IPv4/IPv6 implementado con el software DSTM Border Router. Se conecta la red IPv6 con la red IPv4 y gestiona la asignación de direcciones (IPv6 a direcciones IPv4).

En consecuencia, la arquitectura DSTM se compone de un servidor de direcciones DSTM y nodos DSTM. El servidor DSTM es el responsable de la asignación de las direcciones y la gestión de los nodos de puntos finales (TEP). Los TEP son los encargados de la encapsulación y desencapsulación de paquetes y son gestionados por los controles del servidor DSTM. La Figura 4.20 muestra los componentes de una arquitectura DSTM.

[11] DirectAccess establece una conexión bi-direccional con usuarios de una red empresarial y computadores portátiles conectadas al Internet

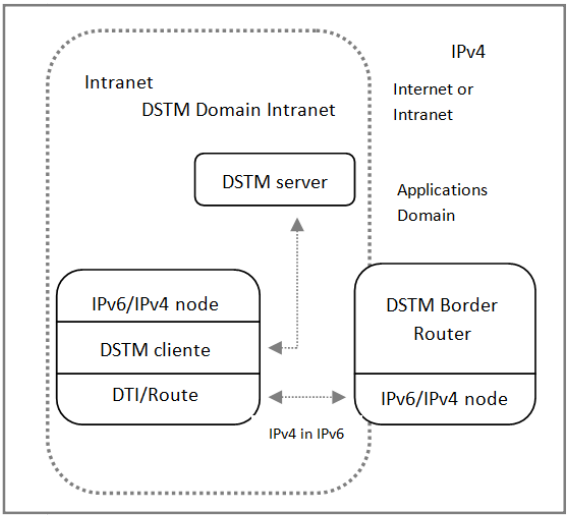


Figura 4.20: Arquitectura DSTM

DSTM es un proyecto en desarrollo, sus últimas actualizaciones fueron publicadas en octubre de 2005 en el Internet-Draft con fecha de expiración 20 de abril de 2006, “Dual Stack IPv6 Dominant Transition Mechanism (DSTM) <draft-bound-dstm-exp-04.txt>”. Sin embargo, existen implementaciones de DSTM para Windows XP, Linux, Free BSD, and Sharp Zaurus.

4.3.14 Tecnología 4rd

4rd es un mecanismo de transición en fase de desarrollo, que proporciona servicios IPv4 sobre la red IPv6 de un ISP. Es una tecnología basada en túneles, similar a 6rd. La diferencia radica en que, 4rd provee conectividad IPv4 en redes IPv6-only, útil para los clientes que aún lo necesitan. Respeta la transparencia de los paquetes IPv4 dentro de las redes end-to-end.

Debido al agotamiento de las direcciones IPv4, 4rd incluye un mecanismo para compartir la misma dirección IPv4 entre varios clientes. Este mecanismo consiste en la asignación de puertos de la capa de transporte. Ninguno de los primeros 4096 es asignado a un cliente, todos los demás se distribuyen. Cada cliente puede tener hasta 4 rangos de puertos, los cuales son derivados algorítmicamente de su prefijo IPv6.

4rd es un proyecto en desarrollo, sus últimas actualizaciones fueron publicadas en abril de 2013 en el Internet-Draft (Experimental) con fecha de expiración en octubre de 2014, “IPv4 Residual Deployment via IPv6-a Stateless Solution (4rd) <draft-ietf-softwire-4rd-08>”.

4.4 Traducción

Permite a los nodos IPv6-only comunicarse con nodos IPv4-only [04]. Aunque no es una técnica deseable a largo plazo. Trabaja realizando traducción de IPv6 a IPv4 y viceversa.

Cuando un traductor IPv4 a IPv6 recibe un datagrama IPv4 dirigida a una isla IPv6, traduce la cabecera IPv4 de ese paquete en una cabecera IPv6. Posteriormente envía el paquete basado en la dirección destino de IPv6. La cabecera IPv4 se retira y se sustituye por una

cabecera IPv6 (a excepción de los paquetes ICMP de la capa de transporte y los datos que se dejan sin alteraciones).

Cuando un traductor IPv6 a IPv4 recibe un datagrama IPv6 dirigida a una isla IPv4, traduce la cabecera IPv6 en una cabecera IPv4. Posteriormente, reenvía el paquete basado en la dirección destino de IPv4. La cabecera IPv6 original es eliminada y sustituida por una cabecera IPv4 (a excepción de los paquetes ICMP de la capa de transporte y los datos que se dejan sin alteraciones).

Los mecanismos de transición de este tipo son los menos recomendados, pues, una traducción no es perfecta y requiere soporte de Application Layer Gateways (ALG). Todas las técnicas de traducción deben utilizarse sólo si no hay otra opción [04].

4.4.1 NAT64/DNS64

NAT64 con estado es un mecanismo de transición que permite traducir paquetes IPv4 a IPv6 y viceversa. DNS64 es un mecanismo para la síntesis de los registros de recursos AAAA (RR). Juntos permiten a un solo cliente IPv6-only iniciar comunicaciones con un servidor IPv4-only. También permiten la comunicación peer-to-peer (con técnicas como ICE: Interactive Connectivity Establishment, RFC 5245 [55]) entre un nodo IPv4 y un nodo IPv6.

La traducción stateful NAT64 se realiza mediante la traducción de los paquetes según el algoritmo de traducción de IP/ICMP definido en el RFC 6145 [42]. Las direcciones IPv4 de host IPv4 son algorítmicamente traducidas desde y hacia IPv6 utilizando el algoritmo definido en el RFC 6052 [40] y un prefijo IPv6 asignado a la stateful NAT64 para este propósito específico.

Las direcciones IPv6 de host IPv4 son traducidas desde y hacia direcciones IPv4 mediante la instalación de las asignaciones normales en el Network Address Port Translation (NAPT, RFC 3022 [48]).

NAT64 cumple con las recomendaciones de cómo debe un NATs manejar los protocolos UDP (RFC 4787 [52]), TCP (RFC 5382 [53]), e ICMP (RFC 5508 [54]). Es compatible con las técnicas de NAT transversal actuales, como ICE (RFC5245 [55]), y con otras técnicas de NAT transversal. Su funcionamiento fue especificado en abril de 2011 mediante la publicación del RFC 6146 [28].

DNS64 es un mecanismo para la síntesis de los registros de recursos AAAA (RR) a partir de los RRs. Un recurso AAAA sintético creado por el DNS64 de un recurso A contiene el mismo nombre de propietario del recurso original A, pero contiene una dirección IPv6 en lugar de una dirección IPv4.

La dirección IPv6 es una representación de la dirección IPv4 del recurso original A. La representación IPv6 de la dirección IPv4 es algorítmica generada a partir de la dirección IPv4 devuelto en el registro A y un conjunto de parámetros configurados en el DNS64 (típicamente, un IPv6 prefijo utilizado por las representaciones IPv6 de direcciones IPv4 y, opcionalmente, otros parámetros) [29]. Su funcionamiento fue especificado en abril de 2011 mediante la es publicación del RFC 6147 [29].

4.4.2 NAT66

Protocolo de tipo experimental, definido en el RFC 6296 [31]. Consiste en la asignación de un Stateless IPv6-to-IPv6 Network Prefix Translation (NPTv6), diseñado para proporcionar la función de independencia asociada con una dirección IPv4-to-IPv4 NAT (NAPT44), y una relación de 1:1 entre las direcciones “dentro” y “fuera” de la red, preservando la accesibilidad de end-to-end en la capa de red.

NPTv6 puede implementarse en un router IPv6 asignando un prefijo NPTv6 a otro prefijo IPv6 por cada datagrama IPv6 que transite por él. Un router que implementa una función de traducción de prefijo NPTv6 se conoce como un NPTv6 Traductor.

El mecanismo de asignación de direcciones NPTv6 es puramente algorítmico, por lo que los traductores NPTv6 no necesitan mantener un estado por nodo o por conexión, permitiendo el despliegue de redes más robustas y adaptables.

NPTv6 implica la modificación de las cabeceras IP en tránsito, por lo que no es compatible con mecanismos de seguridad, tales como AH (Authentication Header) de IPsec, que proporcionan protección de la integridad para el encabezado IP.

4.4.3 SIIT - Stateless IP/ICMP Translation

Mecanismo transición basado en un algoritmo asíncrono sin estado que permite la traducción entre los formatos de cabecera IPv6 y IPv4 (incluyendo las cabeceras ICMP). Puede ser utilizado como parte de una solución para que los hosts IPv6 que no tienen direcciones IPv4 asignadas puedan comunicarse con hosts IPv4-only.

Para el uso de SIIT se asume que en la red IPv6 existen otros mecanismos de transición (como túneles o dual stack) para asegurar la interoperabilidad. Además, se utilizan los siguientes términos:

- **Nodo IPv4-capable:** Un nodo que tiene una pila de protocolo IPv4.
- **Nodo IPv4-enabled:** Un nodo que tiene una pila de protocolo IPv4 y se le asigna una o más direcciones. Los nodos IPv4-only y IPv6/IPv4 son habilitados.
- **Nodo IPv6-capable:** Un nodo que tiene una pila de protocolo IPv6.
- **Nodo IPv6-enabled:** Un nodo que tiene una pila de protocolo IPv6 y se le asigna una o más direcciones. Los nodos IPv6-only y IPv6/IPv4 son habilitados.

El método SIIT define un rango de direcciones IPv6 llamado IPv4-translated (IPv4-traducidas). Para evitar el uso de direcciones IPv4-compatible fuera del uso previsto dentro del túnel automático, SIIT especifica los siguientes formatos de direcciones IPv4-translated:

- **IPv4-Mapped:** Una dirección de la forma 0::ffff:a.b.c.d, que se refiere a un nodo que no es IPv6-capable. Además este protocolo utiliza este tipo de direcciones en los paquetes IPv6 para referirse a un nodo IPv4.
- **IPv4-Compatible:** Una dirección de la forma ::a.b.c.d, que se refiere a un nodo IPv6/IPv4 que apoya el túnel automático. Tales direcciones no se utilizan en este protocolo.

- **IPv4-Translated:** Una dirección de la forma 0::ffff:0:a.b.c.d, que se refiere a un nodo habilitado para IPv6. El prefijo 0::ffff:0:0:0/96 se utiliza para dar un checksum de 0 evitando así cambios en el checksum de la cabecera del protocolo de transporte.

En tal sentido las cabeceras TCP y UDP por lo general no necesitan ser modificadas por el traductor. Las únicas excepciones son los paquetes no fragmentados IPv4 UDP que necesitan una suma de comprobación UDP computarizada pues se requiere checksums pseudo-header UDP en IPv6.

La especificación del protocolo fue publicada inicialmente en febrero de 2000 mediante el RFC 2765 [41] y posteriormente publicado el RFC 6145 [42].

Cabe resaltar que existen dos implementaciones de SIIT (NAT-PT y NAPT-PT) definidas en el RFC 2766 [32] publicado en Febrero de 2000 y posteriormente obsoletas por el RFC 4966 [33] en julio de 2007.

Además NAT-PT tiene dos implementaciones similares (BIS y API) definidas en el RFC 2767 [35] y el RFC 3338 [43], los cuales solo proveen información. Estas Implementaciones se explican a continuación:

- **NAT-PT:** Network Address Translation (NAT) permite traducir la dirección IP sobre TCP, UDP, e ICMP header checksums.

  Network Address Translation and Protocol Translation (NAT-PT) permite traducir un paquete IPv6 en un paquete equivalente IPv4 y viceversa.
- **NAPT-PT:** Network Address Port Translation (NAPT), además de los campos traducidos por el transporte de NAT, identificadores tales como TCP y números de puerto UDP y los tipos de mensajes ICMP pueden ser traducidos. Permite que un conjunto de hosts IPv6 puedan compartir una única dirección IPv4.

  NAPT-PT permite que los hosts IPv6 para comunicarse con hosts IPv4 utilizando una sola dirección IPv4.
- **BIS:** Bump In the Stack (BIS) es similar a NAT-PT pero con la diferencia de que el traductor se encuentra dentro del sistema operativo del hosts. BIS es una interfaz de traducción entre las aplicaciones IPv4 e IPv6, y su objetivo es apoyar aplicaciones IPv4 dentro de una red dominante IPv6 [04].
- **BIA:** Bump in the API (BIA) es el mismo mecanismo que en el BIS pero en este caso la traducción sucede internamente entre la IPv4 y la API IPv6. Cuando una aplicación IPv4 quiere comunicarse con un nodo IPv6, el traductor API intercepta las funciones socket API y llama al correspondiente API IPv6 socket. También utiliza un conjunto de direcciones IPv4 internas [04].

4.4.4 TRT - Transport Relay Translator

Es un mecanismo de traducción diseñado para ser utilizado en la capa de transporte de una red IPv6-only. Un sistema TRT se localiza entre hosts IPv6-only y hosts IPv4. Traduce TCP, UDP sobre IPv6 sobre TCP, UDP a IPv4, viceversa.

En caso de una conexión TCP, el sistema TRT termina la conexión con el cliente y realiza una nueva conexión TCP en el otro lado de la aplicación de IPv4, internamente se realiza la traducción entre las dos sesiones. En caso de una conexión UDP, el traductor simplemente realiza la traducción y reenvía el paquete.

Los traductores de cabecera IPv6 a IPv4 tienen que cuidar de MTU Path y los problemas de fragmentación. Sin embargo, TRT está libre de este problema. TRT sólo admite el tráfico bidireccional. Los traductores de encabezados de IPv6 a IPv4 pueden ser capaces de soportar otros casos, tales como datagramas multicast unidireccionales.

Combinado con una implementación especial del servidor DNS (que traduce direcciones IPv4 a IPv6), los sistemas de apoyo a TRT IPv6-to-IPv4 funcionan muy bien. No requiere ningún cambio en los clientes IPv6 ni en los servidores IPV4 existentes, por lo que el sistema de TRT se puede instalar muy fácilmente a las redes IPv6-capable existentes.

TRT reserva un prefijo IPv6 referido por C6::/64 para la asignación de direcciones el cual debe ser una parte del espacio de direcciones IPv6 unicast asignado al sitio. La información de enrutamiento debe estar configurado de manera que paquetes a C6 :: / 64 se enrutan hacia el sistema TRT [34].

Su funcionamiento fue definido en junio de 2001 mediante la publicación del RFC 3142 [34], el cual solo provee información y no especifica un protocolo.

4.4.5 SOCKS64

Es un mecanismo de puerta de enlace IPv6/IPv4 basado en SOCKS que permite comunicaciones heterogéneas suaves entre los nodos IPv6 y los nodos IPv4 [37]. Se basa en el protocolo SOCKS definido en el RFC 1928 [38].

Es un mecanismo que transmite dos conexiones “terminated” IPv4 e IPv6 en la "Capa de aplicación" (el servidor SOCKS); sus características son heredadas de los mecanismos Relay de conexión de la capa de aplicación y los del mecanismos de SOCKS nativos [37]. Fue publicado en abril de 2001 mediante el RFC 3089 [37] de tipo informacional. Este RFC solo provee información y no especifica un protocolo.

4.4.6 Address plus Port (A+P)

Address plus Port [A+P] utiliza un NAT en la red del cliente (o cerca de él) para acceder la red IPv4. El ISP proporciona una dirección IPv4 y un rango de puertos TCP/UDP para el NAT. Una arquitectura A+P realiza el proceso de encapsulación/desencapsulación, de traducción NAT y señalización en donde el tráfico IPv4 de salida es traducido utilizando el rango de puertos TCP/UDP disponible mediante un NAT.

El ISP es el encargado de realizar el enrutamiento para que el paquete traducido llegue al destino utilizando tanto la dirección IPv4 destino como el puerto TCP/UDP de destino. A+P es un protocolo experimental publicado en agosto de 2011 en el RFC 6346 [44].

5. Trabajos Relacionados

En este capítulo se abordará una serie de estudios relacionados con el proceso de transición IPv4 a IPv6. Estos estudios abarcan desde la evaluación del proceso de implantación y adopción de IPv6 hasta el estudio del rendimiento de los diferentes mecanismos de transición en diferentes sistemas operativos.

Entre los estudios que evalúan el proceso de implantación y la adopción de IPv6 se encuentran:

- **"IPv6 Adoption in the Internet" [56]:** describe una metodología para la evaluación de la adopción, la conectividad, y la latencia de IPv6, desde la perspectiva del operador de un sitio web. Fue aplicada para el sitio web de Google, permitiendo el estudio a gran escala del despliegue en el Internet de IPv6. La metodología de medición se basa en pedir a los clientes web enviar solicitudes HTTP a cualquiera de los host (IPv4-only o host de doble pila) y mediante las solicitudes exitosas comparar los resultados.

  Las pruebas de conectividad fueron realizadas tomando en cuenta la conectividad por ratio (medición de la disponibilidad de conectividad IPv6 entre los usuarios de Google), conectividad por tipo (tipo de conectividad IPv6 que utilizan los clientes: 6to4, Teredo, y ISATAP), conectividad por el tipo de sistema operativo (la mayoría de los clientes IPv6 fueron de Mac OS y Windows Vista que tienen habilitado IPv6 por defecto), conectividad por el país (determinación de la disponibilidad de IPv6 en los diferentes países, la señal más alta de despliegues IPv6 están en Francia y China), conectividad por AS (determinar los sistemas autónomos con mayor tráfico IPv6) y la latencia de IPv6 vs IPv4.

  Los autores se encontraron con que la latencia de IPv6 nativo es comparable a la de IPv4 y pudieron mostrar estadísticas sobre los mecanismos de transición IPv6 utilizados. Mediante los datos recogidos de las pruebas realizadas durante un año, demuestran que la adopción de IPv6 aun cuando crece de manera significativa, es aún baja, varía considerablemente según el país, y está fuertemente influenciada por un pequeño número de grandes despliegues.
- **"Tracking IPv6 Evolution: Data We Have and Data We Need" [57]:** seguimiento de la evolución de la adopción de IPv6 mediante la recolección, medición y análisis de datos, para informar las decisiones técnicas, empresariales y políticas. El autor dispone de datos que han permitido el seguimiento limitado del despliegue de IPv6 (mostrando el análisis de las actividades y los planes de medición de la caída de IPv6), expone los datos DNS del proyecto internacional de red, describe tipos adicionales de datos que apoyarían un mejor seguimiento, y ofrece una perspectiva sobre el futuro de la evolución de IPv6.

Mientras que los estudios anteriores realizan la evaluación de la implantación de IPv6 de forma globalizada, algunos de los trabajos relacionados con la evaluación del rendimiento de las versiones de IP, se enfocan en la comparación de los resultados de métricas, obtenidas de las tecnologías de transición (especialmente las de tunelización) en diferentes sistemas operativos, como es el caso de los siguientes estudios:

- **"Performance Evaluation of IPv4 and IPv6 on Windows Vista and Linux Ubuntu"[58]:** evaluación del rendimiento de IPv4 e IPv6 en los sistemas operativos Windows Vista y Linux Ubuntu para el tráfico TCP y UDP entre dos hosts con hardware similar (CPU: Intel Pentium Core 2 Duo, RAM: 2 GB, NIC: PCI Intel Pro 100, disco duro: Seagate de 160 GB), conectados usando cable cross-over. El throughput, delay, jitter y el uso del CPU fueron las métricas seleccionadas para realizar el estudio. Los resultados muestran que el rendimiento de la red depende no solo de la versión y del tipo de tráfico IP, sino también de la elección del sistema operativo [58], además, de que para Linux Ubuntu se obtuvo un jitter y uso del CPU menor, un delay superior y throughput similar a Windows Vista.
- **"Evaluating IPV6 on Windows and Solaris" [59]:** evaluación del throughput, RTT, uso del CPU, tiempo de creación de sockets, tiempo de conexión TCP e interacciones cliente-servidor para el tráfico TCP y UDP transmitido entre dos host con idéntico hardware (CPU Intel Pentium III 500 MHz, 256 MB de SDRAM PC100, 2 discos duros de 30 GB IBM 7200 RPM IDE, y 100 Mbps PCI Ethernet network adapters) y configuración (Windows 2000 Professional y Solaris 8). Durante las pruebas descubrieron que la pila IPv6 para Windows 2000 no podía manejar bien la fragmentación de mensajes UDP que eran más grandes que el tamaño de MTU de Ethernet 1514 bytes. Con TCP, no había tal problema, aunque la pila podría tener otras deficiencias que no se hayan detectado [59]. Solaris supera significativamente a Windows 2000 en la mayoría de las métricas (throughput y tiempo de creación de sockets) y demuestra su potencial superioridad para los servidores Web.
- **"Performance Comparison of IPv4 and IPv6 using Windows XP and Windows 7 over Gigabit Ethernet LAN"[61]:** comparación del rendimiento de IPv4 e IPv6 en un escenario conformado por dos hosts con hardware similar (CPU Intel Pentium D, RAM 1 GB, NIC Broadcom Net link ™ Gbit Ethernet y un disco duro de 160 GB) y configuración (Windows XP y Windows 7). Los resultados experimentales mostraron que Windows XP proporciona mejores resultados para UDP que Windows 7 y Windows 7 puede proporcionar mejores resultados para TCP que Windows XP. Además, IPv4 tiene resultados superiores en términos de throughput TCP y UDP en comparación con IPv6.
- **"Performance Analysis of IPv4 vs. IPv6 on various Operating Systems using Jumbo Frames" [62]:** evaluación de IPv4 e IPv6 en varios sistemas operativos, mediante el envió de Jumbo Frames. Estos sistemas operativos son: Microsoft Windows Server 2008, Microsoft Windows Server 2003, Microsoft Windows 7 Professional, Linux Fedora, Ubuntu y OpenSUSE. El throughput, delay, jitter, y el uso del CPU fueron las métricas utilizadas. El tamaño de los Jumbo Frames osciló entre 1518 y 9014 bytes. Los resultados concluyeron que para el tráfico TCP, Microsoft Windows Server 2008 y Microsoft Windows 7 tuvieron el más alto rendimiento tanto en IPv4 como en IPv6. Para el tráfico UDP todos los sistemas operativos presentaron resultados similares. Este experimento se llevó a cabo a nivel de aplicación y a nivel de red (DNS, juegos y tráfico VoIP para 5 códecs[12] diferentes) obteniendo resultados para cada uno de ellos.
- **"An Upper Bound Model for TCP and UDP Throughput in IPv4 and IPv6" [63]:** descripción de un modelo de límite superior para calcular el rendimiento del tráfico TCP y UDP de IPv4 e IPv6, en una conexión full-dúplex de tipo punto-a-punto, sobre la

[12] Compressor-Decompressor | Coder-Decoder.

tecnología de Ethernet de 10, 100 o 1000 Mbps. Los experimentos se realizaron en un escenario con la siguiente configuración: dos hosts conectados por un enlace punto-a-punto, con tres particiones en el disco duro para Windows XP SP2, Solaris 10, y Debian 3.1. Equipados con un dual AMD Opteron 246 (2 GHz), 2 GB de RAM, un disco duro de 72 GB, y un adaptador PCI Ethernet full-duplex de 10/100/1000 Mbps. Las métricas calculadas fueron el RTT y el throughput para las cuales hicieron sus propios benchmarks basados en el modelo cliente/servidor, utilizando el lenguaje de programación C.

Los resultados de los experimentos fueron comparados con el rendimiento máximo teórico de TCP/UDP para IPv4 e IPv6, y muestran que Ethernet 10 Mbps es una tecnología muy madura, debido a que los resultados son muy similares a los resultados de modelo de máximo rendimiento. Los resultados experimentales para Fast Ethernet (100 Mbps) dieron un rendimiento cercano a la capacidad máxima teórica, especialmente para TCP y datos UDP mayores de 2000 bytes. En GigaEthernet (1000 Mbps) los resultados son cercanos al modelo para grandes cargas útil de TCP y UDP. Sin embargo, para las cargas útil pequeñas de TCP y UDP, las diferencias entre el modelo de rendimiento máximo y los experimentos son importantes. Los autores suponen que estas diferencias deberían disminuir significativamente con el lanzamiento de tecnologías más rápidas (procesadores y memoria RAM).

- **"Performance Evaluation of Latest Windows Operating Systems" [64]:** evaluación del rendimiento de IPv4 e IPv6 para el tráfico TCP/UDP transmitido entre dos host con hardware idéntico (CPU Intel dual-core 2.50 GHz, 4 GB de DDR2 RAM, controlador LAN Marvell88E8056 PCI-E Gbit) y configuración (Windows XP Professional Service Pack 3, Windows Vista Business Service Pack 2 y Windows 7 Professional) similar, conectados usando cable 1 Gbps Ethernet cross-over. Las métricas utilizadas fueron: OWD (One-Way-Delay) y RTT (Round-Trip-Time), throughput, delay, jitter y el uso del CPU. Los resultados mostraron que el tráfico de red con tamaños de paquetes más pequeños se beneficiarían de Windows Vista y Windows 7. Sin embargo, para el tráfico de red con tamaños de paquete más grandes, Windows XP todavía tiene el mejor rendimiento.
- **"Evaluation of IPv6 and Comparison Study with Different Operating Systems" [71]:** Evaluación, comparación y análisis del rendimiento del protocolo IPv6 en tres sistemas operativos diferentes (Windows 2003, Red hat Linux 9.0 y Free BSD 4.9). Las pruebas fueron realizadas sobre tres escenarios diferentes: El primero, formado por un host (un bucle), el segundo por dos host conectados por un hub Ethernet, y el tercero dos host conectados por un router. El hardware consistía básicamente de tres sistemas Intel Pentium III 733 MHz con 128 MB de RAM y un disco duro de 30 GB. Las métricas evaluadas fueron el throughput y el RTT para tráfico TCP/UDP. Los resultados experimentales muestran que para el primer escenario de pruebas, el throughput en Red hat 9.0 de IPv6 es bueno, en Windows 2003 el más bajo y en Free BSD4.9 más alto. En el segundo escenario, el throughput es similar para los tres sistemas operativos. Sin embargo, se puede decir que Windows 2003 presentó un mejor desempeño.
- **"Deployment and Performance Evaluation of Teredo and ISATAP over Real Test-bed Setup" [65]:** evaluación del rendimiento de los mecanismos de transición IPv4-IPv6 tipo túnel: ISATAP y Teredo. Las pruebas fueron realizadas sobre los sistemas operativos Windows XP, Windows Server 2003 y Linux Fedora Core 10. Las pruebas fueron realizadas en un escenario conformado por 5 nodos (2 hosts, 1 servidor DNS, 2 routers) para transmitir UDP audio streaming, video streaming e ICMP-ping. Las métricas

evaluadas fueron: throughput, End to End Delay (E2ED), RTT y el jitter. Los resultados muestran que ISATAP tiene la ventaja sobre Teredo en todos los parámetros, excepto jitter. Teredo sigue siendo la mejor opción para el tráfico en tiempo real, según los autores.

- **"Network Performance Evaluation of Tunneling Mechanism" [66]:** evaluación de los mecanismos de transición tipo túnel, para el tráfico TCP/UDP. Las métricas evaluadas fueron: throughput, RTT y overhead del túnel para el tráfico TCP/UDP sobre IPv4, IPv6 y IPv6 in IPv4 (túnel), sobre un escenario con la siguiente configuración: sistema operativo Microsoft Windows 7, router Cisco 2821 con IOS 12.2(2) T, switch Cisco Catalyst 2960-24TT 24- Port Ethernet switch, D-ITG, y WireShark 1.2.6. Las pruebas se realizaron 20 veces para asegurar mayor exactitud en los datos, mediante los cuales mostraron para IPv6 in IPv4, un incremento del overhead de túnel y RTT cuando el tamaño de los paquetes crecía, obteniendo la técnica de tunelización un rendimiento inferior que IPv4 y IPv6 nativo en el contexto de la transmisión de datos TCP. En UDP, sin embargo, los rendimientos generales tuvieron valores equivalentes para todos los de tamaños de datos. Esto explica el hecho que el overhead de túnel en UDP, si bien no es menor que en TCP, tampoco está influenciado por el tamaño de los datos [66]. Finalmente, concluyeron: que el mecanismo de transición tipo túnel, sólo es adecuado para ser aplicado a la investigación y experimentación durante el período de transición.
- **"Investigating the IPv6 Teredo Tunnelling Capability and Performance of Internet Clients" [67]:** evaluación del mecanismo de transición Teredo, utilizando mediciones basadas en la web para investigar la capacidad de los clientes Teredos en el Internet y el retardo introducido por la utilización de esta técnica. Se compararon los resultados obtenidos para IPv4 e IPv6 nativo y túnel Teredo para los entornos Windows Vista, Windows 7, Windows XP y otros (Linux y Mac OS X). Las métricas evaluadas fueron capacidad de túnel y delay. Los resultados concluyeron que mientras que poco más 6% de las conexiones de los clientes eran totalmente compatibles con IPv6 usando IPv6 nativo o 6to4, más del doble de conexiones (15-16%) eran de los clientes de Windows que podría haber utilizado IPv6, si Windows Teredo fue totalmente habilitada de forma predeterminada. Sin embargo, el delay en Teredo aumenta en gran medida para traer objetos por 1-1.5 segundos en comparación con IPv4 o IPv6 nativo. Además, más del 80% de las conexiones Teredo en Windows Vista y el 66% en Windows 7, no lograron establecer un túnel.
- **"Performance Comparison of 6to4, 6RD and ISATAP Tunnelling Methods on Real Testbeds" [68]:** evaluación y comparación del rendimiento de las técnicas de transición IPv4-IPv6, tipo túnel, 6to4, ISATAP, y 6RD para tráfico TCP/UDP sobre el sistema operativo Linux Ubuntu. Las métricas evaluadas fueron el throughput y la utilización del CPU, en un escenario conformado por un servidor, tres routers y un host. Los resultados de este estudio mostraron que 6RD exhibió la mayor eficiencia tanto para UDP y TCP. Además, se determinó que los métodos 6to4 y 6RD son más rápidos y más estable que los métodos ISATAP para grandes descargas de archivos basados en TCP. Por último, los autores admiten que este estudio no es suficiente para su uso en el análisis de una red real, ya que, se llevaron a cabo las pruebas bajo un entorno de prueba limitada.
- **"Performance Comparison of ISATAP Implementations on Free BSD Red hat and Windows 2003" [69]:** comparación rendimiento de la implementación del mecanismo de transición tipo túnel ISATAP, en los sistemas operativos Windows 2003, Free BSD 5.3

(con KAME), y Red hat 9.0 (con USAGI). Las métricas evaluadas fueron el throughput, la perdida de paquetes IP/TCP, y el máximo número de paquetes por segundos en el cual el router comience a descartar paquetes. En un escenario formado por 2 hosts y un router con la siguiente configuración: el router: 1.6 GHz Pentium 4 con 512 MB de RAM. El emisor 2.4 GHz Pentium 4 con 128 MB de RAM, y el receptor 2.4 GHz Pentium 4 con 256 MB de RAM. Los resultados experimentales muestran que la implementación de ISATAP en Red hat 9.0 obtuvo un mejor rendimiento.

- **"Tunneling IPv6 through NAT with Teredo Mechanism" [70]:** evaluación del mecanismo de transición tipo túnel Teredo, de los sistema operativo Linux, Solaris y Free BSD (Miredo). En este estudio, los autores explican cómo los hosts candidatos IPv6 que se encuentran detrás de un servidor NAT pueden contar con la ayuda de los servidores Teredo y los Teredo relays para aprender sus direcciones globales y para obtener conectividad, y como clientes, servidores y relays pueden ser organizados en redes Teredo. Además detallan estrategias para la implementación del servidor Teredo y los Teredo relays en Linux y muestran el funcionamiento de diferentes implementaciones de Teredo en dominio público.
- **"IPv4-v6 Configured Tunnel and 6to4 Transition Mechanisms Network Performance Evaluation on Linux Operating Systems" [71]:** evaluación del rendimiento de los mecanismos de transición tipo túnel 6to4 y túnel configurado, implementados en dos diferentes distribuciones de Linux (Fedora 9.10 y Ubuntu 11.0) y Windows Server 2008 y probados mediante tráfico TCP-UDP y evaluados con las métricas: throughput, delay, jitter y uso de la CPU. En un escenario formado por cuatro host (Intel Core 2 Duo E6300, 1.866 GHz: RAM 2 GB), conectados por cables Cat5e. Los resultados obtenidos muestran que los valores del throughput de TCP/UDP de los dos mecanismos son similares, pero el retardo es significativamente diferente dependiendo de la elección del mecanismo de transición y el sistema operativo.
- **"Analytical Performance Evaluation of Native IPv6 and Several Tunneling Technics using Benchmarking Tools" [73]:** comparación analítica de rendimiento de los mecanismos de transición tipo túnel, ISATAP, 6to4, 6rd, Teredo, con el rendimiento de IPv6 nativo, para el tráfico TCP/UDP en las tecnologías Ethernet y Fast Ethernet. Las métricas evaluadas fueron throughput y RTT sobre un escenario con la siguiente configuración de hardware y software: sistema operativo Ubuntu12.10, Kernel 3.5.0-25-generico, 2 AMD Opteron 246 procesadores de 1.99 GHz, 2 GB de RAM, y un Broadcom NeXtreme Gbit Ethernet. Los routers eran Cisco 2811 con la versión Cisco IOS 15.1 (4) M6, 256 MB de memoria flash, y 256 MB de RAM.

  Los resultados obtenidos para el tráfico TCP/UDP, en Ethernet muestran que IPv6 nativo tiene un RTT más bajo comparado con los demás mecanismos de tunelización. El rendimiento de Teredo es el más bajo y el de ISATAP, 6to4, y 6rd es muy similar. Para el tráfico TCP-UDP en Fast Ethernet, los resultados muestran que la red IPv6 nativo presenta el mejor rendimiento, mientras que Teredo representa el más bajo. En consecuencia los autores concluyen que ISATAP presenta el mejor rendimiento y Teredo puede ser visto como una solución de último recurso ya que tiene una alta sobrecarga.

6. Benchmarking

Benchmarking es una técnica empleada para medir el rendimiento de un sistema o sus componentes. Un benchmark es el resultado de la ejecución de un programa o conjunto de programas con el objeto de estimar el rendimiento de un elemento en particular [74]. En el caso de las redes de computadores, el uso de benchmarks puede arrojar resultados más realistas en la evaluación de desempeño en forma analítica, debido a que obtener representaciones matemáticas es difícil de lograr, ya que involucra varios factores en el estudio, incluyendo el hardware empleado, el software, la red de comunicaciones, el sistema operativo, etc.

6.1 Métricas de Benchmarking en Redes de Computadores

Algunas de las métricas más usadas al momento de hacer medición de desempeño en redes de computadores son descritas a continuación.

6.1.1 Throughput

La máxima tasa a la que ninguna de las tramas transmitidas es descartada por los dispositivos de interconexión [75].

6.1.2 Tasa de Pérdida de Paquetes

El porcentaje de paquetes que debieron ser enviados por un dispositivo de red bajo carga constante que no fueron recibidos en el destinatario por falta de recursos [75].

6.1.3 Latencia

Para dispositivos store-and-forward, el intervalo de tiempo desde que el último bit de la trama alcanza al puerto de entrada, finalizando cuando el primer bit de la trama es visto en el puerto de salida. Para dispositivos bit forwarding, el intervalo de tiempo desde que el primer bit de la trama llega al puerto de entrada, finalizando cuando el primer bit es visto en el puerto de salida [75].

6.1.4 Retardo

El retardo puede calcularse de dos formas: (1) de una vía, y (2) de ida y vuelta. El OWD, también llamado OWD (One Way Delay), es calculado como el promedio de las diferencias entre los tiempos de recepción y envío de paquetes [76]. Para obtener un resultado correcto, los relojes del emisor y el receptor deben estar sincronizados. La segunda opción es emplear el RTT (Round Trip Time). El RTT se refiere al tiempo de ida y vuelta de un paquete: desde que sale del emisor hasta que una respuesta es recibida de vuelta. En una red LAN puede estar en el orden de los milisegundos, mientras que en una red WAN puede estar en el orden de los segundos [77]. Esta medida puede tomarse en un mismo equipo (por ejemplo, el emisor del mensaje), eliminando la necesidad de la sincronización.

6.1.5 Jitter

El jitter es la variación del retardo. Se puede decir que el jitter es un problema si diferentes paquetes de datos encuentran diferentes retardos y la aplicación que usa esos datos en el receptor es sensible al retardo (por ejemplo, audio y video). Si el retardo del primer paquete es 20 ms, para el segundo es de 45 ms, y para el tercero es de 40 ms, entonces la aplicación de tiempo real que usa estos paquetes sufre de jitter.

6.2 Herramienta de Evaluación de Desempeño en Redes de Computadores

Existen muchas herramientas de evaluación de desempeño en el mercado, que presentan características muy distintas entre sí. A continuación se presentan diferentes herramientas de medición de desempeño: Iperf[13], Netperf[14], D-ITG[15], NetStress[16], MGEN[17], Rude & Crude[18], y WlanTV[19]. Estas herramientas fueron elegidas por su popularidad y porque representan proyectos activos [78]. De acuerdo a [79], estas herramientas pertenecen al grupo throughput and Bulk Transfer Capacity (BTC), que son benchmarks que emplean transferencias TCP y/o UDP de gran volumen para medir el throughput disponible entre dos procesos remotos (end-to-end).

6.2.1 Iperf

Es una herramienta de benchmark, desarrollada originalmente por Distributed Applications Support Team (DAST) en el National Laboratory for Applied Network Research (NLANR) como una alternativa para calcular el ancho de banda TCP, el rendimiento UDP, la tasa de pérdida de paquetes y el jitter.

Iperf fue diseñada siguiendo el paradigma cliente-servidor y escrita en el lenguaje C++. Su ejecución se inicia vía consola usando diferentes opciones provistas por la herramienta que pueden ser consultadas con la opción `–h`.

Cuenta con un ejecutable utilizable tanto para el cliente (`–c`) como para el servidor (`–s`). Algunos parámetros configurables incluyen el protocolo IP (IPv4 por defecto, `-V` para IPv6), el protocolo de transporte (TCP por defecto, `-u` para UDP), y el tiempo de duración del experimento (`-t`). Cuando se utiliza TCP como protocolo de transporte, se pueden especificar el tamaño de la ventana (`–w`). Cuando se utiliza el protocolo UDP, permite al usuario especificar el tamaño de los datagramas (`-l`) y proporciona resultados del rendimiento, jitter y de los paquetes perdidos. Típicamente la salida de Iperf contiene un informe con marcas de tiempo con la cantidad de datos transmitidos y el rendimiento medido. La comunicación entre los extremos que puede producir de forma unidireccional o bidireccional.

La diferencia en el uso de TCP y UDP, radica en que, TCP utiliza procesos para verificar que los paquetes sean enviados correctamente al receptor mientras que con UDP los paquetes son enviados sin realizar verificaciones dándole la ventaja en velocidad.

Iperf puede ser instalado muy fácilmente en cualquier sistema basado en Unix/Linux o Microsoft Windows. En el sitio web oficial de la herramienta se encuentra disponible la documentación y los archivos para su instalación. La documentación incluye un manual y

[13] https://iperf.fr/
[14] http://www.netperf.org
[15] http://www.grid.unina.it/software/ITG/index.php
[16] http://www.performancewifi.net/performance-wifi/main/netstress.htm
[17] http://mgen.pf.itd.nrl.navy.mil
[18] http://rude.sourceforge.net
[19] http://sourceforge.net/projects/wlantv

varios ejemplos de uso. Como Iperf es una herramienta multiplataforma, puede funcionar en cualquier red y devolver medidas de rendimiento estandarizadas. Esto puede ser útil para comparar equipos de red cableados e inalámbricos de manera imparcial.

6.2.2 Netperf

Netperf es una herramienta de benchmark de código abierto que puede usarse para medir varios aspectos del rendimiento de una red. Su enfoque principal está en la transferencia de grandes volúmenes de datos y el desempeño en tráfico de tipo solicitud/respuesta, usando TCP y UDP con la interfaz de sockets BSD.

Netperf está diseñado siguiendo el paradigma cliente-servidor. Hay dos ejecutables (`netperf` y `netserver`). El programa `netserver` puede ser invocado por `inetd` (el meta-demonio del sistema), o puede ser ejecutado como un demonio standalone. En el primer método, el usuario debe tener privilegios de administrador; el segundo método implica que el usuario debe recordar ejecutar el programa de forma explícita. El programa se puede ejecutar sin opciones, a menos que el usuario desee cambiar el puerto usado por defecto (con la opción `-p`) o desee trabajar con IPv6 (con la opción `-6`) en lugar de IPv4. `netperf` es el cliente de la herramienta, y debe ser invocado por el usuario para iniciar la generación de tráfico en el experimento.

En Debian, `Netperf` puede ser instalado usando `Aptitude`, una utilidad manejadora de paquetes del sistema. Para lograr esto, se debe editar el archivo `/etc/apt/sources.list` (en este paso se necesitan privilegios de administrador), para añadir la rama `non-free` en los repositorios existentes, y que así el paquete pudiese ser ubicado. También es posible la instalación desde archivos fuente cuando la versión binaria no está disponible para el sistema en cuestión.

En el sitio web oficial de la herramienta se encuentra disponible la documentación. Esta incluye información detallada sobre el proceso de instalación basada en los archivos fuente, y una descripción general de la herramienta. Adicionalmente, se describe cómo se pueden llevar a cabo diferentes pruebas. Finalmente, se incluyen una sección de ejemplos y otra de manejo de errores (troubleshooting).

`Netperf` puede ejecutarse desde una consola con el comando `netperf –H <nombreHost|direcciónIP>`, especificando el nombre del host remoto o su dirección IP. Al ejecutar `netperf`, se establece una conexión de control hacia el sistema remoto (`netserver`) para intercambiar parámetros de configuración y resultados hacia y desde el sistema remoto. La conexión de control es una conexión TCP que usa sockets BSD. La duración de la prueba puede ser configurada.

Los resultados finales de la herramienta incluyen tamaño del mensaje enviado en bytes, duración de la prueba en segundos, y throughput. Los resultados se reportan en formato de tabla. Es posible fijar diferentes unidades para la salida reportada, pero estas modificaciones deben ser indicadas al inicializar la prueba.

Netperf es soportado únicamente por plataformas Unix. Como protocolo de capa de red es posible usar IPv4 o IPv6[02][03], con sólo especificar la opción -4 o -6, respectivamente. También es posible usar tanto TCP como UDP como protocolo de capa de transporte.

6.2.3 D-ITG

D-ITG (Distributed Internet Traffic Generator) es una plataforma de código abierto para la generación de tráfico, capaz de producir tráfico IPv4 e IPv6 para paquetes con tamaño y tiempo inter-salida variable. Está concebida para ser usada como una herramienta distribuida de medición de rendimiento, capaz de calcular el retardo de ida (OWD - One Way Delay) y de ida-y-vuelta (RTT - Round Trip Time), la tasa de pérdida de paquetes, el jitter y el throughput [60].

D-ITG sigue el modelo cliente-servidor. Hay cuatro ejecutables básicos que implementan los componentes de la plataforma: ITGSend, ITGRecv, ITGLog, e ITGDec. ITGSend actúa como el cliente, y puede generar varios flujos de datos de forma simultánea siguiendo las especificaciones del archivo de entrada (o archivo de configuración). ITGRecv actúa como el servidor y puede recibir varios flujos de datos de diferentes clientes de forma simultánea. ITGLog es el servidor de almacenamiento (log) de la plataforma, y recibe información de ITGSend e ITGRecv. ITGDec es una utilidad que permite analizar los resultados de los experimentos realizados. Adicionalmente se cuenta con dos ejecutables más: ITGPlot e ITGapi. ITGPlot es una herramienta basada en Octave[20] para graficar los datos contenidos en los archivos log construidos con ITGDec (como delay.dat, bitrate.dat, jitter.dat y packetloss.dat), que contienen un promedio del retardo, tasa de bits, jitter y tasa de pérdida, respectivamente, que son calculados cada milisegundos y reportado en el archivo respectivo. ITGapi es un API de C++ que permite el control remoto de la generación de tráfico.

Cuando se ejecuta ITGSend, se establece una conexión de control al sistema remoto (ITGRecv), para entregar los parámetros de configuración de la prueba, como el protocolo de red a utilizar.

En el sitio web oficial de la herramienta se encuentra disponible la documentación y los archivos para su instalación. La documentación incluye un manual (que también puede ser accedido desde la herramienta con la opción -h) y varios ejemplos de uso.

La herramienta puede trabajar en dos modos: sencillo y script. En el modo sencillo se puede generar un único flujo de paquetes de un cliente (ITGSend) al servidor (ITGRecv). El modo script permite a ITGSend generar de forma simultánea varios flujos desde un mismo cliente. Cada flujo es manejado por un hilo, y un hilo adicional actúa como maestro coordinando al resto de los hilos. Para generar n flujos, el archivo de entrada debe contener n líneas, cada una de las cuales especifica las características de un flujo.

La ejecución se inicia vía consola (no se ofrece una GUI) usando diferentes opciones provistas por la herramienta, que pueden ser consultadas con la opción –h. Algunos

[20] http://www.octave.org

parámetros configurables incluyen TTL (Time to Live), tiempo inter-salida, tamaño del payload, y tipo de protocolo.

D-ITG permite medir el retardo de ida, de ida-y-vuelta, la tasa de pérdida, el jitter, y el throughput. Está disponible para plataformas Unix y Windows. Tanto IPv4 como IPv6 son soportados como protocolo de capa de red; en Linux, para seleccionar uno u otro el usuario debe especificar la dirección IP en el formato apropiado con la opción `-aen ITGSend`. En Windows hay dos binarios diferentes para el soporte de IPv4 e IPv6.

UDP es usado por defecto como protocolo de capa de transporte, pero es posible usar TCP e inclusive ICMP. Otros protocolos de capas superiores soportados incluyen Telnet, DNS, y RTP para aplicaciones de VoIP.

6.2.4 NetStress

NetStress es una herramienta de benchmark sencilla que permite calcular el desempeño de redes cableadas e inalámbricas. Emplea transferencia de grandes volúmenes de datos con TCP. El rendimiento de la red se reporta en términos de throughput. Nuevamente, se emplea el modelo cliente-servidor durante el proceso de pruebas.

La instalación de NetStress es muy sencilla. Una vez descargado el instalador, sólo hace falta ejecutarlo y seguir las instrucciones del asistente de instalación, como es de costumbre en ambientes Windows.

En el sitio web oficial se encuentra un enlace al archivo de ayuda que contiene una breve descripción de la herramienta, los requerimientos del sistema, y explicaciones sobre cómo ejecutar el cliente y el servidor. Adicionalmente, incluye una sección sobre cómo interpretar los resultados reportados. Esta ayuda también puede ser accedida desde la aplicación, en el menú “Help”, seleccionando la opción “Contents”.

Esta herramienta provee una GUI donde el usuario puede seleccionar diferentes opciones. Una vez iniciada la herramienta, se debe seleccionar un modo: cliente o servidor. El servidor recibe paquetes de la red, mientras que el cliente transmite los paquetes a la red.

No hay opción para detener la prueba, ni tampoco una forma de especificar la duración del experimento o la cantidad de datos a transferir, así que el usuario debe cerrar la aplicación para detener el experimento.

La herramienta reporta la cantidad de bytes enviados, bytes recibidos, y el throughput, en modo texto y gráficamente. Es soportada por plataformas Windows. Emplea IPv4 como protocolo de capa de red y TCP como protocolo de capa de transporte. No ofrece soporte para otros protocolos.

6.2.5 MGEN

Multi-Generator (MGEN) es una herramienta de código abierto desarrollada por el grupo de investigación PROTocol Engineering Advanced Networking (PROTEAN) del Laboratorio de Investigación de la Armada (NRL - Naval Research Laboratory) de Estados Unidos. MGEN ofrece la posibilidad de realizar pruebas de rendimiento sobre redes IP y mediciones usando

tráfico UDP/IP. Soporta tanto generación de tráfico unicast como multicast. Sigue el modelo cliente-servidor, usando el mismo programa para ambos extremos.

La distribución contiene archivos HTML con documentación extensa. Allí se explica el uso de la herramienta y sus diferentes opciones. Además ofrece varios ejemplos sobre cómo usar la herramienta para medir el desempeño de la red.

Se puede usar archivos de entrada (scripts) para definir patrones de carga de la red durante el tiempo. También está disponible la opción de líneas de comando. Los archivos de entrada se pueden usar para especificar patrones de tráfico de aplicaciones unicast y/o multicast UDP/IP. Los flujos de datos definidos pueden seguir patrones periódicos (CBR – Constant Bit Rate), Poisson, y por ráfagas. Estos flujos de datos pueden ser modificados durante el experimento, ya que el archivo de entrada permite alterar un flujo dado en un momento determinado.

Algunos campos de la cabecera IP pueden ser fijados por el usuario. Cuando se define un flujo multicast, el usuario puede especificar el TTL. Tanto para flujos unicast como multicast, el valor del campo ToS (Type of Service) también puede ser especificado. Para IPv6, se puede definir el valor del campo Flow Label. Para flujos multicast, el usuario puede controlar cuando unirse o dejar un grupo multicast, indicando la dirección IP del grupo multicast al que unirse o abandonar y el tiempo.

Los resultados son mostrados por la salida estándar, o pueden ser redirigidos a un archivo para su posterior análisis. Estos resultados sólo representan el reporte de los paquetes intercambiados; no se ofrece información adicional.

Para obtener estadísticas, el usuario debe almacenar los resultados en un archivo, que pueda ser usado posteriormente como entrada del programa trpr[21] (Trace Plot Real-time). trpr analiza la salida de MGEN y crea una salida apropiada para su graficación. Además, soporta un rango de funcionalidades para uso específico del programa de graficación gnuplot[22].

Resultados importantes como throughput, latencia, tasa de pérdida, y latencia de unión/abandono de grupos multicast pueden ser calculados a partir de la información en el archivo de salida. Sin embargo, esta tarea es dejada al usuario; MGEN no ofrece resultados estadísticos directos.

6.2.6 Rude & Crude

Rude (Real-time UDP Data Emitter) es un programa que genera tráfico de red, y Crude (Collector for Rude) recibe el tráfico generado por Rude, siguiendo el paradigma cliente-servidor. Actualmente, sólo genera y mide tráfico UDP.

[21] http://pf.itd.nrl.navy.mil/protools/trpr.html
[22] http://www.gnuplot.info

Para instalar Rude se debe descargar el paquete y descomprimirlo. Luego, se usa el script configure para crear el makefile, y usarlo mediante los comandos `make y make install`. Para el proceso de instalación se requieren privilegios de administrador.

La documentación se ofrece a través del manual de la herramienta, que se encuentra disponible una vez que la herramienta se instala. Adicionalmente, en el sitio web se encuentran instrucciones para el proceso de instalación.

El funcionamiento y la configuración son similares a los de MGEN, pero estos proyectos no comparten código. El usuario debe escribir un archivo de entrada (script) que define el experimento. Se pueden definir múltiples flujos en el archivo de entrada. En la documentación se incluye un archivo ejemplo, que también está descrito en el manual.

Los resultados se muestran por defecto en la salida estándar, pero pueden ser redirigidos a un archivo para ser procesados posteriormente.

Los resultados incluyen estadísticas del número de paquetes recibidos, número de paquetes recibidos fuera de secuencia, paquetes perdidos, total de bytes recibidos, retardo promedio, jitter, y throughput. La herramienta se ejecuta en plataformas Unix. Los protocolos soportados son UDP/IPv4.

6.2.7 Wlan Traffic Visualizer

WLAN Traffic Visualizer (WlanTV) permite medir la carga de la red y visualizar secuencias de tramas en redes WLAN IEEE 802.11 [80]. Se encuentra publicada bajo licencia GPL. Está desarrollada en Java, y se encuentran disponibles para la descarga en el sitio web versiones fuente y JAR.

WlanTV requiere Wireshark[23] (Windows) o TShark (Linux), y Java Runtime Environment JRE 1.6 o superior para la ejecución del programa. Sólo se requieren los paquetes estándar de Java. Se usa TShark para decodificar archivos log y para realizar capturas en vivo.Al instalar WlanTV a partir de la distribución JAR, y usando el comando `java –jar wlantv.`jar se abre la GUI desde donde el usuario puede realizar los experimentos.

La documentación no es muy extensa. La distribución incluye un archivo README muy pequeño con pocas instrucciones sobre los prerrequisitos del sistema y pocas instrucciones sobre la instalación y ejecución. También se encuentra disponible para su descarga en el sitio web una captura ejemplo.

Esta herramienta no usa el modelo cliente-servidor. Sólo se necesita un computador para realizar los experimentos. Este computador captura el tráfico de una red IEEE 802.11 y reporta resultados obtenidos en esa captura.

Para los experimentos, el usuario puede utilizar una captura previa, o iniciar una captura en vivo. Esta segunda opción obliga al usuario a detener la captura para poder observar los resultados, que se despliegan en modo texto y gráficamente. Los reportes finales incluyen un

[23] http://www.wireshark.org

conteo de tramas, conteo de bytes, duración de la captura, throughput, información detallada de cada paquete (protocolos, longitud, direcciones, etc.), y distribución del ancho de banda.

WlanTV se encuentra disponible para plataformas Linux y Windows. La herramienta reporta estadísticas para protocolos IEEE 802.11.

6.2.8 Benchmarking Comms1

Es una herramienta de benchmark, desarrollada originalmente por Velásquez y Gamess [81]. Permite reportar el OWD, sin necesidad de utilizar otra herramienta para la sincronización del reloj de los computadores (difícil de lograr al nivel de microsegundos).

Durante el cálculo del OWD, un paquete (IPv4 o IPv6) de longitud fija se intercambia entre el cliente y el servidor un número de veces establecido por el usuario (mediante un archivo de texto). La herramienta lleva una marca de tiempo tomada antes y después del intercambio. La diferencia de las marcas de tiempo se dividen entre el número de paquetes que fue enviado y recibido para obtener un promedio del RTT, este resultado es dividido entre 2 para obtener el OWD promedio. De este modo se reduce el margen de error en los resultados. Los resultados se muestran por defecto en la salida estándar, pero pueden ser redirigidos a un archivo para ser procesados posteriormente.

Benchmarking Comms1 fue diseñada siguiendo el paradigma cliente-servidor y escrita en el lenguaje C. Está disponible para plataformas Unix y Windows. Tanto IPv4 como IPv6 son soportados como protocolo de capa de red;. UDP y TCP son usados como protocolo de capa de transporte. Su ejecución se inicia vía consola usando diferentes opciones provistas por la herramienta.

7. Experimentos y Análisis de Resultados

En este capítulo se presentan las herramientas y técnicas seleccionadas para realizar el estudio, se describen los escenarios y experimentos diseñados para la evaluación del rendimiento de los mecanismos de transición IPv4-IPv6 y se realiza un análisis los resultados obtenidos.

Los mecanismos de transición IPv4-IPv6 seleccionados para la evaluación son: ISATAP, 6to4 y NAT64 los cuales serán comparados entre sí y con IPv4 e IPv6 nativo. Las versiones de IP y los mecanismos de transición IPv4-IPv6 que se estudiarán tienen diferentes características, teorías operativas y disponibilidad dependiente del entorno de red en que serán desplegados. IPv4 e IPv6 se estudiarán de forma nativa. ISATAP y 6to4 son técnicas de túnel, mientras que NAT64 es una técnica de traducción. Por ello, es de suma importancia realizar evaluaciones del desempeño de cada una de estas tecnologías en escenarios lo más realistas posibles y que generen resultados significativos para el posterior uso en redes de gran escala.

Para los diferentes escenarios se instalarán diversos sistemas operativos. De la línea de sistemas operativos producida por Microsoft Corporation se escogieron las versiones Windows 7, Windows 8.1, y Windows 10.De software libre se escogió Linux Debían 7, kernel 3.2.0-4-amd64. Las implementaciones de NAT64 solo están disponibles en Linux, por lo cual, esta tecnología solo será evaluada sobre Debian 7.

Para hacer una evaluación de desempeño equitativa entre las diferentes tecnologías es necesario que las configuraciones de los escenarios sean lo más similares posibles. En general, se tienen 3 PCs conectados point-to-point usando cable cross-over, con las siguientes características de hardware y software: procesador Intel® Core™ 2 Duo CPU E6750 @ 2.66GHz 2.67GHz, 8 GB de RAM, un Broadcom NeXtreme Gigabit Ethernet para cliente (PC1) y servidor (PC2) y dos para el dispositivo intermedio (R1 o T1 según sea el caso). Debido a que solo se permiten 4 particiones primarias en cada PC y el sistema operativo Debian 7, utiliza 2 de ellas para su correcta instalación; cada PC tenía un grub con tres entradas, la primera para el sistema operativo Windows 7, la segunda para Debian 7, y la última para Windows 8.1, el cual fue desinstalado una vez finalizadas las pruebas necesarias en él, para instalar Windows 10.

Las métricas de benchmarking seleccionadas fueron throughput (Sección 6.1.4) y One Way Delay (OWD, Sección 6.1.1). Los resultados obtenidos del cálculo de estas dos métricas permitirán hacer un análisis comparativo entre los mecanismos de transición IPv4-IPv6 con el fin de concluir cual obtiene mejor desempeño, aportando información relevante a los investigadores, los administradores y los usuarios finales, en el área de evaluación de desempeño en redes de computadores.

```
Comandos de consola:
01: <./comms1-tcp-v4-unix|comms1-tcp-v4-windows.exe> client <archivoDatos><direccionServidor><puertoServidor>
02: <./comms1-tcp-v4-unix|comms1-tcp-v4-windows.exe> server <puerto>
03: <./comms1-udp-v4-unix|comms1-udp-v4-windows.exe> client <archivoDatos><direccionServidor><puertoServidor>
04: <./comms1-udp-v4-unix|comms1-udp-v4-windows.exe> server <puerto>
05: <./comms1-tcp-v6-unix|comms1-tcp-v6-windows.exe> client <archivoDatos><direccionServidor><puertoServidor>
06: <./comms1-tcp-v6-unix|comms1-tcp-v6-windows.exe> server <puerto>
07: <./comms1-udp-v6-unix|comms1-udp-v6-windows.exe> client <archivoDatos><direccionServidor><puertoServidor>
08: <./comms1-udp-v6-unix|comms1-udp-v6-windows.exe> server <puerto>
```

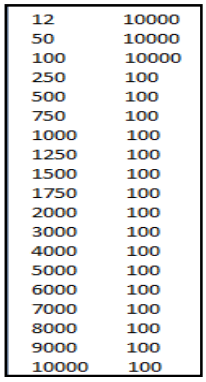
Figura 7.1: Comandos Utilizados para el Cálculo del OWD con el Benchmarks Comms1

Para el cálculo del OWD, fue seleccionada la herramienta de Benchmarking Comms1 (Sección 6.2.8). La ejecución de esta herramienta se inicia vía consola (no se ofrece una GUI). Los comandos propuestos, en forma general, para los diferentes experimentos son los especificados en la Figura 7.1 en donde:

- **`./comms1-tcp-v4-unix y ./comms1-udp-v4-unix:`** son los demonios de la herramienta en Linux para TCP y UDP respectivamente, utilizando el protocolo IPv4.
- **`./comms1-tcp-v6-unix y ./comms1-udp-v6-unix:`** son los demonios de la herramienta en Linux para TCP y UDP respectivamente, utilizando el protocolo IPv6.
- **`comms1-tcp-v4-windows.exe y comms1-udp-v4-windows.exe:`** son los ejecutables de la herramienta en Windows para TCP y UDP respectivamente, utilizando el protocolo IPv4.
- **`comms1-tcp-v6-windows.exe y comms1-udp-v6-windows.exe:`** son los ejecutables de la herramienta en Windows para TCP y UDP respectivamente, utilizando el protocolo IPv6.
- **`client:`** es la palabra reservada para indicar que es el cliente.
- **`server:`** es la palabra reservada para indicar que es el servidor.
- **`<archivoDatos>:`** es el archivo que contiene los valores de carga útil y número de repeticiones de los datos a generar en el experimento. El archivo de datos puede contener una o más líneas con el siguiente formato:

  `<tamañoCargaUtil><cantidadRepeticiones>.`

  La carga útil debe ser especificada en bytes y estar ordenada de forma ascendente; para el correcto funcionamiento de la herramienta de Benchmarking Comms1, el archivo de datos en Windows debe de culminar con una línea en blanco, sin embargo, en Debian 7 la línea al final no debe aparecer. La Figura 7.2 muestra el contenido del archivo de datos: “datos.dat”, que será el mismo para todos los experimentos realizados tanto en Windows como en Debian 7.

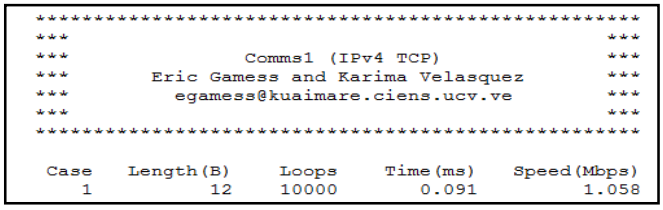


12	10000
50	10000
100	10000
250	100
500	100
750	100
1000	100
1250	100
1500	100
1750	100
2000	100
3000	100
4000	100
5000	100
6000	100
7000	100
8000	100
9000	100
10000	100

Figura 7.2: Formato de Archivo datos.dat

- **<direccionServidor>:** es la dirección IPv4 o IPv6, según sea el caso, del servidor.
- **<puertoServidor> y <puerto>:** es puerto de escucha del servidor.

La herramienta Benchmarks Comms1 reporta resultados con el formato especificado en la Figura 7.3, en donde el valor generado en la columna “Time (ms)” es el OWD del experimento en microsegundos.



```
**************************************************
***                                            ***
***              Comms1 (IPv4 TCP)             ***
***       Eric Gamess and Karima Velasquez     ***
***         egamess@kuaimare.ciens.ucv.ve      ***
***                                            ***
**************************************************

 Case    Length(B)     Loops     Time(ms)    Speed(Mbps)
    1           12     10000        0.091          1.058
```

Figura 7.3: Formato de Resultados Reportados por la Herramienta Benchmarks Comms1

Para calcular el throughput, fue seleccionada la herramienta Iperf (Sección 6.2.1), debido a su flexibilidad en el uso de las versiones IP en cualquiera de los extremos (cliente o servidor). El throughput solo será calculado para el protocolo de transporte UDP, debido a que la herramienta no permite la variación de la carga útil para el protocolo de transporte TCP.

Los comandos propuestos a introducir, en forma general, para los diferentes experimentos son los especificados en la Figura 7.4 en donde:

- **-u:** indica que el protocolo de transporte a utilizar es UDP.
- **-c:** indica que la herramienta se ejecutará en modo cliente.
- **<direccionIPservidor>:** es la dirección IPv4 o IPv6 del servidor.
- **-t 20:** indica que los paquetes sean enviado durante 20 segundos.
- **-i 2:** indica a la herramienta que los reportes de resultados se generen en intervalos de 2 segundos.
- **-b:** indica el tamaño del buffer de lectura y escritura.
- **-l:** indica el tamaño de la carga útil enviado en cada paquete. La carga varía entre 12 bytes y 10000 bytes en el cliente y se mantiene 10000 bytes en el servidor.

- **-V:** indica que la versión del protocolo IP a utilizar es la 6 (IPv6). Iperf permite que el cliente se encuentre en una red IPv4 y el servidor en una red IPv6 y viceversa, en estos casos se debe especificar explícitamente la opción en el extremo que sea necesario.

```
Comandos de consola:
01: iperf -u -c <direccionIPservidor> -t 20 -i 2 -b 100Mbps -l <carga>
02: iperf -u -s –l 10000
03: iperf –V -u -c <direccionIPservidor> -t 20 -i 2 -b 100Mbps -l <carga>
04: iperf –V -u -s –l 10000
```

Figura 7.4: Comandos Utilizados para el Cálculo del Throughput con Iperf

La herramienta Iperf reporta resultados con el formato especificado en la Figura 7.5, en donde el valor generado en la columna “Bandwidth” es el throughput del experimento en megabit por segundo (Mbps).

```
------------------------------------------------------------
Client connecting to 192.168.255.111, UDP port 5001
Sending 250 byte datagrams
UDP buffer size:  224 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.2 port 57093 connected with
192.168.255.111 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec  18.9 MBytes  79.1 Mbits/sec
[  3]  2.0- 4.0 sec  18.8 MBytes  79.0 Mbits/sec
[  3]  4.0- 6.0 sec  18.8 MBytes  79.0 Mbits/sec
[  3]  6.0- 8.0 sec  18.8 MBytes  78.9 Mbits/sec
[  3]  8.0-10.0 sec  18.8 MBytes  79.0 Mbits/sec
[  3] 10.0-12.0 sec  18.8 MBytes  79.0 Mbits/sec
[  3] 12.0-14.0 sec  18.8 MBytes  79.0 Mbits/sec
[  3] 14.0-16.0 sec  18.8 MBytes  79.0 Mbits/sec
[  3] 16.0-18.0 sec  18.8 MBytes  79.0 Mbits/sec
[  3] 18.0-20.0 sec  18.8 MBytes  79.0 Mbits/sec
[  3]  0.0-20.0 sec   188 MBytes  79.0 Mbits/sec
[  3] Sent 790036 datagrams
[  3] Server Report:
[  3]  0.0-20.0 sec   177 MBytes  74.3 Mbits/sec   0.151 ms
46088/790035 (5.8%)
[  3]  0.0-20.0 sec  1 datagrams received out-of-order
```

Figura 7.5 Formato de Resultados Reportados por la Herramienta Iperf

En consecuencia, para las pruebas de evaluación del rendimiento de los mecanismos de transición IPv4-IPv6 (IPv4 nativo, IPv6 nativo, ISATAP, 6to4 y NAT64) se realizará el cálculo del OWD para los protocolos de transporte TCP y UDP con la herramienta Benchmarks Comms1. El throughput será calculado utilizando Iperf solo para UDP, en donde se trasmitirán paquetes IPv4, IPv6, o IPv6 en IPv4, durante 20 segundos para cada tamaño de carga útil, en cada uno de los escenarios diseñados tanto para Ethernet (10 Mbps) como para Fast Ethernet (100 Mbps).

Los paquetes se enviarán desde el cliente hasta el servidor con 19 tamaños de carga útil en bytes especificados en la siguiente lista: 12, 50, 100, 250, 500, 750, 1000, 1250, 1500, 1750, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, y 10000; de los cuales 8 no presentan fragmentación. Los dispositivos intermedios son PCs con Debian 7 o Windows con enrutamiento activo y al menos dos tarjetas de red, configurados como routers o como traductor, según sea el caso.

Una vez seleccionadas las herramientas a utilizar, las métricas a calcular y la cantidad de dispositivos que estarán presente en la redes de prueba, junto con sus características de hardware y software, se procedió a crear escenarios específicos para cada tecnología y poder realizar la evaluación del rendimiento mediante los resultados reportados en cada uno

de ellos. Cabe resaltar que para verificar el correcto funcionamiento de cada escenario se utilizó el analizador de protocolos Wireshark[24]. La Figura 7.6 muestra el esqueleto general de los escenarios de pruebas diseñados.

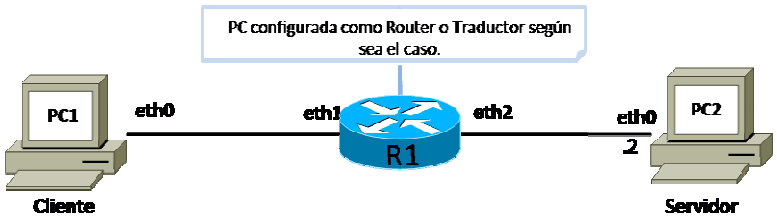


Figura 7.6: Esqueleto General de los Escenarios de Pruebas

7.1 Escenario 1: IPv4 Nativo

La Figura 7.7 muestra el escenario diseñado para realizar los cálculos del OWD y throughput sobre IPv4 nativo. Es una red IPv4 privada, con las siguientes características: 3 PCs conectados point-to-point mediante cross-over, donde PC1 es cliente, PC2 es el servidor y R1 es configurado como enrutador de paquetes.

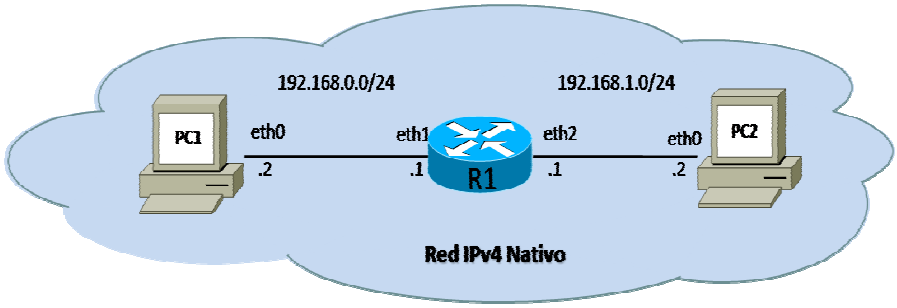


Figura 7.7: Escenario 1, IPv4 Nativo

A continuación se muestra los procedimientos a seguir para realizar la configuración del escenario 1 en Windows 7, Windows 8.1 y Windows 10:

- **Configuración del PC1:** acceder a las propiedades TCP/IPv4 de la interfaz eth0 (adaptador de red “Conexión de área local”), y modificar los campos de la siguiente manera: Dirección IP: 192.168.0.2, Máscara de subred: 255.255.255.0, Puerta de enlace predeterminada: 192.168.0.1.
- **Configuración del PC2:** acceder a las propiedades TCP/IPv4 de la interfaz eth0 (adaptador de red “Conexión de área local”), y modificar los campos de la siguiente manera: Dirección IP: 192.168.1.2, Máscara de subred: 255.255.255.0, Puerta de enlace predeterminada: 192.168.1.1.
- **Configuración del R1:** acceder a las propiedades TCP/IPv4 de la interfaz eth0 (adaptador de red “Conexión de área local 1”), y modificar los campos de la siguiente manera: Dirección IP: 192.168.0.1, Máscara de subred: 255.255.255.0. Proceder de igual forma con la interfaz eth2 (adaptador de red “Conexión de área local 2”) y modificar los campos: Dirección IP: 192.168.1.1, Mascara de subred: 255.255.255.0.

[24] https://www.wireshark.org

Para realizar la configuración del escenario en Debian 7, debe ubicarse el archivo `/etc/network/interfaces` y modificarlo según lo especificado en la Figura 7.8.

PC1:	PC2:	R1:
Archivo /etc/network/interfaces 01: auto eth0 02: iface eth0 inet static 03: address 192.168.0.2 04: netmask 255.255.255.0 05: gateway 192.168.0.1	Archivo /etc/network/interfaces 01: auto eth0 02: iface eth0 inet static 03: address 192.168.1.2 04: netmask 255.255.255.0 05: gateway 192.168.1.1	Archivo /etc/network/interfaces 01: auto eth1 02: iface eth1 inet static 03: address 192.168.0.1 04: netmask 255.255.255.0 05: auto eth2 06: iface eth2 inet static 07: address 192.168.1.1 08: netmask 255.255.255.0

Figura 7.8: Configuración de PC1, PC2 y R1 para IPv4 Nativo en Debian 7

Tanto en Windows como en Debian 7 el dispositivo R1 será el enrutador de paquetes en la red. Por ello, se debe realizar el proceso de activación del forwarding IPv4 en este dispositivo (ver procedimiento en la Sección 7.6.1). Después de modificar las interfaces, reiniciar los PCs o ejecutar los siguientes comandos para que procedan los cambios en las interfaces: `/etc/init.d/network-manager stop` y posteriormente: `/etc/init.d/network-manager start`.

Posteriormente, se procedió a realizar las diferentes pruebas de rendimiento. Durante el proceso de medición del throughput, tanto en Windows como en Debian 7, se ejecutó en PC2 el generador de tráfico en modo servidor: `iperf -u –s –l 10000`. La opción `–l` se estableció en 10000 ya que es la máxima carga útil que será inyectada durante los experimentos.

Una vez el servidor este escuchando peticiones se ingresó en PC1 para Ethernet el comando: `iperf -u -c 192.168.1.2 -t 20 -i 2 -b 10Mbps -l <TamCargaUtil>`, y para Fast Ethernet se utilizó: `iperf -u -c 192.168.1.2 -t 20 -i 2 -b 100Mbps -l <TamCargaUtil>`, en donde, `<TamCargaUtil>` es un entero que indica el tamaño de la carga útil a generar. Todos los experimentos se repitieron de 10 a 20 veces para generar resultados más consistentes.

Para calcular el OWD en Debian 7, con el protocolo de transporte TCP, se utilizó el formato de archivo de datos especificado en la Figura 7.2. Se ejecutó la herramienta Benchmarking Comms1 en modo servidor en PC2 con el comando: `./comms1-tcp-v4-unix server 5001`, y en modo cliente en PC1 de la siguiente manera: `./comms1-tcp-v4-unix client datos.dat 192.168.1.2 5001`.

Para calcular el OWD en Windows con el protocolo de transporte TCP, en PC2 se utilizó el comando: `comms1-tcp-v4-windows.exe server 5001`, y en PC1: `comms1-tcp-v4-windows.exe client datos.dat 192.168.1.2 5001`.

Para calcular el OWD en Debian 7, con el protocolo de transporte UDP, también se utilizó el formato de archivo de datos especificado en la Figura 7.2. Se ejecutó la herramienta Benchmarking Comms1 en modo servidor en PC2 con el siguiente comando: `./comms1-`

`udp-v4-unix server 5001`, y en modo cliente en PC1 de la siguiente manera: `./comms1-udp-v4-unix client datos.dat 192.168.1.2 5001`.

Para calcular el OWD en Windows con el protocolo de transporte UDP, en PC2 se utilizó el comando: `comms1-udp-v4-windows.exe server 5001`, y en PC1: `comms1-udp-v4-windows.exe client datos.dat 192.168.1.2 5001`.

Estos experimentos se realizaron tanto para Ethernet como en Fast Ethernet. Para realizar el cambio de ancho de banda se utilizó el procedimiento mostrado en la Sección 7.6.2.

7.2 Escenario 2: IPv6 Nativo

La Figura 7.9 muestra el escenario diseñado para realizar los cálculos del OWD y Throughput sobre IPv6 nativo. Es una red IPv6 privada, con las siguientes características: 3 PCs conectados point-to-point mediante cross-over, donde PC1 es el host cliente, PC2 es el host servidor y R1 es configurado como enrutador de paquetes.

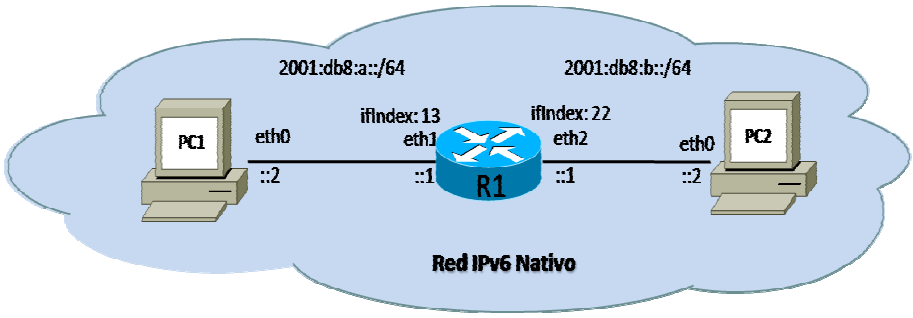


Figura 7.9: Escenario 2, IPv6 Nativo

A continuación se muestra los procedimientos a seguir para realizar la configuración del escenario 2 en Windows 7, Windows 8.1 y Windows 10:

- **Configuración del PC1:** acceder a las propiedades TCP/IPv6 de la interfaz eth0 (adaptador de red "Conexión de área local"), y modificar los campos de la siguiente manera: Dirección IPv6 : 2001:db8:a::2, Longitud del prefijo de subred: 64, Puerta de enlace predeterminada: 2001:db8:a::1.
- **Configuración del PC2:** acceder a las propiedades TCP/IPv6 de la interfaz eth0 (adaptador de red "Conexión de área local"), y modificar los campos de la siguiente manera: Dirección IPv6 : 2001:db8:b::2, Longitud del prefijo de subred: 64, Puerta de enlace predeterminada: 2001:db8:b::1.
- **Configuración del R1:** acceder a las propiedades TCP/IPv6 de la interfaz eth0 (adaptador de red "Conexión de área local 1"), y modificar los campos de la siguiente manera: Dirección IPv6 : 2001:db8:a::1, Longitud del prefijo de subred: 64. Proceder de igual forma con la interfaz eth2 (adaptador de red "Conexión de área local 2") y modificar los campos: Dirección IPv6 : 2001:db8:b::1, Longitud del prefijo de subred: 64.

Para realizar la configuración del escenario en Debian 7, se debe ubicar el archivo `/etc/network/interfaces` y modificarlo según lo especificado en la Figura 7.10.

PC1:	PC2:	R1:
Archivo /etc/network/interfaces 01: auto eth0 02: iface eth0 inet6 static 03: address 2001:db8:a::2 04: netmask 64 05: gateway 2001:db8:a::1	Archivo /etc/network/interfaces 01: auto eth0 02: iface eth0 inet6 static 03: address 2001:db8:b::2 04: netmask 64 05: gateway 2001:db8:b::1	Archivo /etc/network/interfaces 01: auto eth1 02: iface eth1 inet6 static 03: address 2001:db8:a::1 04: netmask 64 05: auto eth2 06: iface eth2 inet6 static 07: address 2001:db8:b::1 08: netmask 64

Figura 7.10: Configuración de PC1, PC2 y R1 para IPv6 Nativo en Debian 7

Tanto en Windows como en Debian 7 el dispositivo R1 será el enrutador de paquetes en la red. Por ello, se debe realizar el proceso de activación del forwarding IPv6 en este dispositivo (ver procedimiento en la Sección 7.6.1).

Después de modificar las interfaces, se debe reiniciar los PCs o ejecutar los siguientes comandos para que procedan los cambios en las interfaces: `/etc/init.d/network-manager stop` y `/etc/init.d/network-manager start`.

Posteriormente, se procedieron a realizar las diferentes pruebas de rendimiento con las herramientas Iperf y Benchmarking Comms1. Durante el proceso de medición del throughput, tanto en Windows como en Debian 7, primero se ejecutó en PC2 el generador de tráfico en modo servidor: `iperf –V -u –s –l 10000`. La opción `–l` se estableció en 10000 ya que es la máxima carga útil que será inyectada durante los experimentos. Una vez el servidor este escuchando peticiones se ingresó el siguiente comando: `iperf –V -u -c 2001:db8:b::2 -t 20 -i 2 -b 10Mbps -l <TamCargaUtil>`, en el PC1 para Ethernet y para Fast Ethernet se utilizó: `iperf –V -u -c 2001:db8:b::2 -t 20 -i 2 -b 100Mbps -l <TamCargaUtil>`, en donde `<TamCargaUtil>` es un entero que indica el tamaño de la carga útil a generar y `–V` indica que se utilizará IPv6. Todos los experimentos se repitieron de 10 a 20 veces para generar resultados más consistentes.

Para calcular el OWD en Debian 7, con el protocolo de transporte TCP, se utilizó el formato de archivo de datos especificado en la Figura 7.2. Se ejecutó la herramienta Benchmarking Comms1 en modo servidor en PC2: `./comms1-tcp-v6-unix server 5001`, y en modo cliente en PC1: `./comms1-tcp-v6-unix client datos.dat 2001:db8::2 5001.`

Para calcular el OWD en Windows con el protocolo de transporte TCP, en PC2 se utilizó el comando: `comms1-tcp-v6-windows.exe server 5001`, y en PC1: `comms1-tcp-v6-windows.exe client datos.dat 2001:db8:b::2 5001`.

Para calcular el OWD en Debian 7, con el protocolo de transporte UDP, también se utilizó el formato de archivo de datos especificado en la Figura 7.2. Se ejecutó la herramienta Benchmarking Comms1 en modo servidor en PC2 con el comando: `./comms1-udp-v6-unix server 5001`, y en modo cliente en PC1 de la siguiente manera: `./comms1-udp-v6-unix client datos.dat 2001:db8::2 5001`.

Para calcular el OWD en Windows con el protocolo de transporte UDP, en PC2 se utilizó el comando: `comms1-udp-v6-windows.exe server 5001`, y en PC1: `comms1-udp-v6-windows.exe client datos.dat 2001:db8:b::2 5001`.

Estos experimentos se realizaron tanto para Ethernet como en Fast Ethernet. Para realizar el cambio de ancho de banda se utilizó el procedimiento mostrado en la Sección 7.6.2.

7.3 Escenario 3: ISATAP

El escenario ISATAP es una red que conecta un sitio ISATAP con una red IPv6 nativa. Este escenario presenta las siguientes características: 3 PCs conectados point-to-point mediante cross-over, donde PC1 es un host dual-stack que actúa como el cliente y tiene activa la interfaz ISATAP (host ISATAP), PC2 es un host IPv6 nativo que actúa como servidor y R1 es el router ISATAP el cual tiene la interfaz ISATAP y posee el enrutamiento activo. El prefijo de la red ISATAP es 2001:db8:fea::/64.

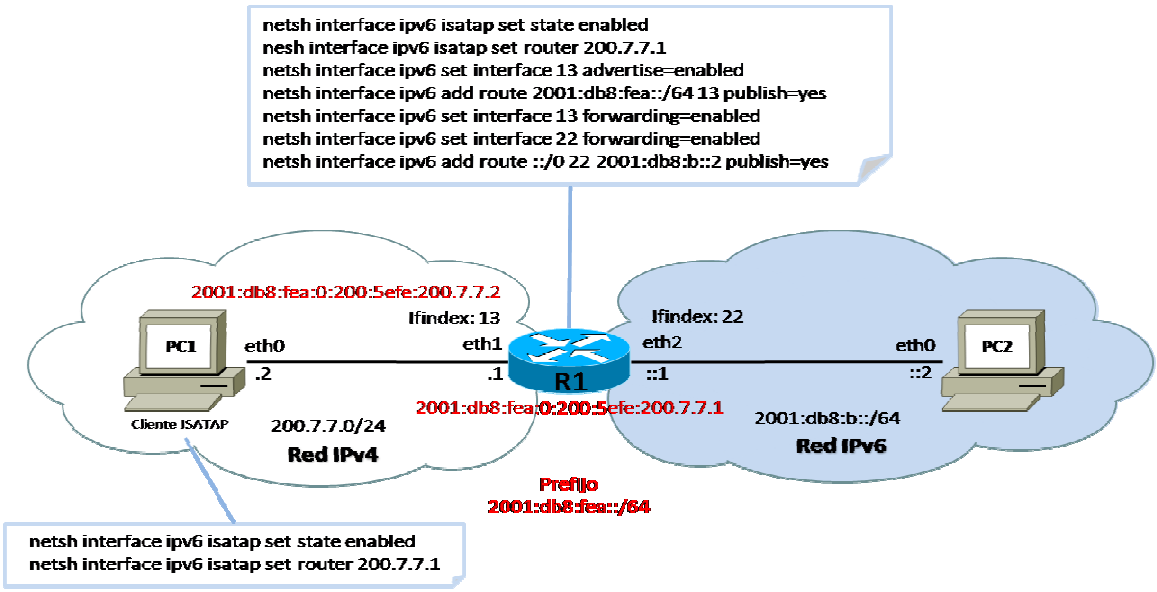


Figura 7.11: Escenario 3, ISATAP (Windows)

A continuación se muestran los procedimientos a seguir para realizar la configuración del escenario 3 en Windows 7, Windows 8.1 y Windows 10 (Figura 7.11):

- **Configuración del PC1:** acceder a las propiedades TCP/IPv4 de la interfaz eth0 (adaptador de red "Conexión de área local"), y modificar los campos de la siguiente manera: Dirección IP: 200.7.7.2, Máscara de subred: 255.255.255.0, Puerta de enlace predeterminada: 200.7.7.1.
- **Configuración del PC2:** acceder a las propiedades TCP/IPv6 de la interfaz eth0 (adaptador de red "Conexión de área local"), y modificar los campos de la siguiente manera: Dirección IPv6: 2001:db8:b::2, Longitud del prefijo de subred: 64, Puerta de enlace predeterminada: 2001:db8:b::1.
- **Configuración del R1:** acceder a las propiedades TCP/IPv4 de la interfaz eth0 (adaptador de red "Conexión de área local 1"), y modificar los campos de la siguiente

manera: Dirección IP: 200.7.7.1, Máscara de subred: 255.255.255.0. Acceder a las propiedades TCP/IPv6 de la interfaz eth2 (adaptador de red "Conexión de área local 2") y modificar los campos: Dirección IPv6 : 2001:db8:b::1, Longitud del prefijo de subred: 64.

Una vez configuradas las interfaces de red, en PC1 se debe realizar la activación de la interfaz ISATAP y asignar el router ISATAP de la red. Para ello se debe acceder a la consola de Windows e ingresar: `netsh interface ipv6 isatap set state enabled` (para activar la interfaz ISATAP) y `netsh interface ipv6 isatap set router 200.7.7.1` (para asignar el router ISATAP de la red).

De igual forma, en R1 se debe realizar la activación de la interfaz ISATAP y realizar las configuraciones necesarias para que R1 actúe como router ISATAP. En este caso se debe ingresar: `netsh interface ipv6 isatap set state enabled` (para activar la interfaz ISATAP), `netsh interface ipv6 isatap set router 200.7.7.1` (para asignar el router ISATAP de la red), `netsh interface ipv6 set interface 13 advertise=enabled` (para activar el envío de mensajes router advertisement por la interfaz), `netsh interface ipv6 add route 2001:db8:fea::/64 13 publish=yes` (para asignar prefijo ISATAP), `netsh interface ipv6 set interface 13 forwarding=enabled` (para activar el forwarding IPv6 en eth1), `netsh interface ipv6 set interface 22 forwarding=enabled` (para activar el forwarding IPv6 en eth2) y `netsh interface ipv6 add route ::/0 22 2001:db8:b::2 publish=yes` (para asignar una ruta estática a la red IPv6).

Por último, ubicar el archivo `/etc/network/interfaces` y modificarlo según lo especificado en la Figura 7.12 y la Figura 7.13 para realizar la configuración del escenario en Debian 7 (Figura 7.14).

```
R1:
Archivo /etc/network/interfaces
01: auto eth1
02: iface eth1 inet static
03: address 192.168.0.1
04: netmask 255.255.255.0
05: auto eth2
06: iface eth2 inet6 static
07: address 2001:db8:b::1
08: netmask 64
09: #Configurar interfaz de túnel ISATAP is0
10: up ip tunnel add is0 mode isatap local 192.168.0.1
11: #Levantar la interfaz is0
12: up link set is0 up
13: #asignar la dirección ISATAP de forma estática a la interfaz
14: up -6 addr add 2001:db8:fea::5efe:192.168.0.1 dev is0
```

Figura 7.12: Configuración de R1 para ISATAP en Debian 7

Adicionalmente en PC1 se debe instalar el demonio del cliente `isatapd`[25] descargándolo de su sitio oficial o ingresando el siguiente comando vía consola: `apt-get install isatapd`. Una vez modificadas las interfaces, reiniciar los PCs o ejecutar los siguientes comandos para

[25]http://www.saschahlusiak.de/linux/isatap.htm

que procedan los cambios en las interfaces: `/etc/init.d/network-manager stop` y posteriormente: `/etc/init.d/network-manager start`.

PC1:	PC2:
Archivo /etc/network/interfaces 01: auto eth0 02: iface eth0 inet static 03: address 192.168.0.2 04: netmask 255.255.255.0 05: gateway 192.168.0.1 06: up isatapd -d -l eth0 -r 192.168.0.1 07: up ip -6 addr add 2001:db8:fea::5efe:192.168.0.2/64 dev is0	Archivo /etc/network/interfaces 01: auto eth0 02: iface eth0 inet6 static 03: address 2001:db8:b::2 04: netmask 64 05: gateway 2001:db8:b::1

Figura 7.13: Configuración de PC1 y PC2 para ISATAP en Debian 7

Es conveniente resaltar, que en Windows se utilizó direcciones IPv4 privadas y en Debian 7 direcciones IPv4 públicas con el fin de mostrar el formato de direcciones ISATAP generadas en cada caso.

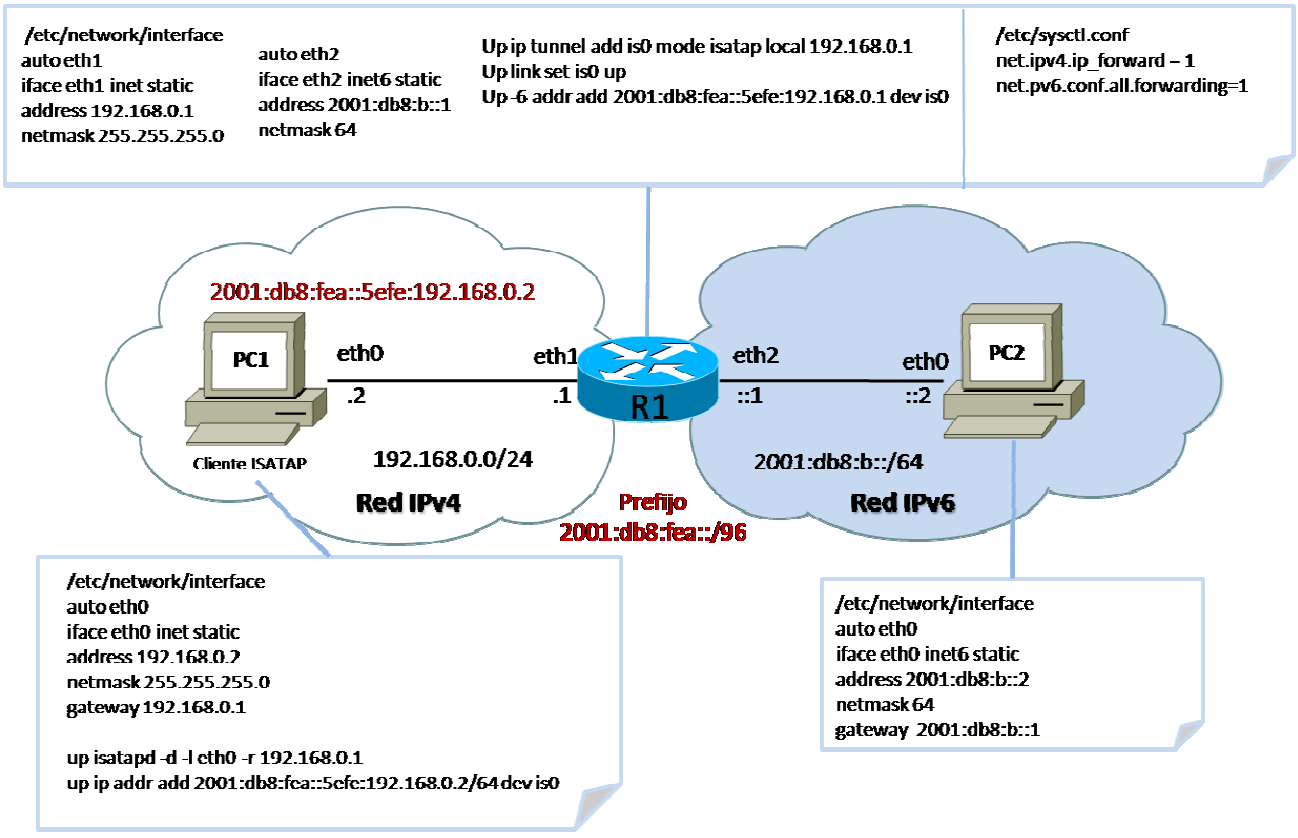


Figura 7.14: Escenario 3, ISATAP (Debian 7)

Después de tener el escenario configurado, se procedieron a realizar las diferentes pruebas de rendimiento con las herramientas Iperf y Benchmarking Comms1.

A pesar de que las configuraciones de los escenarios son diferentes en cada sistema operativo, durante el proceso de medición del throughput, tanto en Windows como en Debian

7, se ejecutaron los mismos comandos. Primero se ejecutó en PC2 el generador de tráfico en modo servidor: `iperf –V -u –s –l 10000`. La opción `–l` se estableció en 10000 ya que es la máxima carga útil que será inyectada durante los experimentos. Una vez el servidor este escuchando peticiones se ingresó el siguiente comando en PC1 para Ethernet: `iperf –V -u -c 2001:db8:b::2 -t 20 -i 2 -b 10Mbps -l <TamCargaUtil>`, y para Fast Ethernet se utilizó: `iperf –V -u -c 2001:db8:b::2 -t 20 -i 2 -b 100Mbps -l <TamCargaUtil>`, en donde, `<TamCargaUtil>` es un entero que indica el tamaño de la carga útil a generar y `–V` indica que se utilizará IPv6. Todos los experimentos se repitieron de 10 a 20 veces para generar resultados más consistentes.

Para calcular el OWD en Debian 7, con el protocolo de transporte TCP, se utilizó el formato de archivo de datos especificado en la Figura 7.2. Se ejecutó la herramienta Benchmarking Comms1 en modo servidor en PC2 con el siguiente comando: `./comms1-tcp-v6-unix server 5001`, y en modo cliente en PC1 de la siguiente manera: `./comms1-tcp-v6-unix client datos.dat 2001:db8::2 5000`.

Para calcular el OWD en Windows, con el protocolo de transporte TCP, en PC2 se utilizó el comando: `comms1-tcp-v6-windows.exe server 5001`, y en PC1: `comms1-tcp-v6-windows.exe client datos.dat 2001:db8:b::2 5000`.

Para calcular el OWD en Debian 7, con el protocolo de transporte UDP, también se utilizó el formato de archivo de datos especificado en la Figura 7.2. Se ejecutó la herramienta Benchmarking Comms1 en modo servidor en PC2 con el siguiente comando: `./comms1-udp-v6-unix server 5001`, y en modo cliente en PC1 de la siguiente manera: `./comms1-udp-v6-unix client datos.dat 2001:db8::2 5001`.

Para calcular el OWD en Windows, con el protocolo de transporte UDP, en PC2 se utilizó el comando: `comms1-udp-v6-windows.exe server 5001`, y en PC1: `comms1-udp-v6-windows.exe client datos.dat 2001:db8:b::2 5001`.

Estos experimentos se realizaron tanto para Ethernet como en Fast Ethernet. Para realizar el cambio de ancho de banda se utilizó el procedimiento mostrado en la Sección 7.6.2.

7.4 Escenario 4: 6to4

El escenario 6to4 es una red donde se conectan dos sitios 6to4 con una red IPv4 nativa. Este escenario presenta las siguientes características: 3 PCs conectados point-to-point mediante cross-over, donde PC1 es un host dual-stack con interfaz 6to4 activa que actúa como el cliente, PC2 también con interfaz 6to4 activa es el servidor y forma el otro sitio 6to4, y R1 que constituye la red IPv4 y es configurado como enrutador de paquetes.

Las direcciones 6to4 para los escenarios diseñados fueron generadas automáticamente en cada sistema operativo, lo cuales toman diferentes criterios para la creación del identificador de interfaz. Mientras que Linux va incrementando desde la dirección “1”, Windows asume que es la misma dirección IPv4.

En consecuencia, el escenario diseñado para Windows y para Debian 7 tiene exactamente la misma topología pero varía en las direcciones IP generadas para cada sitio 6to4. Además, es conveniente resaltar que en Windows se utilizaron direcciones IPv4 privadas y en Debian 7 direcciones IPv4 públicas con el fin de mostrar el formato de direcciones ISATAP generadas en cada caso.

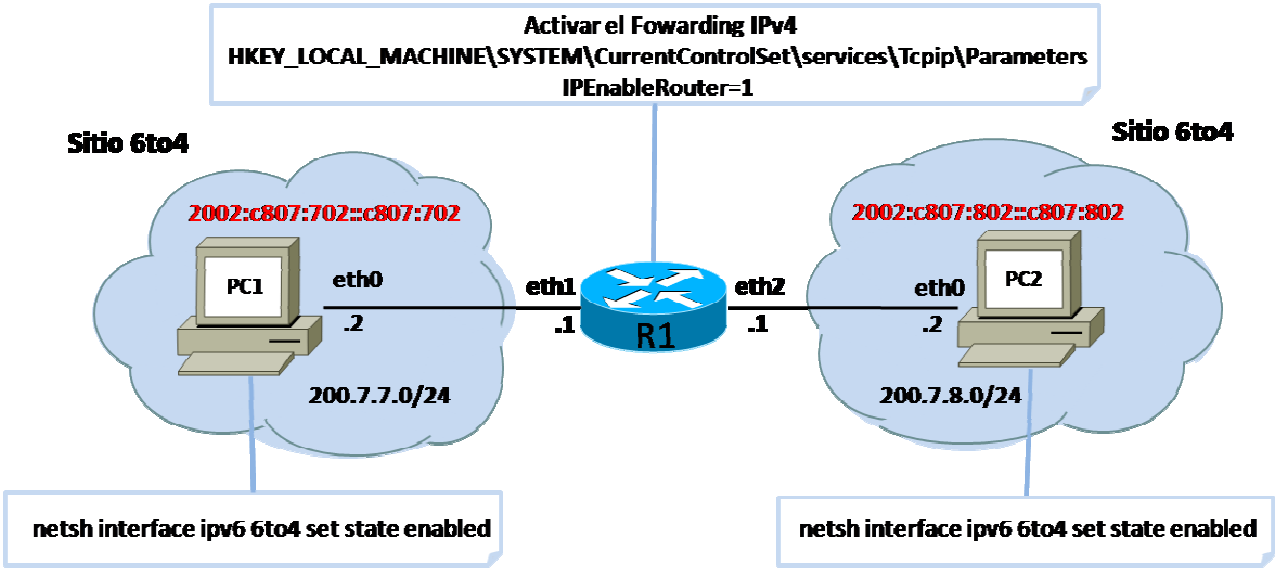


Figura 7.15: Escenario 4, 6to4 (Windows)

A continuación se muestran los procedimientos a seguir para realizar la configuración del escenario 4 en Windows 7, Windows 8.1 y Windows 10 (Figura 7.15):

- **Configuración del PC1:** acceder a las propiedades TCP/IPv4 de la interfaz eth0 (adaptador de red "Conexión de área local"), y modificar los campos de la siguiente manera: Dirección IP: 200.7.7.2, Máscara de subred: 255.255.255.0, Puerta de enlace predeterminada: 200.7.7.1.
- **Configuración del PC2:** acceder a las propiedades TCP/IPv4 de la interfaz eth0 (adaptador de red "Conexión de área local"), y modificar los campos de la siguiente manera: Dirección IP: 200.7.8.2, Máscara de subred: 255.255.255.0, Puerta de enlace predeterminada: 200.7.8.1.
- **Configuración del R1:** acceder a las propiedades TCP/IPv4 de la interfaz eth0 (adaptador de red "Conexión de área local 1"), y modificar los campos de la siguiente manera: Dirección IP: 200.7.7.1, Máscara de subred: 255.255.255.0. Proceder de igual forma con la interfaz eth2 (adaptador de red "Conexión de área local 2") y modificar los campos: Dirección IP: 200.7.8.1, Máscara de subred: 255.255.255.0.

Una vez configuradas las interfaces de red, en PC1 y PC2 se debe realizar la activación de la interfaz 6to4. Para ello se debe acceder a la consola de Windows e ingresar: `netsh interface ipv6 6to4 set state enabled`, para activar la interfaz 6to4 en ambos dispositivos.

Para realizar la configuración del escenario en Debian 7 (Figura 7.17), se debe ubicar el archivo `/etc/network/interfaces` y modificarlo según lo especificado en la Figura 7.16.

```
PC1:
Archivo /etc/network/interfaces
01: auto eth0
02: iface eth0 inet static
03: address 192.168.0.2
04: netmask 255.255.255.0
05: gateway 192.168.0.1
06: #túnel 6to4
07: auto 6to4
08: iface 6to4 inet6 static
09: local 192.168.0.2
```

```
PC2:
Archivo /etc/network/interfaces
01: auto eth0
02: iface eth0 inet static
03: address 192.168.0.2
04: netmask 255.255.255.0
05: gateway 192.168.0.1
06: #túnel 6to4
07: auto 6to4
08: iface 6to4 inet6 static
09: local 192.168.0.2
```

```
R1:
Archivo /etc/network/interfaces
01: auto eth1
02: iface eth1 inet static
03: address 192.168.0.1
04: netmask 255.255.255.0
05: auto eth2
06: iface eth2 inet static
07: address 192.168.1.1
08: netmask 255.255.255.0
```

Figura 7.16: Configuración de PC1, PC2 y R1 para 6to4 en Debian 7

Tanto en Windows como en Debian 7 el dispositivo R1 será el enrutador de paquetes en la red. Por ello, se debe realizar el proceso de activación del forwarding IPv4 en este dispositivo (ver procedimiento en la Sección 7.6.1). Después de modificar las interfaces, reiniciar los PCs o ejecutar los siguientes comandos para que procedan los cambios en las interfaces: `/etc/init.d/network-manager stop` y posteriormente: `/etc/init.d/network-manager start`.

Seguidamente se procedieron a realizar las diferentes pruebas de rendimiento con las herramientas Iperf y Benchmarking Comms1.

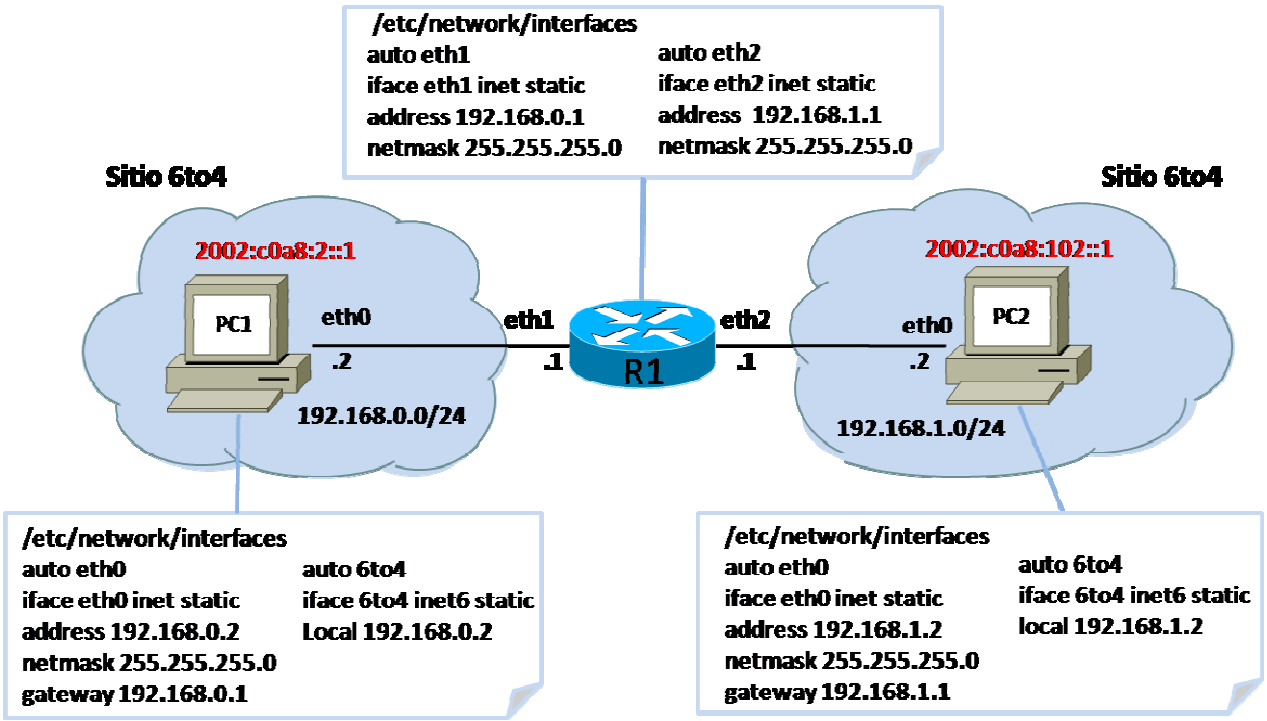


Figura 7.17: Escenario 4, 6to4 (Debian 7)

Durante el proceso de medición del throughput, tanto en Windows como en Debian 7, primero se ejecutó en PC2 el generador de tráfico en modo servidor: `iperf –V -u –s –l 10000`. La opción `–l` se estableció en 10000 ya que es la máxima carga útil que será inyectada durante los experimentos. Una vez el servidor este escuchando peticiones, en Windows se ingresó el siguiente comando: `iperf –V -u -c 2002:c807:802::c807:802 -t 20 -i 2 -b 10Mbps -l <TamCargaUtil>`, en el PC1 para Ethernet y para Fast Ethernet se utilizó: `iperf –V -u -c 2002:c807:802::c807:802 -t 20 -i 2 -b 100Mbps -l <TamCargaUtil>`, y en Debian 7 se ingresó el siguiente comando: `iperf –V -u -c 2002:c0a8:102::1 -t 20 -i 2 -b 10Mbps -l <TamCargaUtil>`, en el PC1 para Ethernet y para Fast Ethernet se utilizó: `iperf –V -u -c 2002:c0a8:102::1 -t 20 -i 2 -b 100Mbps -l <TamCargaUtil>`, en donde, `<TamCargaUtil>` es un entero que indica el tamaño de la carga útil a generar y `–V` indica que se utilizará IPv6. Todos los experimentos se repitieron de 10 a 20 veces para generar resultados más consistentes.

Para calcular el OWD en Debian 7, con el protocolo de transporte TCP, se utilizó el formato de archivo de datos especificado en la Figura 7.2. Se ejecutó la herramienta Benchmarking Comms1 en modo servidor en PC2 con el siguiente comando: `./comms1-tcp-v6-unix server 5001`, y en modo cliente en PC1 de la siguiente manera: `./comms1-tcp-v6-unix client datos.dat 2002:c0a8:102::1`.

Para calcular el OWD en Windows con el protocolo de transporte TCP, en PC2 se utilizó el comando: `comms1-tcp-v6-windows.exe server 5001`, y en PC1: `comms1-tcp-v6-windows.exe client datos.dat 2002:c807:802::c807:802 5001`.

Para calcular el OWD en Debian 7, con el protocolo de transporte UDP, también se utilizó el formato de archivo de datos especificado en la Figura 7.2. Se ejecutó la herramienta Benchmarking Comms1 en modo servidor en PC2 con el siguiente comando: `./comms1-udp-v6-unix server 5001`, y en modo cliente en PC1 de la siguiente manera: `./comms1-udp-v6-unix client datos.dat 2002:c0a8:102::1 5001`.

Para calcular el OWD en Windows con el protocolo de transporte UDP, en PC2 se utilizó el comando: `comms1-udp-v6-windows.exe server 5001`, y en PC1: `comms1-udp-v6-windows.exe client datos.dat 2002:c807:802::c807:802 5001`.

Estos experimentos se realizaron tanto para Ethernet como en Fast Ethernet. Para realizar el cambio de ancho de banda se utilizó el procedimiento mostrado en la Sección 7.6.2.

7.5 Escenario 5: NAT64

NAT64 con estado es un mecanismo de transición que permite traducir paquetes IPv4 a IPv6 y viceversa. El escenario para NAT64 es una red donde se conectan una red Pv4 nativa con una red IPv6 nativa. Este escenario presenta las siguientes características: 3 PCs conectados point-to-point mediante cross-over, donde PC1 es el host cliente y constituye la red IPv4, PC2 es el host servidor y constituye la red IPv6 y T1 es un host dual stack que es configurado como traductor de paquetes.

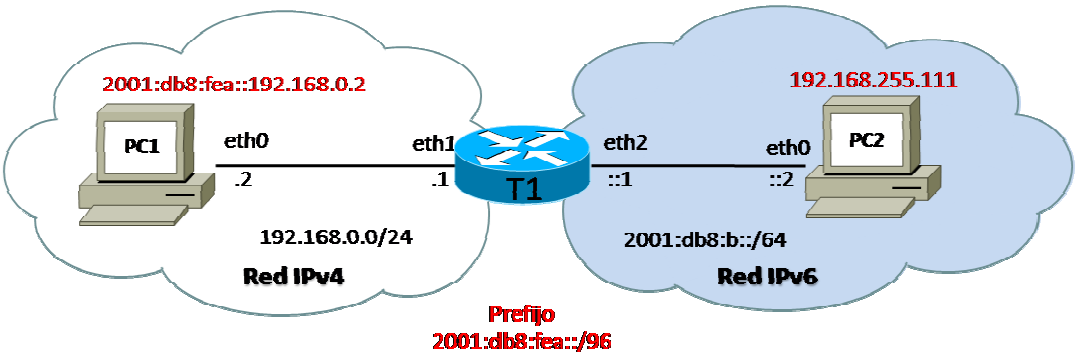


Figura 7.18: Escenario 5, NAT64

Existen varias implementaciones de NAT64 para Debian 7, algunas de ellas se listan a continuación: (1) Jool[26], (2) Tayga[27] (3) WrapSix[28], y (4) Ecdysis[29]. Sin embargo, solo serán evaluadas las implementaciones de NAT64: Jool y Tayga. Esto debido a que Ecdysis es un proyecto muerto y para realizar pruebas en Wrapsix se necesita modificar el código fuente cada vez que se desee cambiar un parámetro, además que los resultados reportados no son consistentes.

7.5.1 Configuración de Jool

Para realizar la configuración del escenario en Debian 7, debe ubicar el archivo `/etc/network/interfaces` y modificar según lo especificado en la Figura 7.19 y la Figura 7.20.

```
PC1:
Archivo /etc/network/interfaces
01: auto eth0
02: iface eth0 inet static
03: address 192.168.0.2
04: netmask 255.255.255.0
05: gateway 192.168.0.1
```

```
PC2:
Archivo /etc/network/interfaces
01: auto eth0
02: iface eth0 inet6 static
03: address 2001:db8:fea::192.168.255.111
04: netmask 120
05: gateway 2001:db8:fea::192.168.255.110
```

Figura 7.19: Configuración de PC1, PC2 para Jool en Debian 7

Una vez modificadas las interfaces, reiniciar los PCs o ejecutar los siguientes comandos para que procedan los cambios en las interfaces: `/etc/init.d/network-manager stop` y posteriormente: `/etc/init.d/network-manager start`.

[26] http://jool.mx/
[27] http://www.litech.org/tayga/
[28] http://www.wrapsix.org/
[29] http://ecdysis.viagenie.ca/

```
T1:
Archivo /etc/network/interfaces
01: auto eth1
02: iface eth1 inet static
03: address 192.168.0.1
04: netmask 255.255.255.0
05: auto eth2
06: iface eth2:0 inet6 static
07: address 2001:db8:fea::192.168.255.110
08: netmask 120
```

Figura 7.20: Configuración de T1 para Jool en Debian 7

Además, T1 es el traductor de paquetes. Por ello, se debe realizar el proceso de activación del forwarding IPv4 e IPv6 (ver procedimiento en la Sección 7.6.1) y realizar la instalación de Jool en este dispositivo.

Para realizar la instalación de Jool, primero es necesario instalar los siguientes requerimientos: `pkg_config`, `libnl-3-dev` y `linux-headers`, ingresando vía consola: `apt-get install pkg_config`, `apt-get install libnl-3-dev`, y `apt-get install linux-headers-$(uname -r)`.

Descargar del sitio oficial el archivo `Jool-3.3.2.zip` y descomprimir ingresando `unzip Jool-3.3.2.zip`. Acceder a `/Jool-3.3.2/mod` y compilar el módulo kernel mediante el comando: `make`, y para instalar: `make modules_install`. Seguidamente, acceder a `/Jool-3.3.2.zip/usr`, compilar el módulo de usuario mediante los comandos: `./configure`, `make`, y `make install`. Correr Jool con `depmod` y agregar el prefijo jool para Siit tradicional: `/sbin/modprobe jool_siit pool6=2001:db8:fea::/96`.

En caso de que Jool no funcione correctamente, si tiene instalado `ethtool`[30], asegurarse de ingresar por consola los comandos especificados en la Figura 7.21, por medio de los cuales se especifica que se inhabiliten la descarga de segmentación TCP (líneas 01 y 06), la descarga de fragmentación UDP (líneas 02 y 07), la segmentación genérica (líneas 03 y 08), la descarga genérica (líneas 04 y 09) y la recepción de descargas largas (líneas 05 y 10).

```
Comandos de Consola:
01: ethtool --offload eth0 tso off
02: ethtool --offload eth0 ufo off
03: ethtool --offload eth0 gso off
04: ethtool --offload eth0 gro off
05: ethtool --offload eth0 lro off
06: ethtool --offload eth1 tso off
07: ethtool --offload eth1 ufo off
08: ethtool --offload eth1 gso off
09: ethtool --offload eth1 gro off
10: ethtool --offload eth1 lro off
```

Figura 7.21: Comandos para ethtool

[30] http://linux.die.net/man/8/ethtool

Otra particularidad a tomar en cuenta sobre la configuración de Jool es la limitación que presenta la implementación en relación al MTU y fragmentación. Existe una diferencia entre IPv4 e IPv6 que el traductor IP por sí solo no puede compensar. La cabecera IPv4 tiene una bandera llamada no fragmentar (DF) la cual determina si el host origen permite que los routers o demás dispositivos fragmenten el paquete. En IPv6, los paquetes no pueden ser fragmentados por los routers, IPv6 sólo se permite la fragmentación en el origen como si la bandera DF estuviera siempre encendida.

Cuando hay un traductor en el medio, un paquete IPv4 que puede ser fragmentado se convierte en un paquete IPv6 que no debe ser fragmentado. Para solventar esta limitación Jool solía tener una bandera llamada –minMTU6. Sin embargo, la disminución del tamaño del MTU a nivel de kernel se considera mejor práctica y ahora se configura ingresando el comando: `ip link set dev eth0 mtu <tamMTU>`, en donde <tamMTU> es el tamaño que se desea colocar al MTU. Si se desconoce la MTU mínima de la red IPv6, se asigna 1280, ya que por defecto, cada nodo IPv6 debe ser capaz de manejar al menos 1280 bytes por paquete. En consecuencia se asignó una MTU de 1474 bytes al PC que actúa de cliente.

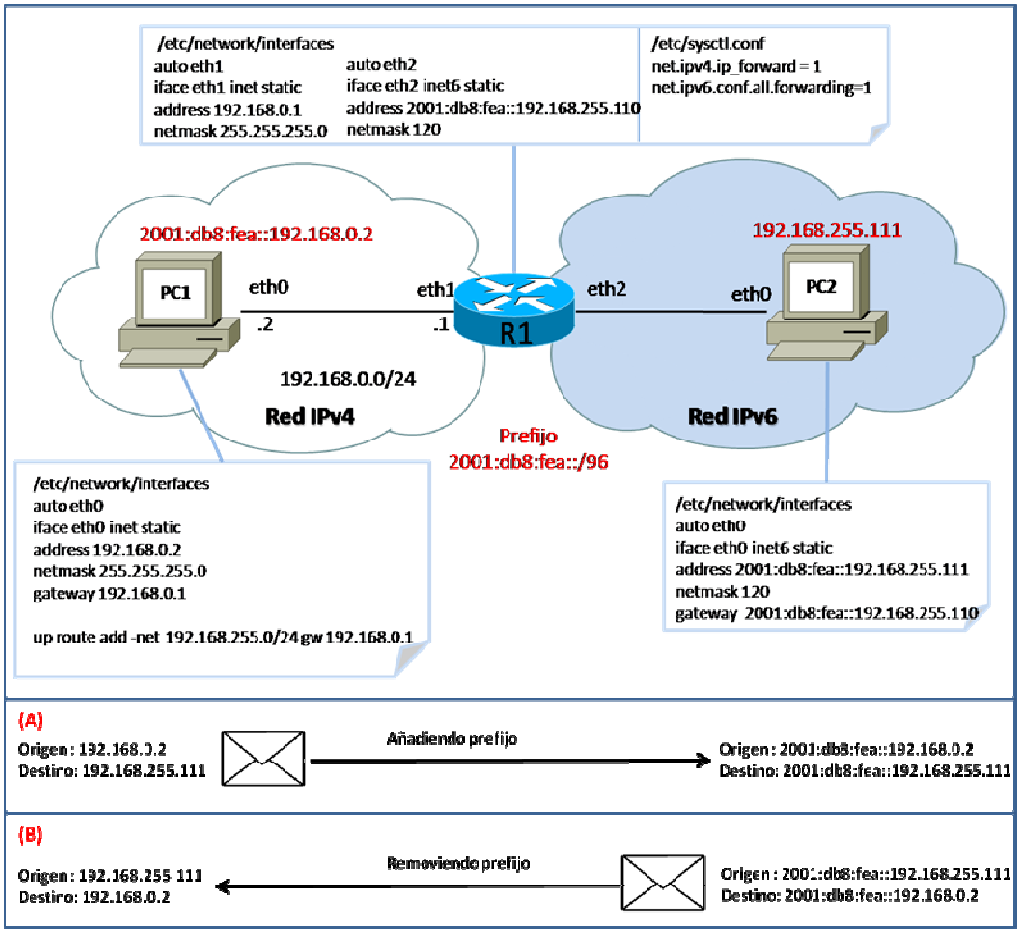


Figura 7.22: Configuración de Jool

Además, se utilizó el método SIIT tradicional que provee Jool. La idea es remover el prefijo IPv6 al traducir de IPv6 a IPv4, y añadir el prefijo IPv4al traducir de IPv4 a IPv6, como se

muestra en la Figura 7.22. En este escenario las pruebas se realizaron tomando como cliente tanto a PC1 como a PC2, es decir, se evaluó la traducción de forma bidireccional, la traducción de IPv4 a IPv6 (caso A) y traducción de IPv6 a IPv4 (caso B). Después de tener el escenario configurado se procedieron a realizar las diferentes pruebas de rendimiento con las herramientas Iperf y Benchmarking Comms1 como se indica en la Sección 7.5.3.

Durante las pruebas los inconvenientes presentados fueron en relación a la limitación de Jool en cuanto al MTU y fragmentación. Estos experimentos se realizaron tanto en Ethernet como en Fast Ethernet. Para realizar el cambio de ancho de banda se utilizó el procedimiento mostrado en la Sección 7.6.2.

7.5.2 Configuración de Tayga

Para realizar la configuración del escenario en Debian 7, se debe ubicar el archivo `/etc/network/interfaces` y modificarlo según lo especificado en la Figura 7.23.

```
PC1:
Archivo /etc/network/interfaces
01: auto eth0
02: iface eth0 inet static
03: address 192.168.0.2
04: netmask 255.255.255.0
05: gateway 192.168.0.1
```

```
PC2:
Archivo /etc/network/interfaces
01: auto eth0
02: iface eth0 inet6 static
03: address 2001:db8:b::2
04: netmask 64
05: gateway 2001:db8:b::1
```

```
T1:
Archivo /etc/network/interfaces
01: auto eth1
02: iface eth1 inet static
03: address 192.168.0.1
04: netmask 255.255.255.0
05: auto eth2
06: iface eth2 inet6 static
07: address 2001:db8:b::2
08: netmask 64
```

Figura 7.23: Configuración de PC1, PC2 y T1 para Tayga en Debian 7

Después de modificar las interfaces, se debe reiniciar los PCs o ejecutar los siguientes comandos para que procedan los cambios en las interfaces: `/etc/init.d/network-manager stop` y posteriormente ejecutar: `/etc/init.d/network-manager start`.

Además, T1 es el traductor de paquetes. Por ello, se debe realizar el proceso de activación del forwarding IPv4 e IPv6 (ver procedimiento en las Sección 7.6.1) y realizar la instalación de Tayga en este dispositivo ingresando vía consola el comando `aptitude install tayga.`

Para realizar la configuración de la herramienta Tayga, se debe ubicar los archivos`/etc/tayga.conf` y `/etc/default/tayga`, modificarlos como se indica en la Figura 7.24.

```
Archivo /etc/tayga.conf
01: tun-device nat64
02: ipv4-addr 192.168.255.1
03: prefix 2001:db8:fea::/96
04: dynamic-pool 192.168.255.0/24
05: data-dir /var/log/tayga/
06: #mapeo que indica la traducción de la dirección Ipv6
07: map 192.168.255.111 2001:db8:b::2
```

```
Archivo /etc/default/tayga
01: RUN="yes"
```

Figura 7.24: Archivos de Configuración de Tayga

Por último, se debe reiniciar T1 ingresando `init 6` o mediante el entorno gráfico para que se levante la interfaz Tayga. De lo contrario, se deben ingresar los comandos indicados en la Figura 7.25, para levantar la interfaz tun nat64 (líneas 01 y 02), asignar las direcciones a la interfaz (líneas 03 y 04) y agregar las rutas estáticas (líneas 05 y 06).

```
Comandos de Consola:
01: tayga --mktun
02: ip link set nat64 up
03: ip addr add 192.168.0.1 dev nat64
04: ip addr add 2001:db8:b::1 dev nat64
05: ip route add 192.168.255.0/24 dev nat64
06: ip route add 2001:db8:fea::/96 dev nat64
```

Figura 7.25: Configuración Manual de la Interfaz Tayga

Cabe resaltar que en este escenario las pruebas se realizaron tomando como cliente tanto a PC1 como a PC2; es decir, se evaluó la traducción de forma bidireccional: la traducción de IPv4 a IPv6 (caso A) y traducción de IPv6 a IPv4 (caso B). Cuando la traducción se realiza de IPv4 a IPv6, se configura en Tayga una dirección IPv4 única a cada host IPv6 que necesita servicio NAT64. Cuando la traducción se realiza de IPv6 a IPv4 se remueve el prefijo NAT64. Después de tener el escenario configurado se procedieron a realizar las diferentes pruebas de rendimiento con las herramientas Iperf y Benchmarking Comms1 como se indica en la Sección 7.5.3.

7.5.3 Experimentos NAT64

Después de tener los escenarios configurados se procedieron a realizar las diferentes pruebas de rendimiento tanto para Tayga como para Jool con las herramientas Iperf y Benchmarking Comms1 explicadas a continuación.

- **Caso A (IPv4 a IPv6):** Durante el proceso de medición del throughput en Debian 7, primero se ejecutó en PC2 el generador de tráfico en modo servidor: `iperf –V -u –s –l 10000`. La opción `–l` se estableció en 10000 ya que es la máxima carga útil que será inyectada durante los experimentos, en donde la opción –V indica que en este extremo se utilizará IPv6. Una vez el servidor estuvo escuchando peticiones, se ingresó el siguiente comando: `iperf -u -c 192.168.255.111 -t 20 -i 2 -b 10Mbps -l <TamCargaUtil>`, en el PC1 para Ethernet y para Fast Ethernet se utilizó: `iperf -u -c 192.168.255.111 -t 20 -i 2 -b 100Mbps -l <TamCargaUtil>`, en donde, `<TamCargaUtil>` es un entero que indica el tamaño de la carga útil a generar y la opción `–V` no es indicada debido a que este extremo trabaja con IPv4. Todos los experimentos se repitieron de 10 a 20 veces para generar resultados más consistentes.

  Para calcular el OWD, con el protocolo de transporte TCP, se utilizó el formato de archivo de datos especificado en la Figura 7.2. Se ejecutó la herramienta Benchmarking Comms1 en modo servidor con el ejecutable para IPv6 en PC2 utilizando el siguiente comando: `./comms1-tcp-v6-unix server 5001`, y en modo cliente con el ejecutable para IPv4 en PC1 con el siguiente comando: `./comms1-tcp-v4-unix client datos.dat 192.168.255.111 5001`.

Para calcular el OWD, con el protocolo de transporte UDP, se utilizó el formato de archivo de datos especificado en la Figura 7.2. Se ejecutó la herramienta Benchmarking Comms1 en modo servidor con el ejecutable para IPv6 en PC2 utilizando el siguiente comando: `./comms1-udp-v6-unix server 5001`, y en modo cliente con el ejecutable para IPv4 en PC1 con el siguiente comando: `./comms1-udp-v4-unix client datos.dat 192.168.255.111 5001`.

- **Caso B (IPv6 a IPv4):** Durante el proceso de medición del throughput en Debian 7, primero se ejecutó en PC2 el generador de tráfico en modo servidor de la siguiente manera: `iperf -u –s –l 10000`. La opción –l se estableció en 10000 ya que es la máxima carga útil que será inyectada durante los experimentos, en donde la opción –V no es indicada debido a que este extremo trabaja con IPv4. Una vez el servidor este escuchando peticiones, se ingresó el siguiente comando: `iperf –V -u -c 2001:db8:fea::192.168.0.2 -t 20 -i 2 -b 10Mbps -l <TamCargaUtil>`, en el PC1 para Ethernet y para Fast Ethernet se utilizó: `iperf –V -u -c 2001:db8:fea::192.168.0.2 -t 20 -i 2 -b 100Mbps -l <TamCargaUtil>`, en donde, `<TamCargaUtil>` es un entero que indica el tamaño de la carga útil a generar y la opción `–V` indica que este extremo trabaja con IPv6. Todos los experimentos se repitieron de 10 a 20 veces para generar resultados más consistentes. Durante las pruebas el contratiempo principal fue generado por la limitación de Jool en cuanto al manejo de MTU y fragmentación.

  Para calcular el OWD, con el protocolo de transporte TCP, se utilizó el formato de archivo de datos especificado en la Figura 7.2. Se ejecutó la herramienta Benchmarking Comms1 en modo servidor con el ejecutable para IPv4 en PC2 utilizando el siguiente comando: `./comms1-tcp-v4-unix server 5001`, y en modo cliente con el ejecutable para IPv6 en PC1con el siguiente comando: `./comms1-tcp-v6-unix client datos.dat 2001:db8:fea::192.168.0.2 5001`.

  Para calcular el OWD, con el protocolo de transporte UDP, se utilizó el formato de archivo de datos especificado en la Figura 7.2. Se ejecutó la herramienta Benchmarking Comms1 en modo servidor con el ejecutable para IPv4 en PC2 utilizando el siguiente comando: `./comms1-udp-v4-unix server 5001`, y en modo cliente con el ejecutable para IPv6 en PC1 con el siguiente comando: `./comms1-udp-v6-unix client datos.dat 2001:db8:fea::192.168.0.2 5001`.

7.6 Configuraciones Generales

En la siguiente Sección se detallan algunos procedimientos que se necesitan para configurar los escenarios diseñados.

7.6.1 Activar Forwarding en Windows y Debian 7

La activación del forwarding es necesaria para configurar un dispositivo de red como enrutador de paquetes. Este procedimiento varía según la versión del protocolo IP que se esté utilizando.

Para activar el forwarding IPv6 en Windows primero se deben ubicar los identificadores (`ifindex`) de las interfaces de red en las que se necesita activar el enrutamiento de paquetes. Para ello, se debe acceder a la consola de Windows y escribir el comando

`ipconfig`. El `ifindex` se encuentra al final de la dirección IPv6 separado por un `%`. Por último ingresar el siguiente comando para la activación del enrutamiento: `netsh interface ipv6 set interface <ifindex> forwarding=enabled`.

Por ejemplo, en la Figura 7.9 el dispositivo R1 tiene dos interfaces de red, la interfaz `eth1` y `eth2` con `ifindex` 13 y 22 respectivamente. Para activar el enrutamiento IPv6 se deben escribir los siguientes comandos en el símbolo de sistema: `netsh interface ipv6 set interface 13 forwarding=enabled` y `netsh interface ipv6 set interface 22 forwarding=enabled`.

Por otro lado, para activar el forwarding en IPv4, se debe acceder a la consola de Windows y escribir `regedit`, y en la ventana desplegada (editor de registros) se debe ubicarla siguiente clave: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters`. Posteriormente, se ubica el parámetro `IPEnableRouter` y se le asigna el valor 1 como se muestra en laFigura 7.26. Finalmente, se debe reiniciar la PC para hacer efectivo los cambios.

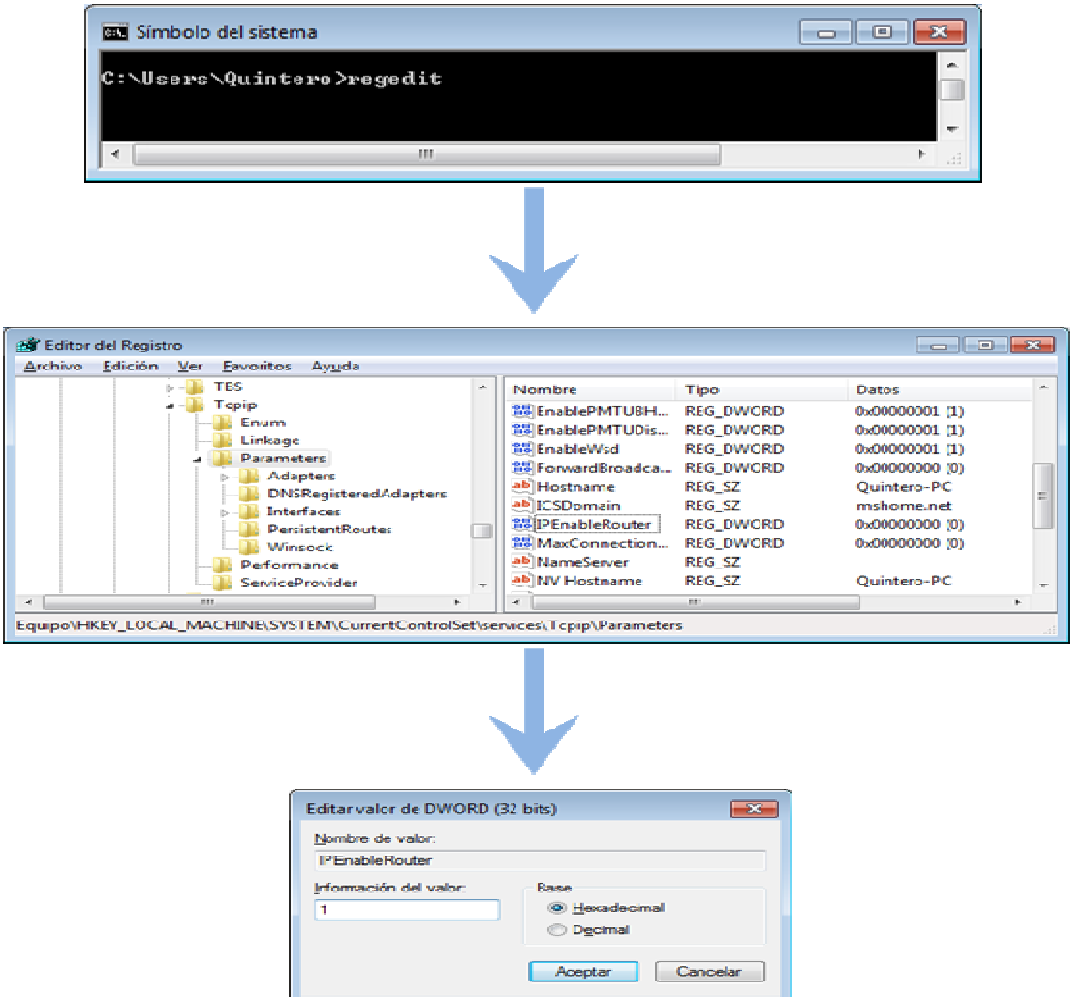


Figura 7.26: Proceso de Activación del Forwarding IPv4 (Windows)

Para activar el forwarding en Debian 7, se debe ubicar el archivo `/etc/sysctl.conf` y descomentar o agregar para IPv4 la línea: `net.ipv4.ip_forward=1` y, para IPv6 se debe agregar la línea: `net.ipv6.conf.all.forwarding=1`. Se debe Ingresar vía consola el comando `sysctl –p` para que se actualicen los cambios en el sistema o reiniciar la PC. Cabe destacar que este procedimiento solo debe aplicarse al dispositivo que actuará como router en la red.

7.6.2 Cambiar de la Velocidad de la Red en Windows y Debian 7

Para especificar explícitamente la velocidad y tipo de comunicación del enlace en Debian 7 primero se debe instalar la herramienta ethtool con el comando: `apt-get install ethtool`. Seguidamente se debe ubicar el archivo `/etc/network/interfaces` y agregar las siguientes líneas a la interfaz: `link-speed <10-100-1000>`, `link-duplex <full-half>`, y `ethernet-autoneg <off-on>`, en donde, solo se debe seleccionar una de las opciones especificadas dentro de `<10-100-1000>`, `<full-half>`, y `<off-on>`, siendo 10-100-1000 las velocidades posibles del enlace, full-half los tipos de comunicación y off-on las opciones de apagado o encendido de la auto negociación. La Figura 7.27 muestra un ejemplo de la configuración en la interfaz con modificación de la velocidad y tipo de comunicación del enlace, en el cual la interfaz eth0 con dirección 192.168.0.2/24 (líneas 01 a 04) utilizará Fast Ethernet (línea 05), en una conexión full dúplex (línea 06) y tendrá la auto negociación apagada (línea 07). También se puede cambiar la velocidad ingresando por consola el siguiente comando: `ethtool -s <interfaz> speed <10-100-1000> duplex <full-half> autoneg <off-on>`

```
Archivo /etc/network/interfaces
01: auto eth0
02: iface eth0 inet static
03: address 192.168.0.2
04: netmask 255.255.255.0
05: link-speed 100
06: link-duplex full
07: ethernet-autoneg off
```

Figura 7.27: Ejemplo de Modificación de la Velocidad del Enlace en una Interfaz

Por otra parte, si se desea modificar la velocidad del enlace en Windows, se debe ubicar el “Panel de Control”, luego el “Centro de redes y recursos compartidos”, y en el menú izquierdo seleccionar "Cambiar configuración del adaptador”. En la lista de resultados seleccionar la interfaz de red que se desea modificar. En los escenarios de pruebas las interfaces de PC1 y PC2 son denominadas “Conexión de área local” (eth0), y en el R1 la interfaz “Conexión de área local 1” (eth1) y “Conexión de área local 2” (eth2).

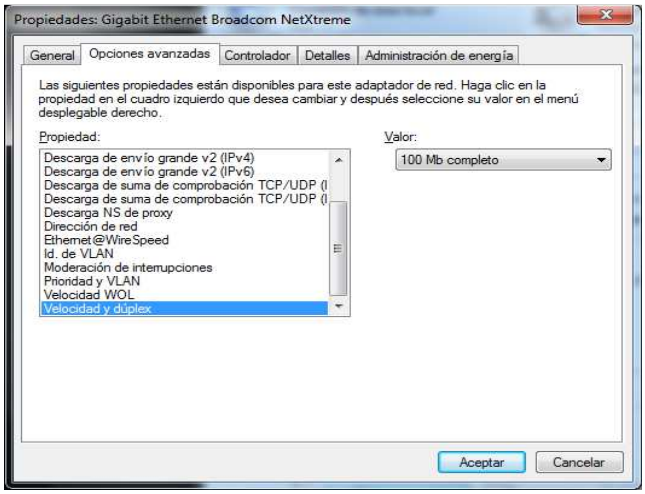


Figura 7.28: Ventana de Propiedades de una Interfaz en Windows

Haga clic con el botón secundario del mouse sobre la interfaz a configurar y, a continuación en “Propiedades”, en la ventana resultante ubicar “Configuración”. Una vez ubicados en la ventana de Configuración de la interfaz ir a la pestaña “Opciones Avanzadas” y en el menú seleccionar la opción “Velocidad y dúplex” y en el campo “Valor” ubicar la velocidad que desee como se muestra en la Figura 7.28.

7.7 Análisis de Resultados

En la siguiente sección se presentan los resultados obtenidos en las diferentes pruebas de rendimiento.

7.7.1 Resultados del Throughput para Ethernet con UDP

La Figura 7.29, muestra los resultados del throughput obtenidos durante los experimentos de cada uno de los mecanismos para Ethernet sobre el sistema operativo Debian 7, en donde teóricamente IPv4 nativo debería presentar un throughput superior o similar al calculado en todas las demás tecnologías.

Como era de esperarse debido a que IPv4 e IPv6 son los protocolos base de los mecanismos de transición y se evaluaron de forma nativa, IPv4 presenta un throughput superior que el de IPv6, ISATAP y 6to4; y similar que el de Jool (IPv4 a IPv6) y Tayga (IPv4 a IPv6). Asimismo, IPv6 presenta un throughput superior que el de ISATAP y 6to4 y similar que el de Jool (IPv6 a IPv4) y Tayga (IPv6 a IPv4) en la mayoría de los casos. Las dos implantaciones de NAT64 (Jool y Tayga) presentan un throughput superior que el de ISATAP y 6to4, mientras que ISATAP y 6to4 tienen un throughput similar. 6to4 presenta un throughput ligeramente superior al de ISATAP. El throughput obtenido para todos los mecanismos se acerca en gran medida a la velocidad de transmisión máxima de Ethernet (10Mbps), siendo 9.64 Mbps el resultado más cercano a la realidad física.

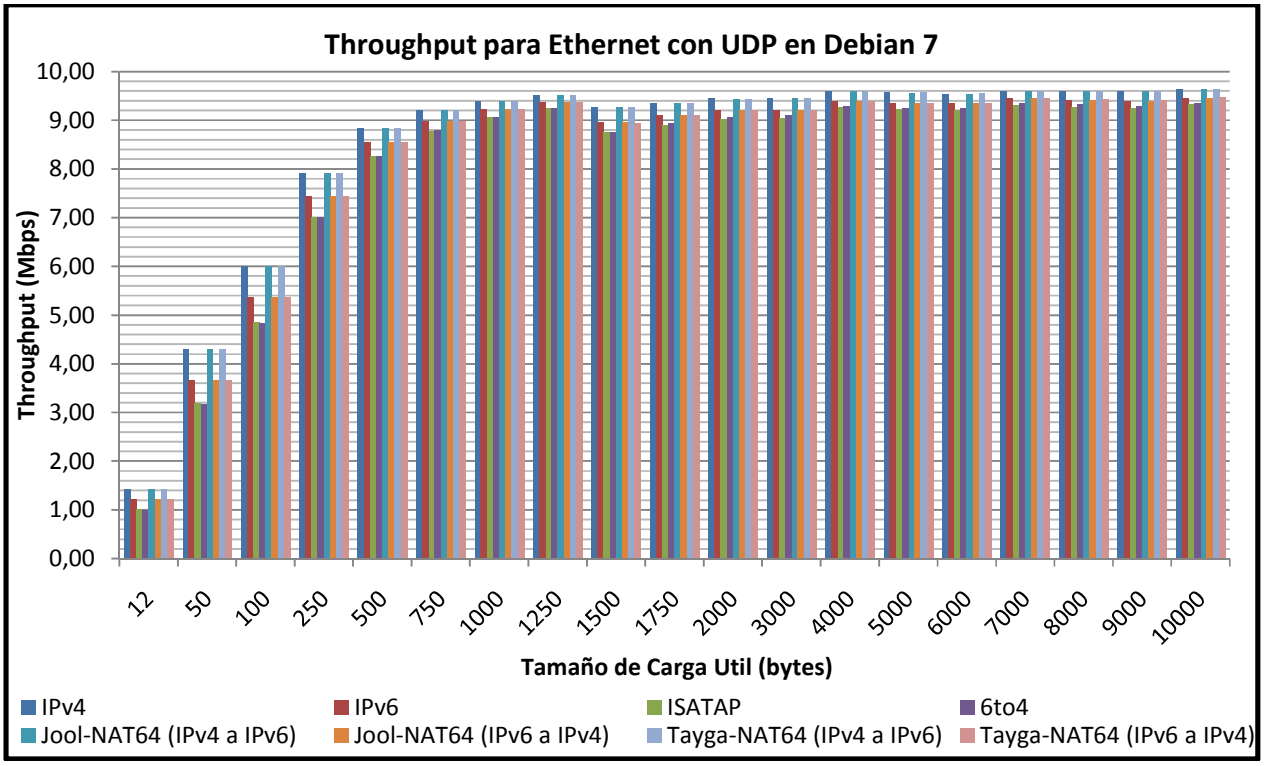


Figura 7.29: Throughput para Ethernet con UDP en Debian 7

La Figura 7.30, muestra los resultados del throughput obtenidos durante los experimentos de cada uno de los mecanismos para Ethernet sobre el sistema operativo Windows 7, en donde se evidencia que IPv4 presenta un throughput superior al calculado en todas las demás tecnologías. El throughput de IPv6 es superior al de ISATAP y 6to4, mientras que el de ISATAP y 6to4 es similar.

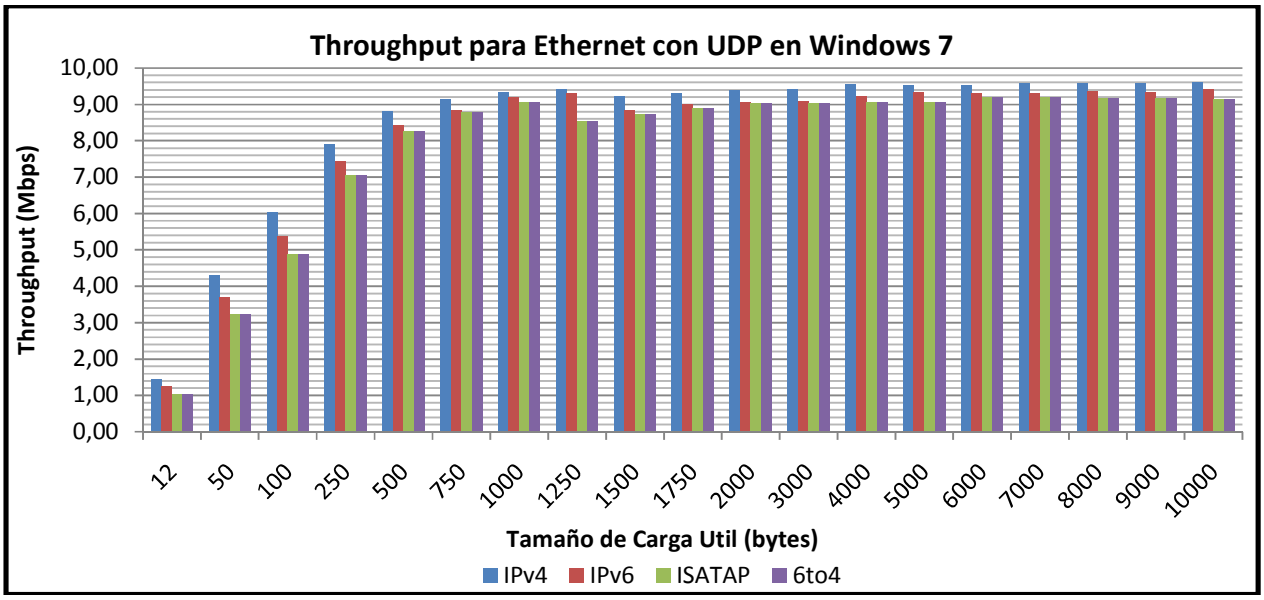


Figura 7.30: Throughput para Ethernet con UDP en Windows 7

De igual forma, en la Figura 7.30 se muestran los resultados del throughput obtenidos durante los experimentos para Ethernet sobre el sistema operativo Windows 8.1. IPv4 presenta el throughput superior en todos los casos, seguido por el de IPv6. Después de IPv6, se encuentra throughput de ISATAP y 6to4, los cuales poseen un comportamiento similar.

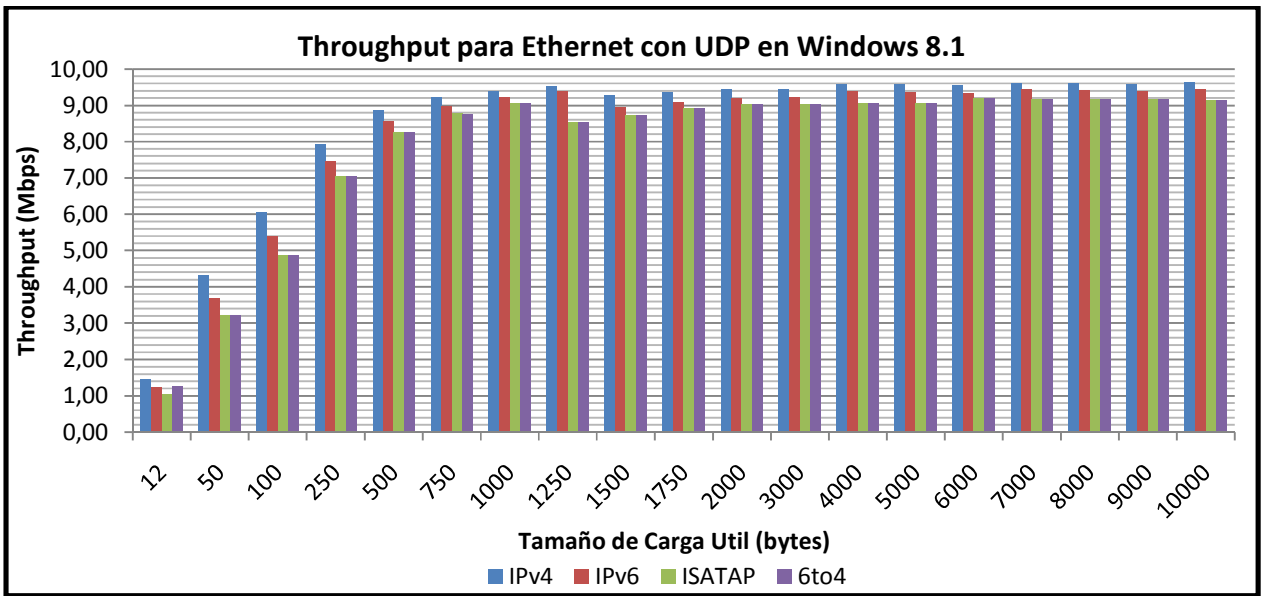


Figura 7.31: Throughput para Ethernet con UDP en Windows 8.1

Por último, en la Figura 7.32, se muestran los resultados del throughput obtenidos durante los experimentos para Ethernet sobre el sistema operativo Windows 10, cuyos resultados son similares a los obtenidos en Windows 7 y Windows 8.1.IPv4 e IPv6 presentan el mayor throughput, seguido de ISATAP y 6to4.

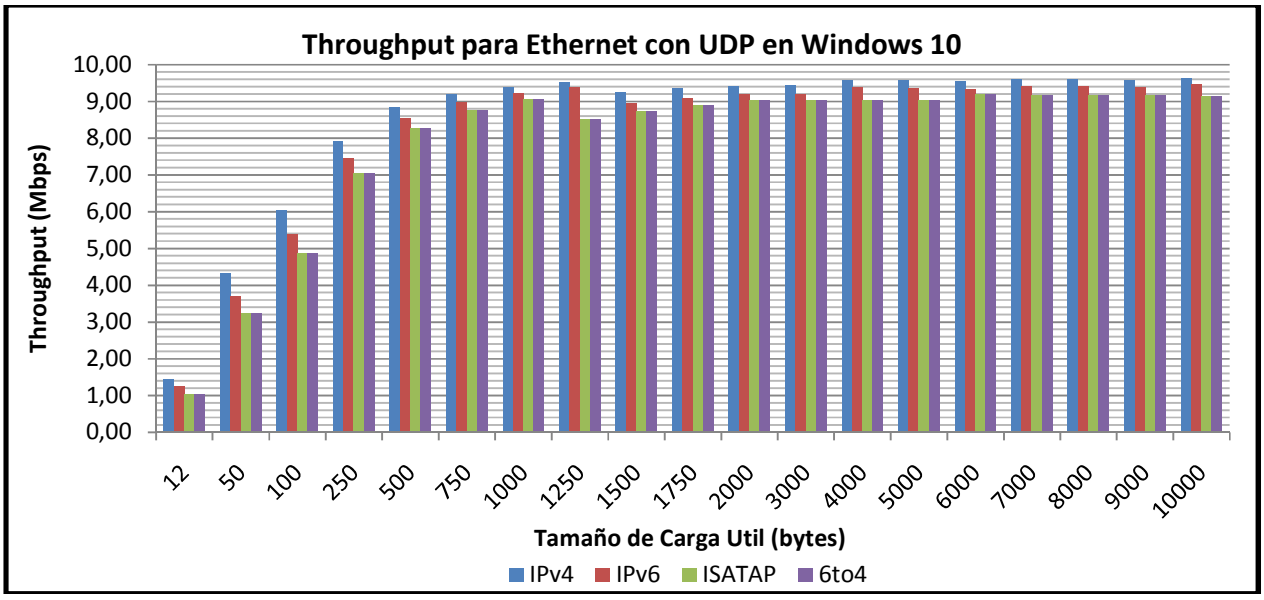


Figura 7.32: Throughput para Ethernet con UDP en Windows 10

De forma general, el throughput obtenido de los mecanismos de transición evaluados sobre las versiones de Windows es bastante cercano a la velocidad máxima de transmisión permitida por Ethernet (10 Mbps), siendo 9.46 Mbps la máxima velocidad reportada.

Al nivel de los sistemas operativos, Debian 7 supera a los otros para casi todos los tamaños de carga útil y todas las tecnologías.

7.7.2 Resultados del Throughput para Fast Ethernet con UDP

La Figura 7.33, muestra los resultados del throughput obtenidos durante los experimentos de cada uno de los mecanismos para Fast Ethernet sobre el sistema operativo Debian 7, en donde se evidencia que IPv4 presenta un throughput superior o similar al calculado en todas las demás tecnologías.

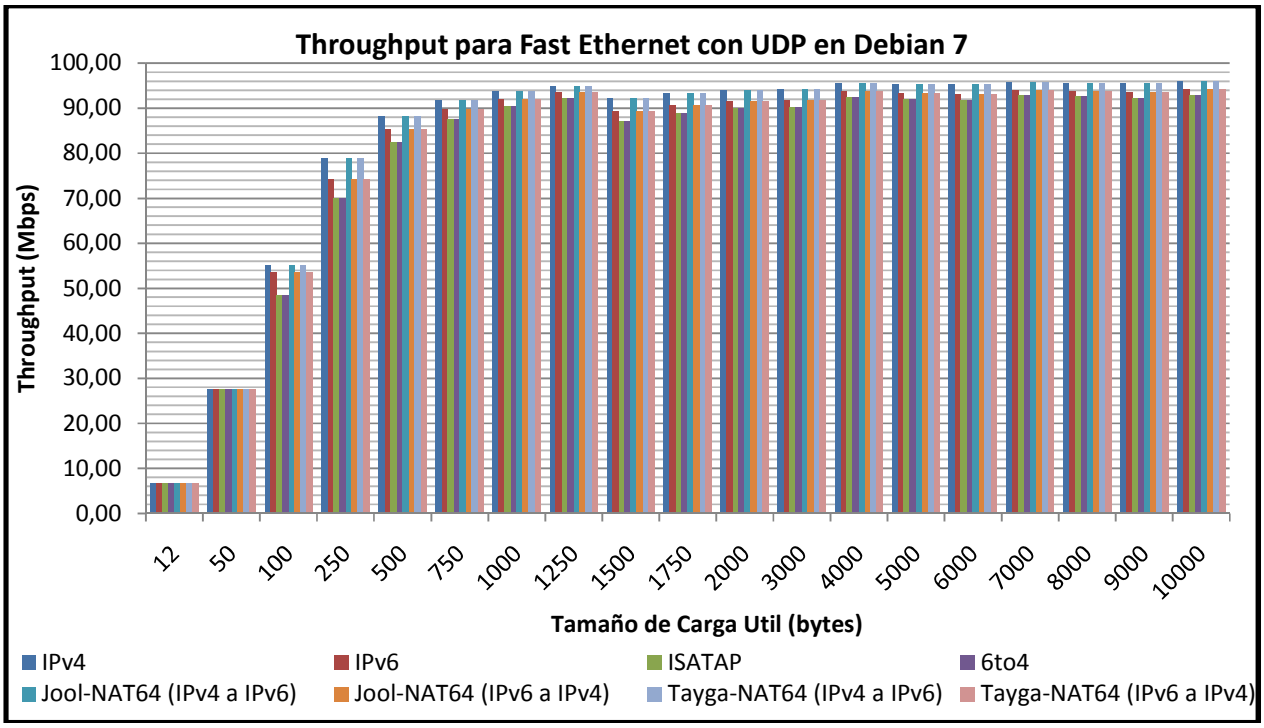


Figura 7.33: Throughput para Fast Ethernet con UDP en Debian 7

Al igual que en Ethernet, debido a que IPv4 e IPv6 son los protocolos base de los mecanismos de transición y se evaluaron de forma nativa, IPv4 presenta un throughput superior que el de IPv6 nativo, ISATAP y 6to4, y es similar al de Jool (IPv4 a IPv6) y Tayga (IPv4 a IPv6). Asimismo, IPv6 presenta un throughput superior al de ISATAP y 6to4 y similar al de Jool (IPv6 a IPv4) y Tayga (IPv6 a IPv4). Además, las dos implementaciones de NAT64 (Jool y Tayga) presentan un throughput superior al de ISATAP y 6to4, mientras que ISATAP y 6to4 tienen un throughput similar. El throughput obtenido para todos los mecanismos se acerca en gran medida velocidad de transmisión máxima de Ethernet (100 Mbps), siendo 96 Mbps el resultado más cercano a la realidad física.

El throughput obtenido para Fast Ethernet sobre el sistema operativo Windows 7 puede ser observado en la Figura 7.34, en donde se evidencia que IPv4 presenta un throughput

superior al calculado en todas las demás tecnologías. El throughput de IPv6 es superior que el de ISATAP y 6to4, mientras que el de ISATAP y 6to4 es similar. Particularmente, la diferencia entre el throughput de IPv4 e IPv6, y el de ISATAP y 6to4 es superior a 10 Mbps para las cargas útil superior a 50 bytes.

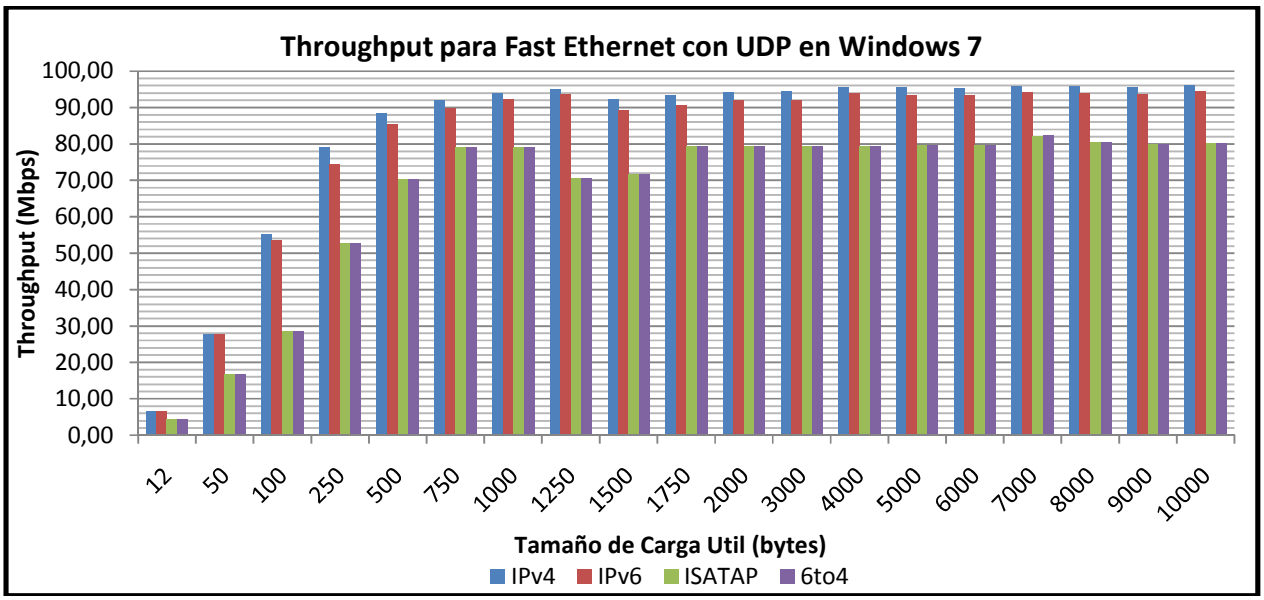


Figura 7.34: Throughput para Fast Ethernet con UDP en Windows 7

De igual forma, en la Figura 7.35 se muestran los resultados del throughput obtenidos durante los experimentos para Fast Ethernet sobre el sistema operativo Windows 8.1. Al igual que en los otros casos, IPv4 muestra el throughput superior. Además, el throughput de IPv6 es superior que el de ISATAP y 6to4, mientras que el comportamiento de ISATAP y 6to4 es similar para todos los tamaños de carga útil.

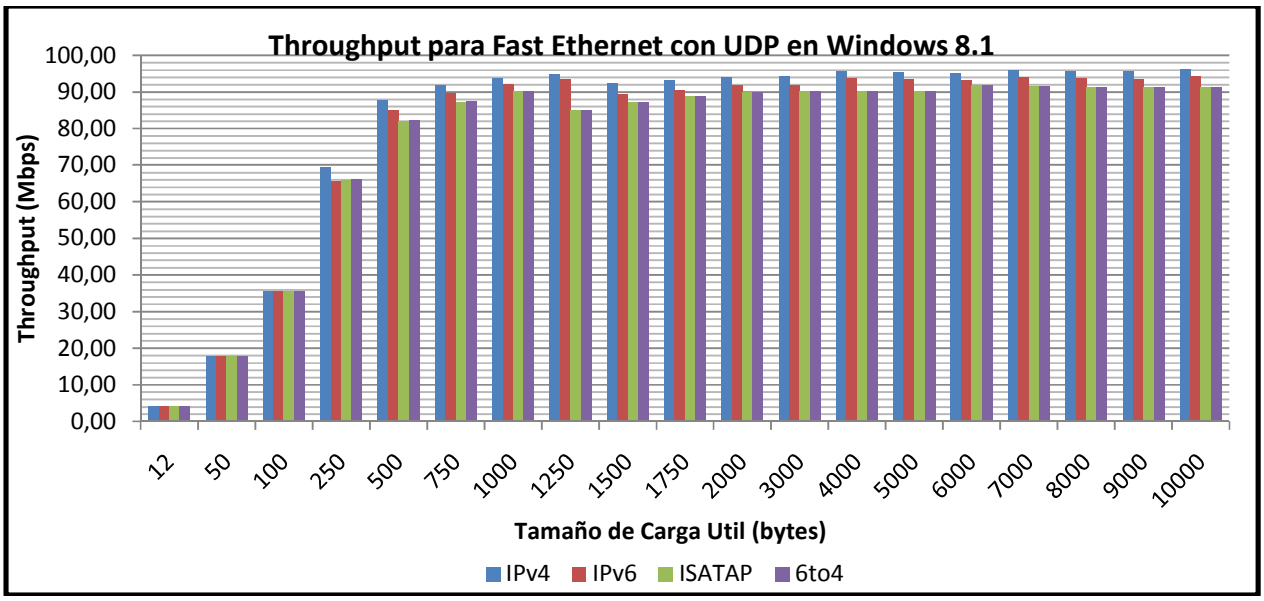


Figura 7.35: Throughput para Fast Ethernet con UDP en Windows 8.1

En la Figura 7.36 se muestran los resultados del throughput obtenidos durante los experimentos para Fast Ethernet sobre el sistema operativo Windows 10, en donde IPv4 presenta un throughput superior al calculado en todas las demás tecnologías. Además, el throughput de IPv6 es superior que el de ISATAP y 6to4, mientras que el de ISATAP y 6to4 es similar para la mayoría de los casos. Estos resultados se asemejan a los obtenidos en Windows 7 y Windows 8.1.

Al nivel de los sistemas operativos al igual que en el caso de Ethernet, Debian 7 supera a los demás para casi todos los tamaños de carga útil.

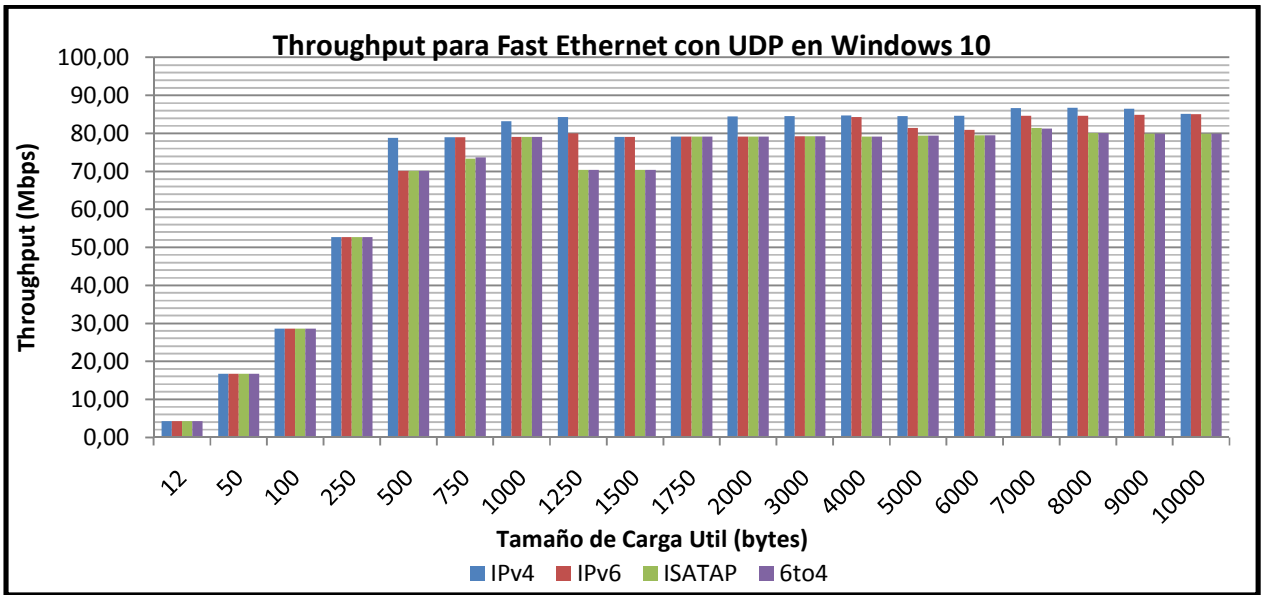


Figura 7.36: Throughput para Fast Ethernet con UDP en Windows 10

7.7.3 Resultados del OWD para Ethernet con UDP

La Figura 7.37 muestra los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Ethernet con el protocolo de transporte UDP sobre el sistema operativo Debian 7, en donde se evidencia que el OWD más alto es presentado por Tayga y el más bajo es presentado por IPv4.

El retardo obtenido de los experimentos de Jool en ambas direcciones es superior que el de IPv4, pero inferior que el de IPv6. Además, para todos los casos Jool posee un mejor rendimiento que Tayga. Hay que resaltar que el retardo obtenido en Tayga supera a las demás tecnologías por casi el doble de microsegundos en las cargas útil mayores a 3000 bytes. Además, ISATAP y 6to4 presentan un retardo similar, que es superior que el de IPv4, IPv6 y Jool, pero inferior que el de Tayga.

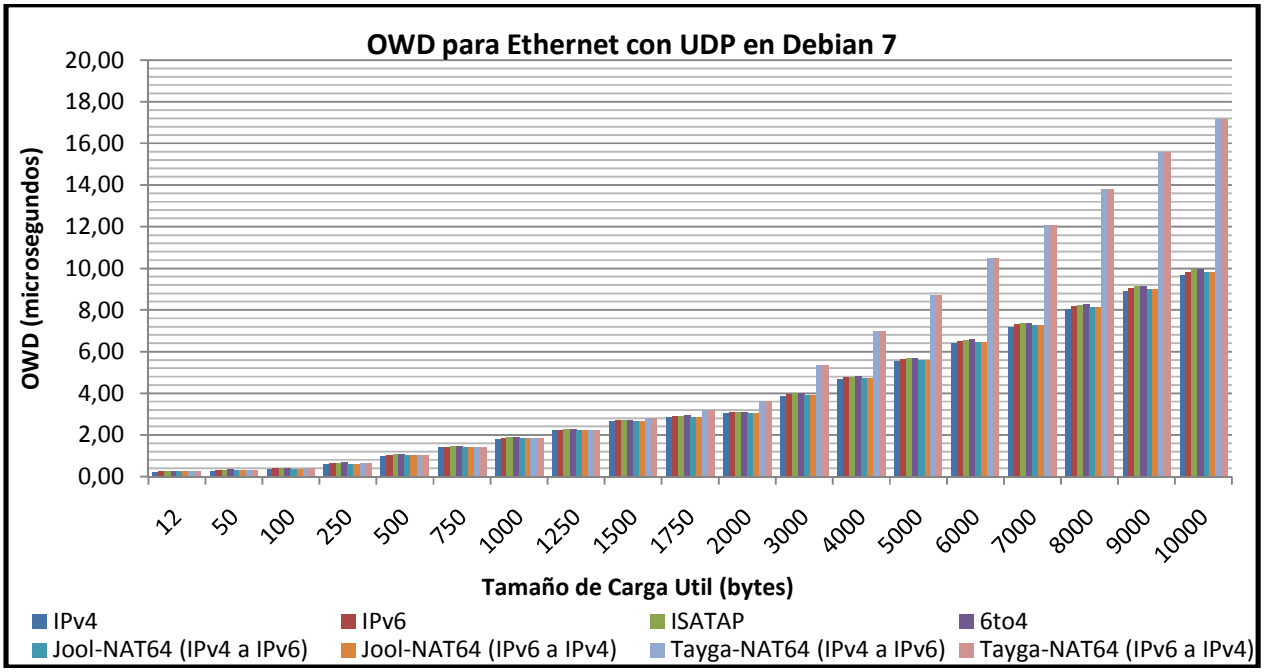


Figura 7.37: OWD para Ethernet con UDP en Debian 7

Por otro lado, la Figura 7.38 muestra los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Ethernet con el protocolo de transporte UDP sobre el sistema operativo Windows 7. Como se puede observar, el OWD de IPv4 siempre es menor o igual a IPv6. Además, es de resaltar algunos casos puntuales donde ISATAP y 6to4 presentaron un retardo inferior que IPv4 e IPv6 (carga útil 1250, 5000, 8000 y 10000 bytes) y donde presentaron tiempos similares (carga útil 750, 1750, y 2000 bytes).

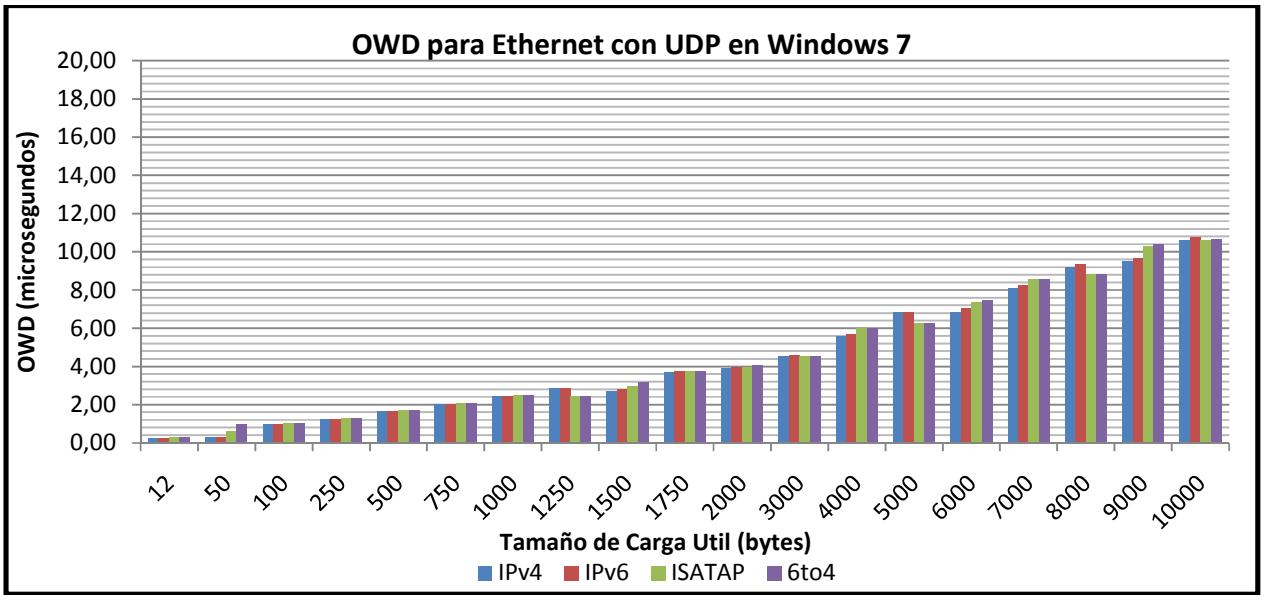


Figura 7.38: OWD para Ethernet con UDP en Windows 7

La Figura 7.39 muestra los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Ethernet con el protocolo de transporte UDP sobre el sistema operativo Windows 8.1, y la Figura 7.40 muestra los resultados obtenidos sobre el sistema operativo Windows 10. Se puede observar que en ambos sistemas operativos el OWD más bajo es presentado por IPv4 nativo, salvo en pequeñas excepciones. El siguiente OWD más alto es presentando por 6to4, seguido por el de ISATAP y 6to4.

Al nivel de los sistemas operativos, Debian 7 obtuvo mejores que los demás para casi todos los tamaños de carga útil.

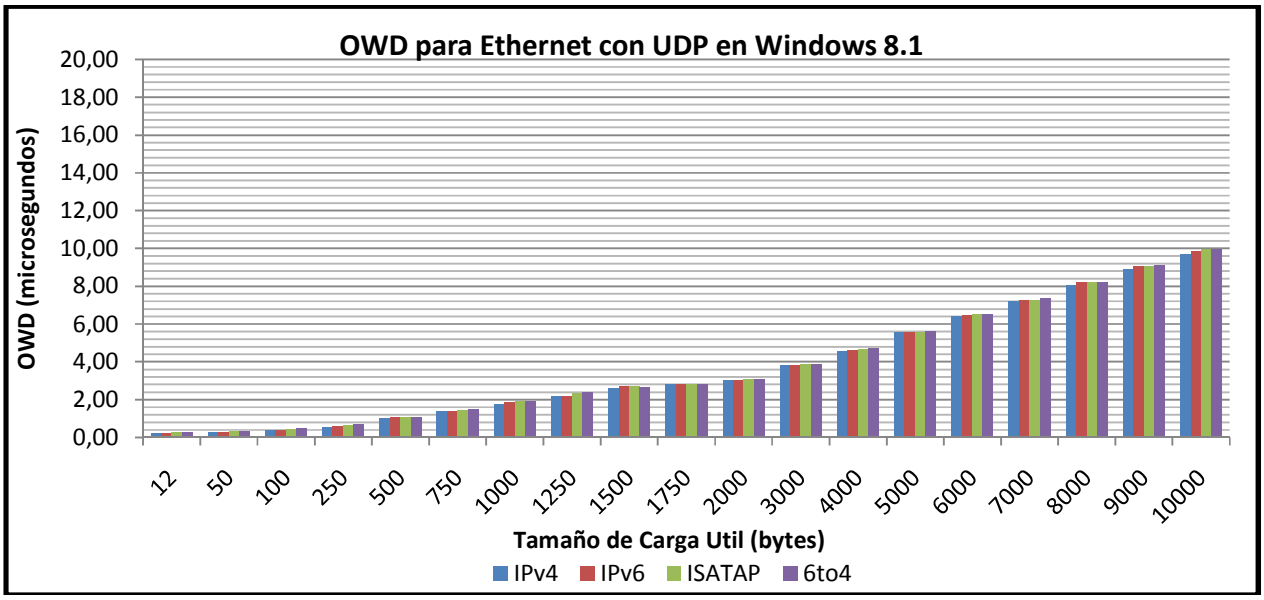


Figura 7.39: OWD para Ethernet con UDP en Windows 8.1

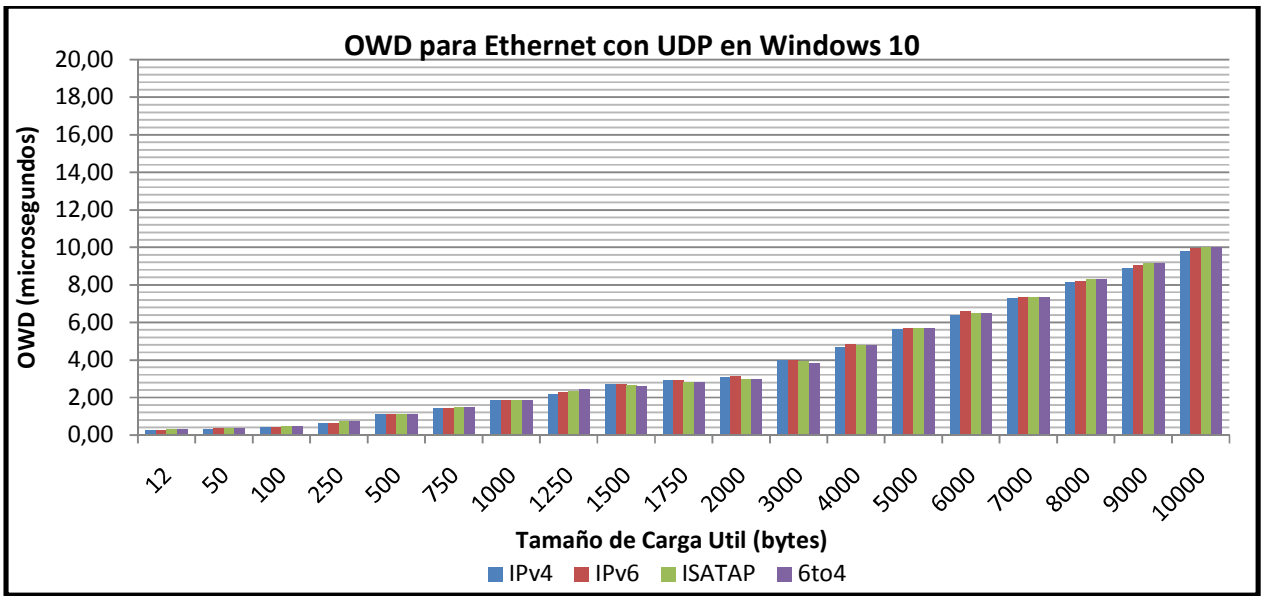


Figura 7.40: OWD para Ethernet con UDP en Windows 10

7.7.4 Resultados del OWD para Ethernet con TCP

La Figura 7.41, muestra los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Ethernet con el protocolo de transporte TCP sobre el sistema operativo Debian 7, donde se evidencia que ISATAP y 6to4 presentaron el retardo más alto mientras que IPv4 reportó el más bajo.

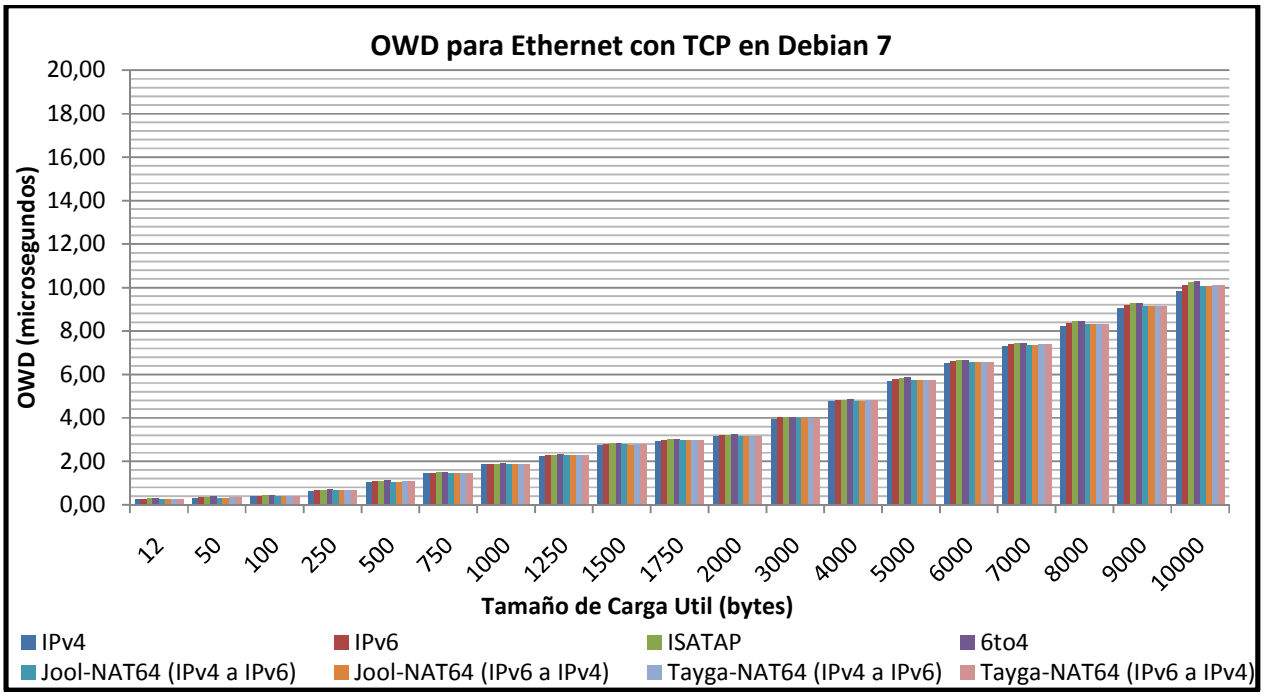


Figura 7.41: OWD para Ethernet con TCP en Debian 7

Además, a pesar de que en Tayga para el sistema operativo Debian 7 utilizando el protocolo de transporte UDP se reportaron valores superiores a todas las demás tecnologías, cuando se utilizó con protocolo de transporte TCP no ocurrió lo mismo. En este caso, Tayga y Jool presentaron retardos similares para ambas direcciones de comunicación, el cual es inferior a los presentados por ISATAP y 6to4, quienes a su vez también presentaron un rendimiento similar.

Los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Ethernet con el protocolo de transporte TCP sobre el sistema operativo Windows 7 son mostrados en la Figura 7.42. Aquí se evidencia que el retardo más bajo fue presentado por IPv4 y el más alto por ISATAP y 6to4. Además, cuando la carga útil es de 5000 bytes la carga útil reportada por ISATAP y 6to4 es inferior a la reportada por IPv4 e IPv6. Cuando la carga útil oscila entre 100 y 1250 bytes ISATAP y 6to4 presentan una elevación en microsegundos en comparación con IPv4 e IPv6 la cual se disminuye para una carga útil superior a 1250 bytes. IPv6 presenta un retardo superior que IPv4 pero inferior que ISATAP y 6to4, para la mayoría de los casos.

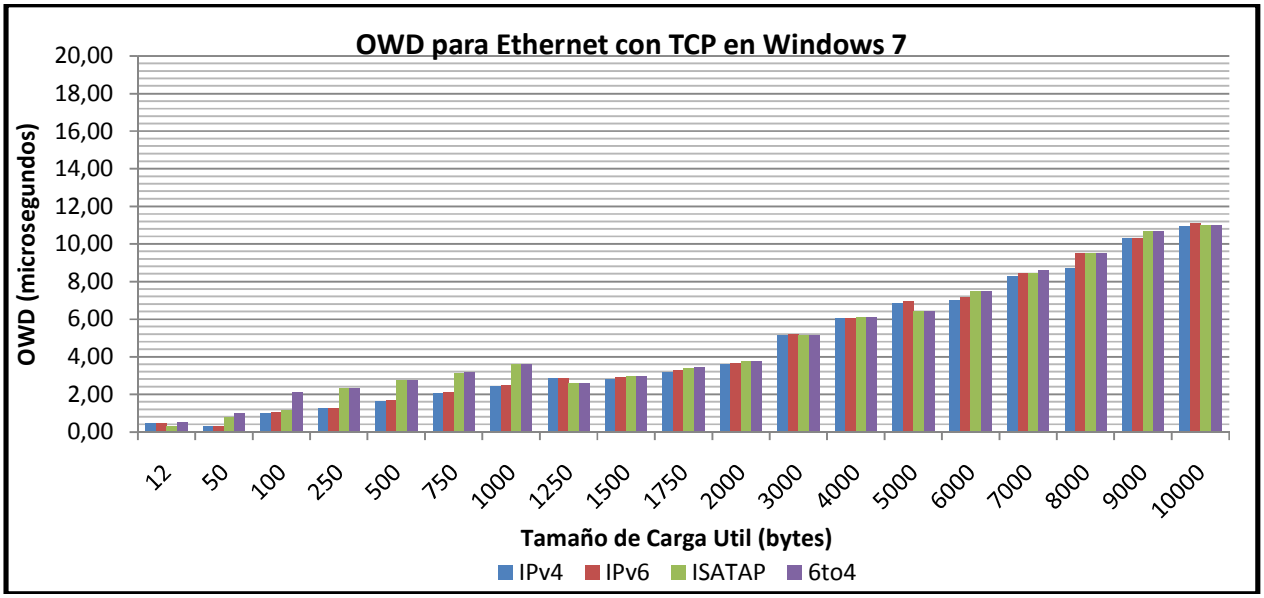


Figura 7.42: OWD para Ethernet con TCP en Windows 7

La Figura 7.43muestra los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Ethernet con el protocolo de transporte TCP sobre el sistema operativo Windows 8.1 y la Figura 7.44, muestra los resultados obtenidos sobre el sistema operativo Windows 10. En ambos casos se evidencia que el OWD más bajo es presentado por IPv4, y el más alto es presentado por ISATAP y 6to4. Además, IPv6 presenta un retardo inferior que ISATAP y 6to4 pero superior que IPv4.

Al nivel de los sistemas operativos, Windows 8.1 obtuvo mejores que los demás para casi todos los tamaños de carga útil.

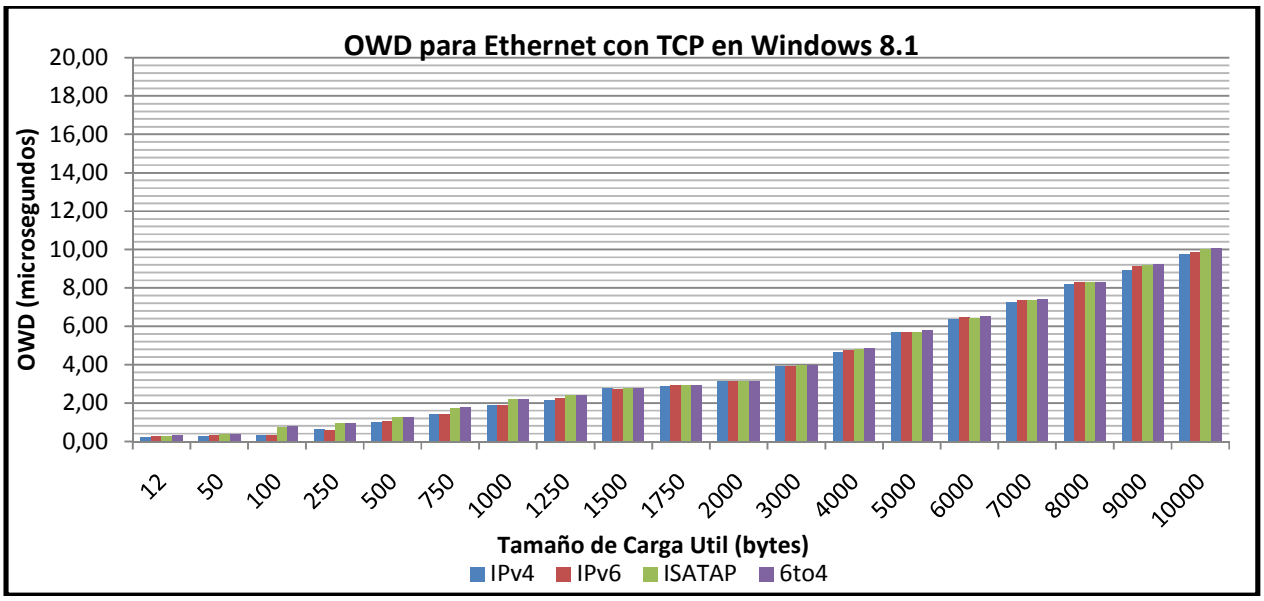


Figura 7.43: OWD para Ethernet con TCP en Windows 8.1

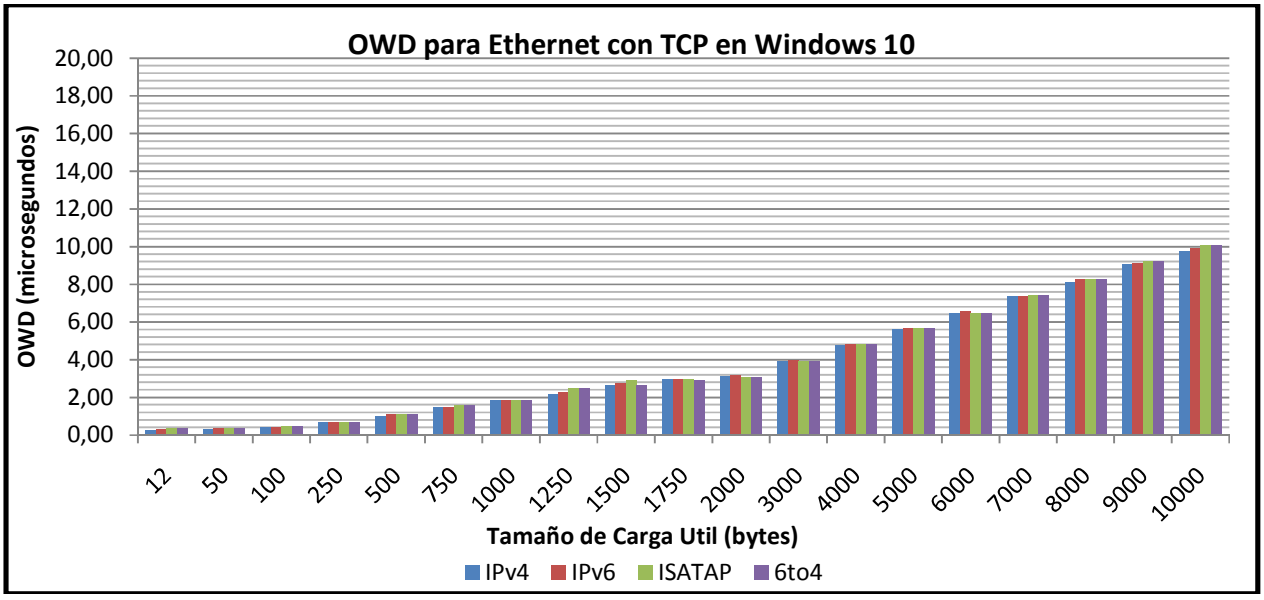


Figura 7.44: OWD para Ethernet con TCP en Windows 10

7.7.5 Resultados del OWD para Fast Ethernet con UDP

La Figura 7.45 muestra los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Fast Ethernet con el protocolo de transporte UDP sobre el sistema operativo Debian 7, en donde se evidencia que el OWD más alto es presentado por Tayga y el más bajo lo es presentado por IPv4. ISATAP y 6to4 presentan un retardo similar, que es superior que el de IPv4, IPv6 y Jool, pero inferior que el de Tayga. IPv6 presenta un retardo superior que el obtenido en Jool pero inferior que el de ISATAP, 6to4 y Tayga.

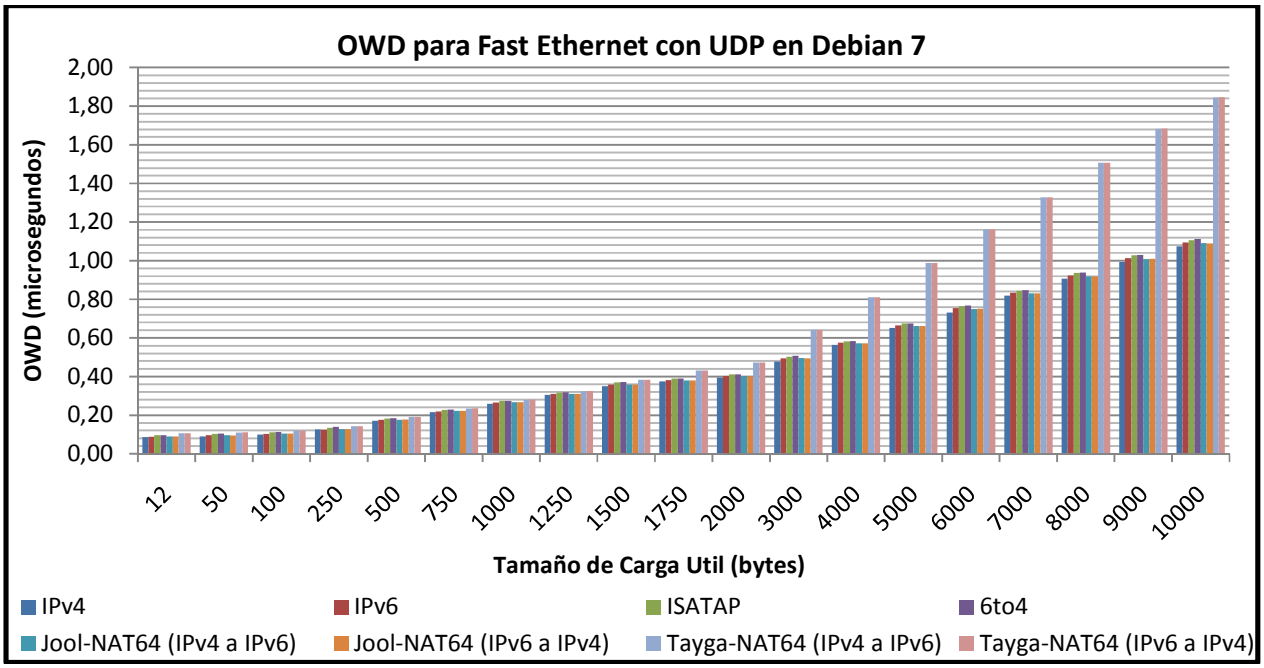


Figura 7.45: OWD para Fast Ethernet con UDP en Debian 7

Al igual que los resultados obtenidos para Ethernet, el retardo obtenido de los experimentos de Jool en ambas direcciones es superior que el de IPv4, pero inferior que el de IPv6, lo que indica que esta implementación de NAT64 presenta un rendimiento superior a Tayga en cuanto a tiempos de transmisión, resaltando que el retardo obtenido en Tayga supera a las demás tecnologías por casi el doble de microsegundos. Estos altos valores de OWD para Tayga representan un comportamiento inesperado.

La Figura 7.46 muestra los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Fast Ethernet con el protocolo de transporte UDP sobre el sistema operativo Windows 7, en donde se evidencia que el OWD más bajo es presentado por IPv4. Además, IPv6 presenta un retardo superior que el de IPv4, mientras que ISATAP presenta un retardo inferior que 6to4 pero superior que IPv6. 6to4 es la tecnología con peor rendimiento en cuanto a tiempos de transmisión ya que presenta el OWD superior.

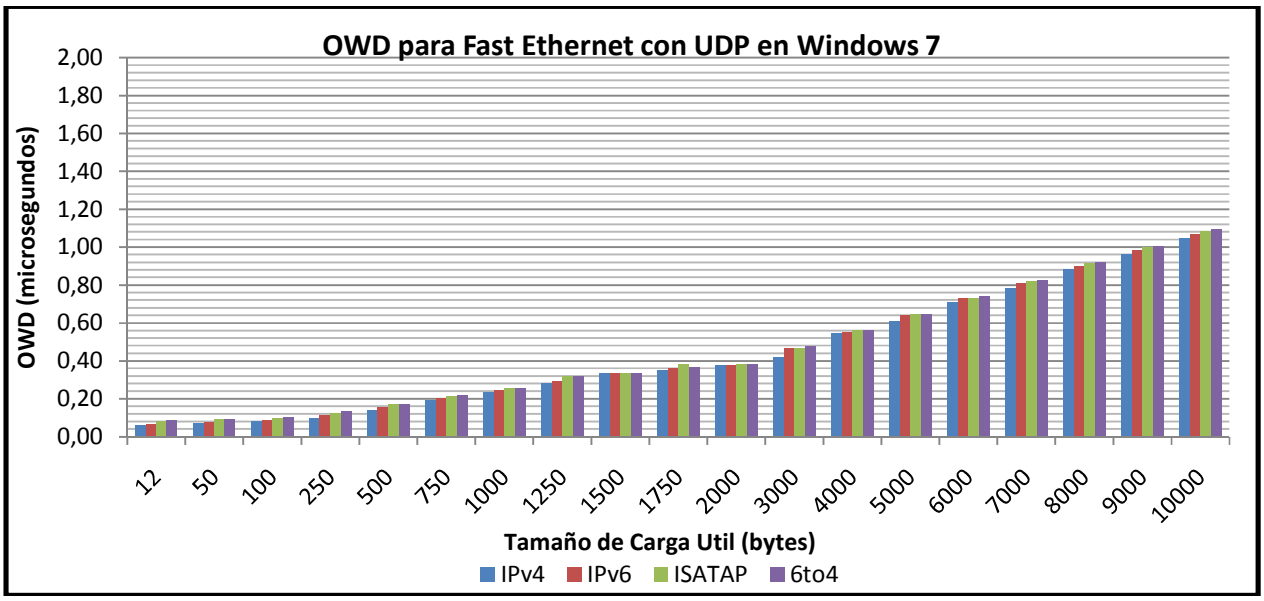


Figura 7.46: OWD para Fast Ethernet con UDP en Windows 7

Por otra parte, la Figura 7.47 muestra los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Fast Ethernet con el protocolo de transporte UDP sobre el sistema operativo Windows 8.1, en donde se evidencia que el OWD más bajo es presentado por IPv4 y el más alto es presentado por 6to4. Además, IPv6 presenta un retardo superior que el de IPv4 e inferior que el de ISATAP, mientras que ISATAP presenta un retardo inferior que 6to4.

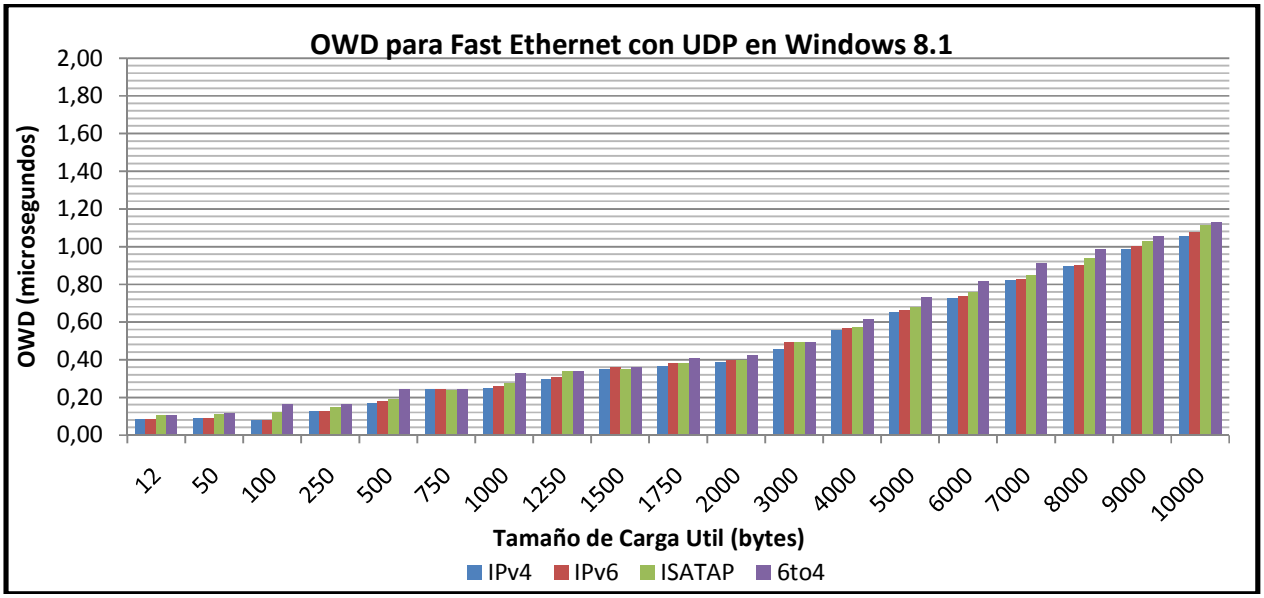


Figura 7.47: OWD para Fast Ethernet con UDP en Windows 8.1

La Figura 7.48 muestra los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Fast Ethernet con el protocolo de transporte UDP sobre el sistema operativo Windows 10, en donde se evidencia que el retardo más bajo es presentado por IPv4 e IPv6, quienes presentan un rendimiento similar. El OWD más alto es presentado por ISATAP y 6to4, quienes también poseen un rendimiento similar, aunque en algunos casos ISATAP supera a 6to4 (por ejemplo, cargas útiles de 750, 1000 y 7000 bytes).

Al nivel de los sistemas operativos, Windows 7 está funcionando mejor que los demás para casi todos los tamaños de carga útil.

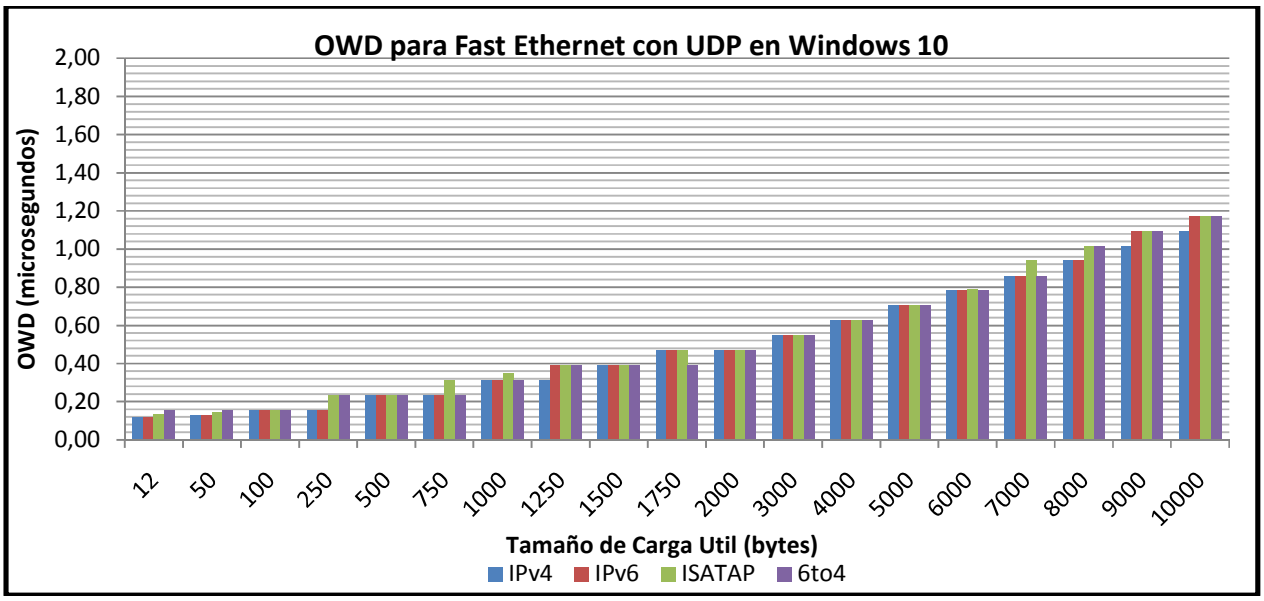


Figura 7.48: OWD para Fast Ethernet con UDP en Windows 10

7.7.6 Resultados del OWD para Fast Ethernet con TCP

Los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Fast Ethernet con el protocolo de transporte TCP sobre el sistema operativo Debian 7 se muestran en la Figura 7.49. Se puede observar que el OWD de ISATAP, 6to4 y Tayga presentan el retardo más alto mientras que IPv4 reporta el OWD más bajo. Además, IPv6 presenta un retardo superior que IPv4 pero inferior que el de ISATAP, 6to4 y Tayga, quienes a su vez presentaron un retardo similar. Jool presenta un retardo superior que IPv4 pero inferior que IPv6.

Por otra parte, la Figura 7.50 muestra los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Fast Ethernet con el protocolo de transporte TCP sobre el sistema operativo Windows 7. Se evidencia que el retardo más bajo es presentado por IPv4 y el más alto por 6to4.También se puede observar que ISATAP posee un retardo superior al de IPv6, pero inferior al de 6to4. IPv6 presenta un retardo superior que IPv4.

Además, cuando la carga útil es de 250, 500, 750 y 1000 bytes, ISATAP y 6to4 presentan una diferencia de casi el doble en microsegundos en comparación con IPv4 e IPv6, y cuando la carga útil es de 1250 bytes el retardo de ISATAP y 6to4 es inferior que el presentado por IPv4 e IPv6.

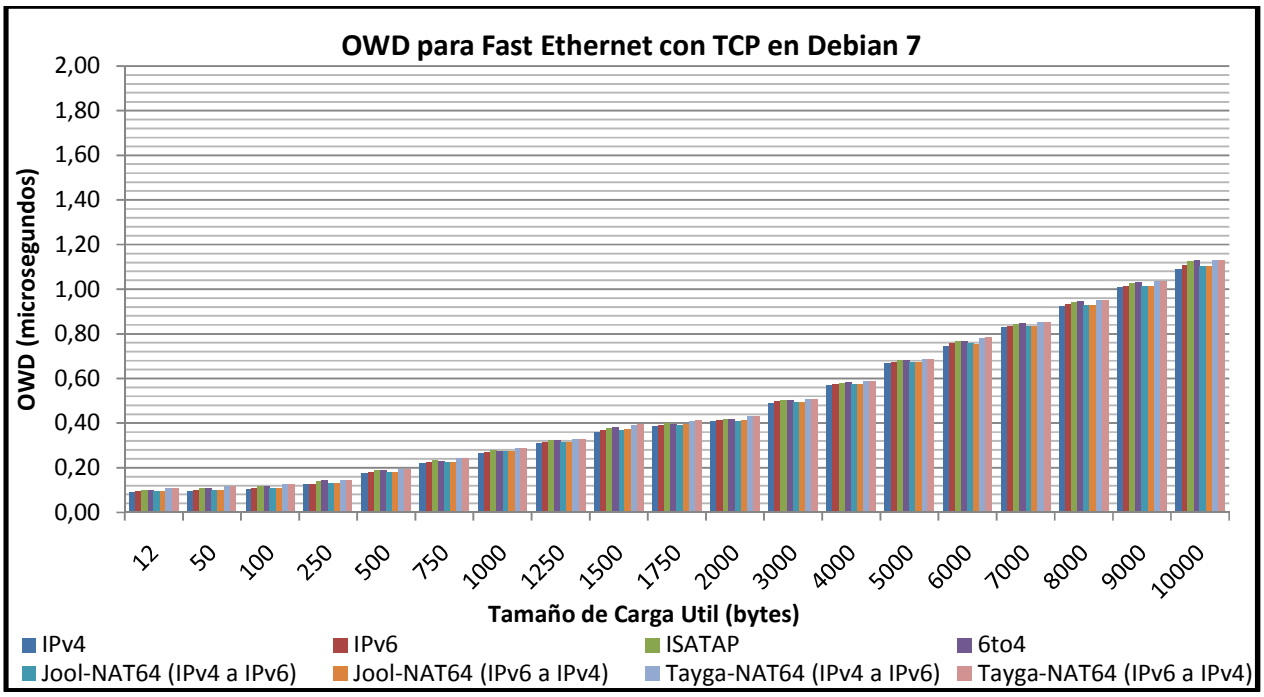


Figura 7.49: OWD para Fast Ethernet con TCP en Debian 7

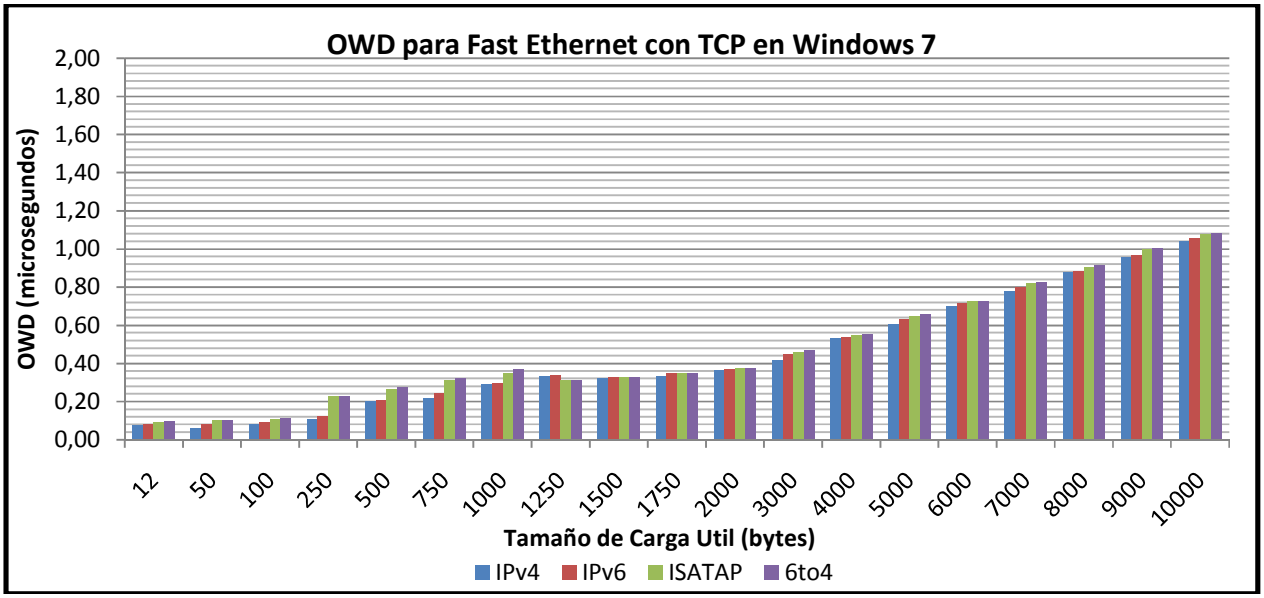


Figura 7.50: OWD para Fast Ethernet con TCP en Windows 7

La Figura 7.51 muestra los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Fast Ethernet con el protocolo de transporte TCP sobre el sistema operativo Windows 8.1, donde se evidencia que 6to4 presenta el retardo más alto. También se puede observar que ISATAP presenta un retardo inferior a 6to4, pero superior que IPv6, mientras que IPv6 tiene un retardo superior que IPv4. Además, cuando la carga útil es de 6000 bytes, ISATAP y 6to4 presentan un retardo inferior que IPv4 e IPv6.

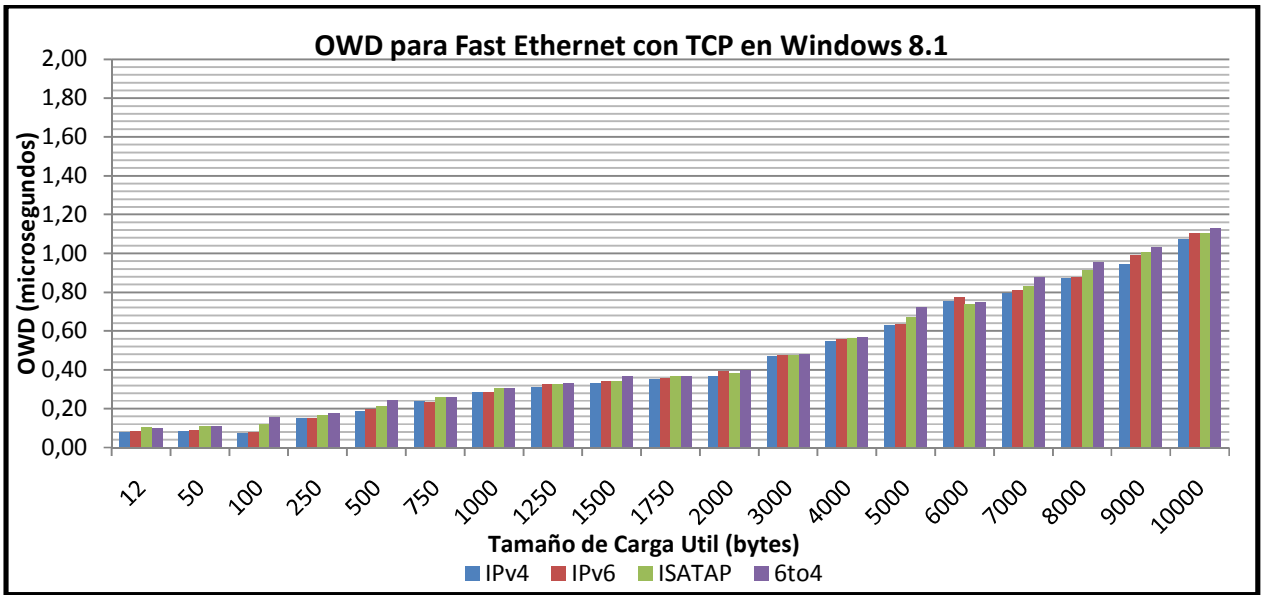


Figura 7.51: OWD para Fast Ethernet con TCP en Windows 8.1

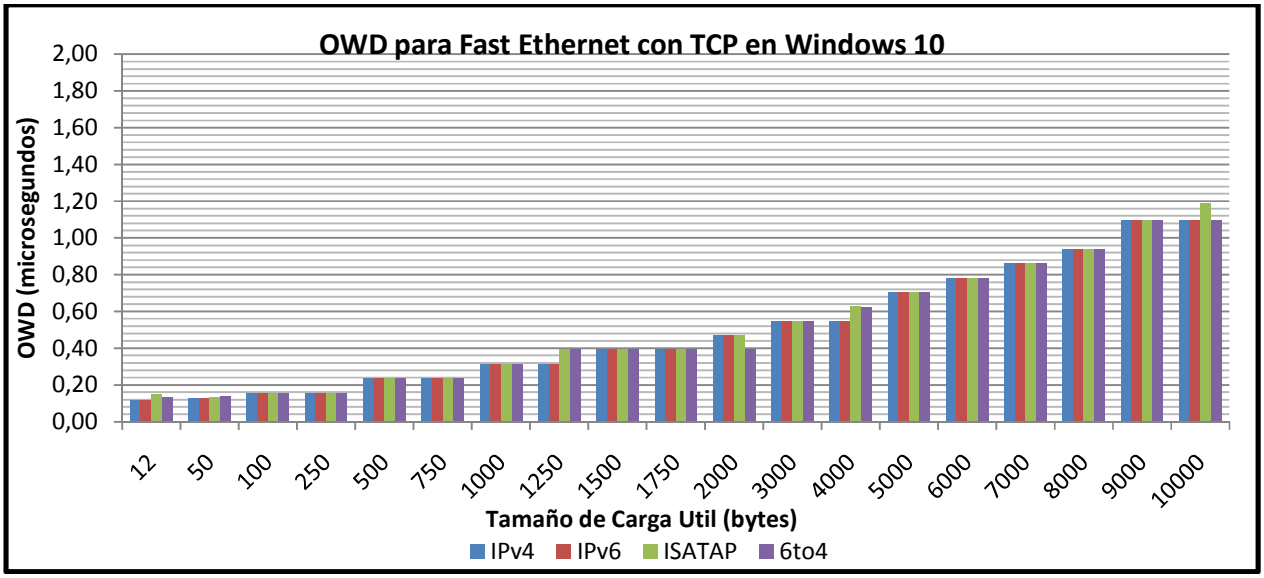


Figura 7.52: OWD para Fast Ethernet con TCP en Windows 10

Finalmente, la Figura 7.52 muestra los resultados del OWD obtenidos durante los experimentos de cada uno de los mecanismos para Fast Ethernet con el protocolo de transporte TCP sobre el sistema operativo Windows 10. En este caso, IPv4 presenta un retardo similar a IPv6, ISATAP y 6to4 en la mayoría de las cargas útiles evaluadas. Hay que resaltar que cuando la carga útil es de 12 y 4000 bytes, ISATAP presenta un retardo superior, seguido por 6to4, cuando la carga útil es de 1250 bytes ISATAP, y 6to4 es superior que IPv4 e IPv6. Particularmente, cuando la carga útil es de 2000 bytes el retardo en ISATAP es inferior al de IPv4, IPv6 y 6to4, además cuando la carga útil es de 10000 bytes el retardo de 6to4 es superior que el de ISATAP.

Al nivel de los sistemas operativos, Windows 7 está funcionando mejor que los demás sistemas operativos para casi todos los tamaños de carga útil.

8. Conclusiones

IPv4 provee los mecanismos necesarios para que la comunicación entre dispositivos sea efectiva. Sin embargo, en febrero de 2011, se agotó el pool de direcciones IPv4 de la IANA, volviéndose indispensable la implementación y pronta adopción de una nueva versión del protocolo de Internet, llamada IPv6.

Esta nueva versión del protocolo de Internet ha de ser implementada globalmente, ya que, provee un amplio pool de direcciones, recurso vital para el correcto funcionamiento del Internet. Por lo tanto es necesario desplegarlo, desarrollar aplicaciones IPv6, y dar soporte de IPv6. Como no se puede pasar de IPv4 a IPv6 en un periodo corto de tiempo, eso obligó a la comunidad a ir a un proceso de transición gradual entre las versiones del protocolo de Internet. De hecho, IPv4 e IPv6 no son “compatibles”, siendo indispensable el diseño de mecanismos de transición que permiten la coexistencia entre ellos.

Se han creado diferentes tecnologías de transición con características, teorías operativas y disponibilidad dependiente del entorno de red en que sea desplegado. Por lo cual, es de suma importancia realizar evaluaciones del desempeño de cada uno de los mecanismos de transición en escenarios lo más realistas posibles y que generen resultados significativos para el posterior uso en redes de gran escala.

En este Trabajo Especial de Grado se propuso un conjunto de escenarios que permitieron realizar una evaluación del desempeño de diferentes mecanismos de transición IPv4-IPv6. Se seleccionaron mecanismos populares de cada categoría, con diferentes rangos de aplicación y con disponibilidad de implementaciones sobre algunas distribuciones de Windows (Windows 7, Windows 8.1 y Windows 10) y Linux Debian 7, en escenarios reales con máximo tres dispositivos involucrados. Los mecanismos seleccionados se listan a continuación:(1) Dual Stack: IPv4 e IPv6 nativo; (2) túnel: ISATAP y 6to4; y (3) traducción: NAT64. En el caso de NAT64, se evaluó su desempeño con las implementaciones Tayga y Jool.

A partir de esta investigación se observó, que tanto en Ethernet como en Fast Ethernet, IPv4 nativo presenta un throughput superior al de IPv6 nativo en todos los casos. Esto es debido a la longitud de las cabeceras IP (20 bytes para IPv4, y 40 bytes para IPv6). Aunque IPv4 es bastante funcional y presenta los mejores resultados tanto en Linux como en Windows, debe ser reemplazado debido al agotamiento de las direcciones IPv4. Su mejor sustituto es IPv6 nativo que tiene el segundo mejor throughput y es la solución que se va a imponer a largo plazo. Además todos los sistemas operativos modernos para PCs evaluados tienen buen soporte para IPv6, un desempeño de red muy similar yal nivel de sistemas operativos, no se puede decir que alguno supera totalmente los demás. Eso depende de la tecnología a escoger.

En el caso de no poder utilizar IPv6 de forma nativa, los experimentos realizados demostraron que tanto ISATAP como 6to4 tienen un throughput similar, que es el más bajo de las tecnologías estudiadas, a causa de la cabecera IPv4 adicional en el túnel. Sin embargo, ambas técnicas de tunelización son buenas posibilidades. Su elección dependerá

del entorno de la red que se desea desplegar tomando en cuenta factores como: las necesidades del sitio, la disponibilidad de la tecnología, y la adquisición de un bloque de direcciones IPv6 nativas. Los administradores de red deben elegir el que mejor se adapte a sus necesidades, o mezclar ambas tecnologías, si es necesario. Vale la pena recordar que, si bien ISATAP tiene mejor rendimiento (tomando en cuentas resultados tanto del throughput como del OWD), cada mecanismo de transición se ejecuta en diferentes escenarios. Si el requisito es conectar dispositivos dispersos a Internet IPv6, probablemente ISATAP es la tecnología más adecuada. Para conectar todo un sitio a IPv6, 6to4 es el más adecuado.

Cuando se requiere un traductor, NAT64 es la mejor solución, ya que es un estándar del IETF y su mayor competidor (NAT-PT) ha sido declarado obsoleto. Dado que NAT64 es una tecnología reciente y está todavía en evolución, es compatible con pocos fabricantes, tales como router recientes Cisco Systems y Linux. A pesar de que en Linux existen varias implementaciones de código abierto de NAT64 (Jool, Tayga, WrapSix y Ecdysis), en este trabajo, solo comparamos dos de ellas: Tayga y Jool. Ecdysis fue descartado debido a que es un proyecto muerto poco funcional, y Wrapsix fue descartado ya se necesita modificar el código fuente cada vez que se desee cambiar un parámetro al momento de hacer las pruebas. Además, con Wrapsix los resultados reportados son inestables y generan errores al momento de la transmisión de datos.

De acuerdo a los experimentos hechos, NAT64 tiene un buen desempeño. En este caso, nuestra recomendación es utilizar Jool porque es un proyecto activo, no mostró problemas de throughput y el OWD presentado es inferior al de IPv6 nativo en casi todos los casos, a pesar de la traducción. La última versión de Tayga fue lanzado en junio de 2011, y parece ser un proyecto muerto aunque funcional. En las pruebas el desempeño de Jool mostró ser ligeramente superior al presentado por Tayga.

La utilización de NAT64 se recomienda como última opción, ya que a pesar de que el throughput de NAT64 es similar al de IPv4 e IPv6 nativo, el OWD es relativamente alto con UDP tanto para Ethernet como para Fast Ethernet y las diferentes implementaciones presentan limitaciones importantes.

Por otro lado, se obtuvieron resultados poco realistas e inestables para los pequeños tamaños de carga útil, que se pudieran producir por la evaluación comparativa hecha por la herramienta utilizada.

Los estudios presentados en este trabajo se realizaron en escenarios de pruebas reales en un entorno controlado. Para trabajos futuros sería conveniente considerar redes más complejas, tecnologías más recientes y la evaluación de otras técnicas de transición tales como Teredo.

Bibliografía

[01] Q. Li, T. Jinmei, and K. Shima. IPv6 Core Protocols Implementation.Morgan Kaufman. Octubre 2006.

[02] S. Deering and R. Hinden. Internet Protocol version 6 (IPv6) Specification. RFC 2460. Diciembre 1998.

[03] J. Davies. Understanding IPv6. 2nd Edition. Microsoft Press. Febrero 2008.

[04] S. Hagen. IPv6 Essentials, 2nd Edition. O'Reilly. Mayo 2006.

[05] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291. Febrero 2006.

[06] B. Carpenter and K. Moore. Connection of IPv6 Domains via IPv4 Clouds. RFC 3056. Febrero 2001.

[07] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303. Diciembre 2005.

[08] B. Stockebrand. IPv6 in Practice. Springer. Febrero 2007.

[09] R.Hinden andB.Haberman. Unique Local IPv6 Unicast Addresses. RFC 4193. Octubre 2005.

[10] E. Nordmark and R. Gilligan. Basic Transition Mechanisms for IPv6 Hosts and Routers. RFC 4213. Octubre 2005.

[11] C. Taffernaberry, G. Mercado, S. Pérez, y R. Moralejo. CODAREC6: Transición. Implementación de TunnelBroker. XIV Congreso Argentino de Computación. Octubre 2008.

[12] F. Templin, T. Gleeson and D. Thaler. Intra-SiteAutomatic Tunnel Addressing Protocol (ISATAP). RFC 5214. Marzo 2008.

[13] C. Huitema. Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs). RFC 4380. Febrero 2006.

[14] A. Conta and S. Deering. Generic Packet Tunneling in IPv6 Specification. RFC 2473. Diciembre 1998.

[15] A. Durand, P. Fasano, I. Guardini and D. Lento. IPv6 Tunnel Broker. RFC 3053. Enero 2001.

[16] B. Carpenter. Advisory Guidelines for 6to4 Deployment. RFC 6343. Agosto 2011.

[17] C. Huitema. An Anycast Prefix for 6to4 Relay Routers. RFC 3068. Junio 2001.

[18] D. Thaler, S. Krishnan and J. Hoagland. Teredo Security Updates. RFC 5991. Septiembre 2010.

[19] P. Savola and C. Patel. Security Considerations for 6to4. RFC 3964. Diciembre 2004.

[20] B. Carpenter and C. Jung. Transmission of IPv6 over IPv4 Domains without Explicit Tunnels. RFC 2529. Marzo 1999.

[21] R. Despres. IPv6 Rapid Deployment on IPv4 Infrastructures (6rd). RFC 5569. Enero 2010.

[22] W. Townsley and O. Troan. IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) Protocol Specification. RFC 5969. Agosto 2010.

[23] B. Storer, C. Pignataro, M Dos Santos, B Stevant, L Toutain anda J. Tremblay. Softwire Hub and Spoke Deployment Framework with Layer Two Tunneling Protocol Version 2 (L2TPv2). RFC 5571. Junio 2009.

[24] M. Blanchet and F. Parent. IPv6 Tunnel Broker with the Tunnel Setup Protocol (TSP). RFC 5572. Febrero 2010.

[25] J. Bound, L. Toutain and JL. Richier. Dual Stack IPv6 Dominant Transition Mechanism (DSTM). Draf-bound-dstm-exp-04.txt. Octubre 2005.

[26] R. Despres, S. Jiang, R. Penno, Y. Lee, G. Chen and M. Chen. IPv4 Residual Deployment via IPv4 – a Stateless Solution (4rd). Draft-ietf-softwire-4rd-08.txt. Abril 2014.

[27] L. Min, J. Mingye and L. Defeng. Tunneling IPv6 with private IPv4 addresses through NAT devices. Draft-liumin-v6ops-silkroad-02.txt. Nobiembre 2004.

[28] M. Bagnulo, P. Matthews and I. van Beijnum. Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. RFC 6146. Abril 2011.

[29] M. Bagnulo, A. Sullivan, P. Matthews and I. van Beijnum. DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers. RFC 6147. Abril 2011.

[30] D. Liu and H. Deng. NAT46 considerations. Draft-liu-behave-nat46-02.txt. Marzo 2010.

[31] M. Wasserman and F. Baker. IPv6-to-IPv6 Network Prefix Translation. RFC 6296. Junio 2011.

[32] G.Tsirtsis and P. Srisuresh. Network Address Translation-Protocol Translation (NAT-PT). RFC 2766. Febrero 2000.

[33] C. Aoun and E. Davies. Reasons to Move the Network Address Translator-Protocol Translator (NAT-PT) to Historic Status. RFC 4966. Julio 2007.

[34] J. Hagino and K. Yamamoto. An IPv6-to-IPv4 Transport Relay Translator. RFC 3142. Junio 2001.

[35] K. Tsuchiya, H. Higuchi and Y. Atarashi. Dual Stack Hosts using the “Bump-In-the-Stack” Technique (BIS). RFC 2767. Febrero 2000.

[36] S. Lee, A-K. Shin, Y-J. Kim, E. Nordmark and A. Durand. Dual Stack Hosts Using “Bump-in-the-API” (BIA). RFC 3388. Octubre 2002.

[37] H. Kitamura. A SOCKS-based IPv6/IPv4 Gateway Mechanism. RFC 3089. Abril 2001.

[38] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas and L. Jones. SOCKS Protocol V5. RFC 1928. Abril 1996.

[39] D. Wing. Transmission of IPv6 over IPv4 Domains without Explicit Tunnels. RFC 2530. Marzo 1999.

[40] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair and X. Li. IPv6 Addressing of IPv4/IPv6 Translators. RFC 6052.Octubre 2010.

[41] E. Nordmark. Stateless IP/ICMP Translation Algorithm (SIIT). RFC 2765. Febrero 2000.

[42] X. Li, C. Bao and F. Baker. IP/ICMP Translation Algorithm. RFC 6145.Abril 2011.

[43] S. Lee, M-K. Shin, Y-J. Kim, E. Nordmark and A. Durand. Dual Stack Hosts Using “Bump-in-the-API” (BIA). RFC 3338. Octubre 2002.

[44] R. Bush. The Address plus Port (A+P) Approach to the IPv4 Address Shortage. RFC 6346. Agosto 2011.

[45] C. Huitema and B. Carpenter. Deprecating Site Local Addresses. RFC 3879.Septiembre 2004.

[46] R. Hinden and S. Deering. IPv6 Multicast Address Assignments. RFC 2375.Julio 1998.

[47] B. Foster and F. Andreasen. Media Geteway Control Protocol (MGCP) Redirect and Reset Package. RFC 3991. Febrero 2005.

[48] P. Srisuresh and K. Egevang. Traditional Ip Network Address Translator (Traditional NAT). RFC 3022. Enero 2001.

[49] A. Durand, R. Droms, J. Woodyatt and Y. Lee. Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion. RFC 6333. Agosto 2011.

[50] D. Hankisns and T. Mrugalski. Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Option for Dual-Stack Lite. RFC 6334. Agosto 2011.

[51] C. Byrne. IPv4 Service Continuity Prefix. RFC 7335. Agosto 2014.

[52] F. Audet and C. Jennings. Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. RFC 4787. Enero 2007.

[53] S. Guha, K. Biswas, B. Ford, S. Sivakumar and P. Srisuresh. NAT Behavioral Requirements for TCP. RFC 5382. Octubre 2008.

[54] P. Srisuresh, B. Ford, S. Sivakumar and S. Guha. NAT Behavioral Requirements for ICMP. RFC 5508. Abril 2009.

[55] J. Rosenberg. Interactive Connectivity Establisment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245. Abril 2010.

[56] L. Colitti, S. H. Gunderson, E. Kline, and T. Refice. Evaluating IPv6 Adoption in the Internet. 11th International Conference on Passive and Active Measurement (PAM'10). Zurich, Switzerland, pp. 141-150.Abril 2010.

[57] Kc. Claffy. Tracking IPv6: Data We Have and Data We Need. ACM SIGCOMM Computer Communication Review. Vol. 41, no. 3,pp. 43-48. Julio 2011.

[58] S. Narayan, P. Shang, and N. Fan. Performance Evaluation of IPv4 and IPv6 on Windows Vista and Linux Ubuntu. 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC'09). Vol. 1. Wuhan, China, pp. 653-656. Abril 2009.

[59] S. Zeadally and I. Raicu. Evaluating IPv6 on Windows and Solaris. IEEE Internet Computing. Vol. 7, no. 3, pp. 51-57. Mayo 2003.

[60] A. Botta, A. Dainotti, and A. Pescapé. Multi-Protocol and Multi-Platform Traffic Generation and Measurement. INFOCOM 2007 Demo Session. Mayo 2007.

[61] P. Jain, S. Singh, G. Singh and C. Goel. Performance Comparison of IPv4 and IPv6 using Windows XP and Windows 7 over Gigabit Ethernet LAN. International Journal of Computer Applications (0975-8887). Vol. 43, no 16. Abril 2012.

[62] P. Raymond. Performance Analysis of IPv4 vs. IPv6 on various Operating Systems using Jumbo Frames. UNITEC New Zealand. 2011.

[63] E. Gamess and R. Surós. An Upper Bound Model for TCP and UDP Throughput in IPv4 and IPv6. Journal of Network and Computer Applications. Vol. 31, no 4, pp. 585-602. Noviembre 2008.

[64] J. Balen, G. Martinovic, and Z. Hocenski. Network Performance Evaluation of Latest Windows Operating Systems. 20th International Conference on Software,

Telecommunications and Computer Networks (SoftCOM). Vol. 7, pp. 1-6. Split, Croatia. Septiembre 2012.

[65] M. Aazam, S. A. H. Shah, I. Khan, and A. Qayyum. Deployment and Performance Evaluation of Teredo and ISATAP over Real Test-bed Setup. International Conference on Management of Emergent Digital EcoSystems (MEDES'10). Bangkok, Thailand. pp. 229-233. Octubre 2010.

[66] N. Bahaman, A. S. Prabuwono, R. Alsaqour and M. Z. Mas`ud. Network Performance Evaluation of Tunneling Mechanism. Journal of Applied Sciences. pp. 459-465. Abril 2012.

[67] S. Zander, L. L. H. Andrew, G. Armitage, G. Huston, and G. Michaelson. Investigating the IPv6 Teredo Tunnelling Capability and Performance of Internet Clients. ACM SIGCOMM Computer Communication Review. Vol. 42, no. 5, pp. 13-20. Octubre 2012.

[68] M. Aazam, S. A. H. Shah, I. Khan, and A. Qayyum. Deployment and Performance Evaluation of Teredo and ISATAP over Real Test-bed Setup. International Conference on Managementof Emergent Digital EcoSystems (MEDES'10). Bangkok, Thailand.Vol. 2, no 5, pp. 229-233. October 2010.

[69] V. Visoottiviseth and N. Bureenok. Performance Comparison of ISATAP Implementations on Free BSDF Red hat and Windows 2003. 22nd International Conference on Advanced Information Networking and Applications - Workshops. Bangkok, Thailand. pp. 547-552. 2008.

[70] S. M. Huang, Q. Wu, and Y.-B. Lin. Tunneling IPv6 through NAT with Teredo Mechanism. 19th International Conference on Advanced Information Networking and Applications. Taipei, Taiwan. Vol. 2, pp. 813-818. Marzo 2005.

[71] S. Narayan and S. Tauch. IPv4-v6 Configured Tunnel and 6to4 Transition Mechanisms Network Performance Evaluation on Linux Operating Systems. 2nd International Conference on Signal Processing Systems (ICSPS). Dalian, China. Vol. 2. Octubre 2010.

[72] S. S. Mohamed, A. Y. M. Abusin and D. Chieng. Evaluation of IPv6 and Comparison Study with Different Operating Systems. Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05). 2005.

[73] F. Sans and E. Gamess. Analytical Performance Evaluation of Native IPv6 and Several Tunneling Technics using Benchmarking Tools.XXXIX Conferencia Latinoamericana en Informática, CLEI 2013, Volume II Club Puerto Azul, Naiguatá, Venezuela. pp.146-154. Octubre 2013.

[74] K. Velásquez y E. Gamess. Análisis Comparativo de Herramientas de Evaluación de Desempeño en Redes de Computadores. Centro de Investigación en Comunicación y Redes (CICORE) Caracas. Noviembre 2009.

[75] E. Gamess. Arquitectura para la Construcción Orientada por Rendimiento de Meta sistemas. Tesis Doctoral. Universidad Central de Venezuela. Junio 2000.

[76] S. Bradner. Benchmark Terminology for Network Interconnection Devices. RFC 1242. Julio 1991.

[77] S. Avallone, A. Botta, A. Dainotti, W. De Donato and A. Pescapé. D-ITG V. 2.6.1d Manual. Mayo 2008.

[78] R. Stevens, B. Fenner and A. Rudoff. UNIX Network Programming: The Sockets Networking API. Volume I. Third Edition. Addison Wesley. 2004.

[79] K. Velásquez and E. Gamess. A Survey of Network Benchmark Tools. Machine Learning and Systems Engeneering. Springer. pp. 465-480. Octobre 2010.

[80] R. Prasad, M. Murray, C. Dovrolis and K. Claffy. Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. IEEE Network. pp. 27-35. Abril 2003.

[81] H. Unandery and W. Wenjuan. Accurate Measurement and Visualization of Traffic Load in IEEE 802.11 WLANs. Master Thesis. University of Agder. Norway. May 2008.

[82] E. Gamess and K. Velásquez. IPv4 and IPv6 Forwarding Performance Evaluation on Different Operating Systems. XXXIV Conferencia Latinoamericana de Informática (CLEI 2008). September 2008.