PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

IMPLEMENTACIÓN DE MECANISMOS DE TRANSICIÓN AL PROTOCOLO IPv6 EN VNUML Y EN UNA RED WINDOWS

Tesis para optar el **Grado de Magister en Ingeniería de las Telecomunicaciones**, que presenta:

JESÚS MARCO VIVAS RUIZ

ASESOR:

Dr. JOSE LUIS MUÑOZ TAPIA

JURADO:

Mg. GUMERCINDO BARTRA GARDINI

Mg. ANTONIO OCAMPO ZÚÑIGA

LIMA - 2017

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El objetivo de la presente tesis es implementar un laboratorio virtual para simular los mecanismos de transición vigentes en redes IPv4/IPv6, aplicándolos en el enrutamiento dinámico, y para facilitar su implementación en una red de área local con usuarios Windows.

La metodología usada en este documento consiste en, por un lado, establecer un enlace a un repositorio remoto con el uso de la herramienta Apache Subversion. En donde, bajo la supervisión del Asesor de la tesis, se realiza el control de versiones de los escenarios virtuales y del documento de la tesis. Así mismo, a través de reuniones presenciales y virtuales con la herramienta de comunicación Skype, se verifica el avance del presente trabajo.

Por otro lado, en cuanto al desarrollo de la tesis, en primer lugar, se diseña la topología de los escenarios para su posterior implementación a través de la lógica del lenguaje de programación de la herramienta de virtualización. En segundo lugar, se realiza la ejecución de pruebas de conectividad y eficiencia de los escenarios virtuales. Y, en tercer lugar, a partir del resultado de las pruebas obtenidas en los escenarios virtuales, se selecciona los mecanismos adecuados para lograr la migración de una red con clientes Windows a IPv6. Esto, sin afectar los servicios y la operatividad de la red.

En el Capítulo I, se describe la necesidad de adoptar el protocolo IPv6. Se exponen las ventajas del protocolo IPv6 sobre IPv4. Además, se analiza los mecanismos de traducción, NAT64+DNS64, y de túneles, 6to4 y 6RD; se describe el protocolo ICMPv6, el papel que cumple en el direccionamiento dinámico IPv6 y en el sistema de nombres de dominio (DNS) en IPv6. Así mismo, se muestra la situación actual de la Red Académica Peruana RAAP y de los operadores de telecomunicaciones en la adopción de IPv6. En relación a la tendencia actual de los mecanismos de transición en las redes móviles, se describe el estado y uso de las tecnologías, como Dual-Stack y 464XLAT.

Por otro lado, el primer aporte de la tesis se muestra en el Capítulo II. La implementación de los escenarios virtuales utilizando una herramienta desarrollada por la Universidad Politécnica de Madrid llamada VNUML. En donde, el único requerimiento que permite ejecutar los escenarios completos es adecuar un solo

equipo anfitrión con capacidades de procesamiento y almacenamiento necesarios para la óptima ejecución de cada máquina virtual. En ese sentido, a través del diseño e implementación de cada escenario virtual que corresponde a los mecanismos de traducción NAT64 y DNS64, y los túneles 6to4 y 6RD, se demuestra la capacidad de la herramienta VNUML en el diseño, ejecución y comprobación de los aspectos teóricos señalados en el primer capítulo de esta tesis.

El siguiente aporte de esta tesis se muestra en el Capítulo III. La migración de una red privada de usuarios Windows a IPv6. En primer lugar, se hace un despliegue dinámico de direcciones IPv6 a través de DHCPv6 + SLAAC. En segundo lugar, el acceso a los servicios e Internet IPv4 se realiza a través de NAT64 stateful y DNS64. Y, en tercer lugar, el acceso a Internet IPv6 se hace posible con el mecanismo NAT66. De esta manera, se demuestra que es factible desplegar IPv6 de forma transparente para los usuarios finales.

INTRODUCCIÓN

La adopción del protocolo de comunicaciones versión 6 (IPv6) es una realidad presente, muy poco difundida, desde hace muchos años. Es entonces que, la creciente demanda de teléfonos inteligentes y la tendencia de tener todo tipo de dispositivo conectado a Internet, a través del Internet de las Cosas (IoT), haya encendido las alarmas por el inminente agotamiento de direcciones del actual protocolo de comunicaciones versión 4 (IPv4). Debido a esto, nace el interés por la elaboración del presente estudio para poner en práctica un conjunto de mecanismos para la adopción de IPv6.

El presente trabajo consiste en estudiar las características de IPv6; desde las más simples, hasta aquellas que comprenden la asignación automática de direcciones. Con lo cual, se proporciona al lector un entorno ilustrado de pruebas basadas en simulaciones y de aplicación real en una red de área local (LAN).

Este estudio se fundamenta en el hecho de que IPv6 no fue diseñado para ser compatible con versiones anteriores. Dicho de otra forma, los nodos concebidos en IPv6 no pueden comunicarse directamente con los nodos IPv4 actuales. En ese sentido, para lograr una convivencia en un entorno heterogéneo, es vital el uso de tecnologías de transición.

Por consiguiente, el objetivo de este estudio es identificar e implementar los siguientes mecanismos de transición para la adopción de IPv6: Túneles 6to4, Túneles 6RD y DNS64 + NAT64. Además, la aplicación de los mismos en el Enrutamiento Dinámico IPv6 en un entorno Windows.

En el capítulo I, se describe el marco teórico. En primer lugar, se analiza el problema, se señala la hipótesis y los objetivos del presente trabajo. En segundo lugar, se sustenta el fundamento teórico. En tercer lugar, se analiza la situación actual de la Red Académica Peruana (RAAP) y el papel de los operadores de Telecomunicaciones, en el proceso de implementación de IPv6 en el Perú. En cuarto lugar, se muestra el estado de la tendencia actual de las tecnologías de transición para la adopción de IPv6. Finalmente, se detalla la metodología a usar en este estudio.

En el capítulo II, se realiza el análisis, en laboratorios virtuales basado en Linux, de escenarios y configuraciones de los mecanismos de transición, como son los Túneles 6to4, Túneles 6RD y DNS64 + NAT64.

En el capítulo III, se muestra una aplicación práctica de migración de una red Windows a IPv6.

En las conclusiones se detalla los alcances del cumplimento de los objetivos propuestos.

Finalmente, en los anexos, se muestra las líneas de programación de cada escenario implementado en el Capítulo II.

CAPITULO I: MARCO TEÓRICO

En el presente capítulo se muestran los aspectos relacionados con la transición a IPv6. Por un lado, se señala el problema actual de la adopción de IPv6; se menciona la posición del regulador de las telecomunicaciones, y cuáles son las acciones de los proveedores de Internet frente al uso creciente de dispositivos móviles. Además, se incluye la hipótesis y los objetivos, y, en el fundamento teórico, se define generalidades sobres los mecanismos para la adopción de IPv6. De otro lado, se describe el estado de las tendencias en cuanto a tecnologías de transición en las redes IPv4/IPv6. Finalmente, se agrega la metodología.

1.1. Problema

Según Google, la adopción del protocolo IPv6 en el Perú alcanza el 15%, como lo muestra la figura Nº1. Telefónica del Perú lidera el despliegue de IPv6 con el 24.01%.[1] Sin embargo, el Organismo Supervisor de Inversión Privada en Telecomunicaciones, en su Propuesta de Reglamento sobre Neutralidad de Red, señala como requisito: La adopción del protocolo IPv6 en ningún caso afectará ni restringirá la normal utilización del acceso a Internet por parte de los usuarios, incluidos el uso de aplicaciones, protocolos, servicios, o cualquier tipo de tráfico.[2]

Ante ello, los proveedores de Internet tienen dos caminos. Por un lado, no hacer nada, y retrasar el despliegue de IPv6, dando lugar a problemas y el incremento de costos para los MNO[1]. Para lo cual, tendrán que seguir resolviendo la dificultad del aumento de la demanda de direcciones IP con CG-NAT[2]. En consecuencia, el crecimiento de la demanda de ancho de banda solo puede ser manejado aumentando la capacidad de CG-NAT. Sin embargo, se espera que en algún momento ciertas páginas web o aplicaciones en Internet solo estarán disponibles en IPv6.[3]

Por otro lado, está el uso de mecanismos de transición para la adopción de IPv6, situación que impacta en el objetivo principal de esta tesis.

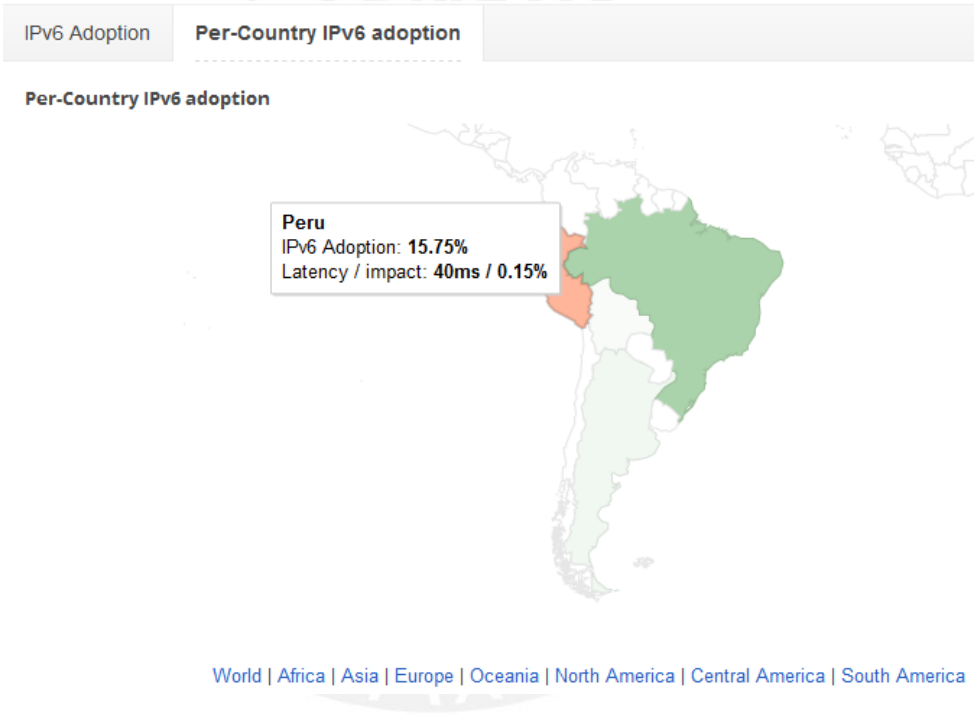


Fig. Nº 1 - Adopción de IPv6 en el Perú [4]

1.2. Hipótesis

Un laboratorio virtual para la simulación de los mecanismos de transición (NAT64, DNS64), y de túneles (6to4, 6RD), que fundamente la migración de una red de área local Windows, de forma transparente para los usuarios y que considere la convivencia de los protocolos IPv4 e IPv6.

[1] Operador de Red Móvil
[2] Carrier Grade – NAT, Traducción de direcciones de red a gran escala

1.3. Objetivos

1.3.1. Objetivo Principal

Implementar un laboratorio virtual para simular los mecanismos de transición vigentes en redes IPv4/IPv6, aplicándolos en el enrutamiento dinámico, y para facilitar su implementación en una red de área local Windows.

1.3.2. Objetivos Específicos

a. Realizar un análisis de la tendencia actual de los países que cuentan con IPv6 desplegado en sus redes.

b. Proponer una arquitectura virtual de red IPv4/IPv6, para asegurar que la convivencia entre ambos protocolos sea fiable.

c. Diseñar y desarrollar un entorno virtual IPv4/IPv6 para implementar los túneles 6to4, túneles 6RD y DNS64 + NAT64, en un entorno Linux.

d. Validar el uso de la asignación de dinámica de direcciones IPv6 y de los mecanismos de transición con el fin de determinar el que mejor se adapte a las necesidades de un entorno real con clientes Windows.

1.4. Fundamento Teórico

1.4.1. Necesidad de IPv6

La tarea de los routers es enviar datos entre las diferentes redes. Junto con IPv4, estos dispositivos proporcionan una ruta de comunicación entre dos hosts, cada uno ubicado en cualquier parte del mundo. Debido a la escasez de espacio de direcciones y mecanismos utilizados para la asignación de direcciones, algunas partes del mundo están empezando a quedarse sin direcciones.[5]
En consecuencia, algunas de las razones que motivan la migración a IPv6 son:

- **Agotamiento de direcciones IPv4.** Las direcciones IPv4 están basadas en un entorno de 32 bits que permite un máximo de 2^{32} direcciones únicas. Sin embargo, el incremento acelerado de dispositivos móviles, como los teléfonos inteligentes, y la tecnología del Internet de las cosas, motivan el agotamiento de las direcciones IPv4. En tal sentido, se desarrolló IPv6, con un direccionamiento de 128 bits, con capacidad de 2^{128} direcciones únicas.[6]

- **Infraestructura punto a punto.** Con IPv6 es posible identificar cada interface con una única dirección IP. No será necesario el uso de NAT, debido a que se elimina el tiempo de procesamiento de la cabecera IP, permitiendo mejorar la eficiencia del enrutamiento. Como resultado, se obtiene una red punto a punto en Internet con soporte nativo a las opciones de QoS[3].[6]

- **Tendencia a la obsolescencia.** IPv4 sobrevive gracias a la aplicación de actualizaciones continuas.[6]

- **Mejora del enrutamiento.** IPv6 adopta una estructura jerárquica de asignación de direcciones para acelerar la búsqueda de rutas y sistemas libres de gestionar, almacenar y actualizar en todo momento la tabla de enrutamiento.[6]

- **Restricción de broadcast.** Broadcast es reemplazado por las funcionalidades de multicast en IPv6; lo que permite una mejor administración de los recursos de red.[6]

- **Configuración automática de direcciones.** Para simplificar la configuración del host, IPv6 admite la configuración de direcciones con estado y sin estado. Lo que le permite al host adquirir una dirección desde un servidor DHCP[4], o crear una propia basada en la información obtenida de otros hosts en la misma capa de enlace.[6]

- **Complementos adicionales y cabeceras de extensión integrados en IPv4.** El paquete IPv6 ya no cuenta con el campo Opciones, en su lugar se incorpora varias cabeceras de extensión. De esta manera brinda soporte a múltiples opciones añadidas a IPv4 mediante complementos (IPsec, IP móvil, etc), y reduce la longitud de la cabecera básica con el fin de aliviar el tiempo de su procesamiento.[6]

- **Eliminación de ARP[5].** El Protocolo de Descubrimiento de Vecinos de IPv6 es un grupo de mensajes ICMPv6[6] que administra el intercambio de información entre nodos vecinos en el mismo enlace. ICMPv6 reemplaza los mensajes ARP,

[3] Calidad de Servicio
[4] Protocolo de configuración de host dinámico
[5] Protocolo de resolución de direcciones
[6] Protocolo de mensajes de control de internet versión 6

los mensajes de descubrimiento de routers ICMPv4 y los mensajes de redirección ICMPv4 para proveer otra serie de funciones.[6]

1.4.2. Tecnologías de Transición IPv6

La clave para una transición exitosa de IPv6 es la compatibilidad con la gran base instalada de hosts y routers IPv4. Al mantener la coexistencia con IPv4, se permite agilizar la tarea de transición a IPv6 en Internet. [7]

La versión 6 del protocolo de Internet no fue diseñado para ser compatible con versiones anteriores. En otras palabras, los nodos IPv6 nativos no pueden comunicarse directamente con nodos IPv4. Consecuentemente, en un entorno heterogéneo son necesarias tecnologías de convivencia y de transición. Para lo cual, en principio, se propusieron tres mecanismos básicos de transición: dual-stack, túneles y traducción.[8]

1.4.2.1. Dual Stack

En la actualidad todos los dispositivos soportan ambas versiones del protocolo IP. Siendo este el escenario ideal para la migración hacia IPv6, sin dejar de lado la conectividad IPv4. El protocolo IPv6 introduce cambios en la capa de internet de TCP/IP, y es transparente para el resto de capas que forman parte de la comunicación.[9]

Los protocolos de la capa de enlace de datos son capaces de transportar paquetes de IPv4 e IPv6.[9] Las características de estos dispositivos son:

- El dispositivo tendrá una dirección en cada pila. No es necesario que las direcciones IPv4 e IPv6 estén relacionadas entre sí. La asignación de las direcciones puede ser estática o dinámica.[10]
- Solo se duplica el nivel IP, no la pila completa.
- Se comunican usando:
  - IPv6 con nodos migrados
  - IPv4 con nodos no migrados

  Por consiguiente, un dispositivo con ambas pilas puede enviar y recibir tráfico a aquellos nodos que solo soportan uno de los dos protocolos (nativos en IPv4 o IPv6). Estos nodos pueden intercambiar tráfico con dispositivos IPv4 usando paquetes IPv4, y también operar con nodos IPv6 usando paquetes IPv6.[10]

- Incluyen bibliotecas de DNS (resolver) con la capacidad de tratar con registros A y AAAA. El DNS es el encargado de reconocer los nodos migrados, con registros AAAA disponibles.[10]

En la figura Nº 2 se muestra el funcionamiento de DualStack:

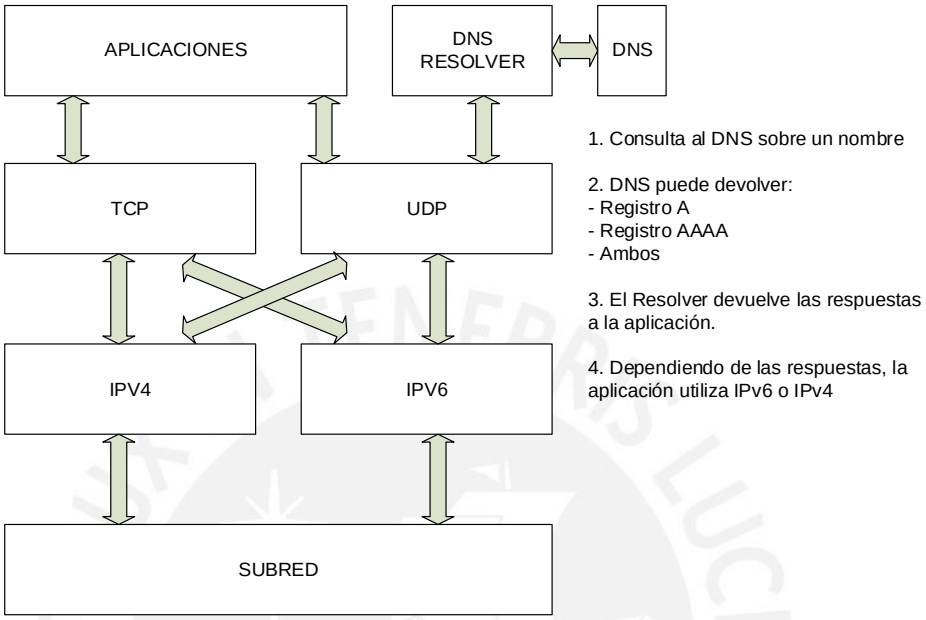


Fig. Nº 2 – Funcionamiento de DualStack [10]

El dispositivo que inicia la comunicación es quien elige el protocolo. La mayoría de los sistemas operativos prefieren IPv6.

Para tener una organización compatible con Dual Stack es necesario que todos los nodos tengan configurado IPv6. Además, el ISP debe proveer conectividad IPv6 para el acceso externo.

Por otro lado, una aplicación práctica de asignación de direcciones es tener una correspondencia entre IPv4 e IPv6. Por ejemplo, se tienen las siguientes redes 172.10.1.0/24, 172.10.2.0/24, 172.10.3.0/24 y 172.10.4.0/24. Para IPv6, la identificación para las redes en IPv6 sería: 2001:db8:7351:1::/64, 2001:db8:7351:2::/64, 2001:db8:7351:3::/64, 2001:db8:7351:4::/64.[9]

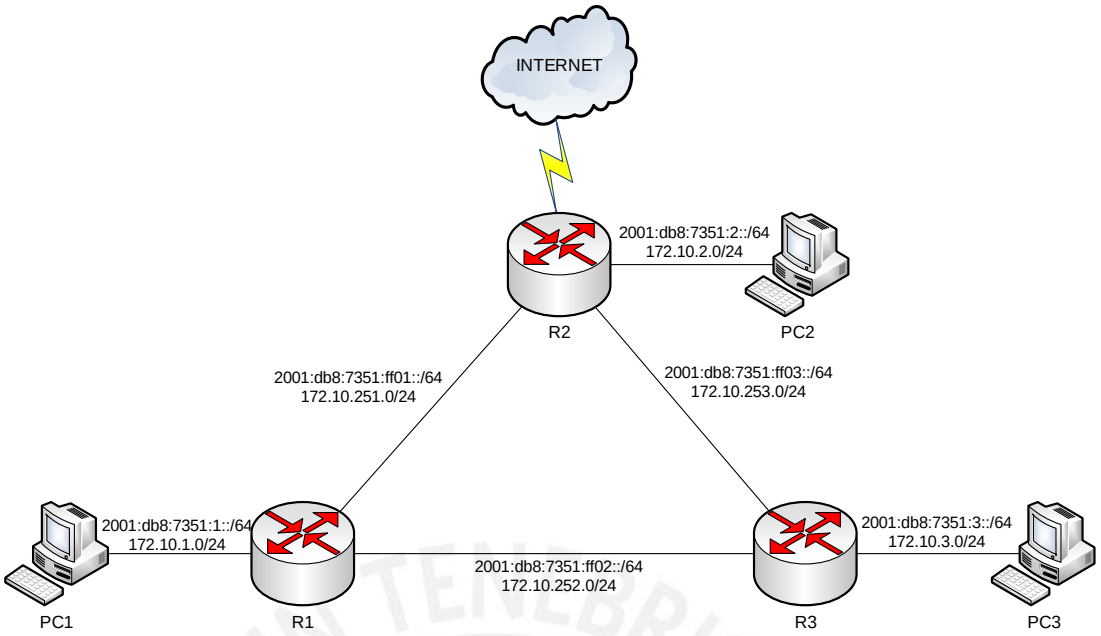


Fig. Nº 3 – Red Dual Stack

En la figura Nº3 se muestra una red habilitada con Dual Stack. Todos los segmentos están identificados con IPv4 e IPv6. Es posible usar protocolos de direccionamiento estático o dinámico. Lo ideal es que ambos protocolos de red utilicen el mismo protocolo de enrutamiento, y con una configuración adecuada las decisiones de enrutamiento serán las mismas.[9]

1.4.2.2. Túneles 6to4

Es un mecanismo definido en el RFC[7] 3056 “Connection of IPv6 Domains via IPv4 Clouds” [11]. El túnel 6to4 usa el prefijo IPv6 común (2002::/16), seguido por la dirección IPv4 pública, de 32 bits, del host 6to4. Por ejemplo, si un host con dirección IPv4 192.0.2.1 desea construir un túnel 6to4, puede obtener una dirección 6to4 que comience con el prefijo 2002:C000:201::/48[8]. En un escenario 6to4, los hosts pueden comunicarse entre sí a través de routers 6to4, figura Nº4 [12].

[7] Solicitud de comentarios
[8] Conversión, decimal a hexadecimal 192→C0, 0→00, 2→02, 1→01

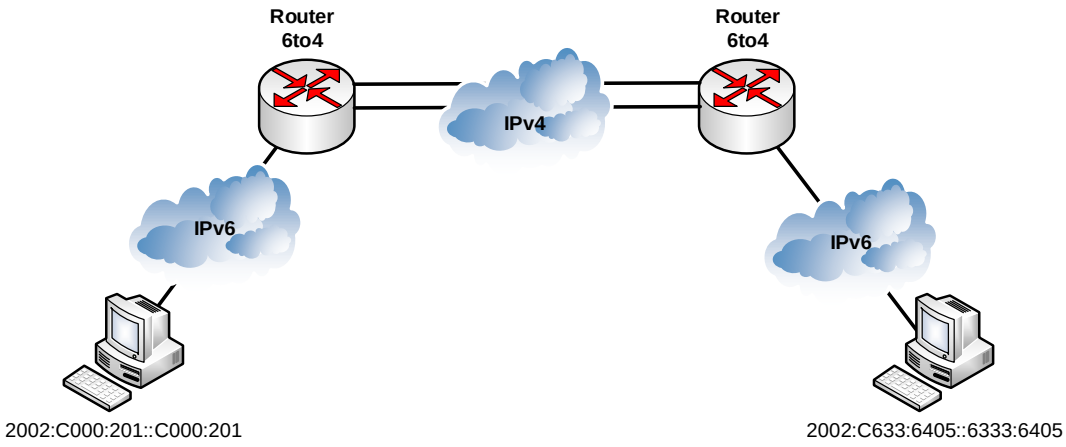


Fig. Nº 4 – Arquitectura comunicación 6to4 [12]

En otro escenario, figura Nº5, para concretar una comunicación entre un host 6to4 y un host IPv6 nativo, se necesita un router de reenvío 6to4. Para permitir que un host 6to4 encuentre fácilmente el router de reenvío 6to4, este router deberá tener asignada la dirección IPv4 anycast 192.88.99.1 [13].

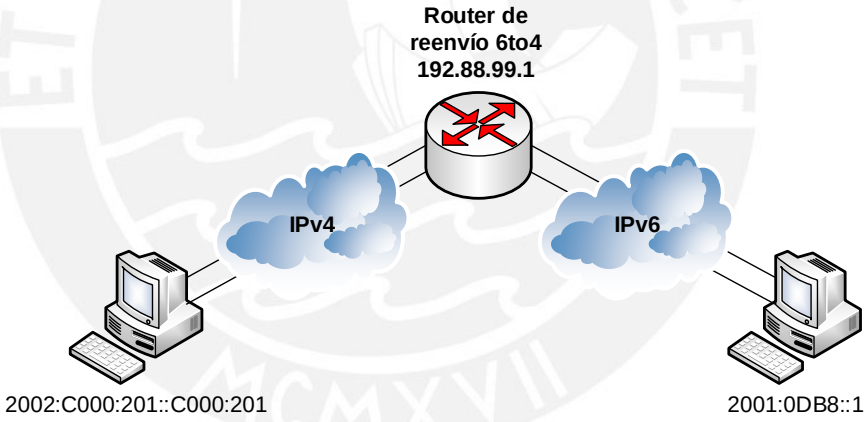


Fig. Nº 5 – Arquitectura comunicación host 6to4 y host IPv6 [12]

Sin embargo, esta característica no es conveniente para los ISPs. Si un proveedor implementa un router de reenvío 6to4 y empieza a anunciar el prefijo 2002::/16, proveerá de conexión libre a todos los hosts 6to4, sin importar si son clientes o no. Esto hace que los ISPs no tengan el control del flujo de paquetes. Si un proveedor trata de filtrar paquetes 6to4 que no provienen de sus propios clientes, el router de reenvío se convierte en un “agujero negro” para otros usuarios. Por consiguiente, se inhabilita la comunicación del mecanismo 6to4.[12]

En la figura Nº6, se describe un escenario típico. Se muestra un host 6to4 “Host H1”, cliente del ISP A. Los routers de la red IPv4 encaminan los paquetes que

corresponden al túnel 6to4 del “Host H1” hacia el router de renvío “Router R”, operado por el ISP A; este, desencapsula los paquetes y los reenvía a la red de destino IPv6. El host IPv6 “Host H2” responde al “Host H1” con la dirección IPv6 de destino que inicia con 2002::/16. Los paquetes, previamente encapsulados, serán enrutados de regreso al router de reenvío “Router R”. Finalmente, serán enviados al “Host H1”.[12]

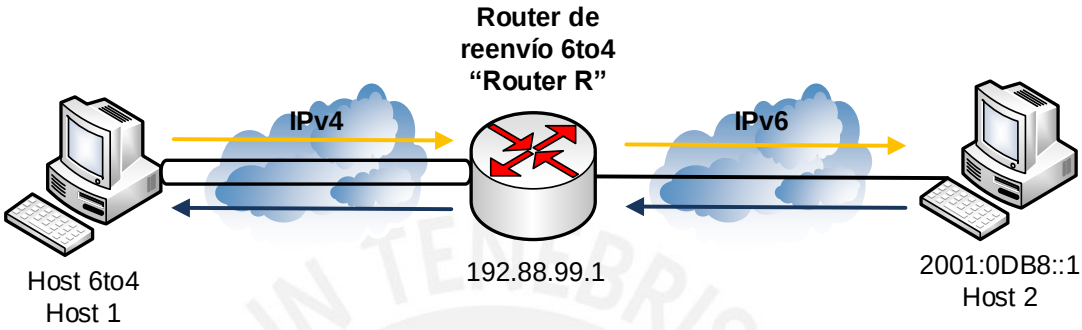


Fig. Nº 6 – Flujo de tráfico túnel 6to4 [12]

6to4 Relay Router Black Hole

Si se da el caso en que el ISP A decide limitar su servicio solo a sus clientes, puede añadir ciertas reglas en su router relay para filtrar los paquetes entrantes. Como se muestra en la figura Nº 7, un host 6to4 “Host 3” que es cliente del ISP B, está cerca del router relay “Router R” del ISP A. El tráfico fluye desde el H3 hacia el host H2, sin ningún problema. Cuando el host H2 en la red IPv6 quiera responder al H3, habrá algunos problemas.[12]

El router relay está anunciando un prefijo 2002::/16. Como resultado, los paquetes de retorno de H2 serán enrutados al router R. Desafortunadamente, estos serán bloqueados ya que las reglas, configuradas previamente, filtrarán los paquetes que no pertenezcan a sus clientes. De este modo, el tráfico nunca llegará al host H3.[12]

De acuerdo al mecanismo de enrutamiento, cada paquete con origen en la red IPv6 y con destino a un host con el prefijo 2002::/16 cerca del router R será enviado al router R. Sin embargo, el router R bloqueará todo el tráfico que no pertenezca a sus clientes. Por lo tanto, el router R se convierte en un “agujero negro” (black hole).[12]

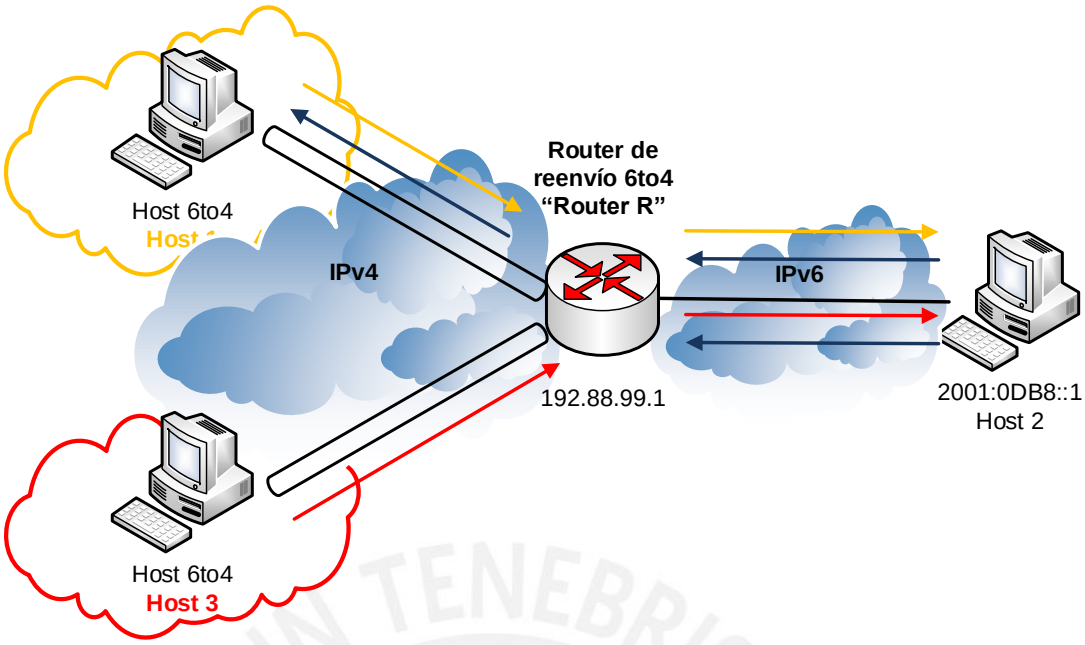


Fig. Nº 7 – 6to4 relay router, Black Hole [12]

Limitaciones

Se requiere que las direcciones IPv4 de los hosts 6to4 sean públicas. Lo que implica que un túnel 6to4 no funciona detrás de un NAT. Sin embargo los NATs son utilizados ampliamente en muchos escenarios, esto impone una limitación poco práctica para el mecanismo 6to4.

1.4.2.3. Túneles de Despliegue Rápido IPv6 (6RD)

Un túnel 6RD es una extensión del mecanismo 6to4. Como se muestra en la figura Nº 8, 6RD permite a un ISP proporcionar servicios de IPv6 unicast a clientes IPv6. Para ello, utiliza encapsulación de IPv6 en IPv4 a través de una red IPv4.[14]

Las principales diferencias entre los túneles 6to4 y 6RD son:

- 6RD no requiere direccionar el tráfico hacia el prefijo 2002::/16; en su lugar, el prefijo puede ser obtenido del propio bloque del ISP. Este mecanismo permite que la operación del mecanismo 6RD este dentro de la red del ISP. Desde la perspectiva del usuario final y, en general, de Internet, el servicio IPv6 proporcionado es equivalente a un IPv6 nativo.[14]

- No es necesario incorporar los 32 bits del destino IPv4 en la cabecera del payload IPv6. El destino IPv4 se obtiene de una combinación de bits en la cabecera del payload y la información en el router. Además, la dirección IPv4 no

está en una ubicación fija en la cabecera IPv6, como sucede en el túnel 6to4 .[14]

El prefijo 6RD del proveedor de servicios (SP) fue seleccionado por el SP para el despliegue IPv6, y se muestra en la figura Nº 8. El prefijo delegado 6RD se deriva del prefijo del SP y de los bits de la dirección IPv4, y es usado por el cliente (CE) para los hosts dentro de la red.[14] Figura Nº8.

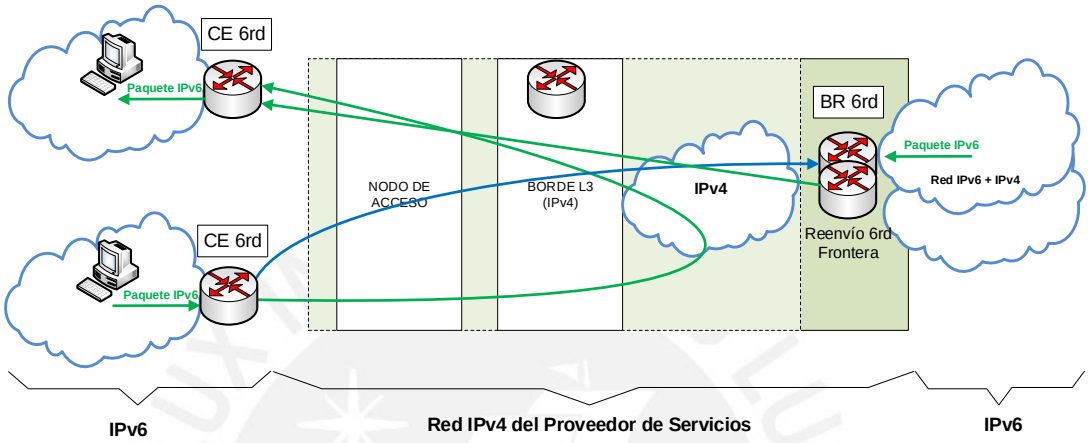


Fig. Nº 8 – Despliegue de 6RD [14]

La siguiente figura muestra cómo se delega un prefijo 6RD:

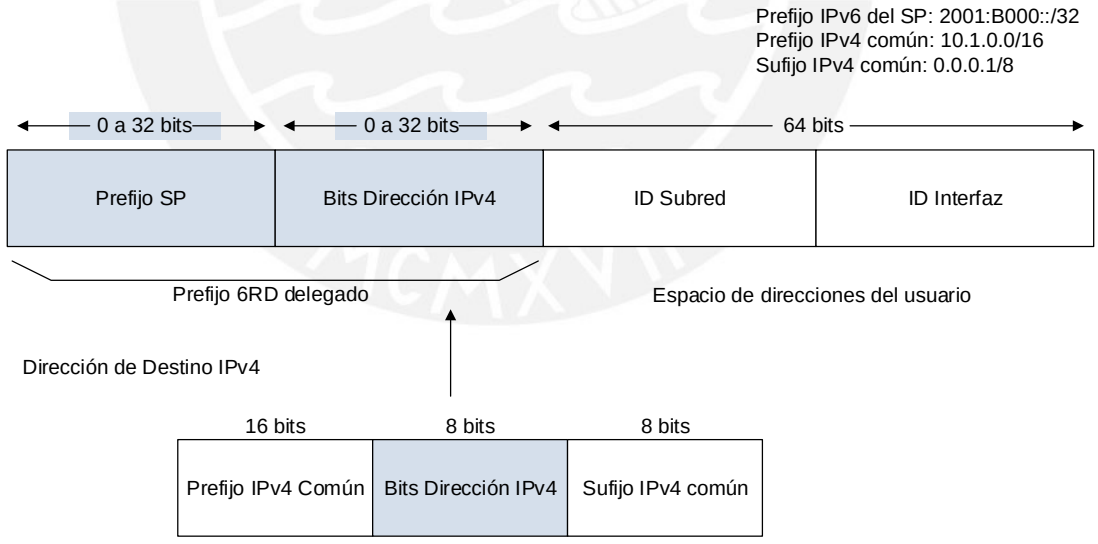


Fig. Nº 9 – Explicación de la delegación del prefijo 6RD [14]

La figura Nº 10 muestra una topología de delegación de prefijo 6RD:

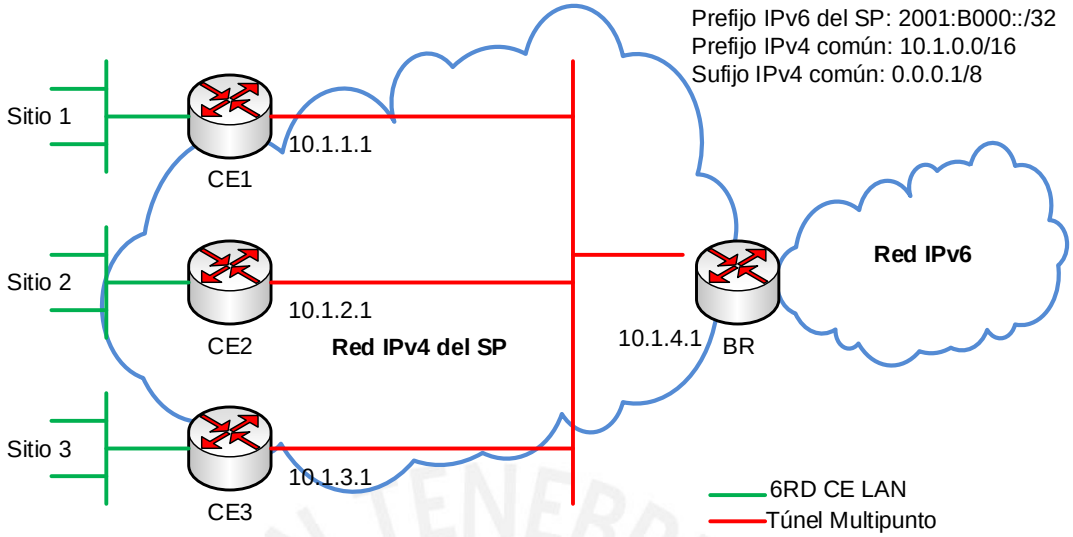


Fig. Nº 10 – Topología de Delegación de Prefijo 6RD [14]

La tabla Nº 1 explica la delegación de prefijo 6RD:

Tabla Nº 1 - Explicación de la delegación del prefijo 6RD [14]

Prefijo del Proveedor de Servicios	2001:B000::/32
Prefijo IPv4 común	10.1.0.0/16
Sufijo IPv4 común	0.0.0.1/8
CE1: Prefijo delegado 6RD	2001:B000:0100::/40
CE2: Prefijo delegado 6RD	2001:B000:0200::/40
CE1 (IPv4), origen del túnel de transporte	10.1.1.1
CE2 (IPv4), origen del túnel de transporte	10.1.2.1
BR (IPv4), origen del túnel de transporte	10.1.4.1

1.4.2.4. Traducción NAT64

La tecnología Network Address Translation IPv6 para IPv4, o NAT64, facilita la comunicación entre nodos nativos IPv4 e IPv6. Esta solución permite a empresas e ISPs acelerar la adopción de IPv6, y a la vez manejar el agotamiento de IPv4.[15]

Todos los escenarios viables de traducción se soportan en NAT64, por lo tanto, se considera como la tecnología de traducción más buscada. La traducción usando la tecnología NAT64 puede lograrse de dos formas: con estado (stateful) y sin estado (stateless):

- Stateless NAT64, definido en el RFC 6145, consiste en un mecanismo de traducción que utiliza algoritmos para el mapeo de direcciones IPv6 a IPv4 y viceversa. No mantiene ningún enlace o sesión mientras realiza la traducción, mientras se realiza la comunicación que es iniciada en IPv4 y aquella que es iniciada en IPv6.[15]
- Stateful NAT64, definido en el RFC 6146, es un mecanismo con estado para la traducción de direcciones IPv6 a IPv4 y viceversa. Es llamado stateful porque crea o modifica enlaces y sesiones mientras realiza la traducción. Este mecanismo soporta comunicaciones iniciadas en IPv4 e IPv6 utilizando asignación de direcciones manuales o dinámicas.[15]

En la figura Nº 11 se muestra los escenarios posibles de traducción IPv4/IPv6, para el caso Stateless y Stateful. La traducción NAT64 stateful es preferida sobre otras tecnologías de traducción y migración IPv6. Debido a que facilita la comunicación usando UDP, TCP, o ICMP entre los host y redes nativas IPv4 e IPv6.[15]

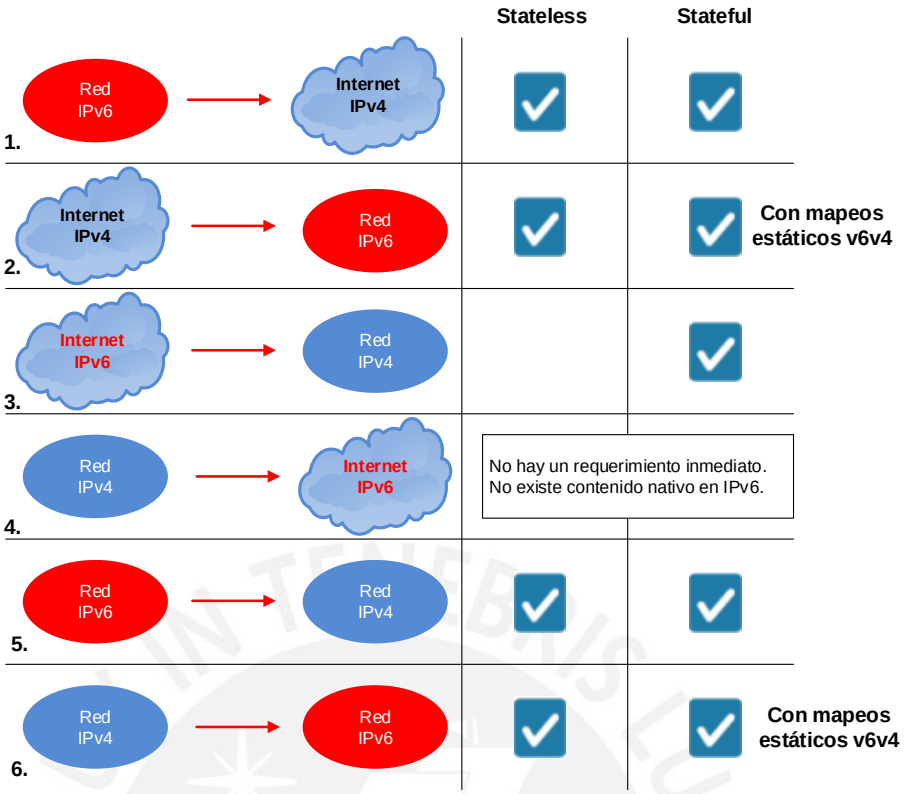


Fig. Nº 11 – Escenarios de traducción IPv4/IPv6 [15]

En la tabla Nº 2 se muestra la terminología usada en NAT64

Tabla Nº 2 – Terminología NAT64 [15]

Terminología	Definición
Well-known prefix (WKP)	El prefijo IPv6 64:ff9b::/96, definido en el RFC 6052, utiliza un algoritmo para el mapeo entre ambos protocolos. Este prefijo no es globalmente enrutable.
Network-specific prefix (NSP)	Un prefijo IPv6 asignado por una organización para usarlo en el mapeo algorítmico entre ambos protocolos. Por lo general, está grabada en el prefijo de una organización puede ser globalmente enrutable. Por ejemplo, 2001:db8:cafe::/96 proviene del prefijo 2001:db8:cafe::/48.
Direcciones IPv4 convertidas a IPv6	Direcciones IPv6 usadas para representar nodos IPv4 en una red IPv6. Por ejemplo, 2001:db8:cafe::c000:0201 usando NSP, ó, 64:ff9b::c000:0201 usando WKP; ambos representan a 192.0.2.1 (hex c000201).

NAT64 Stateless en Linux[9]

Tayga es una implementación de NAT64 sin estado (stateless) que funciona fuera del kernel de Linux. Utiliza el controlador TUN[10] para intercambiar paquetes IPv4 e IPv6 dentro del kernel de Linux.

Para la operación de Tayga es necesario dedicar parte del rango de direcciones IPv6 al prefijo NAT64. Así mismo, se requiere un bloque de direcciones IPv4 para el pool de direcciones dinámico. Tayga asignará direcciones IPv4 de este pool a los hosts IPv6 que requieran el servicio NAT64. El pool dinámico puede ser tomado del espacio de direcciones privadas IPv4 (10.x.x.x, 192.168.x.x) y puede ser de cualquier tamaño. Aunque se requiere que este espacio de direcciones IPv4 sea suficiente para soportar a cada host IPv6 que use NAT64. Tayga requiere de una dirección IPv4 propia, puede ser tomada del pool dinámico IPv4.[16]

Para instalar Tayga, en el caso de Debian, se ejecuta el comando:

```
apt-get install tayga
```

Luego de la instalación, se modifica el archivo /etc/default/tayga, la opción:

```
RUN="no"
```

Se cambia por:

```
RUN="yes"
```

El prefijo de la red IPv6 se utiliza para identificar a los que están en la red IPv4. Los routers de la organización deben configurar las tablas de enrutamiento para que los paquetes destinados a este prefijo se dirijan al nodo que tiene la función de NAT.[9]

El pool dinámico es el prefijo IPv4 que se utiliza para identificar a los equipos IPv6 en la red IPv4. Por defecto el software Tayga enmascara todo el tráfico iniciado en el nodo que realiza NAT64.[9]

En este ejemplo se utiliza un prefijo IPv6 NSP 2001:db8:cafe:ffff::/96. El pool IPv4 es el que viene por defecto en la configuración[16]. Entonces, se modifica el archivo /etc/tayga.conf queda configurado:

[9] Configuración de Tayga en el Escenario Mecanismos de Traducción, Capitulo II
[10] Dispositivo de red virtual punto a punto, diseñado para operar con tramas IP con soporte de kernel de bajo nivel.

```
prefix 2001:db8:cafe:ffff::/96

dynamic-pool 192.168.255.0/24
```

Además, para que el dispositivo NAT64 responda mensajes ICMP se debe configurar una dirección IPv4 y una IPv6.

```
ipv6-address 2001:db8:cafe:a::1

ipv4-address 192.0.2.1
```

Estas direcciones no forman parte de los prefijos, IPv4 e IPv6, reservados para NAT64 y DNS64.

1.4.2.5. DNS64

Una de las características más interesantes de NAT64 es la posibilidad de combinarlo con DNS. Aquellos equipos que residan en redes IPv6 preguntarán siempre al servidor DNS por registros AAAA, pero aquellos equipos que solo disponen de dirección IPv4 no disponen de ningún registro AAAA.[9]

DNS64 sintetiza los registros de recurso AAAA (AAAA RRs) desde un registro de recuro A (A RRs). DNS64 permite a los hosts IPv6 el uso del Nombre de Dominio Completo de un nodo IPv4 para iniciar la comunicación.

En la siguiente tabla se muestra la operación DNS64. Cuando un nodo IPv6 inicia una comunicación, usualmente pregunta por un registro AAAA y espera obtener la dirección IPv6 del nodo objetivo. Para permitir que el servidor responda, se utiliza la función DNS64 para sintetizar un registro AAAA a partir de un registro A (conteniendo la dirección IPv4 real del servidor de respuesta). DNS64 está diseñado como una función adicional del servidor recursivo DNS. De este modo, cuando un servidor DNS64 reciba una consulta AAAA RR iniciada por un nodo IPv6, buscara por un registro AAAA. Si este registro no está disponible (puede ser el caso de un nodo IPv4), el servicio DNS64 realizará la consulta por el registro A. Al descubrirse este registro, el servidor DNS64 creará un registro AAAA añadiendo Pref64::/n de NAT64 a la dirección IPv4 del servidor de respuesta, solo si n es menor que 96. El registro AAAA se retorna al iniciador IPv6, de este modo se inicia la comunicación con la dirección IPv6 asociada al destino IPv4.[17]

Tabla Nº 3 – DNS64 sintetizando un registro A dentro de un registro AAAA

Registro A: example.com	Dirección IPv4: 192.0.2.1
Prefijo NAT64 stateful (Perf64::/96)	2001:db8:cafe::/96
Síntesis DNS64 del registro AAAA de example.com:	
2001:db8:cafe::/96 + 192.0.2.1 = 2001:db8:cafe::192.0.2.1	

DNS64 en Linux[11]

La función DNS64 en Linux está a cargo del servicio BIND[12]. En primer lugar es necesario instalar el software:

```
sudo apt-get install bind9
```

Luego, es necesario editar el archivo de configuración para habilitar DNS64 y permitir las consultas y solicitudes de clientes indirectamente conectados. En el archivo /etc/bind/named.conf.options se añade la siguiente configuración:

```
  dns64 2001:db8:cafe:ffff::/96 {

  };
```

En donde se indica el prefijo utilizado en NAT64.

Adicionalmente, se define una zona DNS para completar la operación de DNS64:

```
$TTL    60000
@       IN      SOA     ns64.example.org. hostmaster.example.org. (
                      20150427          ; Serial
                          28800                 ; Refresh
                          14400                 ; Retry
                          2419200               ; Expire
                          300 )                 ; Negative Cache TTL
;
@           IN  NS      ns64
ns64    IN      A       192.0.2.64
ns64    IN      AAAA    2001:db8:cafe:a::64
sb2     IN      A       198.51.100.22
sb2     IN      AAAA    2001:db8:cafe:b::b2
ha1     IN      AAAA    2001:db8:cafe:a::a1
hb1     IN      AAAA    2001:db8:cafe:b::b1
```

El equipo sb2.example.org dispone de dos direcciones, IPv4 e IPv6. Al ejecutar el comando *host sb2.example.org* se obtiene:

[11] Configuración de BIND en el Escenario Mecanismos de Traducción, Capitulo II
[12] Berkeley Internet Name Daemon

```
root@r64:~# host sb2.example.org
sb2.example.org has address 198.51.100.22
sb2.example.org has IPv6 address 2001:db8:cafe:b::b2
```

El equipo tiene la capacidad de responder las consultas de registros A y AAAA. Para comprobar el funcionamiento de la consulta de registros A se ejecuta el comando *dig sb2.example.org a*

```
;; QUESTION SECTION:
;sb2.example.org.               IN      A

;; ANSWER SECTION:
sb2.example.org.        60000   IN      A       198.51.100.22
```

De la misma manera, para el registro AAAA, se ejecuta el comando *dig sb2.example.org aaaa*

```
;; QUESTION SECTION:
;sb2.example.org.               IN      AAAA

;; ANSWER SECTION:
sb2.example.org.        60000   IN      AAAA    2001:db8:cafe:b::b2
```

Para ver la operación de DNS64 y NAT64 con consultas a hosts nativos en IPv4 que pertenecen a otras zonas; por ejemplo, se tiene la zona example.com

```
$TTL    60000
@       IN      SOA     ns4.example.com. hostmaster.example.com. (
                      20150427          ; Serial
                          28800                 ; Refresh
                          14400                 ; Retry
                          2419200               ; Expire
                          300 )                 ; Negative Cache TTL
;
@           IN  NS      ns4
ns4     IN      A       192.0.2.4
ns4     IN      A       203.0.113.4
h41         IN  A   203.0.113.41
s42         IN  A   203.0.113.42
```

El host h41 tiene una dirección IPv4. Al realizar la consulta desde un equipo que solo cuenta con conectividad IPv4

```
root@s42:~# host h41.example.com
h41.example.com has address 203.0.113.41
root@s42:~#
```

Ahora, si la consulta se realiza desde un equipo IPv6

```
root@ha1:~# host h41.example.com
h41.example.com has address 203.0.113.41
h41.example.com has IPv6 address 2001:db8:cafe:ffff::cb00:7129
root@ha1:~#
```

Se muestra un registro AAAA que permite la comunicación con el host mediante NAT64.

La operación de DNS64 junto a NAT64 transparenta el uso de NAT64. Las respuestas, sean traducidas o no, dependerá del origen de las consultas.

1.4.2.6. Túnel Teredo y NAT66

Ubuntu incluye una implementación de Teredo llamado Miredo [18] . Se trata de un túnel IPv6 que funciona a través de conexiones IPv4, envía tráfico IPv6 sobre tráfico IPv4-UDP. Este servicio permite la operación de NAT66, los equipos IPv6 tienen acceso a Internet IPv6 a través de una dirección IPv6 configurada previamente a través del túnel Teredo.[9]

Operación del túnel Teredo

Teredo está basado en túneles IPv6. Sin embargo, los paquetes IPv6 no están encapsulados directamente en paquetes IPv4, en su lugar, utiliza paquetes UDP [21].

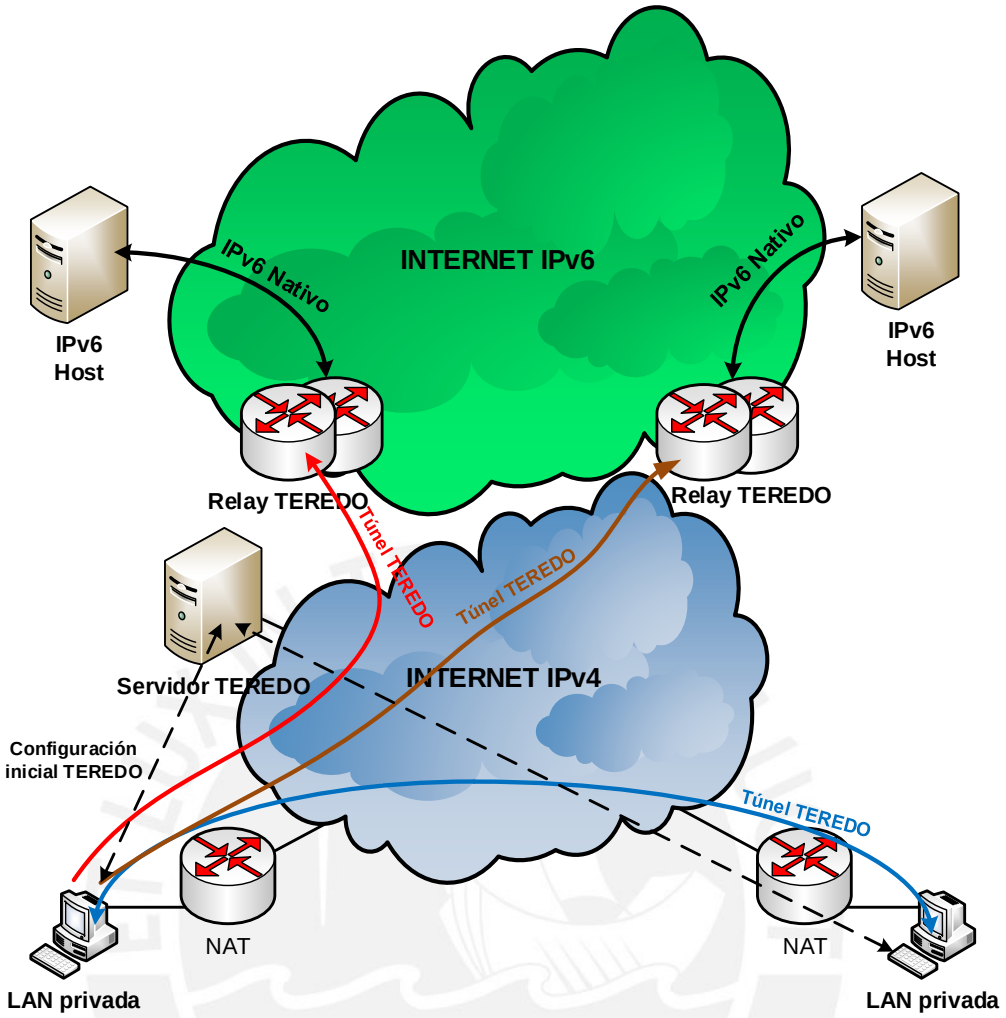


Fig. Nº 12 – Operación de los túneles TEREDO [21]

Por otro lado, existen cuatro agentes Teredo: cliente Teredo, servidor Teredo, relay Teredo y host relay específico Teredo. El cliente Teredo es configurado por el usuario que desea una conexión IPv6.[21]

En la figura Nº 12, la tarea de reenviar los paquetes IPv6 e IPv4 es realizada por los relay Teredo.

El servidor Teredo asigna una dirección IPv6 al cliente Teredo, dependiendo del tipo de NAT que esté usando. Esto es importante porque el tipo de dirección IPv6 y la forma en que se realiza la conexión IPv6 dependen del tipo de NAT en el lado del usuario. Por consiguiente, existe una etapa inicial de conexión con el servidor Teredo en la que el cliente intenta adquirir una dirección IPv6.[21]

En el RFC 3489 “Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)” se definen los detalles sobre los diferentes tipos de NAT44 que pueden configurarse en las redes IPv4 privadas.

Direccionamiento IPv6 en Teredo

El RFC 4380 “Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)”[19] define 5 componentes de la dirección IPv6 de la interfaz *teredo,* tabla Nº4:

Tabla Nº 4 – Componentes de la dirección IPv6 de la interfaz teredo

Bits	**0 - 31**	**32 – 63**	**64 – 79**	**80 – 95**	**96 – 127**
Tamaño	32 bits	32 bits	16 bits	16 bits	32 bits
Descripción	Prefijo	IPv4 del servidor teredo	Flags	Puerto UDP (bits invertidos)	IPv4 del cliente (bits invertidos)
Parte	2001:0000	53aa:064c	24a2	C5a5	4cf8:8935
Decodificado		83.170.6.76	Seguridad Teredo [20]	14938	179.7.118.202

Los bits 80 – 95 contienen el puerto UDP; este es el número de puerto que NAT asigna al cliente Teredo, con todos los bits invertidos. En la tabla Nº4, el puerto UDP es 14938 (en hexadecimal, C5A5 invertido = 3A5A, o en decimal 14938).

Los bits 96 – 127 contienen la dirección IPv4 del cliente; esta es la dirección IPv4 pública con todos los bits invertidos. En la tabla Nº4, la dirección IPv4 pública es 179.7.118.202 (en hexadecimal, 4cf8:8935 invertido = B307:76CA, o en decimal 179.7.118.202).

Configuración del Túnel Teredo

La implementación del túnel Teredo se inicia con la instalación del paquete Miredo en el servidor NAT64/DNS64 a través del siguiente comando en la consola:

```
sudo apt-get install miredo
```

Una vez instalado el programa Miredo se habilita la interfaz *teredo* en el sistema. No se requiere de configuración adicional. Al ejecutar el comando *ifconfig* en la consola del servidor se tiene:

```
teredo    Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet6 addr: 2001:0:53aa:64c:24a2:c5a5:4cf8:8935/32 Scope:Global
          inet6 addr: fe80::ffff:ffff:ffff/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1280  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:380 (380.0 B)
```

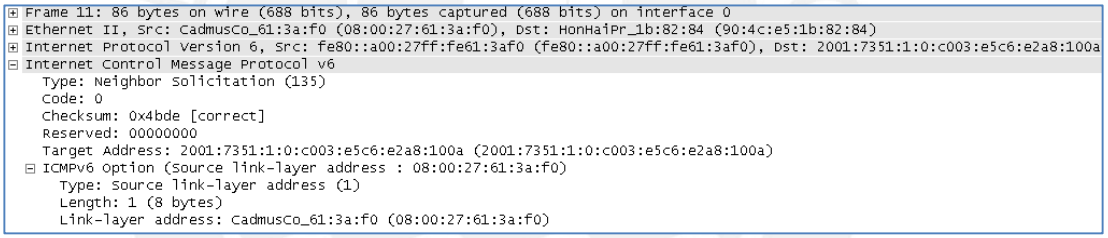
1.4.3. ICMPv6 Neighbor Discovery

Neighbor Discovery es un protocolo que permite a diferentes nodos de una misma red anunciarse, y, además, para conocer de la existencia de equipos vecinos.

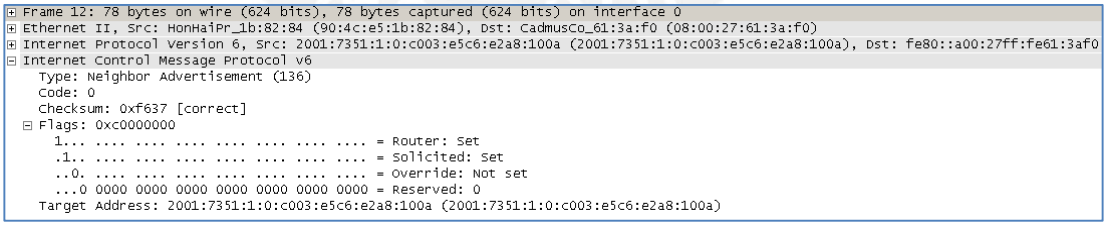
Routers y hosts (nodos) usan mensajes Neighbor Discovery (ND) para determinar las direcciones de la capa de enlace de vecinos que residen en los enlaces adjuntos y sobrescribir las entradas de cache no válidos. Los hosts también utilizan ND para encontrar routers vecinos que puedan reenviar sus paquetes.[22]

Además, los nodos utilizan activamente ND para contactarse con equipos vecinos. Cuando un router (o ruta de acceso a un router) falla, los nodos activan la búsqueda de rutas alternativas para llegar al destino.[22]

En la siguiente captura Wireshark, se muestra la trama tipo 135, Neighbor Solicitation. En donde, la dirección de destino es 2001:7351:1:0:c003:e5c6:e2a8:100a la dirección MAC del nodo solicitante es 08:00:27:61:3a:f0 y la dirección IP de origen es fe80::a00:27ff:fe61:3af0.



Se muestra la captura del mensaje unicast de respuesta tipo 136, Neighbor Advertisement:



1.4.3.1. Mejoras sobre IPv4

Neighbor Discovery de IPv6 involucra un conjunto de protocolos IPv4: ARP, ICMP, Router Discovery, y ICMP Redirect. Sin embargo, ND provee mejoras sobre IPv4, las cuales cubren lo siguiente:[22]

- Descubrimiento de routers: Cuando un host ubica routers que residen en un enlace adjunto.[22]

- Descubrimiento de prefijos: Cuando un host descubre prefijos de dirección para destinos que residen en un enlace adjunto. Los nodos usan prefijos para distinguirse entre destinos que residen en un enlace adjunto y aquellos destinos que solo pueden ser alcanzados a través de un router.[22]
- Descubrimiento de parámetros: Cuando un nodo aprende varios parámetros (de enlace y de Internet) que se colocan en los paquetes salientes.[22]
- Resolución de direcciones: Cuando un nodo usa solo una dirección de destino IPv6 para determinar una dirección de la capa de enlace de destinos en un enlace adjunto.[22]
- Determinación del siguiente salto: El algoritmo que un nodo usa para mapear una dirección de destino IPv6 en una dirección IPv6 vecina al cual planea enviar tráfico para el destino.[22]
- Detección de vecino inaccesible: Cuando se determina que un nodo ya no está disponible.[22]
- Detección de dirección duplicada: Cuando un nodo determina que una dirección está en uso.[22]

Neighbor Discovery utiliza los siguientes mensajes ICMPv6: router solicitation, router advertisement, neighbor solicitation, neighbor advertisement, y redirect.[22]

IPv6 Neighbor Discovery reemplaza los siguientes protocolos IPv4: router discovery (RDISC), Address Resolution Protocol (ARP), y ICMPv4 redirect.[22]

1.4.4. Direccionamiento Dinámico IPv6

En la presente tesis se hace uso de dos métodos de direccionamiento automático IPv6, direccionamiento con estado (Stateful) y sin estado (Stateless). El primero, es una analogía al método utilizado en IPv4 que involucra al Protocolo de Configuración Dinámica de Hosts (DHCP). En el direccionamiento Stateless, el host genera su propia dirección IPv6 a partir de un prefijo anunciado en la red y la dirección física de la interfaz de red del host.

1.4.4.1. Autoconfiguración de Direcciones sin Estado (SLAAC)

Este mecanismo está diseñado para evitar la configuración manual de direcciones cuando un host se conecta una red IPv6.[23] SLAAC funciona de la siguiente manera:

- Los routers IPv6 envían periódicamente a la red mensajes de anuncio conteniendo un prefijo de subred (típicamente un /64).[23]
- Los hosts conectados generan de forma autónoma direcciones globales basadas en el prefijo de subred anunciado y sus identificadores EUI-64[23].

SLAAC, al contrario de DHCP, es un mecanismo descentralizado y sin estado. Es descentralizado porque no depende de un nodo central, además del router anunciando el prefijo (o prefijos). En efecto, un router no tiene idea de las direcciones que están siendo asignadas (y no hay una forma sencilla de saberlo). De otro lado, SLAAC es sin estado porque no hay ninguna información que pueda mantenerse centralizada por un largo periodo por los routers.[6]

Una de las ventajas de SLAAC es que la modificación de prefijos es sencilla; los routers solo tienen que anunciar el nuevo prefijo o prefijos. SLAAC es uno de los componentes clave de IPv6, dado que permite la asignación de direcciones a una gran cantidad de dispositivos, y no requiere de algún esfuerzo en particular de parte del router. En este contexto, es un componente clave para el despliegue de una gran cantidad de teléfonos IP, sensores, televisores inteligentes, etc. Sin embargo, no se recomienda el uso de SLAAC en una red corporativa, dado que cada componente de la misma requiere ser identificado y controlado. [6]

La topología mostrada en la siguiente figura muestra la operación de SLAAC.

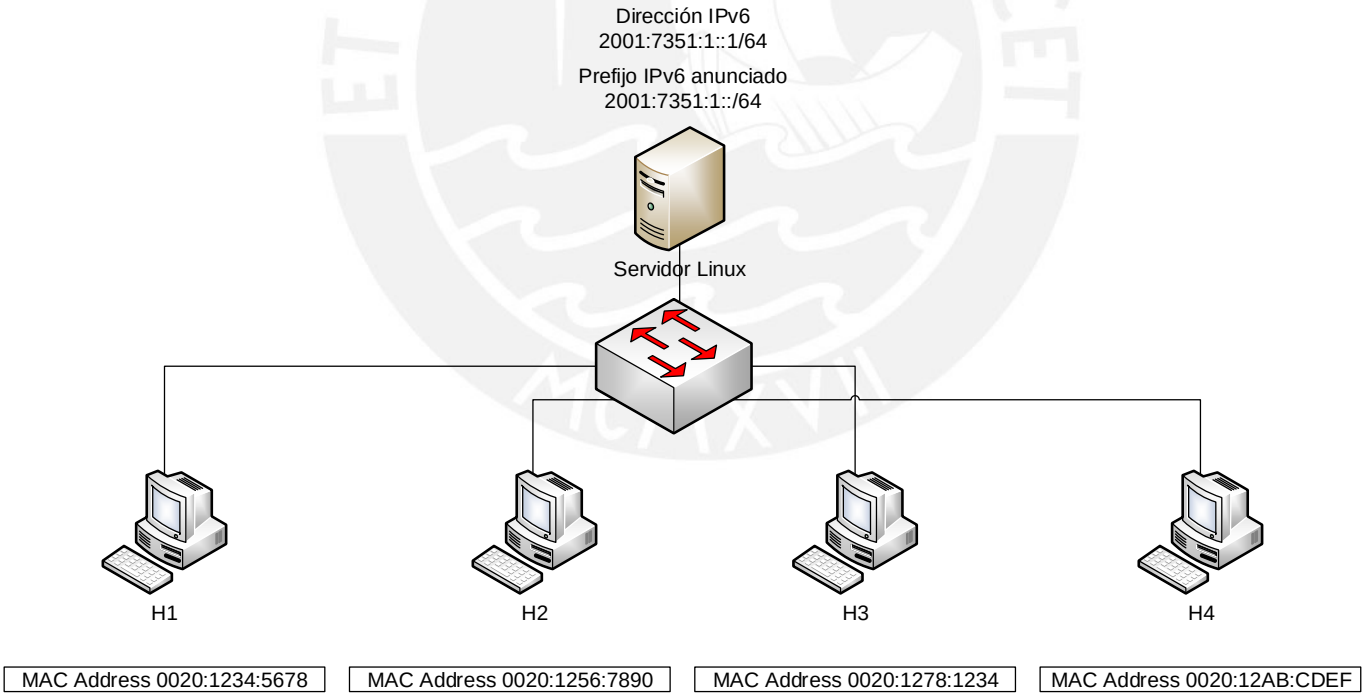


Fig. Nº 13 - Operación de SLAAC

Los equipos (H1 – H4) mostrados en la figura Nº 13 usan el método de identificación de host EUI-64 [23], las direcciones IPv6 creadas usando SLAAC quedarían de la siguiente manera:

- H1 → 2001:7351:1::12FF:FE34:5678
- H2 → 2001:7351:1::12FF:FE56:7890
- H3 → 2001:7351:1::12FF:FE78:1234
- H4 → 2001:7351:1::12FF:FEAB:CDEF

En H1, el prefijo 2001:7351:1::/64 es extraído de los mensajes RA del Servidor Ubuntu y se convierte en el prefijo inicial. El identificador del cliente es creado a partir de la dirección MAC (0020:1234:5678). El primer paso de la conversión EUI-64 es dividir en la mitad la dirección MAC y ubicar FF:FE en el medio, lo que resulta en 0020:12FF:FE34:5678. Luego, se toma el séptimo bit y se convierte a 0, en este caso los primeros 8 bits son 00000010 (0x02); dando como resultado 00000000 (0x00). Finalmente, el identificador de host resultante es 000012FF:FE34:5678. Cuando el prefijo y el identificador del host se combinan, resulta en una dirección IPv6 que es usada por H1, 2001:7351:1:0000:0000:12FF:FE34:5678, la cual puede ser resumida en 2001:7351:1::12FF:FE34:5678.

El demonio RADVD (Router Advertise Daemon) es utilizado por el servidor Linux para anunciar el prefijo 2001:7351:1::1764. La configuración del demonio se encuentra en el archivo /etc/radvd.conf.

```
interface eth0
{
#Enable Advertisements
AdvSendAdvert on ;
#Lifetime adv
AdvDefaultLifetime 340;
#Time between Adv
MaxRtrAdvInterval 120;
MinRtrAdvInterval 60;

prefix 2001:7351:1::/64
{
#Lifetime Adv
        AdvPreferredLifetime 300;
        AdvValidLifetime 400;
};
};
```

De donde:

- **Interface eth0:** Es la interfaz configurada con una dirección IPv6 y de donde se anuncia el prefijo IPv6 a la red.[24]

- **AdvSendAdvert on|off:** Indica el envío y respuesta periódica a los anuncios del router. Si está habilitada, permite los anuncios desde la interfaz configurada previamente.[24]

- **AdvDefaultLifetime (segundos):** Tiempo de vida asociado con el router, en segundos. El valor máximo es de 18.2 horas. Un tiempo de vida igual a 0 indica que un router no es la puerta de enlace por defecto y no debe figurar en la lista de routers. Esta vida útil solo se aplica a la utilidad del router como puerta de enlace por defecto. [24]

- **MaxRtrAdvInterval (segundos):** El tiempo máximo permitido entre el envió de anuncios multicast no solicitados del router desde la interfaz. No debe ser menor que 4 segundos, y no mayor a 1800 segundos. Al usar extensiones IPv6 móvil, el valor mínimo es de 0.07 segundos. Valor por defecto: 600 segundos. [24]

- **MinRtrAdvInterval (segundos):** El tiempo mínimo permitido entre el envío de anuncios multicast no solicitados del router desde la interfaz. No debe ser menor que 3 segundos, y no mayor a 0.75 * MaxRtrAdvInterval. Al usar extensiones IPv6 móvil, el valor mínimo es de 0.03 segundos. [24]
  Valor por defecto: 0.33 * MaxRtrAdvInterval.

- **Prefix:** Prefijo IPv6 que se anuncia a la red.

- **AdvPreferredLifetime** (**seconds**|**infinity**): El tiempo en segundos (desde que el paquete es enviado) en que la dirección se genera a partir del prefijo a través de la configuración automática de direcciones sin estado (stateless).
  Se recomienda que este valor sea suficientemente largo: 7 días. Este valor no debe ser mayor que AdvValidLifetime. El valor por defecto es 14400 segundos (4 horas). [24]

- **AdvValidLifetime (seconds|infinity)**: El tiempo en segundo (desde que el paquete es enviado) en que el prefijo es válido, con el objetivo de validar el enlace. Se recomienda que este valor sea suficientemente largo: 30 días. El valor por defecto es 86400 (1 dia). [24]

Las actualizaciones serán enviadas cada 60 y 120 segundo siendo válidas por 340 segundos. En este caso, el prefijo que se anuncia es 2001:7351:1::/64 y es válida por 400 segundos, como máximo.

Dado que el identificador de 64 bits es único y se mantiene a pesar de que el host se conecte desde distintas redes, no se recomienda que la dirección global se autogenere a través de la dirección MAC. El RFC 4941 “Privacy Extensions for Stateless Address Autoconfiguration in IPv6” describe un método para mejorar la privacidad de un nodo utilizando direcciones IPv6 globales, generadas aleatoriamente y temporales; y que además son desechadas y reemplazadas por otras direcciones con regularidad. Al igual que las direcciones públicas autogeneradas, las direcciones temporales son autogeneradas a partir de la información del prefijo global proporcionado por un router en la misma subred. Sin embargo, donde las direcciones públicas utilizan un identificador de interfaz fija, las direcciones temporales utilizan un identificador de interfaz separado que es generado aleatoriamente y de constante cambio [25].

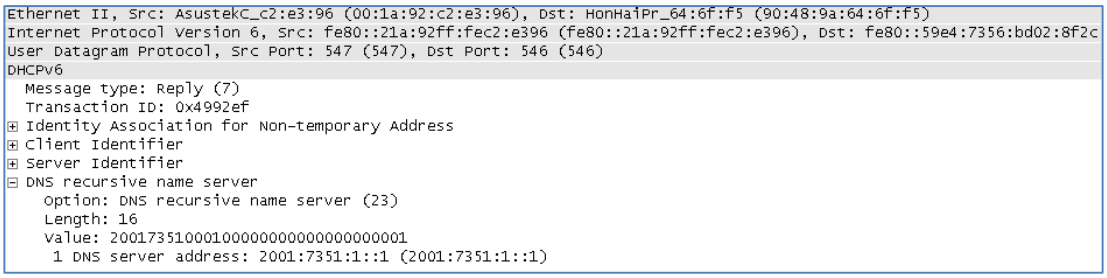
1.4.4.2. SLAAC y DHCPv6 stateless

Se trata de una combinación de un modo de gestión de direcciones con estado, y, otra, sin estado. Es decir, la dirección IPv6 se adquiere, en primer lugar, a través de SLAAC. El resto de parámetros de red, como DNS, puerta de enlace, sufijo de nombre de dominio, etc., se obtiene de DHCPv6. En este modo híbrido, la verificación de direcciones IPv6 es realizada por SLAAC [26]. El modo híbrido está definido en el RFC 3736 “Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6”.

Para indicar que se requiere parámetros adicionales en la configuración de la red, se activa el flag “Other Configuration” en el mensaje tipo 134, Router Advertisement:

```
Ethernet II, Src: AsustekC_c2:e3:96 (00:1a:92:c2:e3:96), Dst: IPv6mcast_01 (33:33:00:00:00:01)
Internet Protocol Version 6, Src: fe80::21a:92ff:fec2:e396 (fe80::21a:92ff:fec2:e396), Dst: ff02::1
Internet Control Message Protocol v6
  Type: Router Advertisement (134)
  Code: 0
  Checksum: 0x4bcc [correct]
  Cur hop limit: 64
⊟ Flags: 0xc0
    1... .... = Managed address configuration: Set
    .1.. .... = Other configuration: Set
    ..0. .... = Home Agent: Not set
    ...0 0... = Prf (Default Router Preference): Medium (0)
    .... .0.. = Proxy: Not set
    .... ..0. = Reserved: 0
  Router lifetime (s): 180
  Reachable time (ms): 0
  Retrans timer (ms): 0
⊞ ICMPv6 Option (Source link-layer address : 00:1a:92:c2:e3:96)
```

En la siguiente captura Wireshark, se muestra la respuesta del servidor DHCP con parámetros de red solicitados, en este caso información del servidor DNS.



El servicio más utilizado en entornos Linux y Unix es ISC DHCP. Para efectos de anunciar la dirección IPv6 del servidor DNS , es necesario agregar la siguiente línea en el archivo *etc/dhcp/dhcpd6.conf*:

```
option dhcp6.name-servers 2001:7351:1::1;
```

En una red con pocos equipos, como la doméstica, donde no es prioritario registrar las direcciones asignadas a los equipos, se recomienda este modo de configuración híbrido.

1.4.4.3. Autoconfiguración Stateful DHCPv6

En el modelo de autoconfiguración con estado DHCPv6[27] los nodos obtienen, además de la dirección IPv6, información y parámetros de configuración de un servidor central. Los servidores mantienen una base de datos que controlan las direcciones asignadas y de los hosts que las poseen. Las direcciones IPv6 son entregadas por un tiempo limitado. Cuando este periodo expira, si el host no muestra actividad en la red, la dirección es liberada y puede ser reasignada a otra interfaz de red. Para manejar la expiración de una dirección, esta experimenta dos fases distintas mientras es asignada a una interfaz. En primer lugar, una dirección es preferida cuando su uso en la comunicación se da manera irrestricta. En segundo lugar, luego de un cierto periodo de tiempo, la dirección se considera obsoleta y su uso no es recomendable, pero no prohibido.[6]

DUID (DHCP Unique Identifier): Cada servidor y cliente DHCP tiene un DUID. Los servidores DHCP usan DUIDs para identificar clientes para la selección de los parámetros de configuración y en la asociación de identidad con clientes. Los clientes DHCP usan DUIDs para identificar a un servidor en mensajes donde requiere ser identificado.[27]

IAID (Identity Association ID): Una IA es una implementación a través de la cual un servidor y un cliente pueden identificar, agrupar y gestionar un conjunto de direcciones IPv6 relacionadas. Cada IA consiste de un IAID y una información de configuración asociada.[27]

Un cliente debe asociarse con al menos una IA distinta con cada una de sus interfaces de red para lo cual se va a solicitar la asignación de direcciones IPv6 de un servidor DHCP. Los clientes usan IAs asociados a una interface para obtener la información de configuración de un servidor en dicha interface. Cada IA debe asociarse solo con una interfaz.[27]

DHCPv6 opera de manera distinta a IPv4, como es el uso de multicast y de las direcciones link-local durante el proceso de negociación. En IPv4 el equipo que solicita la dirección IP lo hace con la IP asignada o con la dirección 0.0.0.0.[9]

1.4.5. Sistema de Nombre de Dominio (DNS) en IPv6

Para indicar una dirección IPv4 se utiliza el registro A, de 32 bits de tamaño. En el caso de IPv6, el registro tiene un tamaño de 128 bits y se denota como AAAA. Esta solicitud DNS es hecha por el host cliente. Los registros A y AAAA, son requeridos por separado. Las redes que tienen en funcionamiento IPv6 e IPv4, no requieren del servicio DHCP, la dirección la adquieren de SLAAC. Además, para resolución de una dirección IPv6 recurren al DNS utilizado para IPv4.[9]

La configuración es mínima. Basta con indicar una dirección IPv6 para los registros AAAA. La configuración que se muestra a continuación es exclusiva de un servidor DNS Bind9, de uso frecuente en entornos Linux y UNIX, y corresponde al archivo /etc/bind/named.conf.local. Donde se hace referencia a la zona “lanip6.local[13]”.

```
zone "lanip6.local"  {
        type master;
        file "/var/lib/bind/lanip6.local";
};
```

Luego, se especifica la zona en el archivo /var/lib/bind/lanip6.local.

[13] Segmento de red IPv6

```
;
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     ns1.local. root.myhost.lanip6.local. (
                              61         ; Serial
                          604800         ; Refresh
                           86400         ; Retry
                         2419200         ; Expire
                          604800 )       ; Negative Cache TTL
;
@       IN      NS      ns1.local.

myhost  IN      A       10.10.11.250
        IN      AAAA    2001:7351:1::1
blah    IN      CNAME   myhost

servidor IN      A       10.10.11.11
        IN      AAAA    2001:7351:1:ffff::a0a:b0b
```

El archivo inicia con la configuración de un tiempo de espera predeterminado. El caracter "@" es automáticamente reemplazado por el nombre de la zona (lanip6.local). Para el dominio se definen directamente dos registros de recursos (RR): Inicio de Actividad (SOA) RR y Servidor de Nombres (NS) RR.[28]

El registro SOA especifica que este servidor es autorizativo para esta zona. Un servidor autorizativo es la mejor fuente de datos de una zona. El registro SOA contiene información general acerca de la zona y reglas de recarga para servidores secundarios. Solamente puede haber un registro SOA por zona. Este tipo de registro se define en RFC 1035.[28] El resto de valores de SOA son:

- Serial: Es un valor que incrementa, en cada cambio que se hace en el archivo.
- Refresh: Indica el tiempo luego del cual los servidores DNS secundarios se actualizan a partir del servidor primario.
- Retry: Indica el tiempo de reintento, si la actualización falla.
- Expire: Indica el tiempo luego del cual un servidor secundario se detiene y deja de recibir actualizaciones del primario.
- Negative cache TTL: Indica el tiempo que la respuesta "this host does not exist" es válida en un dispositivo de resolución.

Los hosts "myhost" y "servidor" tienen registros A y AAAA, pueden ser resueltos vía IPv4 e IPv6. No se recomienda exponer dominios de red en internet a través de direcciones privadas, como 192.168.0.0/16 o fd00::/8.

El host “blah” es un alias (CNAME) para “myhost”, y también se resuelve con los mismos registros A y AAAA.

El registro NS especifica un servidor de nombres autorizado para este sistema principal. Este tipo de registro se define en RFC 1035.[28]

1.5. Estado de la tendencia Actual de IPv6 en el Perú

1.5.1. La Red Académica Peruana (RAAP)

La Red Académica Peruana es una asociación civil sin fines de lucro que tiene por objetivo promover la investigación, la capacitación y el intercambio de conocimiento a través del uso intensivo de las tecnologías de información y comunicación.

La RAAP está conformada por 8 instituciones académicas, públicas y privadas, dedicadas a la investigación, y su propósito es convocar, según determinados estándares técnicos, a todas las instituciones académicas peruanas dispuestas a participar de sus proyectos e infraestructura. La RAAP es integrante de la red de Cooperación Latino Americana de Redes Avanzadas (CLARA).[29]

Informados de esta iniciativa, se crea en el Perú, el 30 de abril de 2003, la Red Académica Peruana (RAAP) bajo el auspicio de CONCYTEC; como una Asociación Civil sin fines de lucro, constituyéndose como la Red Nacional de Investigación y Educación (NREN) del Perú firmando en México, en Junio del mismo año, los estatutos de CLARA, conjuntamente con otros 16 países de América Latina.

Conformada inicialmente por la Universidad Nacional Mayor de San Marcos, la Universidad Nacional de Ingeniería, la Universidad Peruana Cayetano Heredia, la Universidad Nacional Agraria La Molina, la Pontificia Universidad Católica del Perú, el Instituto Peruano de Energía Nuclear y el Instituto Nacional de Investigación y Capacitación en Comunicaciones (INICTEL), la RAAP espera congregar a todas las instituciones educativas y de investigación del país.[30]

a. Características

Es una red de comunicaciones de arquitectura abierta, con soporte multiprotocolo que permite servicios de banda ancha. Su objetivo es mantener una independencia de conexión con la red Internet actual que está orientada al ámbito comercial.

La arquitectura de red integrará las universidades e institutos de investigación del país en una sola red.

Actualmente usa los nuevos protocolos y arquitecturas de red IPV6 garantizando una adecuada calidad de servicio a las nuevas aplicaciones de I+D y permitiendo la creación de redes virtuales privadas (VPN) para la creación de grupos de investigación.

Es la red que da acceso a toda la comunidad académica y de investigación nacional independientemente de sus proveedores actuales de Internet.

La arquitectura de la red actual está conformada por una red IP VPN a nivel nacional, con una Cabecera principal.

El Cabecera Principal; además de constituirse en el NAP académico, tiene comunicación hacia las redes internacionales de investigación basada en IPv6.

Los equipos de red (routers), tienen la capacidad de enrutar tráficos de paquetes IPv4 e IPv6, así como voz y video sobre ambos protocolos IP. [30]

b. Topología de Red

La Red Académica Peruana comprende, por un lado, el desarrollo de actividades de investigación, docencia e intercambio de información; por otro lado, la implementación de una arquitectura de conectividad de última generación y alta velocidad que hacen posible dicho intercambio mucho más eficiente. [31] En la figura Nº 14 se muestra la topología de red.

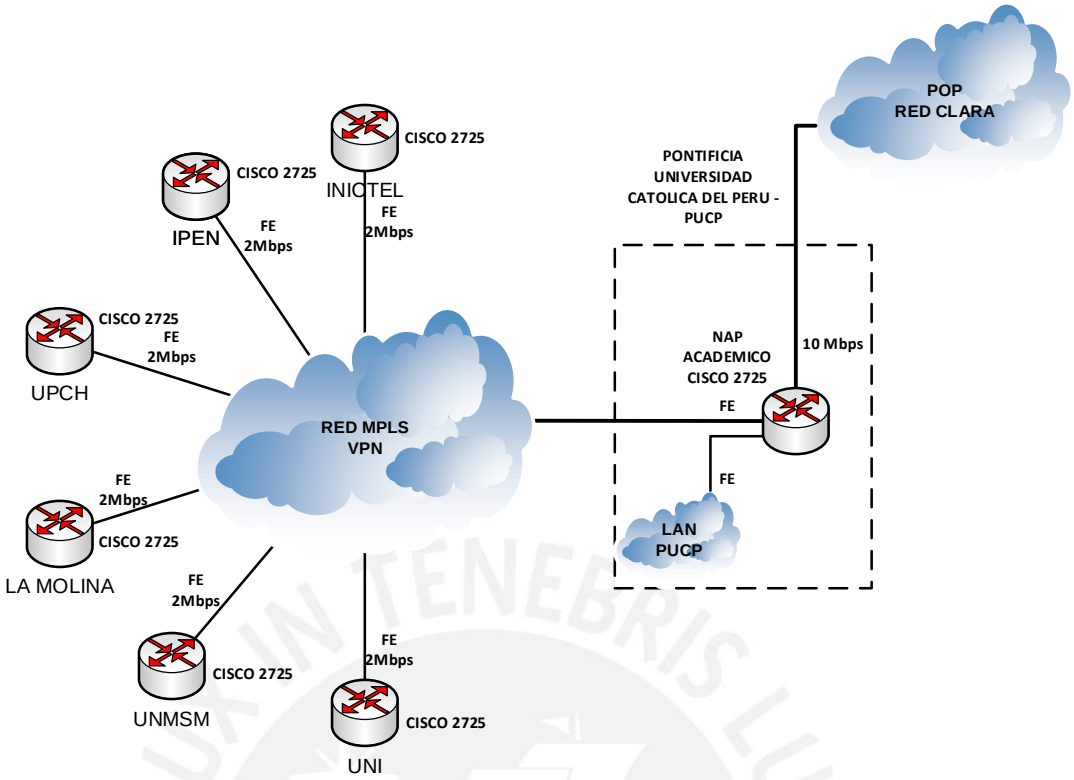


Fig. Nº 14 - Mapa de la Topología de la Troncal [31]

La Red Académica Peruana cuenta con la siguiente configuración [32]:

- Numero de Sistema Autónomo[14] (ASN): 28095
- Dirección IPv4 asignada: 190.124.48.0/21
- Dirección IPv6 asignada: 2001:13A0::/32
- Formato de distribución de direcciones: 2001:13A0:tzzz dddd ccci iiii::/48

Dónde:

- t: tipo de red:

  Red RAAP = 0

  Otra red = 1

- zzz: Zona geográfica (Tabla Nº 5):

Tabla Nº 5 - Identificador de Zona Geográfica

Zona Geográfica	**zzz**
Reservado	000
Capital	001

[14] Referido a un grupo de redes IP que son gestionadas por uno o más operadores que poseen una política de rutas; cada uno cuenta con un número que es utilizado para el intercambio de rutas con otros sistemas.

Norte	010
Oriente	011
Sur	100
Sur-oriente	101
Centro	110
Reservado	111

- ccc: Clase (Tabla Nº 6)

Tabla Nº 6 - Identificador de Clase

Clase	**Ccc**
Reserva	000
Universidad Publica	001
Universidad Privada	010
Instituto de investigación	011
Proyectos	100
Universidad Pública/usuario	101
Universidad Privada/usuario	110
Instituto de investigación	111

- dddd: Departamento (Tabla Nº 7)

Tabla Nº 7 - Identificador de Departamento

Departamento		**dddd**
Norte	Tumbes	= 000x
	Piura	= 001x
	Lambayeque	= 010x
	Cajamarca	= 011x
	La Libertad	= 100x
Oriente	Amazonas	= 000x
	Loreto	= 001x
	San Martín	= 010x
	Ucayali	= 011x
Capital	Lima	= 000x
	Callao	= 001x
Centro	Ancash	= 000x
	Huánuco	= 001x
	Pasco	= 010x
	Junín	= 011x
	Huancavelica	= 100x
Sur	Ica	= 000x
	Arequipa	= 001x
	Ayacucho	= 010x
	Tacna	= 011x
	Apurimac	= 100x
	Moquegua	= 101x
Sur Oriente	Cusco	= 000x
	Madre de Dios	= 001x
	Puno	= 010x

x = 0: Sede principal
x = 1: Sede de otro departamento

- iiiii: Institución (Tabla Nº 8)

Tabla Nº 8 - Identificador de Institución

Institución	iiiii
UNALM	= 00001
UNI	= 00010
UNMSM	= 00011
PUCP	= 00001
UPCH	= 00010
INICTEL	= 00001
IPEN	= 00010

En la tabla Nº 9 se muestra la asignación de direcciones IPv6 para cada institución miembro de la RAAP.

Tabla Nº 9 - Asignación de direcciones IPv6 a instituciones miembros de RAAP

Institución	Prefijo	Prefijo abreviado
UNMSM	2001:13A0:0001 0000 0010 0001::/48	2001:13A0:1021::/48
UNI	2001:13A0:0001 0000 0010 0010::/48	2001:13A0:1022::/48
UNALM	2001:13A0:0001 0000 0010 0011::/48	2001:13A0:1023::/48
PUCP	2001:13A0:0001 0000 0100 0001::/48	2001:13A0:1041::/48
UPCH	2001:13A0:0001 0000 0100 0010::/48	2001:13A0:1042::/48
INICTEL	2001:13A0:0001 0000 0110 0001::/48	2001:13A0:1061::/48
IPEN	2001:13A0:0001 0000 0110 0010::/48	2001:13A0:1062::/48

c. Estado Actual

Actualmente, el estado de la RAAP es el siguiente [33]:

- No cuenta con dominios alojados activos
- No cuenta con servidores de dominio activos
- No cuenta con prefijos IPv4 anunciados
- No cuenta con prefijos IPv6 anunciados

La figura Nº 15 muestra el estado actual del sistema autónomo de la Red Académica Peruana según la base de datos de la RIPE.

Fig. Nº 15 - Estado del Sistema Autónomo de la RAAP [34]

1.5.2. Los Operadores de Telecomunicaciones y la Adopción de IPv6

De acuerdo al reporte mundial de Sistemas Autónomos, Perú tiene asignados 37, de los cuales 27 están activos, cada cual con uno o más prefijos IPv4 configurados. Así mismo, 9 ASNs tienen uno o más prefijos IPv6 configurados. Además, el 33.3% de los ASNs activos tienen un prefijo IPv6. Cabe resaltar que Telefónica del Perú S.A.A., con ASN 6147, es el más grande; y está ubicado en el puesto 267 del ranking global. [35]

Por otro lado, en lo que se refiere al tiempo de permanencia, el 19 de noviembre de 1993 se asignó el ASN 264684 a la Red Científica Peruana, convirtiéndose en el registro más antiguo del Perú. Así mismo, el último registro de AS se dio el 30 de marzo de 2006 a Telefónica del Perú S.A.A con el número 264684.[35]

La tabla Nº 10 muestra la lista actualizada de operadores e instituciones en el Perú que cuentan con prefijos IPv6 activos.

Tabla Nº 10 - Sistemas Autónomos asignados a Perú ordenados según su posición en el ranking global [30]

ASN	Nombre	Fecha de Registro	Ranking Global	Nº IPv4 enrutables	Nº Prefijos IPv4	Nº Prefijos IPv6
6147	Telefónica del Perú S.A.A.	08/12/1995	267	1,542,656	1,108	27
12252	América Móvil del Perú S.A.C.	17/11/2000	509	629,248	185	2
19180	Americatel del Perú	01/12/2001	1,535	135,680	53	1
3132	Red Científica Peruana	19/11/1993	2,007	99,072	10	2
27843	Optical Technologies S.A.C.	01/08/2006	2,737	67,840	174	2
262253	Econocable Media S.A.C.	03/12/2012	9,153	9,472	37	1
28032	Internexa Perú S.A	23/01/2009	13,721	4,608	18	2
262182	Media Networks Latin America SAC	05/09/2011	16,796	3,584	11	8
263692	DirecTV Perú S.R.L	12/11/2014	27,167	1,024	4	1

La tabla Nº 10 muestra la lista actualizada de operadores e instituciones en el Perú que cuentan con prefijos IPv6 activos.

1.6. Estado de la Tendencia Actual de los Mecanismos de Transición de IPv6

A continuación se realizará un análisis de la tendencia de los mecanismos de traducción utilizados en la actualidad.

1.6.1. Tecnologías de Transición en redes móviles

En la actualidad, los operadores pueden considerar múltiples enfoques para abordar el despliegue de IPv6, que incluyen implementaciones nativas de dual-stack, 6RD, Mapeo de Dirección y Puerto (MAP), y el Traductor de Direcciones Cliente/Proveedor (464XLAT). En la tabla Nº 11 se muestra la aplicación de cada tecnología.

Tabla Nº 11 - Aplicación de Tecnologías de transición en redes móviles [32]

Tecnología	Aplicación
Infraestructura Nativa de Dual-stack	El proveedor habilita un stack para IPv6 e IPv4 a través de su infraestructura.
6RD	El proveedor habilita dual-stack en la infraestructura y núcleo del cliente (CPE), conservando un punto de borde o acceso IPv4. Con la finalidad de crear un túnel para IPv6 a través de la red de acceso IPv4.
MAP	El proveedor habilita dual-stack en el CPE, pero deshabilita IPv4 en el borde. Los algoritmos de MAP permiten el intercambio de direcciones IPv4 entre suscriptores residenciales sin necesidad de un LSN[15] con estado centralizado.
464XLAT	EL proveedor solo ejecuta IPv6 en su infraestructura pero, usando un código de detección de intrusiones basado en host, ofrece una interface dual-stack para aplicaciones, sin la necesidad de cambiar a IPv4 en ningún momento.

1.6.2. Uso de Dual-Stack

A nivel corporativo, la mayoría de las empresas prefieren el modelo dual-stack. Para ello, se asigna un espacio de direcciones tanto para IPv4 e IPv6 a usuarios finales, aplicaciones y equipos de red. Por tanto, le corresponde al terminal del usuario decidir que protocolo utilizar; dado que la mayoría de sistemas operativos tienen a IPv6 como ruta por defecto cuando esta se encuentra operativa.[36]

En la actualidad, los sistemas operativos tienen pre-establecidos controles específicos para asegurar conexiones rápidas y fiables cuando operan en modo dual-

[15] Traducción de direcciones de red a gran escala

stack, basado en el algoritmo descrito en RFC 6555.[37] Esta tecnología tiene dos características principales:

- Provee una conexión inmediata a los usuarios, si al intentar conectarse usando IPv4, la conexión a IPv6 no es exitosa, esta se dará de manera rápida.
- Evita conexiones simultáneas, al no hacer intentos repetitivos tanto en IPv6 e IPv4.

Sin embargo, el modelo dual-stack no será sostenible en el tiempo cuando el espacio de direcciones IPv4 esté totalmente agotado.[36]

1.6.3. 464XLAT en Redes Móviles

Los operadores de redes móviles en EE.UU. y Korea utilizan 464XLAT para desplegar IPv6 en sus redes LTE.

Descrito en RFC 6877 [38], es una arquitectura que provee conectividad IPv4 a través de una red IPv6. Para lo cual combina los siguientes protocolos:

- Protocolo de traducción con estado [39], en CG-NAT64
- Protocolo de traducción sin estado [40], en el terminal del usuario (UE)
- IPv4 embebido dentro de direcciones IPv6 [41]

464XLAT es una técnica simple y escalable para desplegar el servicio de acceso IPv4 hacia redes móviles IPv6. La figura Nº 16 muestra los elementos de red de la solución. Cabe señalar que 464XLAT también puede ser usado en la telefonía fija.[3]

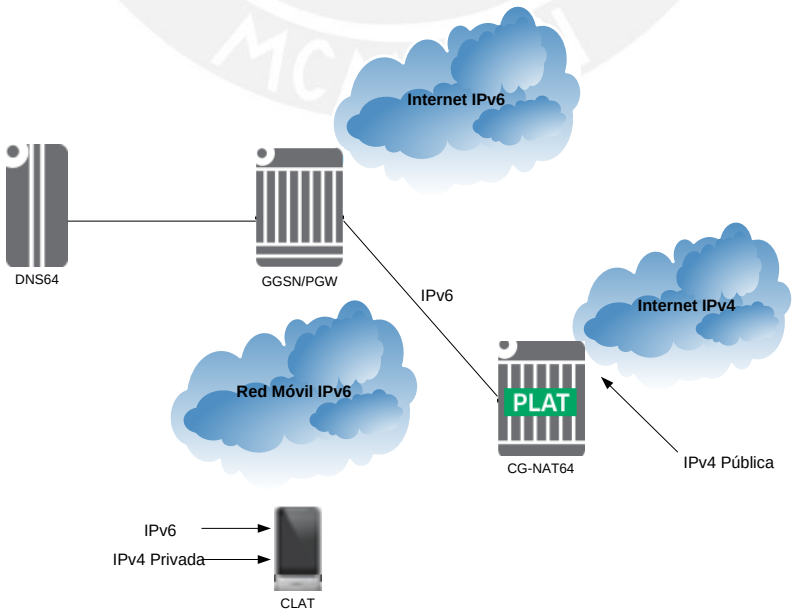


Fig. Nº 16 - 464XLAT en Redes Móviles [3]

IPv6 nativo

Un prefijo simple IPv6/64 es asignado al terminal (UE) en la red móvil. Las aplicaciones y páginas web habilitados en IPv6 son enrutados sobre GGSN/PGW hacia Internet IPv6.[3]

DNS64

Es un mecanismo para la síntesis de recursos de registro (AAAA) de direcciones IPv6 desde registros (A) de direcciones IPv4. Si una petición AAAA resulta en uno o más registros, el resultado es devuelto al cliente de acuerdo a una semántica de DNS normal. Si un registro A esta disponible para una consulta AAAA, el componente DNS64 incrusta las direcciones IPv4 del registro A en una dirección IPv6, y será usado en respuesta a la consulta AAAA.[3]

CLAT

El componente de traducción del lado cliente (CLAT) es una parte del software que se ejecuta dentro del terminal UE. Así mismo, implementa un protocolo de traducción sin estado y ofrece una dirección IPv4 privada y una ruta por defecto IPv4 hacia aplicaciones IPv4 en el terminal UE. El tráfico con un destino IPv4 es traducido a IPv6 por el componente CLAT.[3]

PLAT

El componente de traducción del lado del proveedor (PLAT), implementa un protocolo de traducción con estado. PLAT traduce direcciones IPv6 globales a direcciones IPv4 públicas, y viceversa. Todo el tráfico dirigido a este prefijo IPv6 debe ser enrutado hacia PLAT. Así mismo, PLAT deriva la dirección de destino IPv4 desde la dirección de destino IPv6.[3]

PLAT implementa puertas de enlace de capa de aplicación (ALG) para permitir a ciertos protocolos atravesar el componente CG-NAT. Estos protocolos incluyen: FTP, SIP, RTSP y PPTP. Además, implementa un mecanismo de registro para todos los enlaces CG-NAT64 para efectos legales. Por ejemplo, Syslog, SNMP, Radius, o IPFIX.[3]

1.7. Metodología

El estudio busca involucrar los conceptos revisados en la literatura y de herramientas de virtualización utilizadas en la actualidad, las que permiten enriquecer el marco teórico. Este se complementa con un conjunto de pruebas basadas en la virtualización de elementos y dispositivos de red. Con lo cual se puede elaborar un modelo que permita ejecutar diversos escenarios de convivencia entre las redes IPv4 e IPv6, a través de los mecanismos de traducción descritos en el punto 1.4.2 del presente capítulo.

1.7.1. Diseño de Investigación

El diseño de la investigación es experimental y transversal. Es experimental dado que se realiza con la manipulación de elementos de red virtuales, es decir se observan los resultados en su estado modificado, de este modo se explota las características que ofrecen las herramientas de virtualización a utilizar. Es transversal porque la toma de los datos se realiza en un momento único.

1.7.2. Herramientas de Virtualización

En la actualidad, algunas de las herramientas de virtualización muy conocidas son: Qemu[16], Xen[17], Servidor WMware[18], Virtual Box[19], y UML[20]. Sin embargo, estas herramientas no proporcionan un mecanismo específico para facilitar la implementación de escenarios de redes virtuales. Por esta razón, son requeridas algunas iniciativas orientadas a simplificar el despliegue y configuración de infraestructuras de red virtual, entre ellas; Imunes [42], Velnet [43], My Linux Network (MLN) [44], Netkit [45], Dynamips [46] y Virtual Network User-Mode Linux (VNUML) [47]. Su uso en entornos de investigación y educación ha sido estudiado en [48].[49]

Por lo tanto, VNUML representa una opción interesante para poner a prueba conceptos de redes informáticas avanzados. En tal sentido, permite ejecutar una versión reducida del núcleo de Linux y un archivo de sistema, necesario por las

[16] Emulador y virtualizador genérico de computadoras de código abierto.
[17] Monitor de máquinas virtuales de código abierto que permite ejecutar varios sistemas operativos en paralelo en una sola computadora o host.
[18] Proporciona un software de virtualización disponible para ordenadores compatibles X86
[19] Paquete de Software de virtualización x86 desarrollado por Oracle
[20] Permite que múltiples sistemas operativos basados en el kernel de Linux (conocidos como invitados) se ejecuten como una aplicación dentro de un sistema Linux normal (conocido como anfitrión).

computadoras que son parte del escenario virtualizado. De este modo, las configuraciones utilizadas pueden ser utilizadas en equipos reales.[49]

1.7.2.1. VNUML

Es una herramienta de gestión de escenarios de código abierto de uso general diseñado para la implementación de pruebas en redes virtuales de forma automática. Además, permite ejecutar e interactuar con entornos conformados por máquinas virtuales GNU/Linux interconectadas. Así mismo, los escenarios creados pueden ser interconectados con equipos y redes externos.[48]

VNUML está compuesta por dos componentes: un lenguaje basado en XML que permite describir el escenario de la red virtual (lenguaje especificado VNUML); y el intérprete de lenguaje que analiza la descripción, cuya función es construir, ejecutar y administrar el escenario (analizador VNUML).[50]

Se proporcionan tres operaciones básicas de administración: construcción de escenarios, liberación de escenarios y ejecución de secuencia de comandos. La forma de funcionamiento se muestra en la figura Nº 17.[50]

- **Construcción de escenarios:** El analizador VNUML procesa la secuencia de comandos con la finalidad de crear el escenario de red virtual mediante el arranque de las máquinas virtuales y la creación de las redes virtuales para interconectarlas de acuerdo a la topología especificada. El usuario puede interactuar de varias maneras (ej. Acceso directo a través de una consola gráfica, login SSH, etc.) y utilizarlos como sistemas GNU/Linux convencionales con el propósito de ejecutar las pruebas deseadas. [50]

- **Liberación de escenarios:** Una vez completada la fase de experimentación, esta operación desmantela el escenario, de este modo se liberan los recursos. [50]
- **Ejecución de secuencia de comandos:** Aunque se puede interactuar directamente con los equipos virtuales, VNUML permite automatizar la ejecución de la secuencia de comandos y la copia de archivos. Para lo cual se definen dos etiquetas, <exec> y <filetree>, que permiten especificar los comandos a ser ejecutados y los directorios a ser copiados, del host a las máquinas virtuales. Ambas etiquetas utilizan el atributo <seq>, que permite agrupar los elementos en una secuencia, especificado como un parámetro adicional al invocar VNUML.

La ejecución de secuencia de comandos se utiliza normalmente para iniciar o detener los servicios dentro de las máquinas virtuales. [50]

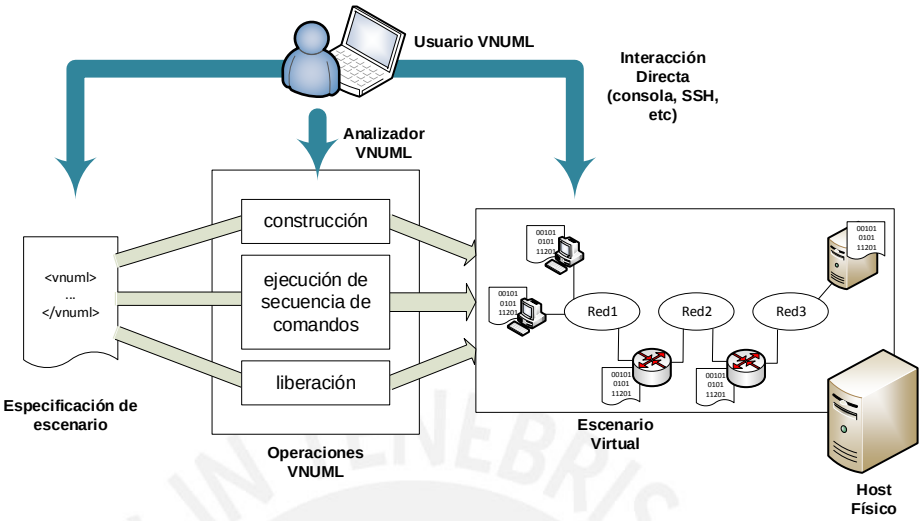


Fig. Nº 17 - Flujo de Operación VNUML [50]

CAPÍTULO II: IMPLEMENTACIÓN DE ESCENARIOS VIRTUALES

En el presente capítulo se implementa y analiza los mecanismos de transición a IPv6 descritas en el capítulo I. Además, se describe la secuencia de pruebas a través de la emulación de elementos de red usando tecnologías de emulación como VNUML.

Uno de los objetivos de este estudio es mostrar una arquitectura que asegure la convivencia de IPv4 e IPv6.

2.1. VNUML

De acuerdo a la metodología, en esta tesis se hace uso del simulador VNUML. A continuación se muestra una descripción breve de la estructura del lenguaje utilizado:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/share/xml/vnuml/vnuml.dtd">
<vnuml>
    <!-- Global definitions -->
    <global>
        .....
    </global>
    <!--Network definitions -->
    <net name="Net0" .... />
    ...
    <!-- Virtual machines definition -->
    <vm name="uml1">
        ...
    </vm>
    ....
</vnuml>
```

Las etiquetas <vnuml> y </vnuml> agrupan toda la definición del escenario. Las definiciones globales se encuentran entre las etiquetas <global> y </global>. Las redes se definen con la etiqueta <net>, y las máquinas virtuales con <vm> y </vm>.

La definición básica de una máquina virtual es como sigue:

```
<vm name="h41">
  <console id="1">pts</console>
  <if id="0" net="Net1">
      <ipv4 mask="255.255.255.0">192.168.100.104</ipv4>
  </if>
  <if id="1" net="Net2">

  </if>

      <exec seq="start" type="verbatim">ip route add default via 192.168.100.106 </exec>
</vm>
```

La etiqueta <console> define el acceso a la consola de la máquina. La configuración IP de la máquina está definida en la etiqueta <if>.

A continuación se describe los comandos más usados en VNUML:

- Para exportar el archivo vnuml al entorno del escenario virtual:

```
$ export DIRPRACT=.
```

- Puesta en marcha del escenario

```
$ simctl dnsnat64 sh
```

```
dnsnat64-> start
```

- Verificación del escenario:

```
dnsnat64-> vms
```

- Acceso a consola de una máquina virtual:

```
dnsnat64-> get dns64
```

- Parada del escenario:

```
dnsnat64-> stop
```

- Comandos adicionales [51]:

```
dnsnat64-> help
    start                          Starts scenario
    stop                           Stops scenario
    status                         State of the selected simulation
                                   (running | stopped)
    vms                            Shows vm's and tty's from simulation
    labels [vm]                    Shows sequence labels for ALL vm's or for vm
    exec label [vm [vm [ ... ]]]   Exec label in the vm's where label is defined
                                   or exec the label only in the vm list
    netinfo                        Provides info about network connections
    get [-t term_type] vm [pts]    Gets a terminal for vm
                                   term_type selects the terminal
                                   (xterm | gnome | kde | local)
                                   pts is an integer to select the console
    forcestop                      Destroys a simulation scenario
                                   Use with caution
    exit                           Exits interactive mode
    help                           To show this info
```

2.2. Escenario Mecanismos de Traducción

2.2.1. Topología

El escenario describe dos redes públicas nativas en IPv6 e IPv4 respectivamente. El objetivo es permitir la comunicación entre ambas redes. Para ello se pone a prueba la operación de los mecanismos DNS64 y NAT64 descritos en los puntos 1.4.2.4 y 1.4.2.5 del capítulo 1, respectivamente. La topología de red se muestra en la figura N° 18.

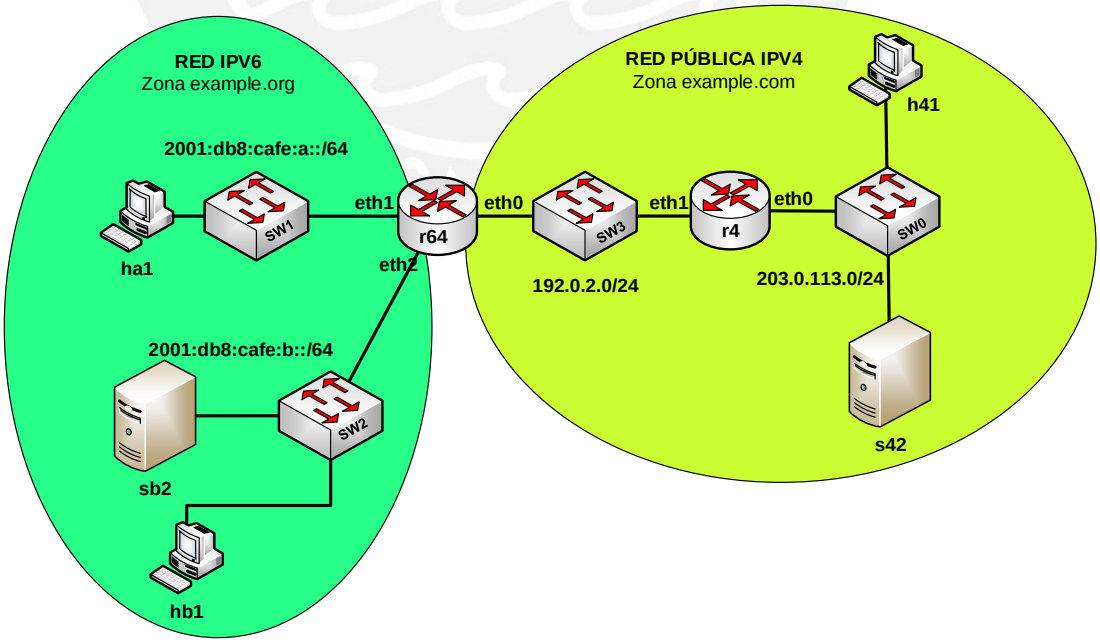


Fig. Nº 18 - Topología DNS64+NAT64

Los nodos mostrados en la figura Nº 17 son:

- **r64:** es el router principal del escenario, encargado de comunicar entre si las redes IPv4 e IPv6. Con funciones de proveer los servicios NAT64 y DNS64.
- **r4:** es un router con la función de proveer el servicio DNS a la red pública IPv4.
- **ha1:** es un cliente IPv6 con sistema operativo Debian.
- **hb1:** es un cliente IPv6 con sistema operativo Debian.
- **sb1:** es un servidor web IPv6 con sistema operativo Debian.
- **h41:** es un cliente IPv4 con sistema operativo Debian.
- **s42:** es un servidor web IPv4 con sistema operativo Debian.

Los equipos están conectados a las siguientes redes, tabla Nº 12:

Tabla Nº 12 - Redes utilizadas en la virtualización

Nombre	**Dirección IPv4**	**Dirección IPv6**	**Descripción**
SW0 (Net0)	203.0.113.0/24	-	Red de clientes
SW1 (Net1)	-	2001:db8:cafe:a::/64	Red de clientes
SW2 (Net2)	-	2001:db8:cafe:b::/64	Red de clientes
SW3 (Net3)	192.0.2.0/24		Red de transición IPv4 – IPv6

Las características de todos los sistemas utilizados en la virtualización se muestran en la tabla Nº 13:

Tabla Nº 13 - Sistemas utilizados en la virtualización

Nombre	**Dirección IPv4**	**Dirección IPv6**	**Zona de dominio DNS**	**Sistema Operativo**
r64	192.0.2.64	2001:db8:cafe:a::64 2001:db8:cafe:b::64	example.org example.com	Debian 6
r4	203.0.113.4 192.0.2.4		example.com	Debian 6
ha1		2001:db8:cafe:a::a1	example.org	Debian 6
hb1		2001:db8:cafe:b::b1	example.org	Debian 6

sb1		2001:db8:cafe:b::b2	example.org	Debian 6
h41	203.0.113.41		example.com	Debian 6
s42	203.0.113.42		example.com	Debian 6

2.2.2. Construcción del Escenario

La construcción del escenario se da en el archivo VNUML **dnsnat64.vnuml**. En donde se implementa cada nodo y se especifica la configuración respectiva dependiendo de la función que desempeñará dentro del escenario principal conforme a la topología mostrada en la figura Nº 17, y las tablas Nº12 y Nº13 (el archivo dnsnat64.vnuml completo se muestra en el anexo A de esta tesis).

2.2.2.1. Configuración del Router r64

Enrutamiento Estático

Para que un sistema Debian tenga funciones de un router IPv4/IPv6 es necesario ejecutar los siguientes comandos en el terminal:

```
sysctl -w net.ipv6.conf.all.forwarding=1
sysctl -w net.ipv4.conf.all.forwarding=1
```

De acuerdo a la topología de la figura Nº 17, se requiere alcanzar la red de los clientes IPv4. Esto es posible con la siguiente ruta estática:

```
ip route add 203.0.113.0/24 via 192.0.2.4
```

Servicio NAT64 Stateless

En el router r64 el programa Tayga gestiona el servicio NAT64 stateless. Para su operación se crea el archivo */usr/local/etc/tayga.conf* con los siguientes parámetros:

```
tun-device nat64
ipv4-addr 198.51.100.1
prefix 2001:db8:cafe:ffff::/96
dynamic-pool 198.51.100.0/24
map 198.51.100.2 2001:db8:cafe:a::a1
map 198.51.100.3 2001:db8:cafe:b::b1
map 198.51.100.4 2001:db8:cafe:b::b2
data-dir /var/db/tayga
```

Además, se crea un script con los siguientes parámetros y debe ser ejecutado en modo privilegiado al inicio del sistema:

```
mkdir -p /var/db/tayga
tayga --mktun
ip link set nat64 up
ip route add 198.51.100.0/24 dev nat64
ip -6 route add 2001:db8:cafe:ffff::/96 dev nat64
tayga
```

Servicio DNS64

En Debian, el programa Bind gestiona el servicio DNS64. Se edita el archivo */etc/bind/named.conf.options* con los siguientes parámetros:

```
options {
        directory "/var/cache/bind";

        ...

    # Our conf for IPv6 and DNS64
    listen-on-v6 { any; };

        # DNS64: Note that you can write multiple of these if you need
        # more IPv6 prefixes.
        # 2001:db8:cafe:ffff::/96 has to be the same as Tayga pool
        dns64 2001:db8:cafe:ffff::/96 {

        };
};
```

Servicio DNS

De acuerdo a la topología de la figura Nº 17 en la red IPv6 se estable la zona example.org. Para ello, el router r64 requiere de los siguientes archivos:

- Archivo de configuración DNS maestro */etc/bind/named.conf.local:*

```
zone "example.org" {
        type master;
        file "/etc/bind/db.example.org";
};

zone "100.51.198.in-addr.arpa" {
        type master;
        file "/etc/bind/db.198.51.100";
};
```

- Archivo de zona de búsqueda directa */etc/bind/db.example.org:*

```
$TTL    60000
@       IN      SOA     ns64.example.org. hostmaster.example.org. (
                      20150427          ; Serial
                         28800                  ; Refresh
                         14400                  ; Retry
                         2419200                ; Expire
                         300 )                  ; Negative Cache TTL
;
@           IN  NS      ns64
ns64    IN      A           192.0.2.64
sb2     IN      A       198.51.100.4
        IN      AAAA    2001:db8:cafe:b::b2
ns64    IN      AAAA    2001:db8:cafe:a::64
ha1     IN      AAAA    2001:db8:cafe:a::a1
hb1     IN      AAAA    2001:db8:cafe:b::b1
```

2.2.2.2. Configuración del Router r4

Enrutamiento Estático

Para habilitar las funciones del router IPv4, se ejecuta el siguiente comando en el terminal:

```
sysctl -w net.ipv4.conf.all.forwarding=1
```

Por otro lado, para alcanzar la red del pool dinámico del router r64, se crea una ruta con el siguiente comando:

```
ip route add 198.51.100.0/24 via 192.0.2.64
```

Servicio DNS

De acuerdo a la topología de la figura Nº 17 , en la red IPv4 se estable la zona example.com. Para ello, el router r4 requiere de los siguientes archivos:

- Archivo DNS de nivel superior */etc/bind/db.root:*

```
$ORIGIN .
$TTL 60000      ; 16 hours 40 minutes
@             IN SOA  localhost. hostmaster.localhost. (
                                20150428   ; serial
                                28800      ; refresh (8 hours)
                                14400      ; retry (4 hours)
                                2419200    ; expire (4 weeks)
                                300        ; minimum (5 minutes)
                                )
                        NS      ROOT-SERVER-A

ROOT-SERVER-A               3600000  IN  A     192.0.2.4

example.com.                3600000  IN  NS    ns4.example.com.
ns4.example.com.            3600000  IN  A     203.0.113.4
113.0.203.in-addr.arpa 3600000  IN  NS    ns4.example.com.

example.org.                 3600000  IN  NS    ns64.example.org.
ns64.example.org.            3600000  IN  A     192.0.2.64
100.51.198.in-addr.arpa 3600000  IN  NS    ns4.example.org.
```

- Archivo de configuración DNS maestro */etc/bind/named.conf.local:*

```
zone "." {
        type master;
        file "/etc/bind/db.root";
};

zone "example.com" {
        type master;
        file "/etc/bind/db.example.com";
};

zone "113.0.203.in-addr.arpa" {
        type master;
        file "/etc/bind/db.203.0.113";
};
```

- Archivo de búsqueda directa */etc/bind/db.example.com:*

```
$TTL    60000
@       IN      SOA     ns4.example.com. hostmaster.example.com. (
                      20150427          ; Serial
                         28800                ; Refresh
                         14400                ; Retry
                         2419200              ; Expire
                         300 )                ; Negative Cache TTL
;
@           IN  NS      ns4
ns4     IN      A       192.0.2.4
ns4     IN      A       203.0.113.4
h41         IN  A   203.0.113.41
s42         IN  A   203.0.113.42
```

2.2.2.3. Configuración de los Clientes IPv6

Configuración, a través de líneas de comandos en el terminal, de las direcciones IPv6, Gateway y DNS, respectivamente:

Host ha1

```
ip -6 a a 2001:db8:cafe:a::a1/64 dev eth0
ip -6 route add default via 2001:db8:cafe:a::64
echo -e "search example.org\nnameserver 2001:db8:cafe:a::64\n" > /etc/resolv.conf
```

Host hb1

```
ip -6 a a 2001:db8:cafe:b::b1/64 dev eth0
ip -6 route add default via 2001:db8:cafe:b::64
echo -e "search example.org\nnameserver 2001:db8:cafe:a::64\n" > /etc/resolv.conf
```

Host sb2

```
ip -6 a a 2001:db8:cafe:b::b2/64 dev eth0
ip -6 route add default via 2001:db8:cafe:b::64
echo -e "search example.org\nnameserver 2001:db8:cafe:a::64\n" > /etc/resolv.conf
```

2.2.2.4. Configuración de los Clientes IPv4

Configuración, a través de lineas de comandos en el terminal, de las direcciones IPv6, Gateway y DNS, respectivamente:

Host h41

```
ip a a 203.0.113.41/24 dev eth0
ip route add default via 203.0.113.4
echo -e "search example.com\nnameserver 203.0.113.4\n" > /etc/resolv.conf
```

Host s42

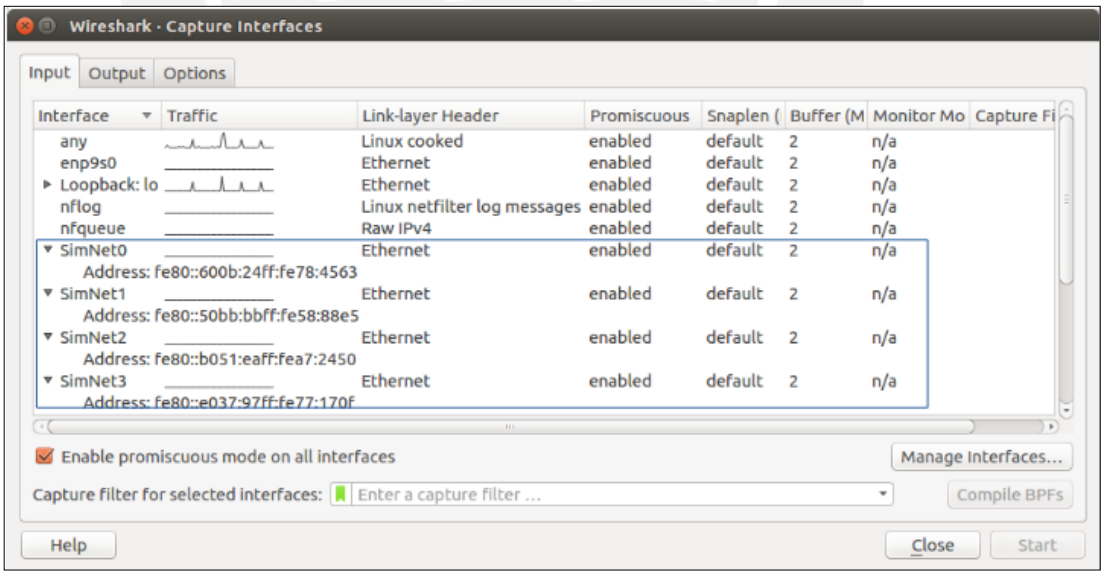
```
ip a a 203.0.113.42/24 dev eth0
ip route add default via 203.0.113.4
echo -e "search example.com\nnameserver 203.0.113.4\n" > /etc/resolv.conf
```

2.2.3. Verificación del Escenario

Una vez iniciado el escenario es necesario verificar las redes y nodos virtuales. Con el comando *netinfo* se muestra las cuatro redes virtuales, SimNet0, SimNet1, SimNet2 y SimNet3, con las que será posible capturar los paquetes usando la aplicación Wireshark.

```
dnsnat64-> netinfo
UML               IFACE           NET
HOST              SimNet0         Net0
HOST              SimNet1         Net1
HOST              SimNet2         Net2
HOST              SimNet3         Net3
-------------------------------------
h41               eth0            Net0
ha1               eth0            Net1
hb1               eth0            Net2
r4                eth0            Net0
r4                eth1            Net3
r64               eth0            Net3
r64               eth1            Net1
r64               eth2            Net2
s42               eth0            Net0
sb2               eth0            Net2
```

Dentro de Wireshark las redes virtuales SimNet se muestran de la siguiente manera:



Con la ayuda del comando vms se verifica los nodos iniciados dentro del escenario dnsnat64:

```
dnsnat64-> vms
Virtual machines from dnsnat64:
num     vms        enabled tty's Id
  1     r64        0 1
  2     r4         0 1
  3     ha1        0
  4     hb1        0
  5     sb2        0
  6     h41        0
  7     s42        0
```

2.2.4. Operación del Escenario DNS64+NAT64

En la figura Nº 19 se muestra el diagrama de flujo del escenario. Las pruebas que se presentan a continuación hacen referencia al RFC 7269 con respecto al manejo de paquetes ICMPv6 cuando superan los límites de MTU[21] de los enlaces principales.

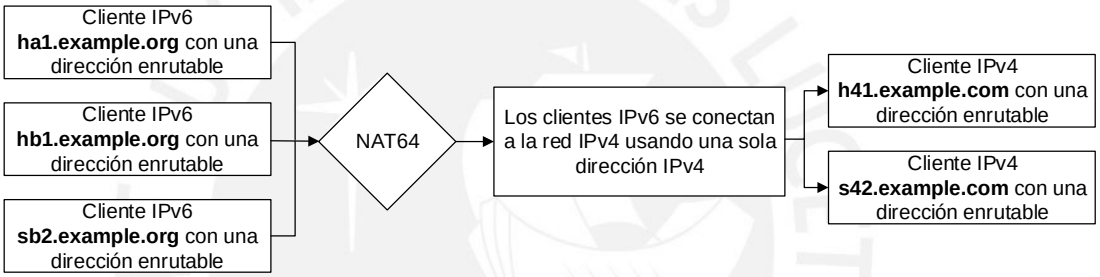


Fig. Nº 19 - Diagrama de flujo del escenario NAT64

2.2.4.1. Fragmentación de Paquetes

El RFC 7269 “NAT64 Deployment Options and Experience” resalta que IPv6 requiere que cada enlace en Internet tenga un MTU de 1280 bytes, o superior. Los nodos IPv6 recibirán mensajes de error PTB (Packet Too Big) si el siguiente salto en enlace tiene un MTU inferior a 1280 bytes. [52]

En IPv6, a diferencia de IPv4, no se permite la fragmentación de los paquetes en el router. Si un paquete requiere fragmentación debido a un MTU inferior en el enlace, el origen es responsable de fragmentarlo. Al igual que en IPv4, cada parte de un paquete fragmentado contiene un único valor de ID, de esta forma, el destino puede identificar fácilmente los fragmentos del paquete original para re-ensamblarlo.

[21] Unidad máxima de transferencia

En las siguientes pruebas de ICMPv6 se utilizan los nombres FDQN de los hosts. De esta forma se comprueba la operación de los servidores DNS.

Para la captura de las tramas utiliza el programa Wireshark. Sin embargo, en esta prueba de fragmentación de paquetes es necesario desactivar la opción "Reassemble fragmented IP datagrams", en IPv4 e IPv6. Con el objetivo de tener una captura "real" de los paquetes. En la siguiente figura se muestra la configuración a cambiar en Wireshark.

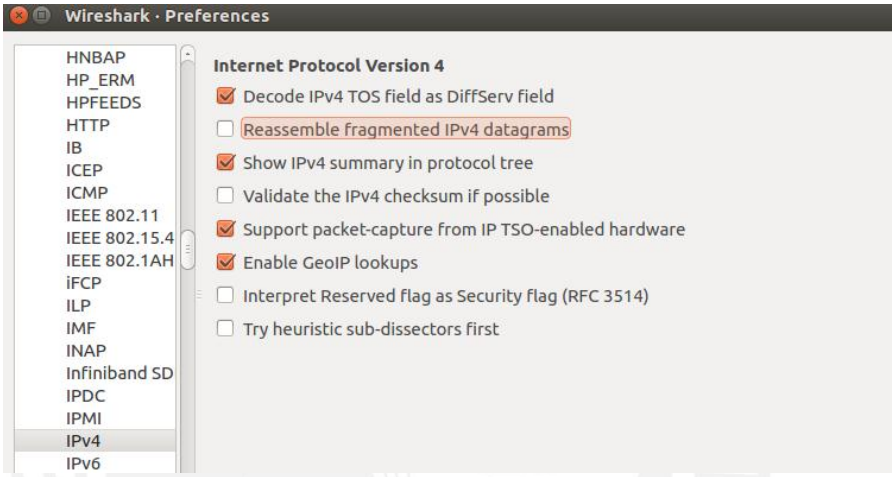


Fig. Nº 20 - Wireshark Fragmentación de paquetes

Envío de Paquetes: de IPv4 a IPv6

Para esta prueba de ICMPv6 se utiliza los nombres FDQN de los hosts, lo que comprueba la operación de DNS64. Además, se requiere un mapeo estático de las direcciones IPv6 a IPv4, como parte de las funciones del servicio NAT64 través del servicio Tayga. En el archivo */usr/local/etc/tayga.conf* se agregan los siguientes parámetros:

```
map 198.51.100.2 2001:db8:cafe:a::a1
map 198.51.100.3 2001:db8:cafe:b::b1
map 198.51.100.4 2001:db8:cafe:b::b2
```

Desde el host IPv4 h41.example.com (IP: 203.0.113.41/24), se realiza una prueba de conectividad al host IPv6 sb2.example.org, cuya dirección IPv6: 2001:db8:cafe:b::b2 esta mapeada a la dirección IPv4: 198.51.100.4. Además, el host h41 cuenta con los registros A y AAAA en el servidor DNS.

Por otro lado, en Linux, para determinar el tamaño de MTU a largo de la ruta entre ambos hosts se ejecuta el comando tracepath -n sb2.example.org:

```
root@h41:~# tracepath -n sb2.example.org
 1:  203.0.113.41                                          0.192ms pmtu 1500
 1:  203.0.113.4                                           0.508ms
 1:  203.0.113.4                                           0.258ms
 2:  192.0.2.64                                            0.433ms
 3:  198.51.100.1                                          0.453ms
 4:  198.51.100.1                                          0.453ms pmtu 1480
 4:  no reply
 5:  198.51.100.4                                          1.316ms reached
     Resume: pmtu 1480 hops 5 back 60
```

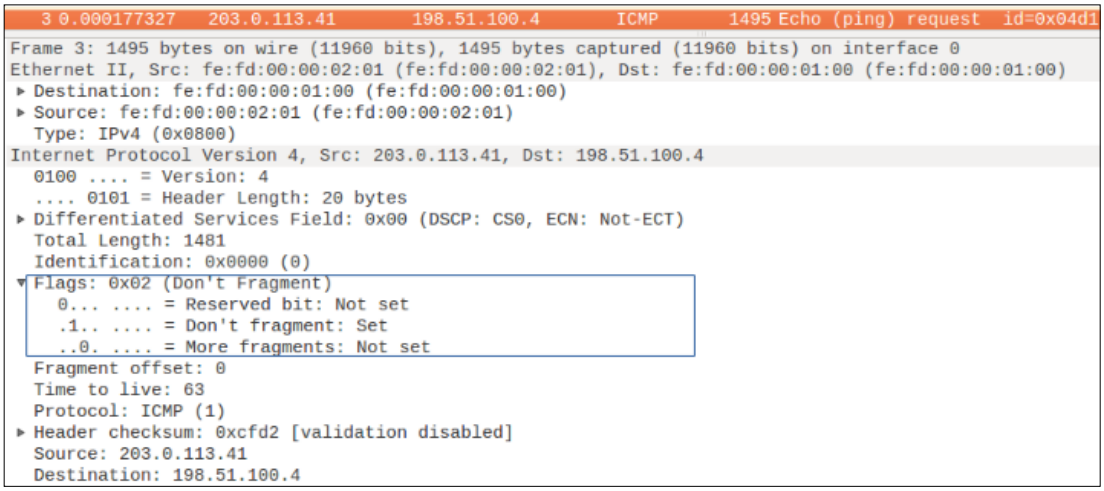
El resultado muestra un MTU de 1480 bytes en la interfaz NAT64 (IP: 198.51.100.1) gestionado por el servicio Tayga.

Si se envía un ping con un payload de 1453 bytes el tamaño total de datos del paquete IPv4 será:

1481bytes = 20 IP_header_bytes + 8 ICMP_header_bytes + 1453_payload_bytes

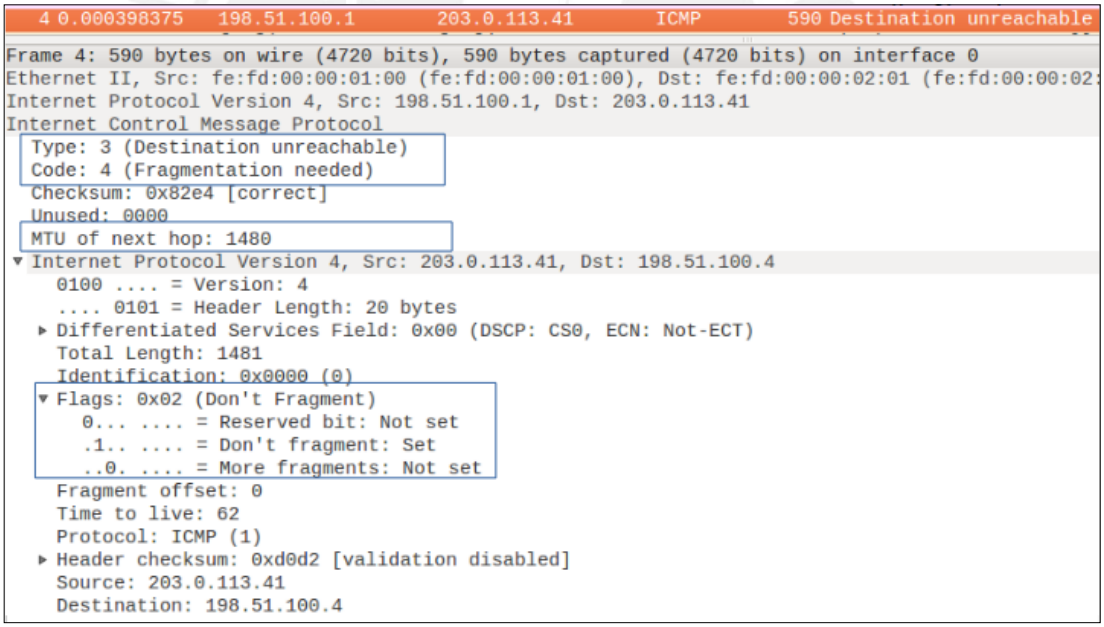
```
root@h41:~# ping -s 1453 -c1 sb2.example.org
PING sb2.example.org (198.51.100.4) 1453(1481) bytes of data.
From 198.51.100.1 icmp_seq=1 Frag needed and DF set (mtu = 1480)

--- sb2.example.org ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

El ping no se llega a completar y se produce el mensaje “Frag needed and DF set”, Para obtener más información de este error se realiza una captura Wireshark del mensaje “Echo (ping) request”, en la red SimNet3, entre el router 64 y el router r4:



En la captura anterior el paquete tiene el flag 0x02 “No fragmentar”.

Sin embargo, en la respuesta de la interfaz NAT64 (IP: 198.51.100.1) se tiene un mensaje tipo 3 “Destino no alcanzado”, y código 4: “Se requiere fragmentación”:



Por otro lado, si el paquete que se envía es menor o igual al tamaño de 1480 bytes de MTU:

```
root@h41:~# ping -s 1452 -c1 sb2.example.org
PING sb2.example.org (198.51.100.4) 1452(1480) bytes of data.
1460 bytes from 198.51.100.4: icmp_req=1 ttl=60 time=2.12 ms

--- sb2.example.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.126/2.126/2.126/0.000 ms
```

Envío de Paquetes: de IPv6 a IPv4

Conforme al procedimiento de la prueba anterior, se determina el tamaño de MTU. Del host sb2.example.org al host h41.example.com se ejecuta el comando tracepath6 -n h41.example.com:

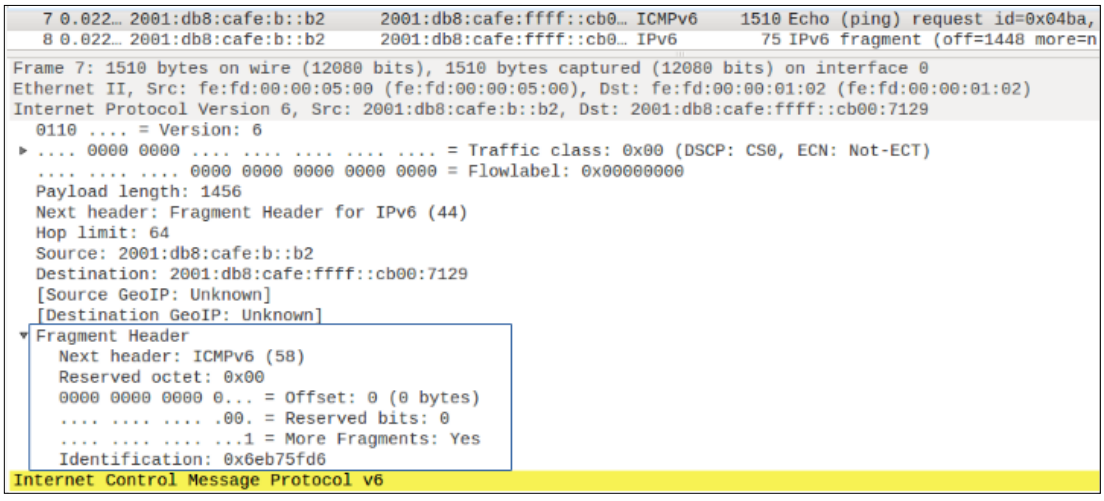
```
root@sb2:~# tracepath6 -n h41.example.com
 1?: [LOCALHOST]                      0.180ms pmtu 1500
 1:  2001:db8:cafe:b::64                                0.423ms
 1:  2001:db8:cafe:b::64                                0.382ms
 2:  2001:db8:cafe:ffff::c633:6401                      0.460ms
 3:  2001:db8:cafe:ffff::c000:240                       0.642ms
 4:  2001:db8:cafe:ffff::c000:204                       1.042ms
 5:  2001:db8:cafe:ffff::cb00:7129                      1.230ms reached
     Resume: pmtu 1500 hops 5 back 60
```

En esta ocasión se detecta un MTU de 1500 bytes en toda la ruta.
Al enviar un ping6 con tamaño de payload de 1453 bytes, el tamaño total de datos del paquete es:

1501bytes = 40 IPv6_header_bytes + 8 ICMP_header_bytes + 1453_payload_bytes

```
root@sb2:~# ping6 -s 1453 -c1 h41.example.com
PING h41.example.com(2001:db8:cafe:ffff::cb00:7129) 1453 data bytes

--- h41.example.com ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

La captura Wireshark en el segmento de red SimNet2 es:



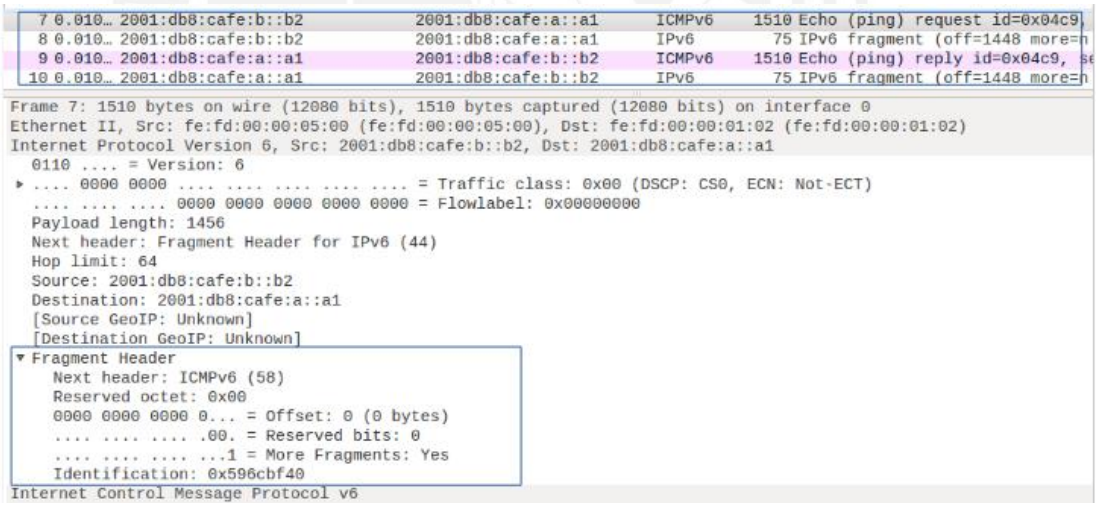
El MTU del enlace es 1500 bytes. El paquete que se intenta enviar tiene un tamaño de 1501 bytes. Las tramas 7 y 8 corresponden al paquete fragmentado por el host que realiza la petición ICMPv6.

Se envía un paquete, con el mismo tamaño de datos dentro de la red IPv6, desde el host sb2.example.org al host ha1.example.org:

```
root@sb2:~# ping6 -s1453 -c1 ha1.example.org
PING ha1.example.org(2001:db8:cafe:a::a1) 1453 data bytes
1461 bytes from 2001:db8:cafe:a::a1: icmp_seq=1 ttl=63 time=0.968 ms

--- ha1.example.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.968/0.968/0.968/0.000 ms
```

En este caso, se obtiene respuesta del host destino. Ambos equipos están en la misma zona (example.org) pero pertenecen a segmentos de red IPv6 distintos.

La captura Wireshark en el segmento de red SimNet2:



En esta oportunidad, la captura Wireshark muestra que la fragmentación del paquete es realizada por ambos hosts. Es decir, el host de origen fragmenta el paquete (tramas 7 y 8) para luego enviarlo; el host de destino, debe esperar la llegada del fragmento con el bit MF=0, para re-emsamblar el paquete, fragmentarlo y enviar la respuesta (tramas 9 y 10).

Prueba ICMPv6 PTB (Packet Too Big)

Se modifica el tamaño de MTU a 1280 bytes en la interfaces eth0 y eth1 de los routers r64 y r4, respectivamente:

```
root@r64:~# ip link set mtu 1280 dev eth0
```

```
root@r4:~# ip link set mtu 1280 dev eth1
```

Se ejecuta el comando tracepath6 -n desde los hosts sb2.example.org y h41.example.com para determinar el tamaño de MTU en ambos sentidos:

```
root@sb2:~# tracepath6 -n h41.example.com
 1?: [LOCALHOST]                      0.096ms pmtu 1500
 1:  2001:db8:cafe:b::64                                0.385ms
 1:  2001:db8:cafe:b::64                                0.316ms
 2:  2001:db8:cafe:ffff::c633:6401                      0.686ms
 3:  2001:db8:cafe:ffff::c000:240                       0.457ms
 4:  2001:db8:cafe:ffff::c000:240                       0.418ms pmtu 1300
 4:  2001:db8:cafe:ffff::c000:204                       0.586ms
 5:  2001:db8:cafe:ffff::cb00:7129                      1.037ms reached
     Resume: pmtu 1300 hops 5 back 60
```

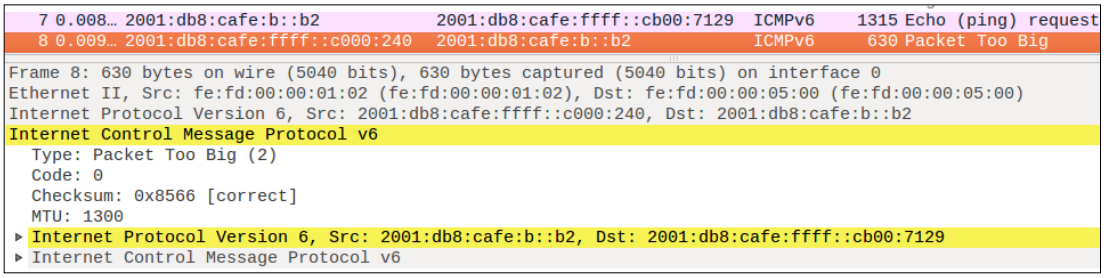
```
root@h41:~# tracepath -n sb2.example.org
 1:  203.0.113.41                                       0.136ms pmtu 1500
 1:  203.0.113.4                                        0.260ms
 1:  203.0.113.4                                        0.218ms
 2:  203.0.113.4                                        0.216ms pmtu 1280
 2:  192.0.2.64                                         0.351ms
 3:  198.51.100.1                                       0.441ms
 4:  no reply
 5:  198.51.100.4                                       2.447ms reached
     Resume: pmtu 1280 hops 5 back 60
```

Al enviar un ping6 con un tamaño de payload de 1253 bytes de datos desde el host sb2.example.org, el tamaño total de datos del paquete es:

1301bytes = 40 IPv6_header_bytes + 8 ICMP_header_bytes + 1253_payload_bytes

```
root@sb2:~# ping6 -s1253 -c1 h41.example.com
PING h41.example.com(2001:db8:cafe:ffff::cb00:7129) 1253 data bytes
From 2001:db8:cafe:ffff::c000:240 icmp_seq=1 Packet too big: mtu=1300

--- h41.example.com ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

La siguiente captura Wireshark muestra que el mensaje PTB es enviado por la interfaz eth0 del router r64 que tiene un MTU de 1300 bytes.



En las pruebas realizadas se demuestra que el traductor NAT64 stateless gestionado por el programa Tayga no tiene la capacidad de mantener la comunicación punto a punto cuando:

- Un paquete IPv4 que puede ser fragmentado se convierte en un paquete IPv6 que no debe ser fragmentado.

- Un paquete IPv6 que no debe ser fragmentado se convierte un paquete IPv4 que puede ser fragmentado.

2.2.4.2. Destino No alcanzable

Los mensajes Destination Unreachable, o Destino no alcanzable, por lo general son mostrados cuando un paquete no puede ser entregado. Y existen otras razones, además de la congestión. En NAT64 se configura un pool dinámico y limitado de direcciones IPv4 que son usadas para traducir las direcciones de los hosts IPv6 con la finalidad de alcanzar los nodos IPv4. Si este pool alcanza el máximo de traducciones permitidas se generan mensajes del tipo Destination Unreachable.

Las razones de este mensaje de error pueden ser codificadas de acuerdo a la tabla N°14.

Tabla Nº 14 – Códigos de Destino No Alcanzable [53]

Código	Nombre	Descripción
0	No route to destination	El paquete no puede entregado, el router no conoce la ruta hacia el destino
1	Communication with destination administratively prohibed	El paquete fue bloqueado debido a un filtro
2	Beyond scope of source address	Error generado cuando el origen es una dirección de enlace local y el destino es una dirección global. [RFC 4443]
3	Address unreachable	El destino no puede ser alcanzado.
4	Port unreachable	El puerto de destino TCP/UDP no existe, o el destino no está mostrado en tal puerto.
5	Source address failed ingress/egress policy	El paquete fue bloqueado debido a un filtro. [RFC 4443]
6	Reject route to destination	Los paquetes con un prefijo específico son filtrados y bloqueados. [RFC 4443]

En esta prueba se fuerza un error código 5. Primero, en el router r64, se modifica el pool dinámico IPv4 para permitir la traducción de direcciones de solo 3 equipos IPv6. Para ello, en el archivo /etc/tayga.conf se modifica el siguiente parámetro:

```
dynamic-pool 198.51.100.0/30
```

Luego, se envía un paquete ICMPv6 desde cada uno de los hosts de la red IPv6 hacia uno de los hosts en la red IPv4. En el host ha1 se ejecuta el comando ping6:

```
root@ha1:~# ping6 -c1 s42.example.com
PING s42.example.com(2001:db8:cafe:ffff::cb00:712a) 56 data bytes
64 bytes from 2001:db8:cafe:ffff::cb00:712a: icmp_seq=1 ttl=60 time=0.806 ms

--- s42.example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.806/0.806/0.806/0.000 ms
root@ha1:~#
```

En el host sb2 se ejecuta el comando ping6:

```
root@sb2:~# ping6 -c1 s42.example.com
PING s42.example.com(2001:db8:cafe:ffff::cb00:712a) 56 data bytes
64 bytes from 2001:db8:cafe:ffff::cb00:712a: icmp_seq=1 ttl=60 time=1.39 ms

--- s42.example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.392/1.392/1.392/0.000 ms
root@sb2:~#
```

Finalmente, en el host hb1 se ejecuta el comando ping6:

```
root@hb1:~# ping6 -c1 s42.example.com
PING s42.example.com(2001:db8:cafe:ffff::cb00:712a) 56 data bytes
From 2001:db8:cafe:ffff::c633:6401 icmp_seq=1 Destination unreachable: Unknown c
ode 5

--- s42.example.com ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

root@hb1:~# _
```

El envío de paquetes ICMPv6 fue exitoso desde los host ha1 y sb2. Sin embargo, cuando se envía el ping6 desde el host hb1 se muestra el error código 5. En la siguiente captura de paquetes se observa que el mensaje de respuesta fue generado por la interfaz nat64:

La IP 198.51.100.1 de la interfaz nat64 pertenece a la subnet 192.51.100.0/30 y que a la vez es el pool dinámico utilizado para la traducción de las direcciones de los host de la red IPv6. Entonces:

Del pool dinámico IPv4 (198.51.100.0/30) se han asignado tres direcciones:

198.51.100.1 → interfaz tun64, asignación fija

198.51.100.2 → host sb2 (2001:db8:cafe:b::b2), asignación dinámica

198.51.100.3 → host ha1 (2001:db8:cafe:a::a1), asignación dinámica.

La interfaz tun64 (dirección IPv4 198.51.100.1, traducida a IPv6 2001:db8:cafe:ffff::c633:6401 en formato hexadecimal) del router r64, envía un mensaje de respuesta ICMPv6 Destination Unreachable al host hb1 (dirección IPv6 2001:db8:cafe:b::b1).

Se trata de un mensaje tipo=1, código=5. En donde se indica que no existe ninguna dirección IPv4 disponible para ser asignada al host hb1 en el proceso de NAT64.

2.3. Escenario 6to4

2.3.1. Topología

La figura Nº 21 muestra la topología del escenario 6to4. Se configura un túnel 6to4 en cada router frontera de las redes privadas IPv6. Todo el tráfico IPv6 destinado a la red IPv6 nativa es enrutado sobre IPv4 a través de un túnel hacia uno de los router relays 6to4. El tráfico desde la red IPv6 nativa hacia los hosts IPv6 es enrutado sobre IPv4 a través del túnel hacia el router frontera de la red privada IPv6, y luego hacia el host de destino IPv6.

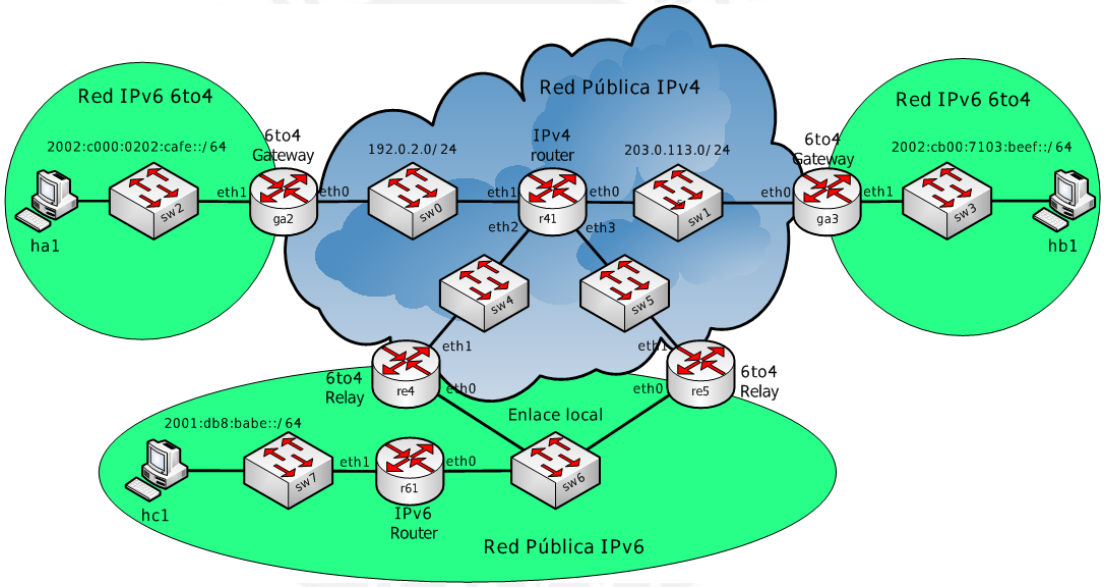


Fig. Nº 21 - Topología 6to4

El escenario 6to4 está conformado por los siguientes componentes, tabla Nº 15:

Tabla Nº 15 – Redes utilizadas en el escenario 6RD

Nombre	Dirección IPv4	Dirección IPv6	Descripción
SW0 (Net0)	192.0.2.0/24		Red IPv4 pública
SW1 (Net1)	203.0.113.0/24		Red IPv4 pública
SW2 (Net2)		2002:c000:0202:cafe::/64	Red IPv6 privada
SW3 (Net3)		2002:cb00:7103:beef::/64	Red IPv6 privada

SW4 (Net4)	10.0.0.0/24		Red IPv4 privada
SW5 (Net5)	10.0.1.0/24		Red IPv4 privada
SW6 (Net6)		fe80::/64	Enlace local
SW7 (Net7)		2001:db8:babe::/64	Red IPv6 pública

Las características de todos los sistemas utilizados en la virtualización se muestran en la tabla Nº 16:

Tabla Nº 16 – Descripción de los nodos

Nombre	Dirección IPv4 pública	Dirección IPv6 global	Dirección IPv6 local	Sistema Operativo
r41	192.0.2.1/24 203.0.113.1/24 10.0.0.1/24 10.0.1.1/24	-	-	Debian 6
ga2	192.0.2.2/24	2002:c000:0202:cafe::a2/64	fe80::fcfd:ff:fe00:201/64	Debian 6
ga3	203.0.113.3/24	2002:cb00:7103:beef::a3/64	fe80::fcfd:ff:fe00:401/64	Debian 6
re4	10.0.0.4/24	-	fe80::fcfd:ff:fe00:600/64	Debian 6
re5	10.0.1.5/24	-	fe80::fcfd:ff:fe00:700/64	Debian 6
r61	-	2001:db8:babe::61/64	fe80::fcfd:ff:fe00:801/64 fe80::fcfd:ff:fe00:800/64	Debian 6
ha1	-	2002:c000:202:cafe:fcfd:ff:fe00:300/64	fe80::fcfd:ff:fe00:300/64	Debian 6
hb1	-	2002:cb00:7103:beef::b1/64	fe80::fcfd:ff:fe00:500/64	Debian 6
hc1	-	2001:db8:babe::c1/64	fe80::fcfd:ff:fe00:900/64	Debian 6

2.3.2. Construcción del Escenario

Este escenario se construye a través del archivo 6to4.vnuml (Anexo B). La disposición de las redes y nodos es la siguiente:

```
6to4-> netinfo
UML             IFACE           NET
HOST            SimNet0         Net0
HOST            SimNet1         Net1
HOST            SimNet2         Net2
HOST            SimNet3         Net3
HOST            SimNet4         Net4
HOST            SimNet5         Net5
HOST            SimNet6         Net6
HOST            SimNet7         Net7
--------------------------------------
ga2             eth0            Net0
ga2             eth1            Net2
ga3             eth0            Net1
ga3             eth1            Net3
ha1             eth0            Net2
hb1             eth0            Net3
hc1             eth0            Net7
r41             eth0            Net1
r41             eth1            Net0
r41             eth2            Net4
r41             eth3            Net5
r61             eth0            Net6
r61             eth1            Net7
re4             eth0            Net6
re4             eth1            Net4
re5             eth0            Net6
re5             eth1            Net5
```

2.3.2.1. Configuración del túnel 6to4

Una de las características del mecanismo 6to4 es el uso del prefijo 2002::/16 en la configuración del túnel. Los siguientes 32 bits se concatenan a partir de la dirección IPv4 pública. En el caso de la primera red privada IPv6, el router frontera (ga2) tiene el prefijo:

2002:c000:0202::/16 → c000:0202 (hexadecimal) = 192.0.2.2 (decimal)

```
root@ga2:~# ifconfig 6to4
6to4      Link encap:IPv6-in-IPv4
          inet6 addr: 2002:c000:202:cafe::/16 Scope:Global
          inet6 addr: ::192.0.2.2/128 Scope:Compat
          UP RUNNING NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

En la segunda red privada IPv6, el router frontera (ga3) tiene el siguiente prefijo:
2002:cb00:7103::/16 → cb00:7103 (hexadecimal) = 203.0.113.3 (decimal)

```
root@ga3:~# ifconfig 6to4
6to4      Link encap:IPv6-in-IPv4
          inet6 addr: 2002:cb00:7103:beef::/16 Scope:Global
          inet6 addr: ::203.0.113.3/128 Scope:Compat
          UP RUNNING NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Los router relays 6to4 utilizan la dirección anycast 192.88.99.1 con el objetivo de responder las solicitudes de conectividad nativa IPv6 de los routers de borde 6to4. Para permitir la propagación de este enrutamiento, es necesario alcanzar el prefijo 192.88.99.0/24. Dado que hay dos router relays con el mismo prefijo, habrá una ruta hacia cada router; entonces, la tabla de enrutamiento del router r41 es:

```
root@r41:~# route -n -4
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 eth2
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 eth3
192.0.2.0       0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.88.99.0     10.0.0.4        255.255.255.0   UG    1      0        0 eth2
192.88.99.0     10.0.1.5        255.255.255.0   UG    2      0        0 eth3
```

En este caso, se toma la primera a través del Gateway 10.0.0.4 pues tiene una métrica igual a 1. Además, en cada router frontera 6to4 se configura una ruta para que todo el tráfico IPv6 nativo se propague hacia el primer router relay (::192.88.99.1) disponible a través del túnel 6to4:

```
root@ga2:~# route -n -6
Kernel IPv6 routing table
Destination                    Next Hop                   Flag Met Ref Use If
::/96                          ::                         Un   256 0     1 6to4
2002:c000:202:cafe::/64        ::                         U    256 0     0 eth1
2002::/16                      ::                         U    256 0     0 6to4
fe80::/64                      ::                         U    256 0     0 eth0
fe80::/64                      ::                         U    256 0     0 eth1
fe80::/64                      ::                         U    256 0     0 6to4
::/0                           ::192.88.99.1              UG   1   0     0 6to4
```

```
root@ga3:~# route -n -6
Kernel IPv6 routing table
Destination                    Next Hop                   Flag Met Ref Use If
::/96                          ::                         Un   256 0     1 6to4
2002:cb00:7103:beef::/64       ::                         U    256 0     0 eth1
2002::/16                      ::                         U    256 0     0 6to4
fe80::/64                      ::                         U    256 0     0 eth0
fe80::/64                      ::                         U    256 0     0 eth1
fe80::/64                      ::                         U    256 0     0 6to4
::/0                           ::192.88.99.1              UG   1   0     0 6to4
```

Cada router relay 6to4 se anuncia al exterior con el prefijo 2002:c058:6301::/16: c058:6301 (hexadecimal) = 192.88.99.1 (decimal)

```
root@re4:~# ifconfig 6to4
6to4      Link encap:IPv6-in-IPv4
          inet6 addr: 2002:c058:6301::/16 Scope:Global
          inet6 addr: ::192.88.99.1/128 Scope:Compat
          UP RUNNING NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
root@re5:~# ifconfig 6to4
6to4      Link encap:IPv6-in-IPv4
          inet6 addr: 2002:c058:6301::/16 Scope:Global
          inet6 addr: ::192.88.99.1/128 Scope:Compat
          UP RUNNING NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

2.3.3. Operación de 6to4

2.3.3.1. Comunicación TCP asimétrica

Desde el host 6to4 (h41) se establece una conexión TCP a través de una solicitud http al host IPv6 nativo (hc1):

```
root@ha1:~# wget http://[2001:db8:babe::c1]
--2016-11-20 18:34:46--  http://[2001:db8:babe::c1]/
Connecting to 2001:db8:babe::c1:80... conectado.
Petici◆n HTTP enviada, esperando respuesta... 200 OK
Longitud: 177 [text/html]
Saving to: `index.html'

100%[======================================>] 177         --.-K/s   in 0s

2016-11-20 18:34:46 (1,24 MB/s) - `index.html' saved [177/177]
```

La captura Wireshark en la red sw2 muestra lo siguiente:

	Time	Source	Destination	Protoco	Length	Info
1	0.000000000	2002:c000:202:cafe:fcfd:ff:fe00:300	2001:db8:babe::c1	TCP	94	49580 → 80 [SYN]
2	0.001031416	fe80::fcfd:ff:fe00:201	ff02::1:ff00:300	ICMPv6	86	Neighbor Solicita
3	0.001108311	2002:c000:202:cafe:fcfd:ff:fe00:300	fe80::fcfd:ff:fe00:201	ICMPv6	86	Neighbor Advertis
4	0.001160972	2001:db8:babe::c1	2002:c000:202:cafe:fcfd:ff:fe00:300	TCP	94	80 → 49580 [SYN,
5	0.001251556	2002:c000:202:cafe:fcfd:ff:fe00:300	2001:db8:babe::c1	TCP	86	49580 → 80 [ACK]
6	0.003008902	2002:c000:202:cafe:fcfd:ff:fe00:300	2001:db8:babe::c1	HTTP	203	GET / HTTP/1.0
7	0.004371226	2001:db8:babe::c1	2002:c000:202:cafe:fcfd:ff:fe00:300	TCP	86	80 → 49580 [ACK]
8	0.018633445	2001:db8:babe::c1	2002:c000:202:cafe:fcfd:ff:fe00:300	HTTP	576	HTTP/1.1 200 OK
9	0.018702239	2002:c000:202:cafe:fcfd:ff:fe00:300	2001:db8:babe::c1	TCP	86	49580 → 80 [ACK]
10	0.038778116	2002:c000:202:cafe:fcfd:ff:fe00:300	2001:db8:babe::c1	TCP	86	49580 → 80 [FIN,
11	0.040070389	2001:db8:babe::c1	2002:c000:202:cafe:fcfd:ff:fe00:300	TCP	86	80 → 49580 [FIN,
12	0.040254979	2002:c000:202:cafe:fcfd:ff:fe00:300	2001:db8:babe::c1	TCP	86	49580 → 80 [ACK]

Captura Wireshark en la red sw4:

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000				42	<Ignored>
2	0.000096800				42	<Ignored>
3	0.000158819	2002:c000:202:cafe:fcfd:ff:fe00:300	2001:db8:babe::c1	TCP	114	49579 → 80 [SYN]
4	0.002410013	2002:c000:202:cafe:fcfd:ff:fe00:300	2001:db8:babe::c1	TCP	106	49579 → 80 [ACK]
5	0.003923823	2002:c000:202:cafe:fcfd:ff:fe00:300	2001:db8:babe::c1	HTTP	223	GET / HTTP/1.0
6	0.023761471	2002:c000:202:cafe:fcfd:ff:fe00:300	2001:db8:babe::c1	TCP	106	49579 → 80 [ACK]
7	0.035746236	2002:c000:202:cafe:fcfd:ff:fe00:300	2001:db8:babe::c1	TCP	106	49579 → 80 [FIN,
8	0.036776045	2002:c000:202:cafe:fcfd:ff:fe00:300	2001:db8:babe::c1	TCP	106	49579 → 80 [ACK]

Captura Wireshark en la red sw5:

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	2001:db8:babe::c1	2002:c000:202:cafe:fcfd:ff:fe00:300	TCP	114	80 → 49581 [SYN, ACK]
2	0.002581474	2001:db8:babe::c1	2002:c000:202:cafe:fcfd:ff:fe00:300	TCP	106	80 → 49581 [ACK] Seq=1
3	0.004777912	2001:db8:babe::c1	2002:c000:202:cafe:fcfd:ff:fe00:300	HTTP	596	HTTP/1.1 200 OK (text
4	0.012255958	2001:db8:babe::c1	2002:c000:202:cafe:fcfd:ff:fe00:300	TCP	106	80 → 49581 [FIN, ACK]
5	5.017124315				42	<Ignored>

Se demuestra que el tráfico con destino al host hc1 lo encamina el router relay re4; el tráfico de respuesta lo encamina el router relay re5. Dando como resultado, caminos de ida y vuelta asimétricos.

2.4. Escenario IPv6 Rapid Deployment (6RD)

2.4.1. Topología

Como se muestra en la figura Nº22, se establecen túneles entre los routers relay de frontera (6RD BR) y los routers de frontera (6RD Gateway). Los routers de frontera están desplegados en las instalaciones del cliente. Los routers relay de frontera permiten la conexión hacia La red IPv6 nativa. Las direcciones IPv6 son asignadas a través de mecánicos de direccionamiento dinámico sin estado (SLAAC), y estático. La diferencia más importante entre los mecanismos 6RD y 6to4 es el prefijo. Los hosts 6RD utilizan el prefijo IPv6 asignado por su propio ISP, mientras que los hosts 6to4 comparten el prefijo común 2002::/16.

En esta topología se usan los prefijos IPv4 147.83.0.0/16 e IPv6 2001:40b0::/32, ambos son parte de un conjunto de dominios gestionado en el Sistema Autónomo (AS) numero 13041 asignado por la RIPE y con registro en España.

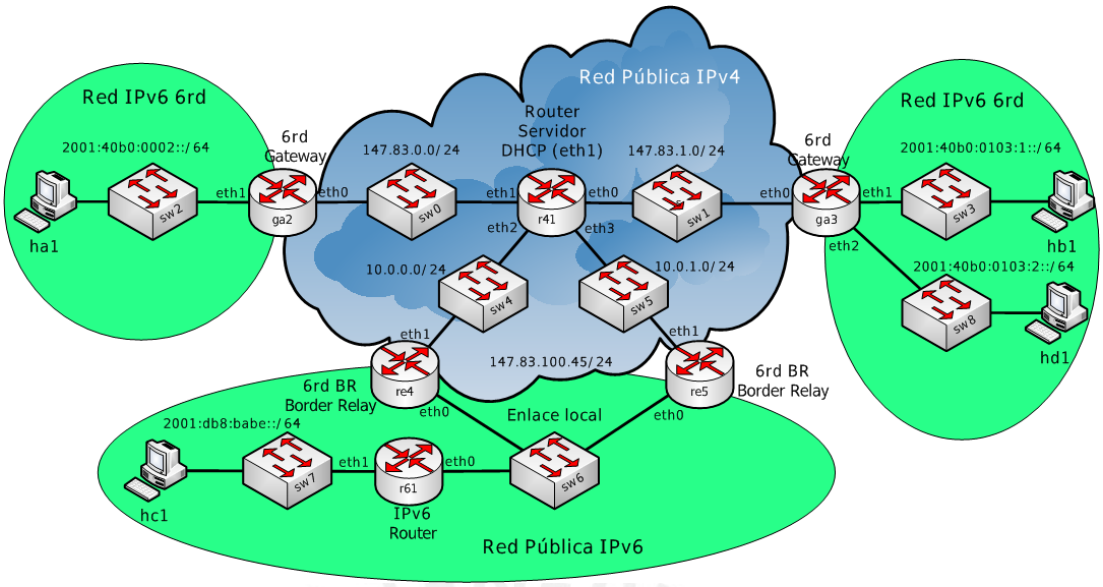


Fig. Nº 22 – Topología 6RD

La topología mostrada en la figura Nº21 está conformada por los siguientes componentes, tabla Nº 17:

Tabla Nº 17 – Redes utilizadas en el escenario 6RD

Nombre	Dirección IPv4	Dirección IPv6	Descripción
SW0 (Net0)	147.83.0.0/24	-	Red IPv4 pública
SW1 (Net1)	147.83.1.0/24	-	Red IPv4 pública
SW2 (Net2)	-	2001:40b0:0002::/64	Red IPv6 privada
SW3 (Net3)	-	2001:40b0:0103:1::/64	Red IPv6 privada
SW4 (Net4)	10.0.0.0/24	-	Red IPv4 privada
SW5 (Net5)	10.0.1.0/24	-	Red IPv4 privada
SW6 (Net6)	-	fe80::/64	Enlace local
SW7 (Net7)	-	2001:db8:babe::/64	Red IPv6 pública
SW8 (Net8)	-	2001:40b0:0103:2::/64	Red IPv6 privada

Las características de todos los sistemas utilizados en la virtualización se muestran en la siguiente tabla:

Tabla Nº 18 – Descripción de los nodos

Nombre	Dirección IPv4	Dirección IPv6 global	Dirección IPv6 local	Sistema Operativo
r41	147.83.0.1/24 147.83.1.1/24 10.0.0.1/24 10.0.1.1/24	-	-	Debian 6
ga2	147.83.0.2/24	2001:40b0:0002::a2/64	fe80::fcfd:ff:fe00:201/64	Debian 6
ga3	147.83.1.3/24	2001:40b0:0103:1::a3/64 2001:40b0:0103:2::a3/64	fe80::fcfd:ff:fe00:401/64 fe80::fcfd:ff:fe00:402/64	Debian 6
re4	10.0.0.4/24	-	fe80::fcfd:ff:fe00:700/64	Debian 6
re5	10.0.1.5/24	-	fe80::fcfd:ff:fe00:800/64	Debian 6
r61	-	2001:db8:babe::61/64	fe80::fcfd:ff:fe00:900/64 fe80::fcfd:ff:fe00:901/64	Debian 6
ha1	-	2001:40b0:2:0:fcfd:ff:fe00:300/64	fe80::fcfd:ff:fe00:300/64	Debian 6
hb1	-	2001:40b0:103:1::b1/64	fe80::fcfd:ff:fe00:500/64	Debian 6
hc1	-	2001:db8:babe::c1/64	fe80::fcfd:ff:fe00:a00/64	Debian 6
hd1	-	2001:40b0:103:2::d1/64	fe80::fcfd:ff:fe00:600/64	Debian 6

2.4.2. Construcción del Escenario

Este escenario se construye a través del archivo 6rd.vnuml. La disposición de las redes y nodos es la siguiente:

```
6rd-> netinfo
UML               IFACE           NET
HOST              SimNet0         Net0
HOST              SimNet1         Net1
HOST              SimNet2         Net2
HOST              SimNet3         Net3
HOST              SimNet4         Net4
HOST              SimNet5         Net5
HOST              SimNet6         Net6
HOST              SimNet7         Net7
HOST              SimNet8         Net8
---------------------------------------
ga2               eth0            Net0
ga2               eth1            Net2
ga3               eth0            Net1
ga3               eth1            Net3
ga3               eth2            Net8
ha1               eth0            Net2
hb1               eth0            Net3
hc1               eth0            Net7
hd1               eth0            Net8
r41               eth0            Net1
r41               eth1            Net0
r41               eth2            Net4
r41               eth3            Net5
r61               eth0            Net6
r61               eth1            Net7
re4               eth0            Net6
re4               eth1            Net4
re5               eth0            Net6
re5               eth1            Net5
```

2.4.2.1. Configuración del túnel 6RD

La topología de la figura Nº 24 tiene en común el prefijo IPv4 147.83.0.0/16 y el prefijo IPv6 2001:40b0::/32. Esto permite crear túneles 6RD con un prefijo IPv6 ::/48. Por lo tanto, se tiene 16 bits por sitio para segmentar la red local, permitiendo 65536 subredes con un prefijo ::/64 para asignar a los hosts.

Configuración del túnel 6RD en el router de frontera ga2 tabla Nº 19

La dirección IPv4 se obtiene de forma dinámica en la interfaz WAN (eth0), a través del cliente DHCP. El servidor DHCP es administrado en el router **r4,** conectado a la interfaz eth1. El programa ISC-DHCP gestiona el servicio DHCP en ambos casos.

Tabla Nº 19 – Parámetros de configuración del túnel 6RD en el router ga2

Parámetro	Valor
Prefijo IPv6 del ISP	2001:40b0::/32
Prefijo IPv4 en común	147.83.0.0/16
Sufijo IPv4	Cliente DHCP
Prefijo IPv6 delegado	2001:40b0:{sufijo IPv4 hex}::/48
Dirección IPv6 de la interfaz túnel 6RD	2001:40b0:{sufijo IPv4 hex}::1/32
Subredes	2001:40b0:{sufijo IPv4 hex}::/64

Configuración dinámica obtenida en la interfaz **eth0** del router **ga2:**

```
root@ga2:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr fe:fd:00:00:02:00
          inet addr:147.83.0.2  Bcast:147.83.0.255  Mask:255.255.255.0
          inet6 addr: fe80::fcfd:ff:fe00:200/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1543 (1.5 KiB)  TX bytes:1996 (1.9 KiB)
          Interrupt:5
```

Configuración del túnel **6RD** en router **ga3**:

```
root@ga2:~# ifconfig 6rd
6rd       Link encap:IPv6-in-IPv4
          inet6 addr: ::147.83.0.2/128 Scope:Compat
          inet6 addr: 2001:40b0:2::1/32 Scope:Global
          UP RUNNING NOARP  MTU:1480  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:208 (208.0 B)  TX bytes:208 (208.0 B)
```

Configuración del túnel 6RD en el router de frontera ga3 tabla Nº 20

Tabla Nº 20 - Parámetros de configuración del túnel 6RD en el router ga3

Parámetro	Valor
Prefijo IPv6 del ISP	2001:40b0::/32
Prefijo IPv4 en común	147.83.0.0/16
Sufijo IPv4	0.0.1.3/16
Prefijo IPv6 delegado	2001:40b0:0103::/48
Dirección IPv6 de la interfaz túnel 6RD	2001:40b0:0103::1/32
Subredes	2001:40b0:0103:1::/64 2001:40b0:0103:2::/64

Configuración del túnel **6RD**:

```
root@ga3:~# ifconfig 6rd
6rd       Link encap:IPv6-in-IPv4
          inet6 addr: ::147.83.1.3/128 Scope:Compat
          inet6 addr: 2001:40b0:103::1/32 Scope:Global
          UP RUNNING NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Configuración del túnel 6RD en los routers relay de frontera re4 y re5 tabla Nº 21

Tabla Nº 21 - Parámetros de configuración del túnel 6RD en los routers re4 y re5

Parámetro	Valor
Prefijo IPv6 del ISP	2001:40b0::/32
Prefijo IPv4 en común	147.83.0.0/16
Sufijo IPv4	0.0.100.45/16
Prefijo IPv6 delegado	2001:40b0:642d::/48
Dirección IPv6 de la interfaz túnel 6RD	2001:40b0:642d::1/32

Configuración del túnel **6RD** en el router relay de frontera **re4**:

```
root@re4:~# ifconfig 6rd
6rd       Link encap:IPv6-in-IPv4
          inet6 addr: ::147.83.100.45/128 Scope:Compat
          inet6 addr: 2001:40b0:642d::1/32 Scope:Global
          UP RUNNING NOARP  MTU:1480  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:104 (104.0 B)  TX bytes:0 (0.0 B)
```

Configuración del túnel **6RD** en el router relay de frontera **re5**:

```
root@re5:~# ifconfig 6rd
6rd       Link encap:IPv6-in-IPv4
          inet6 addr: ::147.83.100.45/128 Scope:Compat
          inet6 addr: 2001:40b0:642d::1/32 Scope:Global
          UP RUNNING NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

2.4.3. Operación del Escenario 6RD

2.4.3.1. Comunicación ICMPv6 Asimétrica

El host IPv6 **hc1** envía una petición ICMPv6 hacia el host IPv6 **hd1**. Las tablas de enrutamiento de los routers **r61** y **r41** muestra el sentido del flujo del tráfico.

El router **r61** tiene como prioridad el router **re5 (fe80::fcfd:ff:fe00:800)**:

```
root@r61:~# route -n -6
Kernel IPv6 routing table
Destination                    Next Hop                   Flag Met Ref Use If
2001:db8:babe::/64             ::                         U    256 0     0 eth1
2001:40b0::/32                 fe80::fcfd:ff:fe00:800     UG   1   0     0 eth0
2001:40b0::/32                 fe80::fcfd:ff:fe00:700     UG   2   0     0 eth0
```

El router **r41** tiene como prioridad el router **re4 (10.0.0.4)**:

```
root@r41:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 eth2
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 eth3
147.83.0.0      0.0.0.0         255.255.255.0   U     0      0        0 eth1
147.83.1.0      0.0.0.0         255.255.255.0   U     0      0        0 eth0
147.83.100.0    10.0.0.4        255.255.255.0   UG    1      0        0 eth2
147.83.100.0    10.0.1.5        255.255.255.0   UG    2      0        0 eth3
```

El flujo del tráfico se muestra en la figura Nº 23:

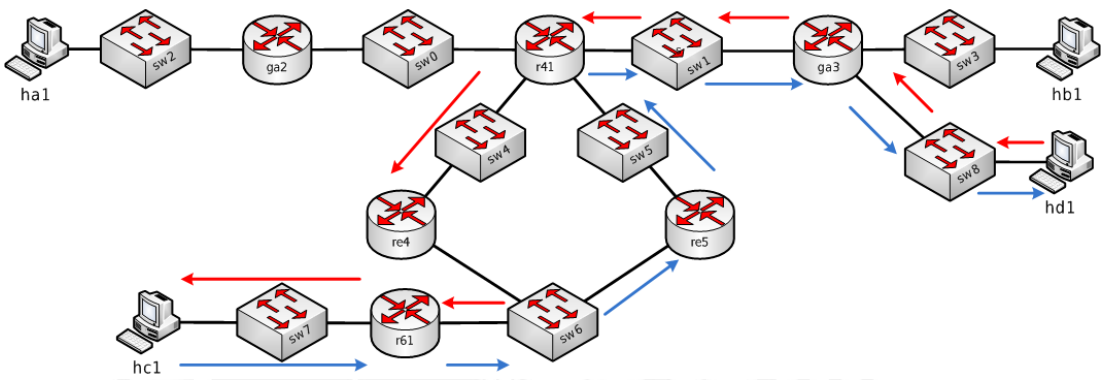


Fig. Nº 23 – Flujo de tráfico en la red 6RD

2.1.1.1. Unidad Máxima de Transmisión (MTU)

Se configura la interfaz túnel 6RD de los routers relay de frontera, re4 y re5, con un MTU de 1280 bytes.

```
root@re4:~# ip link set mtu 1280 dev 6rd
root@re4:~# ifconfig 6rd
6rd       Link encap:IPv6-in-IPv4
          inet6 addr: ::147.83.100.45/128 Scope:Compat
          inet6 addr: 2001:40b0:642d::1/32 Scope:Global
          UP RUNNING NOARP  MTU:1280  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:208 (208.0 B)  TX bytes:0 (0.0 B)
```

```
root@re5:~# ip link set mtu 1280 dev 6rd
root@re5:~# ifconfig 6rd
6rd       Link encap:IPv6-in-IPv4
          inet6 addr: ::147.83.100.45/128 Scope:Compat
          inet6 addr: 2001:40b0:642d::1/32 Scope:Global
          UP RUNNING NOARP  MTU:1280  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:208 (208.0 B)
```

Desde el host **hc1** se realiza una traza al host **hd1** para descubrir el tamaño de MTU a lo largo del enlace:

```
root@hc1:~# tracepath6 -n 2001:40b0:103:2::d1
 1?: [LOCALHOST]                      0.085ms pmtu 1500
 1:  2001:db8:babe::61                                0.751ms
 1:  2001:db8:babe::61                                0.336ms
 2:  2001:40b0:642d::1                                0.488ms
 3:  2001:40b0:642d::1                               10.685ms pmtu 1280
 3:  2001:40b0:103::1                                 1.169ms
 4:  2001:40b0:103:2::d1                              1.178ms reached
     Resume: pmtu 1280 hops 4 back 61
```

El host IPv6 **hc1** envía una petición ICMPv6 con 1500 bytes de datos hacia el host IPv6 **hd1**.

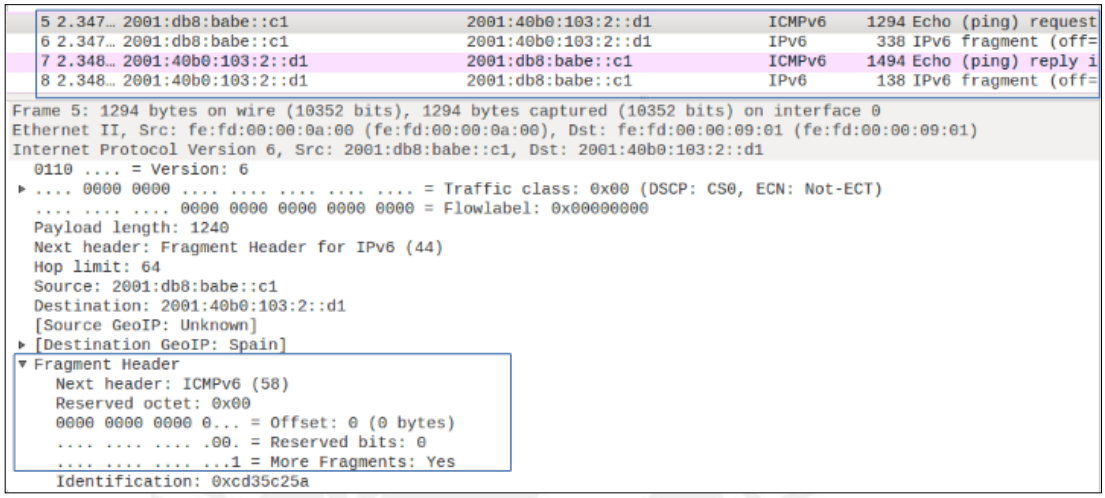
```
root@hc1:~# ping6 -c1 -s1500 2001:40b0:103:2::d1
PING 2001:40b0:103:2::d1(2001:40b0:103:2::d1) 1500 data bytes
From 2001:40b0:642d::1 icmp_seq=1 Packet too big: mtu=1280

--- 2001:40b0:103:2::d1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

```
Frame 5: 1294 bytes on wire (10352 bits), 1294 bytes captured (10352 bits) on interface 0
Ethernet II, Src: fe:fd:00:00:09:01 (fe:fd:00:00:09:01), Dst: fe:fd:00:00:0a:00 (fe:fd:00:00:0a:00)
Internet Protocol Version 6, Src: 2001:40b0:642d::1, Dst: 2001:db8:babe::c1
Internet Control Message Protocol v6
  Type: Packet Too Big (2)
  Code: 0
  Checksum: 0x5b82 [correct]
  MTU: 1280
▼ Internet Protocol Version 6, Src: 2001:db8:babe::c1, Dst: 2001:40b0:103:2::d1
    0110 .... = Version: 6
  ▶ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
  ▶ Payload length: 1456
```

En la captura Wireshark se muestra que el router **re5** envía el mensaje de error **Packet too big** al host **hc1**.

El host **hc1** fragmenta el siguiente paquete ICMPv6 y lo envía al host **hd1**:

```
root@hc1:~# ping6 -c1 -s1500 2001:40b0:103:2::d1
PING 2001:40b0:103:2::d1(2001:40b0:103:2::d1) 1500 data bytes
1508 bytes from 2001:40b0:103:2::d1: icmp_seq=1 ttl=61 time=1.19 ms

--- 2001:40b0:103:2::d1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.195/1.195/1.195/0.000 ms
```



La captura Wireshark muestra que, luego de fragmentar los paquetes ICMPv6, el flujo de tráfico se realiza sin problemas tanto en la petición como en la respuesta.

Por lo tanto, se demuestra que el que los routers relay de frontera **6RD** cuentan con un mecanismo para permitir el flujo de paquetes ICMPv6 fragmentados independientemente del MTU de la interfaz túnel 6RD.

CAPÍTULO III: MIGRACIÓN DE UNA RED WINDOWS A IPv6

En el presente capítulo se describe la migración de una red IPv4 a IPv6. La ventaja de este caso práctico es la posibilidad de demostrar la capacidad de acceso a recursos, nativos en IPv4, desde equipos con diferentes características en cuanto al hardware y software pero habilitados solo con IPv6. Además, se ponen a prueba los mecanismos de traducción descritos en el capítulo 2, y se verifica la diferencia operativa con los servicios usados en IPv4.

La migración es un proceso complejo. Se inicia con la implementación de un servidor con los servicios necesarios para la operación de los mecanismos DNS64+NAT64. Para esto, se comparte los recursos a nivel de capa I del modelo OSI con IPv4, con la finalidad de que no se altere la operatividad de la red IPv4 durante el proceso de transición. Luego, se separa el segmento de red que será de uso exclusivo de los equipos IPv6.

La migración al protocolo de red IPv6 necesita una serie de requisitos que deben cumplirse de forma transparente para los equipos terminales:

- Acceso de todos los equipos a la red local e Internet.
- Acceso a los servicios en IPv6 e IPv4
- Servicio de gestión del direccionamiento

3.1. Escenario

El método de traducción NAT64 facilita el acceso de una red IPv6 a equipos y recursos de Internet IPv4 (ya sea en redes de tránsito, acceso o de borde) mediante la traducción de la cabecera IP y la dirección entre ambos protocolos.

Los métodos de traducción no representan una solución a largo plazo; se trata de una estrategia de coexistencia a mediano plazo que puede facilitar un programa de transición a largo plazo por parte de empresas e ISPs.

En este escenario se implementará el método de traducción NAT64 stateful usando el software Jool.

3.1.1. Topología IPv4

La figura 24 muestra el escenario con la actual configuración en IPv4.El segmento utiliza la red privada 10.10.11.0/24 para el enrutamiento interno y cuenta con los siguientes nodos, tabla Nº 22:

Tabla Nº 22 – Servidores y equipo de comunicaciones

Equipo	Sistema Operativo	Descripción	Dirección IP
Intel Pentitum D, 2 GB RAM, Pfsense 3.2.1	FreeBSD (10.3)	Router Proxy Firefall	10.10.11.5
Intel Core 2, 4 GB RAM,	Ubuntu 12.10 LTS	Servidor de backup NFS (imágenes)	10.10.11.9
Intel Core 2, 4 GB	Ubuntu 12.10 LTS	Servidor de archivos Servidor Web	10.10.11.11

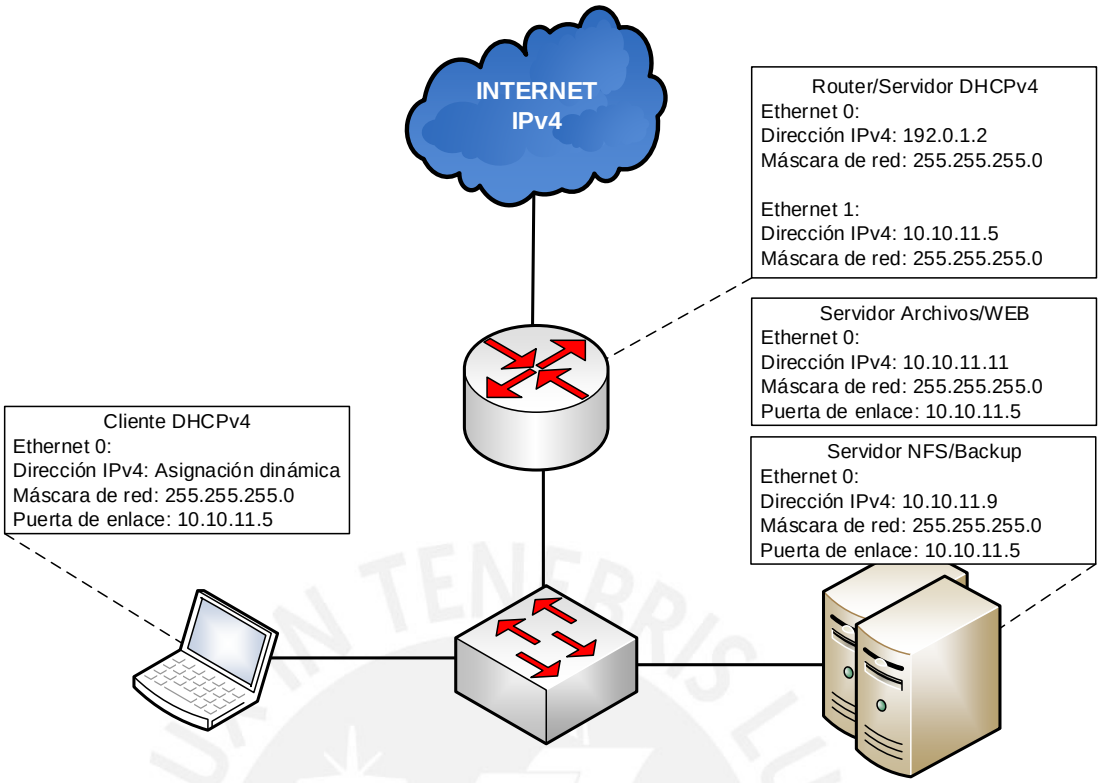


Fig. Nº 24 – Topología IPv4

Los equipos que se encuentran en la red de transito; se conectan a través de puntos de acceso fijo e inalámbrico para obtener los siguientes servicios:

- Acceso a la intranet local para el registro de la reparación
- Acceso a la intranet para ejecutar pruebas de benchmarking
- Acceso al servidor de archivos NFS para el proceso de restauración del sistema operativo.
- Acceso a Internet para actualizar el sistema operativo y los controladores de dispositivos.

El diagrama de flujo de la figura Nº 25 describe la operación de la topología IPv4. Los clientes tienen una dirección IPv4 no enrutable asignada dinámicamente por el servidor DHCPv4 dentro de la red local. Los clientes utilizan NAT44, lo que significa que acceden a Internet IPv4 a través de una única puerta de enlace.

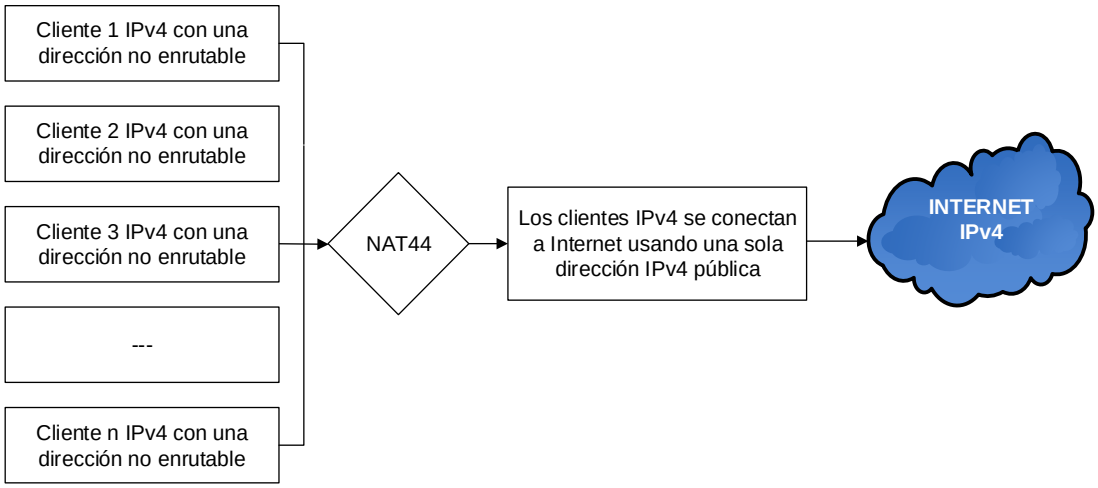


Fig. Nº 25 – Diagrama de flujo de la red IPv4

Las características de los equipos que se conectan son (tabla Nº 23) :

Tabla Nº 23 – Características de los equipos

Equipo	Procesador	Sistema Operativo	Tipo de acceso
PC de escritorio Laptop Tablet	Intel, AMD, ARM	Windows 7 Windows 8 Windows 8.1 Windows 10 Linux Ubuntu Desktop Linux Suse Desktop Android	Fijo, Inalámbrico

El sistema operativo predominante se muestra en la tabla Nº 24:

Tabla Nº 24 - Sistema Operativo de mayor uso [54]

Sistema Operativo	% de Ingresos
Windows 7	5
Windows 8	15
Windows 8.1	30
Windows 10	40
Linux Ubuntu Desktop	5
Linux Suse Desktop	3
Android	2
Total	100

3.1.2. Migración a IPv6

Teniendo como base la infraestructura de red y los servicios en IPv4 se procede a migrar a IPv6. Lo primordial de este proceso es no alterar la configuración de fábrica de los equipos. La figura Nº 26 muestra la topología IPv4/IPv6.

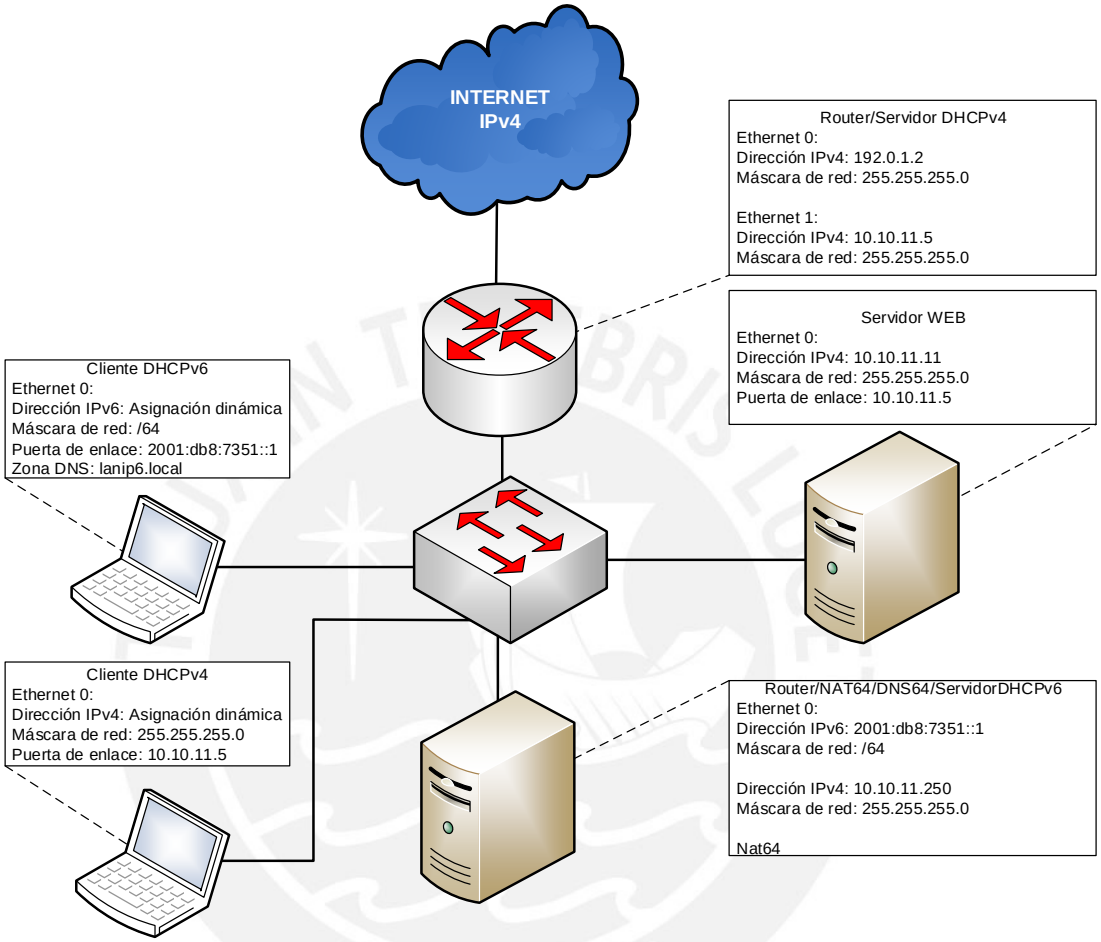


Fig. Nº 26 – Topología IPv6/IPv4

El servidor encargado de conectar la red IPv6 tiene las siguientes características (tabla 25):

Tabla Nº 25 - Características del servidor Linux NAT64/DNS64

Equipo	Sistema Operativo	Descripción	Dirección IP
Intel Core2 Duo, 2 GB RAM	Ubuntu Server 14.04 LTS	-Router IPv6 -Servidor DHCPv6 Sateful -NAT64 -DNS64	10.10.11.250 2001:db8:7351::1

Para esta prueba piloto, se dispone de un segmento de conexión IPv4/IPv6. En donde los hosts configurados en IPv6 comparten la misma red física de IPv4. Además, se desactiva el servicio DHCPv4; servidores e impresoras contarán con una IPv4 fija con la finalidad de comprobar que no se altere el funcionamiento de los servicios.
La siguiente figura muestra el diagrama de flujo de la operación NAT64:

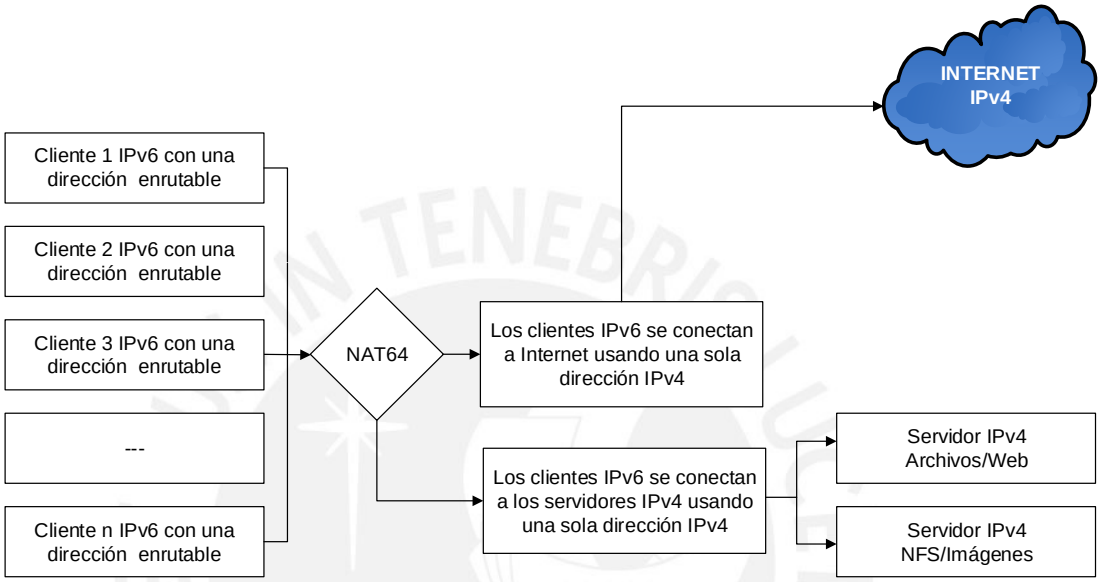


Fig. Nº 27 – Diagrama de flujo de NAT64

En el punto 1.4.4 se describe los métodos de direccionamiento dinámico. En esta migración se utiliza la configuración DHCP stateful, a través de la combinación de los programas RADVD y ISC-DHCPv6, para tener un control de las direcciones IPv6 que se asignan dinámicamente y para el anuncio del servidor DNS64 a la red IPv6.

3.1.3. Despliegue de Direcciones IPv6

El documento Manual de Políticas de LACNIC define la política de direcciones en América Latina. LACNIC realizará una distribución de un prefijo /32 al recibir una solicitud de direccionamiento IPv6 por parte de un ISP que cuenten con distribuciones previas de IPv4. [55]

Por otro lado, LACNIC realizará distribuciones experimentales del prefijo 2001:db8::/32 con el objetivo de fomentar la investigación y el desarrollo en la región. Estas distribuciones abarcarán todos los recursos que LACNIC posee (direcciones IPv4, IPv6, ASN).[55]

Por lo tanto, para la implementación de este caso práctico de migración en Windows, el prefijo a usarse es 2001:db8::/32.

3.1.4. Configuración del Servidor Linux para IPv6

En primer lugar se configura la dirección global IPv6 en la interface del servidor Linux. Esto se logra agregando los siguientes parámetros en el archivo */etc/network/interfaces* :

```
iface eth0 inet6 static
    address 2001:db8:7351::1
    netmask 64
```

3.1.4.1. Servicio DHCPv6 Stateful

Un router IPv6 basado en Linux, a diferencia de IPv4, anuncia su presencia en la red a través de mensajes ICMPv6 RA (Router Advertisement), enviados por el servicio RADVD. Sin embargo, el servicio RA solo puede proveer a los hosts un prefijo de red, un Gateway y la dirección de un servidor DNS en modo Stateless. Esta asignación dinámica de direcciones es compatible con clientes Linux. Para los clientes con Windows y Apple es necesario que el servicio RADVD opere junto con el servicio DHCPv6 para el despliegue de direcciones en modo Stateful.

Así mismo, el RFC 3736 “Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6” define un método mixto de configuración, en el que el cliente obtiene la dirección del gateway por SLAAC y utiliza el servicio DHCPv6 para obtener el resto de parámetros de red [56] .

En consecuencia, la asignación dinámica del rango de direcciones IPv6, el anuncio del servidor DNS IPv6 y del dominio IPv6 está a cargo del programa ISC-DHCP-SERVER. El programa RADVD es el encargado de anunciar la dirección del gateway IPv6 y de permitir las opciones de configuración dinámica de DHCPv6.

Configuración de RADVD

El archivo de configuración es */etc/radvd.conf* , y se modifica con los siguientes parámetros:

```
iface eth0
{
    AdvManagedFlag on;
    AdvSendAdvert on;
    AdvDefaultLifetime 1800;
    AdvOtherConfigFlag on;
    UnicastOnly on;
    RDNSS 2001:db8:7351::1 {
    };
}
```

- El prefijo IPv6 se anuncia a través de la interfaz eth0 del servidor Linux.
- La opción **AdvManagedFlag** anuncia al host final que se le otorgará una dirección IPv6 con estado (stateful) a través de DHCPv6.
- La opción **AdvSendAdvert** indica que la interfaz eth0 enviará periódicamente mensajes RA y responderá mensajes de solicitud de router.
- La opción **AdvDefaultLifetime** indica el tiempo de presencia del router como puerta de enlace (Gateway) por defecto.
- La opción **AdvOtherConfigFlag** se usa para informar al host final que configuración adicional como direcciones de servidor DNS, SIP o NTP será entregada a través de DHCP.
- La opción **UnicastOnly** evita que se envíe “anuncios no solicitados”, solo se responderá a través de mensajes unicast al nodo que lo solicite.
- La opción **RDNSS** anuncia a los clientes Linux la dirección IPv6 del servidor DNS.

Configuración de DHCPv6

Para complementar el despliegue de direcciones IPv6 en modo stateful es necesario la instalación del programa ISC-DHCPv6-Server. En Linux el comando es:

```
sudo apt-get install isc-dhcp-server
```

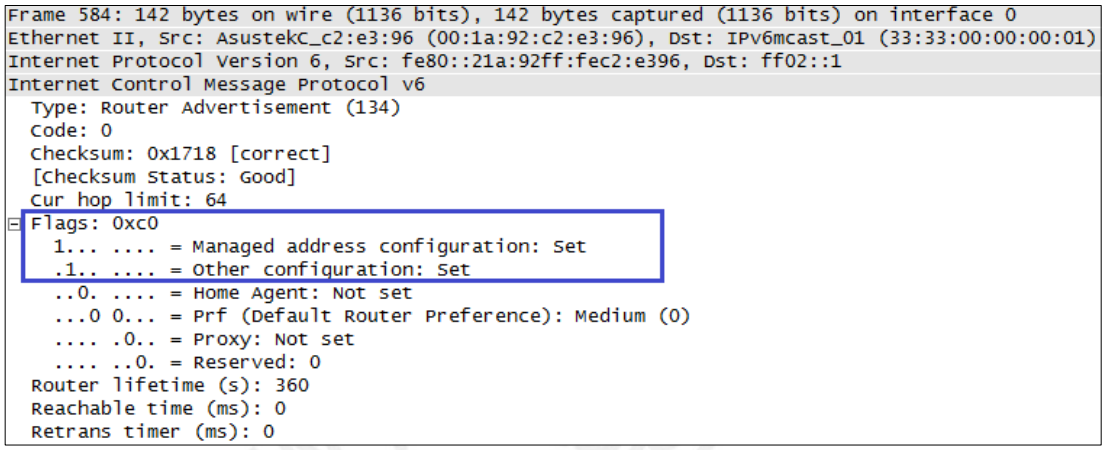
Luego, se crea el archivo */etc/dhcpd/dhcpd6.conf*, y se configura con los siguientes parámetros:

```
ddns-update-style none;
ddns-updates on;

# Option definitions common to all supported networks...
default-lease-time 300; # 1 hour
max-lease-time 300; # 1 hour

# This DHCP server is the official DHCP server for the local network
authoritative;
option domain-name "lanip6.local";
option dhcp6.name-servers 2001:db8:7351::1;
option dhcp6.domain-search "lanip6.local";

log-facility local7;

zone lanip6.local. {
    primary 127.0.0.1;
}
}
# Subnet declaration
subnet6 2001:db8:7351::/64 {
    range6 2001:db8:7351::130 2001:db8:7351::150;
}
```

Dentro de las opciones dhcpv6 configuradas se tiene lo siguiente:[57]

- La opción **ddns_updates (Dynamic DNS)** permite actualizar automáticamente los registros DNS.
- La opción **default-lease-time**; el tiempo (segundos) de vida por defecto de asignación de una dirección IPv6 a un cliente.
- La opción **max-lease-time**; el tiempo (segundos) de vida máximo de asignación de una dirección IPv6 a un cliente.
- La opción **authoritative**; indica que el servidor DHCP define la correcta configuración de la red y envía mensajes DHCPNAK para corregir y completar parámetros de red incompletos de los clientes.
- La opción **domain-name**; indica el dominio **lanip6.local** especificado en la configuración del servicio DNS.
- La opción **dhcp6.name-servers**; especifica la dirección IPv6 del servidor DNS.
- La opción **dhcp6.domain-search**; especifica el dominio **lanip6.local** en el que se hará la búsqueda de hosts específicos.
- La opción **log-facility**; permite al servidor DHCPv6 registrarse completamente una vez que el archivo dhcpd6.conf haya sido leído.
- La opción **zone**; establece la zona lanip6.local y su identificado

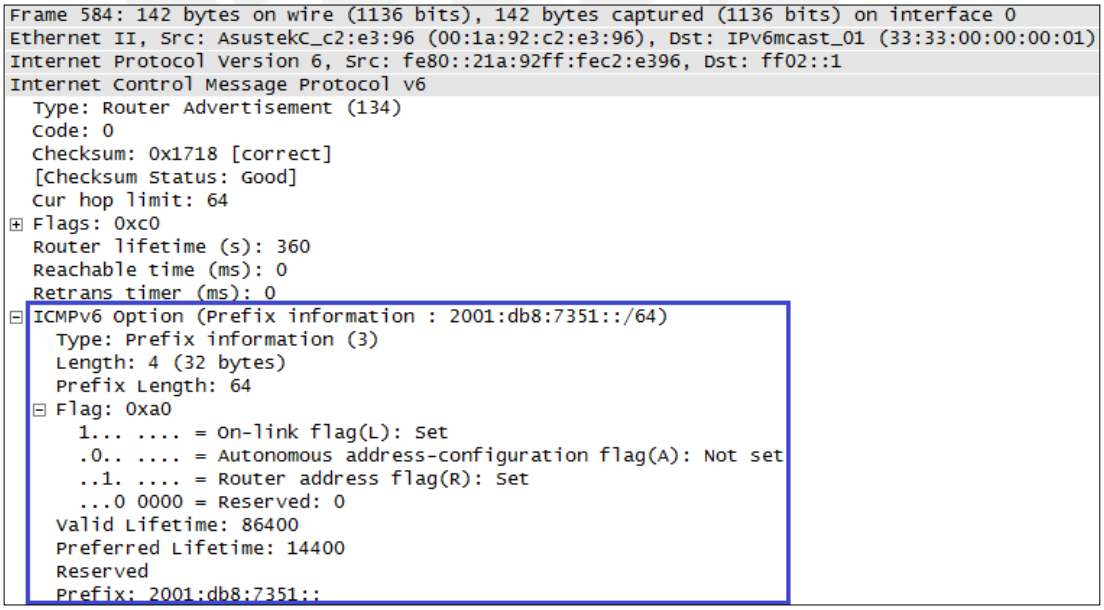
La siguiente captura Wireshark de la trama 584 muestra el envío de un mensaje ICMPv6 desde el servidor, en donde se aprecia la configuración SLAAC/DHCPv6 establecida con los parámetros ejecutados anteriormente:



Como se aprecia en la captura anterior, en el mensaje se tiene los siguientes datos:

- Typo: **Router Advertisement (134)**
- Flags: 0xc0, con los parámetros **Managed address configuration** y **Other configuration** activados.

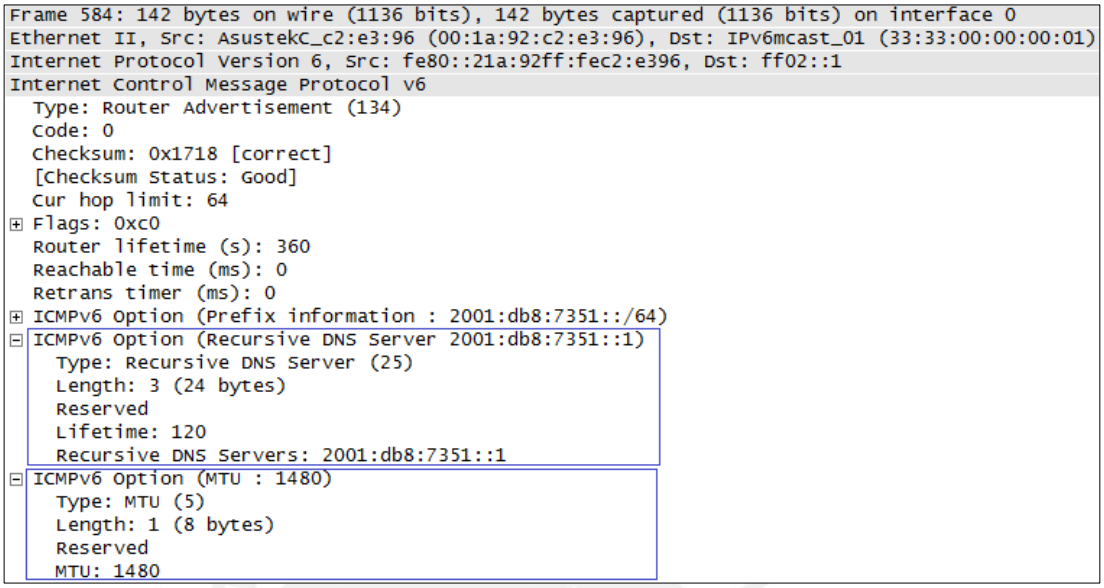
La siguiente captura Wireshark es parte de la trama 584, muestra información del prefijo:



Como se aprecia en la captura anterior, en el mensaje se tiene los siguientes datos:

- Typo: **Prefix information (3)**
- Flags: 0xa0, con los parámetros **On-Link flag (L)** y **Router address** activados.

La siguiente captura Wireshark es parte de la trama 584, muestra información del servidor DNS y del MTU:



3.1.4.2. Servicio DNS

El servicio DNS está gestionado por el programa BIND versión 9.8.

Configuración de DNS

En el archivo */etc/bind/named.conf.options* se agrega el siguiente parámetro:

```
options {
[...]
    dns64 2001:db8:7351:ffff::/96 {};
};
```

Se crea el archivo */etc/bind/lanip6.local* y se agregan los registros A y AAAA de los servidores DNS64, Web e Image:

```
$TTL    604800
@       IN      SOA     ns1.local. root.myhost.lanip6.local. (
                              61         ; Serial
                          604800         ; Refresh
                           86400         ; Retry
                         2419200         ; Expire
                          604800 )       ; Negative Cache TTL
;
@       IN      NS      ns1.local.

dns64   IN      A       10.10.11.250
        IN      AAAA    2001:db8:7351::1

web     IN      A       10.10.11.11
        IN      AAAA    2001:db8:7351:ffff::a0a:b0b

image   IN      A       10.10.11.9
        IN      AAAA    2001:db8:7351:ffff::a0a:b09
```

La zona **lanip6.local** se crea al modificar el archivo */etc/bind/named.conf.local* con los siguientes parámetros:

```
zone "lanip6.local"  {
        type master;
        file "/etc/bind/lanip6.local";
};
```

Configuración del firewall con iptables[22]

Para permitir las consultas IPv6 hacia el servidor de nombres de dominio es necesario habilitar el puerto 53 en TCP y UDP en el servidor NAT64/DNS64:

```
iptables -A INPUT -p udp -m udp --dport 53 -j ACCEPT
iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 53 -j ACCEPT
iptables -A INPUT -m state --state NEW -m udp -p udp --dport 53 -j ACCEPT
```

3.1.5. Mecanismo de Traducción NAT64 Stateful

Una de las ventajas de NAT64 stateful es la capacidad de usar una sola dirección IPv4 para traducir a un grupo de direcciones IPv6. En este proceso de migración, la cantidad de clientes IPv6 es variable. Según el RFC 6146 “Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers” [39], una dirección IPv4 es el requerimiento mínimo para la operación de traducción.

Jool es un programa de código abierto que implementa NAT64 con estado (stateful); entre sus características, permite mapear una sola dirección IPv4 a un grupo de direcciones IPv6. En este escenario, Jool utiliza el rango de puertos 61001 – 65535

[22] Herramienta de cortafuegos de Linux que permite filtrar paquetes

para las conexiones UDP, TCP e ICMP de la dirección IPv4 (10.10.11.250) de la interfaz eth0, en el servidor NAT64/DNS64.[58]

3.1.5.1. Configuración del paquete Jool

Preparación del kernel de Linux

Para instalar el programa Jool revisión 3.5.2, el servidor Linux debe contar con la versión 3.2 del kernel, o superior [58]. En la consola del servidor se ejecuta el siguiente comando para comprobar la versión:

```
uname -r
```

```
hp@hp-RQ425AA-AKV-m7730la:~$ uname -r
4.2.0-42-generic
```

En el caso de nuestro servidor, la versión (4.2.0-42) cumple con el requerimiento.

Por otro lado, se instala el conjunto de paquetes esenciales para la compilación del paquete Jool:

```
sudo apt-get install build-essential
```

Se instala el gestor del kernel DKMS, con la finalidad de facilitar la compilación e instalación del programa Jool:

```
sudo apt-get install dkms
```

Compilación e Instalación

Se descarga el paquete Jool, con el siguiente comando:

```
wget https://github.com/NICMx/releases/raw/master/Jool/Jool-3.5.1.zip
```

Los siguientes comandos permiten instalar el paquete Jool:

```
unzip Jool-3.5.1.zip
```

```
sudo dkms Jool-3.5.1
```

Configuración

Se habilita el enrutamiento IPv4 e IPv6:

```
net.ipv6.conf.default.forwarding=1

net.ipv4.conf.default.forwarding=1
```

Se especifica el pool IPv6 para la traducción NAT64 stateful:

```
modprobe jool pool6=2001:db8:7351:ffff::/96
```

3.1.6. Acceso Internet IPv6

En figura Nº 28 muestra la topología IPv4/IPv6 que incluye el acceso a Internet IPv6 usando el mecanismo NAT66.

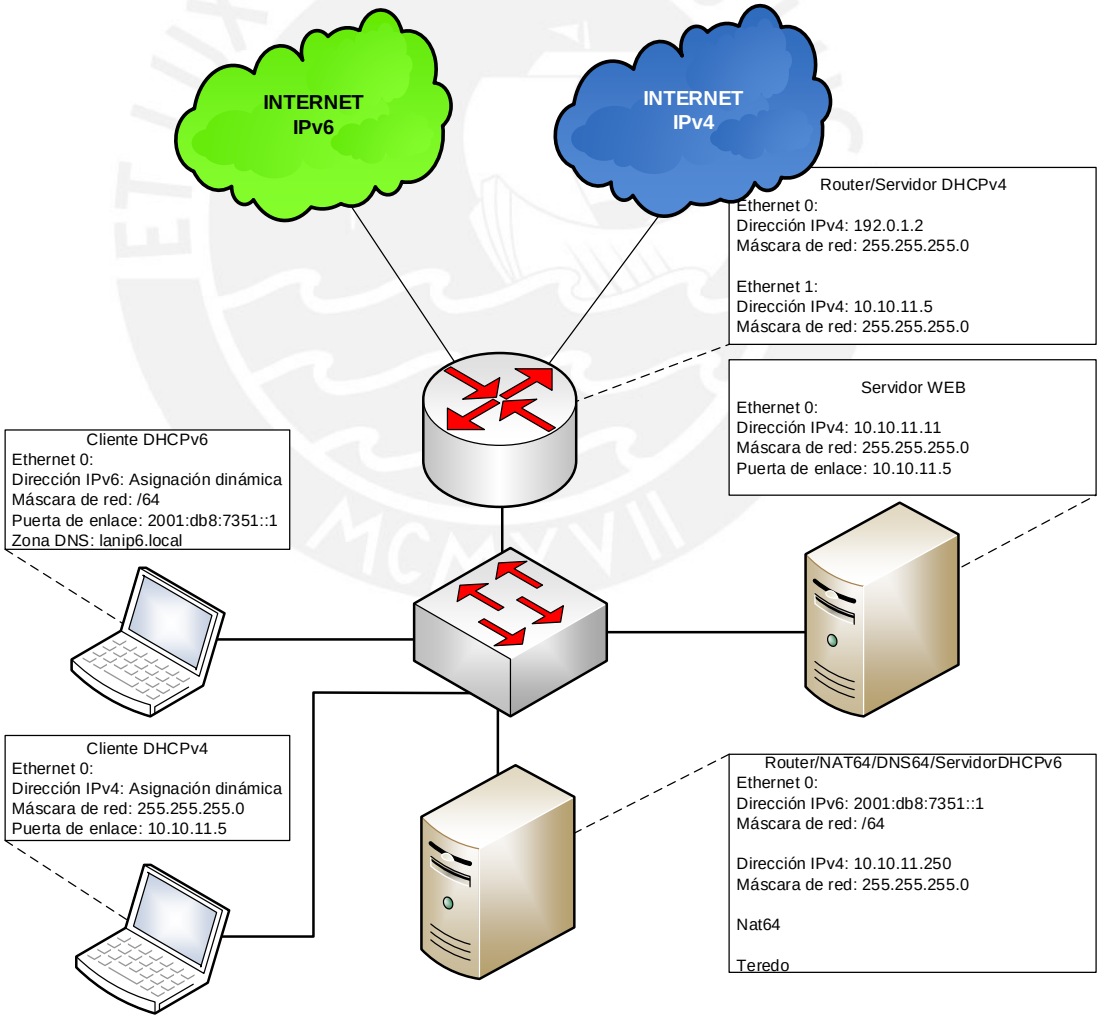


Fig. Nº 28 – Acceso a Internet IPv4 e Internet IPv6

En el punto 1.4.2.6 del Capítulo I se describe el mecanismo de traducción NAT66. En el diagrama de la figura Nº 29 de flujo que se muestra a continuación muestra la operación de la red NAT64 + NAT66:

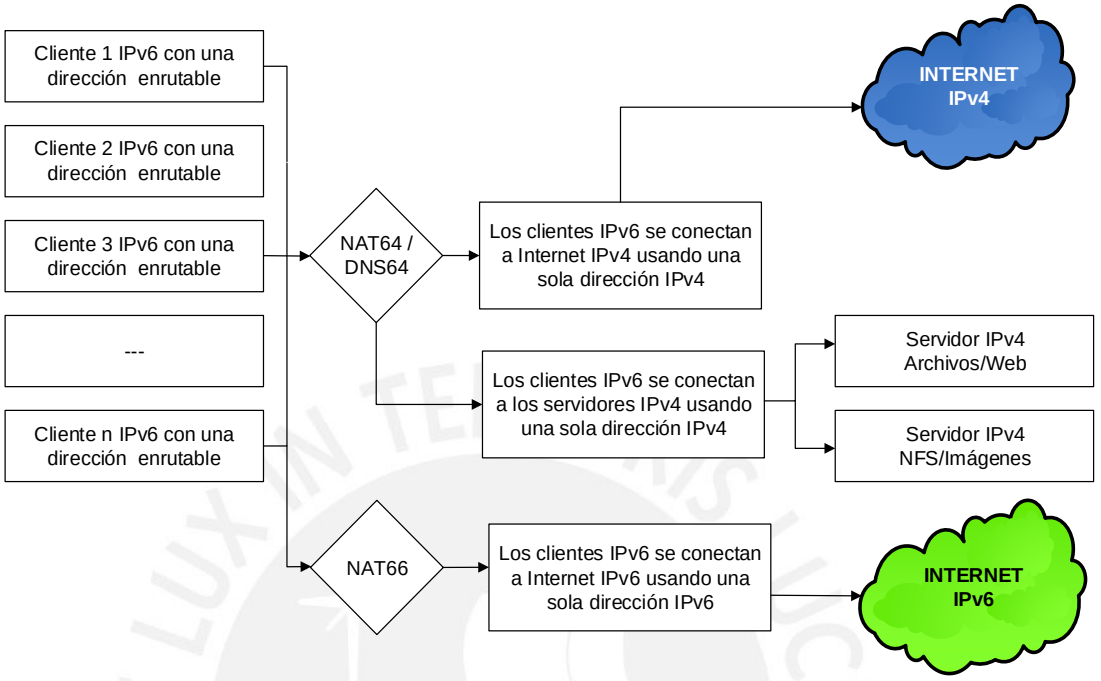


Fig. Nº 29 – Diagrama de flujo NAT64 y NAT66

Para encaminar el tráfico de la red IPv6 hacia Internet IPv6 se agrega el siguiente parámetro al archivo */etc/rc.local* del servidor NAT64/DNS64:

```
ip6tables -t nat -A POSTROUTING -o teredo -j MASQUERADE
```

De esta forma, todo el tráfico destinado a Internet IPv6 será gestionado por la interfaz *teredo*. Al ejecutar el comando route -n -6 en la consola del servidor se tiene la tabla de enrutamiento IPv6:

```
Kernel IPv6 routing table
Destination                    Next Hop                   Flag Met Ref Use If
2001::/32                      ::                         U    256 0     0 teredo
fe80::/64                      ::                         U    256 0     0 wlp12s0
fe80::/64                      ::                         U    256 0     0 teredo
::/0                           ::                         U    1029 2    73 teredo
::/0                           ::                         !n   -1  1   434 lo
::1/128                        ::                         Un   0   3     7 lo
2001::/128                     ::                         Un   0   1     0 lo
2001:0:53aa:64c:24a2:c5a5:4cf8:8935/128 ::                      Un   0   1     0 lo
fe80::/128                     ::                         Un   0   1     0 lo
fe80::/128                     ::                         Un   0   1     0 lo
fe80::ffff:ffff:ffff/128       ::                         Un   0   1     0 lo
fe80::5d9d:67db:53d3:e591/128  ::                         Un   0   1     0 lo
ff00::/8                       ::                         U    256 2    34 wlp12s0
ff00::/8                       ::                         U    256 0     0 teredo
```

Además, para resolver los nombres de los sitios Web a las direcciones IPv6 en Internet IPv6 se modifica el archivo */etc/bind/named.conf.options* :

```
forwarders {
 2001:4860:4860::6464;
};
```

El parámetro anterior permite redirigir las peticiones DNS de los clientes IPv6 hacia uno de los servidores DNS IPv6 públicos de Google.

De esta forma, los clientes IPv6 tendrán acceso a Internet IPv6 a través del mecanismo NAT66 gestionado por el programa Teredo en el servidor Linux.

3.1.7. Configuración de los Clientes

En el lado de los clientes, para adquirir una dirección IPv6 desde el servidor Linux, se activa el modo automático en la interfaz de red de los equipos con Windows, versiones 7/8/8.1/10

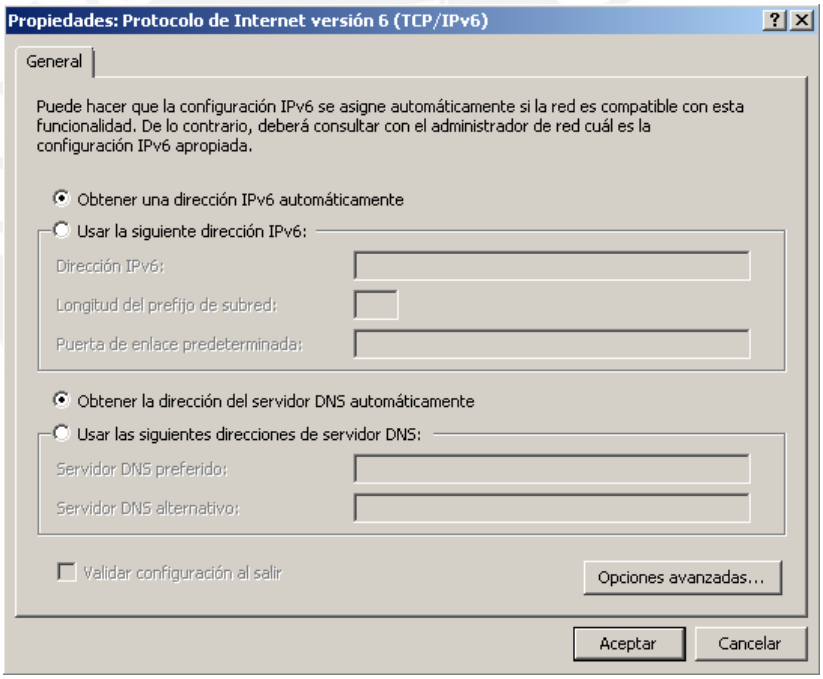


Fig. Nº 30 – Cliente IPv6 en Windows

La figura Nº 31 muestra la configuración de la interfaz con los parámetros obtenidos a través de SLAAC y DHCPv6.

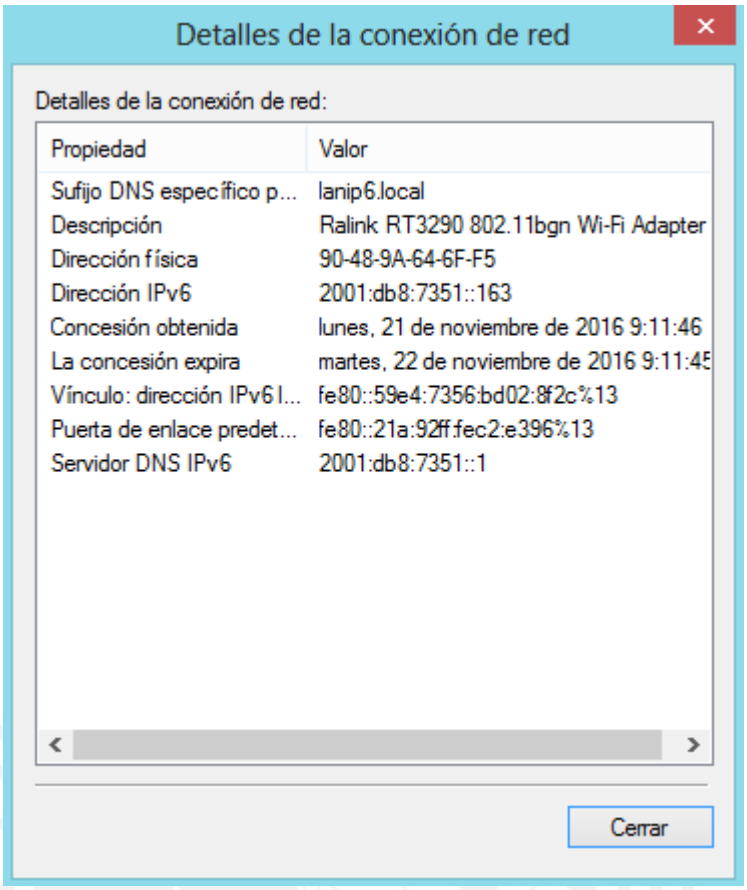


Fig. Nº 31 - Conexión del Cliente IPv6 en Windows

3.2. Pruebas de Operación NAT66, NAT64 y DNS64

3.2.1. Prueba de Conectividad ICMPv6

Desde un host IPv6 con sistema operativo Windows se envía paquetes ICMPv6 a los dominios IPv4 e IPv6 de Google. Los siguientes escenarios ponen a prueba la capacidad de traducción de direcciones del servidor NAT64 y el acceso a Internet IPv6 a través del mecanismo NAT66. Además, la operatividad del servidor DNS64.

Desde un equipo con IPv6 se envía un paquete ICMPv6 al sitio Web IPv4 de Google:

```
C:\Users\Admin>ping -6 -n 1 ipv4.google.com

Haciendo ping a ipv4.l.google.com [2001:db8:7351:ffff::acd9:126e] con 32 bytes de datos:
Respuesta desde 2001:db8:7351:ffff::acd9:126e: tiempo=815ms

Estadísticas de ping para 2001:db8:7351:ffff::acd9:126e:
    Paquetes: enviados = 1, recibidos = 1, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 815ms, Máximo = 815ms, Media = 815ms
```

La IPv4 de la dirección Web *ipv4.google.com* es 172.217.18.110. El servidor Linux entiende que se trata de una IPv4 y utiliza el mecanismo DNS64 para asignar el prefijo 2001:db8:7351:ffff::/96. De esta forma se crea un registro AAAA.

Si se envía un paquete ICMPv6 al sitio Web IPv6 de Google:

```
C:\Users\Admin>ping -6 -n 1 ipv6.google.com

Haciendo ping a ipv6.l.google.com [2a00:1450:400a:804::200e] con 32 bytes de datos:
Respuesta desde 2a00:1450:400a:804::200e: tiempo=360ms

Estadísticas de ping para 2a00:1450:400a:804::200e:
    Paquetes: enviados = 1, recibidos = 1, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 360ms, Máximo = 360ms, Media = 360ms
```

La IPv6 de la dirección Web *ipv6.google.com* es 2a00:1450:400a:804::200e. El servidor Linux entiende que se trata de una dirección IPv6 y reenvía la petición ICMPv6 a la interfaz teredo del servidor. La dirección IPv6 de Google se obtiene al realizar la consulta al DNSv6 público de Google.

En ese sentido, para comprobar el funcionamiento del servidor DNS64 local y el DNS público de Google se ejecuta el comando *nslookup* desde la consola CMD de Windows.

Los siguientes dominios no cuentan con IPv6, el servicio DNS64 crea el registro AAAA:

```
C:\Users\Admin>nslookup pucp.edu.pe
Servidor:  UnKnown
Address:  2001:db8:7351::1

Respuesta no autoritativa:
Nombre:  pucp.edu.pe
Addresses:  2001:db8:7351:ffff::c810:1c2
          200.16.1.194
```

```
C:\Users\Admin>nslookup sunedu.gob.pe
Servidor:  UnKnown
Address:  2001:db8:7351::1

Respuesta no autoritativa:
Nombre:  sunedu.gob.pe
Addresses:  2001:db8:7351:ffff::c830:4d83
          200.48.77.131
```

```
C:\Users\Admin>nslookup hp.com
Servidor:  UnKnown
Address:  2001:db8:7351::1

Respuesta no autoritativa:
Nombre:  hp.com
Addresses:  2001:db8:7351:ffff::fc9:e10a
          2001:db8:7351:ffff::fd8:f112
          2001:db8:7351:ffff::fd9:e8f5
          2001:db8:7351:ffff::ff0:3cee
          15.240.60.238
          15.216.241.18
          15.217.232.245
          15.201.225.10
```

Los siguientes dominios tienen IPv6 nativa, la consulta DNS se redirige al servidor DNSv6 de Google para obtener la dirección IPv6:

```
C:\Users\Admin>nslookup facebook.com
Servidor:  UnKnown
Address:  2001:db8:7351::1

Respuesta no autoritativa:
Nombre:  facebook.com
Addresses:  2a03:2880:f11c:8083:face:b00c:0:25de
          31.13.93.36
```

```
C:\Users\Admin>nslookup ieeexplore.ieee.org
Servidor:  UnKnown
Address:  2001:db8:7351::1

Respuesta no autoritativa:
Nombre:  ieeexplore.ieee.org
Addresses:  2620:104:c000:8193::112
          140.98.193.112
```

```
C:\Users\Admin>nslookup google.com
Servidor:  UnKnown
Address:  2001:db8:7351::1

Respuesta no autoritativa:
Nombre:  google.com
Addresses:  2404:6800:400a:808::200e
          172.217.26.110
```

3.2.2. Evaluación del Rendimiento de la red con Iperf3

Para esta evaluación, se toma como referencia las pruebas realizadas en el documento “Study of packet level UDP performance of NAT44, NAT64 and IPv6 using iperf in the context of IPv6 migration” [56]. En la topología mostrada en la figura Nº 28 se tienen las siguientes configuraciones experimentales de red: NAT44, NAT64 y

NAT66. Para evaluar el rendimiento de la red se utiliza el paquete Iperf [60]. Iperf puede utilizarse con TCP y UDP.

En esta parte de la evaluación realiza el envío de paquetes a un servidor remoto a través de UDP con la finalidad de comparar los siguientes parámetros de red:

- Tiempo (segundos)
- Ancho de banda (mbits/seg)
- Variación en la latencia, Jitter (ms)
- Perdida de paquetes (%)

Servidor IPERF versión 3 público

Iperf se ejecuta en modo cliente-servidor. Para el entorno de pruebas, existen servidores Iperf disponibles a través de diferentes proveedores alrededor del mundo [61].

Para la región Américas, Tabla Nº 26, se tienen los siguientes servidores:

Tabla Nº 26 - Servidores Iperf3 disponibles en la región Américas [61]

Americas							
iPerf3 server	Characteristics	Localization	Datacenter	Hosting	Speed	Port	IP version
iperf.scottlinux.com		USA California	Hurricane Fremont 2	Scott	1 Gbit/s	5201 TCP/UDP	IPv4 and IPv6
iperf.he.net		USA California	Hurricane Fremont 1	HURRICANE ELECTRIC INTERNET SERVICES		5201 TCP/UDP	IPv4 and IPv6

Para verificar la disponibilidad de cada uno de los servidores Iperf se ejecuta el siguiente comando desde la consola CMD de Windows:

```
iperf3.exe -c <servidor> -u
```

Servidor: **iperf.scottlinux.com:**

```
Client connecting to iperf.scottlinux.com, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 63.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.4 port 60979 connected with 45.33.39.39 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] WARNING: did not receive ack of last datagram after 10 tries.
```

Servidor: **iperf.he.net:**

```
Client connecting to iperf.he.net, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 63.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.4 port 54555 connected with 216.218.227.10 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.6 sec  1.09 MBytes   869 Kbits/sec   4.071 ms  113/  893 (13%)
```

Solo el servidor **iperf.he.net** presenta un reporte final con la información requerida para evaluar el rendimiento de la red:

- **10.6 sec**: Es el tiempo de duración de la prueba, en segundos.
- **1.09 MBytes:** Es el tamaño de información enviada, en Mbytes.
- **869 kbits/s:** Es el ancho de banda utilizado, en Kbits/s
- **4.071 ms:** Es el jitter, en milisegundos.
- **13%:** Es el porcentaje de paquetes perdidos

Determinación de la carga útil (payload)

Para evitar que se produzcan incrementos en la tara del CPU y la memoria del servidor remoto al momento de reensamblar los fragmentos de los paquetes enviados, es necesario ajustar el tamaño del payload de acuerdo al tipo de enlace de transmisión.

- Se habilitan los servicios DHCP y DHCPv6+SLAAC para la asignación de las direcciones IPv4 e IPv6.
- Se envían paquetes por un tiempo de 10 segundos al servidor remoto **iperf.he.net.**
- Se utiliza el ancho de banda en modo UDP de 1 Mbits/s.

Desde la consola CMD de Windows se ejecuta el siguiente comando:

En NAT44:

```
iperf3.exe -c iperf.he.net -u -4
```

```
Connecting to host iperf.he.net, port 5201
[  4] local 10.10.11.21 port 64059 connected to 216.218.227.10 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec   152 KBytes  1.24 Mbits/sec  19
[  4]   1.00-2.00   sec   144 KBytes  1.18 Mbits/sec  18
[  4]   2.00-3.00   sec   136 KBytes  1.11 Mbits/sec  17
[  4]   3.00-4.00   sec   136 KBytes  1.11 Mbits/sec  17
[  4]   4.00-5.00   sec   136 KBytes  1.11 Mbits/sec  17
[  4]   5.00-6.00   sec   136 KBytes  1.11 Mbits/sec  17
[  4]   6.00-7.00   sec   144 KBytes  1.18 Mbits/sec  18
[  4]   7.00-8.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   8.00-9.00   sec   144 KBytes  1.18 Mbits/sec  18
[  4]   9.00-10.00  sec   128 KBytes  1.05 Mbits/sec  16
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.00  sec  1.35 MBytes  1.13 Mbits/sec  16.677 ms  0/171 (0%)
[  4] Sent 171 datagrams

iperf Done.
```

Captura Wireshark:

Source	Destination	Protocol	Length	Info
10.10.11.21	216.218.227.10	UDP	1514	54040→5201 Len=8192
10.10.11.21	216.218.227.10	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=1480, ID=2b8c)
10.10.11.21	216.218.227.10	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=2960, ID=2b8c)
10.10.11.21	216.218.227.10	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=4440, ID=2b8c)
10.10.11.21	216.218.227.10	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=5920, ID=2b8c)
10.10.11.21	216.218.227.10	IPv4	834	Fragmented IP protocol (proto=UDP 17, off=7400, ID=2b8c)

En NAT64:

```
iperf3.exe -c 2001:db8:7351:ffff::216.218.227.10 -u
```

```
Connecting to host 2001:db8:7351:ffff::216.218.227.10, port 5201
[  4] local 2001:db8:7351::165 port 53265 connected to 2001:db8:7351:ffff::d8da:e30a port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec   136 KBytes  1.11 Mbits/sec  17
[  4]   1.00-2.00   sec   144 KBytes  1.18 Mbits/sec  18
[  4]   2.00-3.00   sec   120 KBytes   983 Kbits/sec  15
[  4]   3.00-4.00   sec   144 KBytes  1.18 Mbits/sec  18
[  4]   4.00-5.00   sec   136 KBytes  1.12 Mbits/sec  17
[  4]   5.00-6.00   sec   144 KBytes  1.18 Mbits/sec  18
[  4]   6.00-7.00   sec   120 KBytes   983 Kbits/sec  15
[  4]   7.00-8.00   sec   144 KBytes  1.18 Mbits/sec  18
[  4]   8.00-9.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   9.00-10.00  sec   136 KBytes  1.11 Mbits/sec  17
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.00  sec  1.32 MBytes  1.11 Mbits/sec  7.718 ms  0/168 (0%)
[  4] Sent 168 datagrams

iperf Done.
```

Captura Wireshark:

Source	Destination	Protocol	Length	Info
2001:db8:7351::165	2001:db8:7351:ffff::d8da:e30a	UDP	1494	53265→5201 Len=8192
2001:db8:7351::165	2001:db8:7351:ffff::d8da:e30a	IPv6	1494	IPv6 fragment (off=1432 more=y
2001:db8:7351::165	2001:db8:7351:ffff::d8da:e30a	IPv6	1494	IPv6 fragment (off=2864 more=y
2001:db8:7351::165	2001:db8:7351:ffff::d8da:e30a	IPv6	1494	IPv6 fragment (off=4296 more=y
2001:db8:7351::165	2001:db8:7351:ffff::d8da:e30a	IPv6	1494	IPv6 fragment (off=5728 more=y
2001:db8:7351::165	2001:db8:7351:ffff::d8da:e30a	IPv6	1102	IPv6 fragment (off=7160 more=n

En NAT66:

```
iperf3.exe -c iperf.he.net -u -6
```

```
Connecting to host iperf.he.net, port 5201
[  4] local 2001:db8:7351::165 port 63538 connected to 2001:470:0:238::2 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec   152 KBytes  1.24 Mbits/sec  19
[  4]   1.00-2.00   sec   144 KBytes  1.18 Mbits/sec  18
[  4]   2.00-3.00   sec   136 KBytes  1.11 Mbits/sec  17
[  4]   3.00-4.00   sec   144 KBytes  1.18 Mbits/sec  18
[  4]   4.00-5.00   sec   144 KBytes  1.18 Mbits/sec  18
[  4]   5.00-6.00   sec   120 KBytes   984 Kbits/sec  15
[  4]   6.00-7.00   sec   152 KBytes  1.24 Mbits/sec  19
[  4]   7.00-8.00   sec   144 KBytes  1.18 Mbits/sec  18
[  4]   8.00-9.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   9.00-10.00  sec   112 KBytes   921 Kbits/sec  14
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.00  sec  1.34 MBytes  1.13 Mbits/sec  2.471 ms  1/171 (0.58%)
[  4] Sent 171 datagrams

iperf Done.
```

Captura Wireshark:

Source	Destination	Protocol	Length	Info
2001:db8:7351::165	2001:470:0:238::2	UDP	1294	63538→5201 Len=8192
2001:db8:7351::165	2001:470:0:238::2	IPv6	1294	IPv6 fragment (off=1232 more=y
2001:db8:7351::165	2001:470:0:238::2	IPv6	1294	IPv6 fragment (off=2464 more=y
2001:db8:7351::165	2001:470:0:238::2	IPv6	1294	IPv6 fragment (off=3696 more=y
2001:db8:7351::165	2001:470:0:238::2	IPv6	1294	IPv6 fragment (off=4928 more=y
2001:db8:7351::165	2001:470:0:238::2	IPv6	1294	IPv6 fragment (off=6160 more=y
2001:db8:7351::165	2001:470:0:238::2	IPv6	870	IPv6 fragment (off=7392 more=n

Las capturas Wireshark muestran fragmentación de paquetes debido al exceso en el tamaño del payload enviado al servidor remoto (8192 bytes). Cada mecanismo soporta un tamaño máximo de payload:

- NAT44 → 1472 bytes de payload
  1500 bytes (MTU) = 20 IPv4 header_bytes + 8 UDP header_bytes + **1472 payload_bytes**
- NAT64 → 1432 bytes de payload
  1480 bytes (MTU) = 40 IPv6 header_bytes +8 UDP header_bytes + **1432 payload_bytes**.
- NAT66 → 1232 bytes de payload
  1280 bytes (MTU) = 40 IPv6 header_bytes +8 UDP header_bytes + **1232 payload_bytes**.

3.2.2.1. Ancho de banda de la red

Se realizan 5 pruebas para estimar el ancho de banda del ISP que servirá de escenario para la evaluación del rendimiento de los enlaces NAT44, NAT64 y NAT66. Para ello, se ejecuta la aplicación SpeedTest[23] desde un dispositivo conectado a la red WiFi. Los resultados de las mediciones se muestran en la siguiente tabla:

[23] Aplicación desarrollada por OOKLA

Tabla Nº 27 – Medición del ancho de banda con SpeedTest

Hora	Subida (Mbps)	Descarga (Mbps)
8:25	5,3	3,34
8:39	5,75	3,3
13:55	5,13	3,31
14:16	5,02	3,56
14:42	6,24	3,87
Promedio	5,488	3,476

3.2.2.2. Envío de Paquetes a través UDP

Para esta secuencia de pruebas se toman 10 muestras. La configuración de Iperf tendrá los siguientes parámetros:

- **l** tamaño de payload; 1472 bytes en NAT44, 1232 bytes en NAT66 y 1432 bytes en NAT64
- **b** ancho de banda de envío; 5 Megabits/s (tabla Nº 27)
- **t** tiempo de duración de la prueba, 60 segundos

Los comandos Iperf a ejecutarse en la consola CMD de Windows para cada una de las 10 muestras:

NAT44:

```
iperf3.exe -c iperf.he.net -u -4 -l 1472 -b 5M -t 60
```

NAT64:

```
iperf3.exe -c 2001:db8:7351:ffff::216.218.227.10 -u -l 1432 -b 5M -t 60
```

NAT66:

```
iperf3.exe -c iperf.he.net -u -6 -l 1232 -b 5M -t 60
```

Luego de ejecutar las pruebas, los resultados son registrados y comparados en los siguientes gráficos:

a. Diferencia en el jitter NAT44, NAT64 y NAT66

Tabla Nº 28 – Test de jitter envío de información UDP

	Pruebas											
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Minimo	Maximo
NAT44	6,096	1,171	2,334	1,479	4,872	5,208	1,891	1,663	3,089	1,783	1,171	6,096
NAT64	3,611	1,243	1,913	3,323	1,434	5,634	8,087	1,389	1,185	2,108	1,185	8,087
NAT66	1,076	2,658	2,236	1,109	3,127	4,275	1,112	2,726	1,010	2,148	1,010	4,275

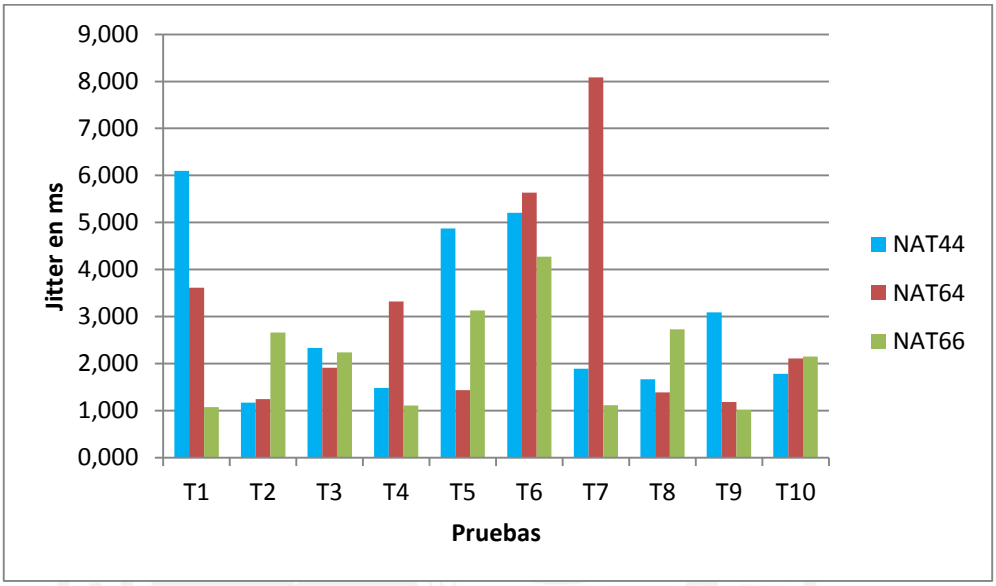


Fig. Nº 32 - Diferencia en jitter, envío UDP en enlaces NAT44, NAT64 y NAT66

De la tabla Nº 28 y figura Nº 32, NAT66 presenta un tener un jitter inferior a NAT44 y NAt64.

b. Diferencia en pérdida de paquetes NAT44, NAT64 y NAT66

Tabla Nº 29 – Test de pérdida de paquetes envío de información UDP

	Pruebas											
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Minimo	Maximo
NAT44	20,203%	0,000%	0,405%	0,000%	4,649%	2,968%	0,051%	0,028%	0,358%	0,193%	0,000%	20,203%
NAT64	0,772%	0,000%	0,967%	0,000%	0,451%	0,126%	0,348%	0,000%	0,413%	0,000%	0,000%	0,967%
NAT66	2,401%	1,944%	0,737%	0,796%	0,915%	0,924%	1,569%	0,763%	1,128%	1,174%	0,737%	2,401%

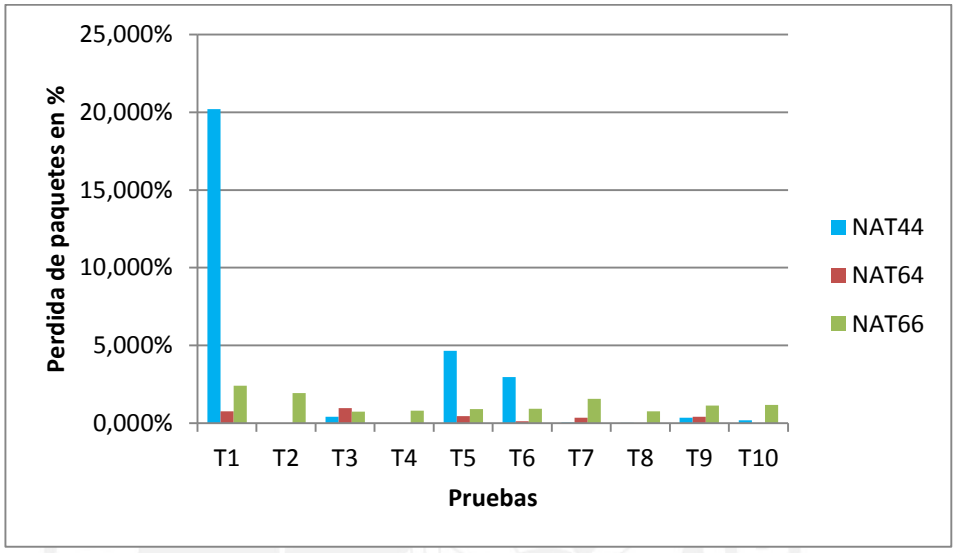


Fig. Nº 33 – Diferencia en pérdida de paquetes, envío UDP en enlaces NAT44, NAT64 y NAT66

De la tabla Nº 29 y figura Nº 33, NAT64 y NAT66 tienen menor pérdida de paquetes en comparación con NAT44.

3.2.2.3. Descarga de Paquetes a través UDP

Se toma en cuenta la configuración de los comandos ejecutados en el punto 3.2.2.1 y se agrega la opción –R que permite la recepción de paquetes desde el servidor remoto **iperf.he.net**. Además, se ajusta el ancho de banda para la descarga a 3Mbits/s, de acuerdo a la tabla Nº 27**.**

Los comandos a ejecutarse son:

NAT44:

```
iperf3.exe -c iperf.he.net -u -4 -l 1472 -b 3M -t 60 -R
```

NAT64:

```
iperf3.exe -c 2001:db8:7351:ffff::216.218.227.10 -u -l 1432 -b 3M -t 60 -R
```

NAT66:

```
iperf3.exe -c iperf.he.net -u -6 -l 1232 -b 3M -t 60 -R
```

a. Diferencia en el jitter NAT44, NAT64 y NAT66

Tabla Nº 30 – Test de jitter descarga UDP

	Pruebas											
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Minimo	Maximo
NAT44	2,925	3,938	1,711	1,792	1,694	5,706	0,573	2,908	2,908	13,495	0,573	13,495
NAT64	3,695	0,927	2,011	1,066	5,087	9,779	1,388	3,184	3,184	6,952	0,927	9,779
NAT66	1,944	1,846	0,765	1,195	1,694	2,043	2,505	2,672	2,672	15,118	0,765	15,118

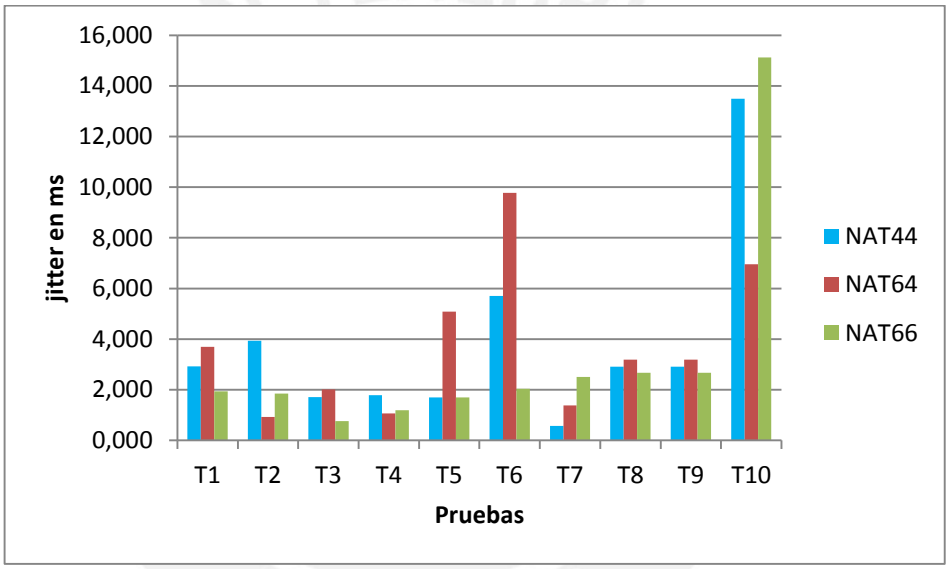


Fig. Nº 34 - Diferencia en jitter, descarga UDP en enlaces NAT44, NAT64 y NAT66

De la tabla Nº 30 y figura Nº 34, NAT64 presenta un menor jitter.

b. Diferencia en pérdida de paquetes NAT44, NAT64 y NAT66

Tabla Nº 31 - Test de pérdida de paquetes descarga UDP

	Pruebas											
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Minimo	Maximo
NAT44	5,706%	0,142%	0,630%	1,872%	7,395%	21,935%	0,046%	5,061%	5,061%	7,939%	0,046%	21,935%
NAT64	1,975%	0,309%	2,817%	1,544%	2,546%	59,298%	6,982%	7,107%	7,107%	7,390%	0,309%	59,298%
NAT66	2,640%	0,722%	1,209%	4,913%	1,258%	14,404%	10,505%	11,795%	11,795%	8,286%	0,722%	14,404%

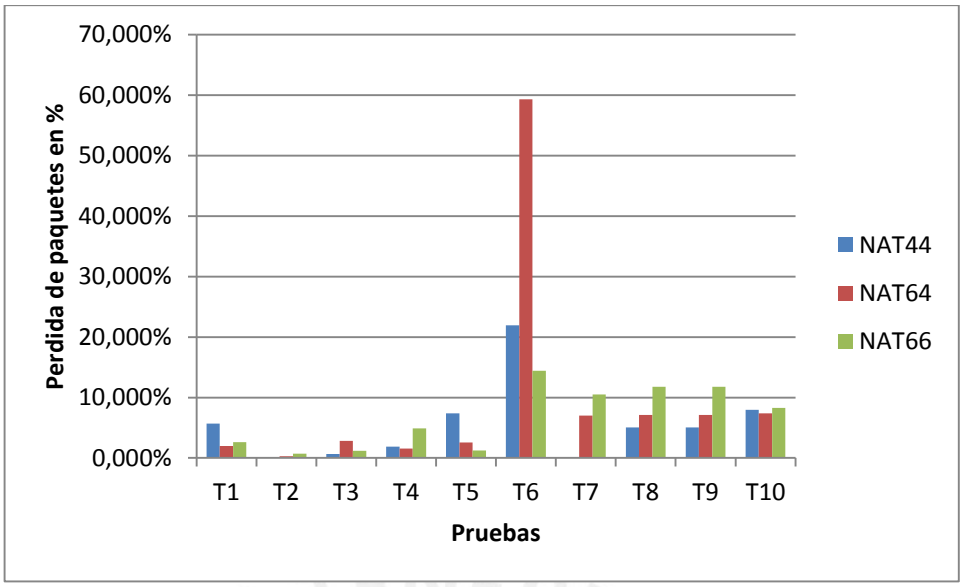


Fig. Nº 35 - Diferencia en pérdida de paquetes descarga UDP en enlaces NAT44, NAT64 y NAT66

De la tabla Nº 31 y figura Nº 35, NAT66 tiene una menor perdida de paquetes.

En conclusión, el uso de NAT64 y NAT66 para acceder a recursos disponibles en Internet no impacta en la operación de la red. Además, al comparar el desempeño de ambos mecanismos con NAT44, se cumple con el objetivo de no degradar el desempeño de la red. Y su operación es transparente para los usuarios finales.

CONCLUSIONES

La elaboración de la presente tesis abarca en su totalidad el entorno de simulación planteado en la hipótesis. Se alcanzó el objetivo principal de implementar un ambiente de red virtual y de migrar una red con usuarios Windows a IPv6. El estudio de los conceptos teóricos permitió comprender la operación del protocolo IPv6. Con lo cual, se diseñaron los escenarios para poner a prueba los mecanismos de traducción NAT64, DNS64, y los túneles 6to4 y 6RD.

A pesar de que la topología de cada mecanismo de transición a IPv6 requiere de la implementación de redes complejas, esto fue cubierto en su totalidad por el generador de ambientes virtuales VNUML desarrollado por la Universidad Politécnica de Madrid. Para ello, fue necesario entender la lógica de los algoritmos usados en esta herramienta. En ese sentido, los requerimientos de la herramienta VNUML en cuanto al procesamiento, almacenamiento y de red del host anfitrión, fueron satisfechos por una computadora portátil Dell Latitude E5400.

Para el desarrollo de este trabajo de tesis, los elementos de red fueron simulados en máquinas virtuales Linux, específicamente la distribución Debian versión 6. Así mismo, la captura de los paquetes en IPv4 e IPv6 fue realizado con el programa Wireshark. En el primer escenario se implementaron los mecanismos NAT64 y DNS64. Para ello, fue necesario instalar y configurar el servicio de traducción de direcciones Tayga y el servicio de resolución de nombres de dominio BIND. Las pruebas de operación de este escenario demostraron que el mecanismo NAT64 stateless es una buena alternativa de transición para redes domésticas. Además, se comprueba que el MTU mínimo en los enlaces de traducción debe ser 1280 bytes. Y, el MTU de un enlace NAT64 es de 1480 bytes; si se envía un paquete con un payload superior a este valor se genera mensajes PTB (Packet Too Big).

Se comprueba que se requiere una dirección IPv4 por cada dirección IPv6 traducida, lo que constituye una desventaja si se desea implementar NAT64 stateless en redes empresariales con centenares de usuarios IPv6 que deseen acceder a recursos, servicios e información disponible solo en IPv4.

En los escenarios virtuales de mecanismos de túneles 6to4 y 6RD se demostró que es posible encaminar el tráfico de petición y respuesta desde un host dual stack

ubicado en una red privada IPv6 hacia un host nativo en IPv6 por dos o más routers relay ubicados en la frontera de la red pública IPv6 a la que pertenece dicho host. Dando como resultado, caminos de ida y vuelta asimétricos. Así mismo, se demostró que los túneles 6RD son una evolución del mecanismo 6to4. Debido a que es posible asignar un prefijo 6RD a partir de la asignación dinámica, a través de DHCP, de direcciones publicas IPv6 e IPv4 en la interfaz WAN.

En el caso práctico de migración de una red Windows a IPv6 se dedicó como servidor de migración a un equipo con Linux Ubuntu. Se tomó en cuenta la inestabilidad del mecanismo NAT64 stateless gestionado por servicio Tayga y, en su defecto, se implementó el mecanismo NAT64 stateful a través del programa Jool. Se demostró que es posible asignar una sola dirección IPv4 a un grupo de direcciones IPv6. Para la distribución ordenada y dinámica de las direcciones IPv6 se implementó el servicio DHCPv6+SLAAC. Además, se demostró que es posible acceder a Internet IPv6 e Internet IPv4 desde los equipos IPv6 a través de la combinación de servicios NAT64, DNS64 y NAT66. Este último gestionado por un túnel Teredo implementado en el servidor de migración.

Teredo es la aplicación de los túneles 6to4 más simple de implementar en Linux. Sin embargo, requiere que se ejecute una nueva instalación del programa cada vez que la IPv4 pública cambia, debido a la asignación dinámica del ISP local.

Así mismo, en las pruebas de rendimiento de la red realizadas con el programa Iperf, se concluye que no existe una marcada diferencia entre los mecanismos NAT44, NAT64 y NAT66. Debido a que los tres mecanismos utilizaron el mismo acceso IPv4 del ISP local para la transferencia de paquetes con el servidor Iperf remoto.

REFERENCIAS

[1] APNIC, «Use of IPv6 for Peru (PE)», *APNIC IPv6 Measurements*, 2016. [En línea]. Disponible en: http://stats.labs.apnic.net/ipv6. [Accedido: 29-may-2016].

[2] OSIPTEL, «Medidas que requieren autorización previa», en *Neutralidad de Red*, 2015, p. 12.

[3] Alcatel-Lucent, «IPv6 migration strategies for mobile networks», en *464XLAT in mobile networks*, 2015, pp. 1-7.

[4] Google, «Google IPv6 Statics», 2016. [En línea]. Disponible en: http://www.google.com/intl/en/ipv6/statistics.html#tab=per-country-ipv6-adoption&tab=per-country-ipv6-adoption. [Accedido: 25-dic-2016].

[5] J. Govil, J. Govil, N. Kaur, y H. Kaur, «An examination of IPv4 and IPv6 networks: constraints and various transition mechanisms», en *Conference Proceedings - IEEE SOUTHEASTCON*, 2008, pp. 178-185.

[6] J. S. Bernial y J. L. Muñoz, «IPv6», en *Texto no publicado, propiedad de la Universidad Politécnica de Cataluña, Barcelona*.

[7] R. Gilligan y E. Nordmark, «Basic Transition Mechanisms for IPv6 Hosts and Routers», en *RFC 4213 (Proposed Standard)*, 2005, p. 2.

[8] M. Georgescu, H. Hazeyama, T. Okuda, Y. Kadobayashi, y S. Yamaguchi, «Benchmarking the Load Scalability of IPv6», en *20th IEEE Symposium on Computers and Communication (ISCC)*, 2015, p. 329.

[9] R. Falguera, L. Javier, y C. Llopis, «Fundamentos teorico-practicos del protocolo IPv6», en *Texto no publicado, propiedad de la Universidad Politécnica de Cataluña, Barcelona*, 2016, vol. 6, pp. 1-193.

[10] R. Garcia, «DISEÑO DE ESCENARIOS DE TRANSICIÓN A IPV6», en *Tesis, Universidad Politécnica de Madrid*, 2011, pp. 1-83.

[11] K. Moore, «Connection of IPv6 domains via IPv4 clouds», en *RFC 3056*, 2001, pp. 1-23.

[12] P.-K. Chen, C.-W. Lu, y Q. Wu, «IPv6 Rapid Deployment in Taiwan Academic Network (TANet)», en *International Conference on Advanced Communication Technology, ICACT*, 2012, pp. 694-697.

[13] C. Huitema, «An Anycast Prefix for 6to4 Relay Routers», en *RFC 3068*, 2001, pp. 1-9.

[14] CISCO, «IPv6 Rapid Deployment», en *Interface and Hardware Component Configuration Guide, Cisco IOS XE Release 3S*, 2015, pp. 1-8.

[15] Cisco Systems Inc., «NAT64 Technology : Connecting IPv6 and IPv4 Networks», en *Webtorials*, 2011, vol. 4, n.º June, pp. 1-22.

[16] N. Lutchansky, «Tayga», 2011. [En línea]. Disponible en: http://www.litech.org/tayga/. [Accedido: 26-oct-2016].

[17] A. Sullivan, M. Bagnulo, P. Matthews, y I. Beijnum van, «DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers», en *RFC 6147*, 2011, pp. 1-32.

[18] R. Denis-Courmont, «Miredo: Teredo IPv6 tunneling for Linux and BSD», 2013. [En línea]. Disponible en: https://www.remlab.net/miredo/intro.shtml.en. [Accedido: 25-dic-2016].

[19] C. Huitema, «Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)», en *RFC 4380*, 2006, pp. 1-53.

[20] D. Thaler, S. Krishnan, y J. Hoagland, «Teredo Security Updates», en *RFC 5991*, 2010, pp. 1-10.

[21] Consulintel, «Teredo», *The IPv6 Portal*. [En línea]. Disponible en: http://www.ipv6tf.org/index.php?page=using/connectivity/teredo. [Accedido: 03-ene-2017].

[22] J. Networks, «Junos ® OS», en *IPv6 Neighbor Discovery Feature Guide*, 2016, pp. 17-18.

[23] IEEE, «Guidelines for 64-bit global identifier (EUI-64)», en *EUI-64 Guidelines*, 2007.

[24] W. Simpson, T. Narten, E. Nordmark, y H. Soliman, «Neighbor Discovery for IP version 6 (IPv6)», en *RFC 2461*, 2007, pp. 1-97.

[25] T. Narten, R. Draves, y S. Krishnan, «Privacy Extensions for Stateless Address Autoconfiguration in IPv6», en *RFC 4941*, 2007, pp. 1-23.

[26] K. Lu, G. Song, y X. Hu, «Research on internet of things' support for Ipv6 addressing strategy under the platform of cloud computing», en *Proceedings - 2013 International Conference on Computational and Information Sciences, ICCIS 2013*, 2013, n.º 2, pp. 1361-1364.

[27] R. Droms, J. Bound, B. Volz, C. Perkins, T. Lemon, y M. Carney, «Dynamic Host Configuration Protocol for IPv6 (DHCPv6)», en *RFC 3315*, 2003, pp. 1-101.

[28] IBM, «Registros de recursos de DNS». [En línea]. Disponible en: https://www.ibm.com/support/knowledgecenter/es/ssw_i5_54/rzakk/rzakkconceptresourcerec.htm.

[29] «RAAP». [En línea]. Disponible en: www.lamolina.edu.pe/siglo21/marzo/raap.ppt. [Accedido: 26-jun-2016].

[30] «RED ACADEMICA PERUANA». [En línea]. Disponible en: http://www.raap.org.pe. [Accedido: 26-jun-2016].

[31] M. J. López, «Meta 10 eLAC2007 : RedCLARA y las Redes Nacionales de Investigación y Educación», 2008, pp. 36-39.

[32] G. Ríos Kruger, «Plan de Direccionamiento RAAP», en *PUCP - Oficina de Infraestructura Informática Académica*, 2016.

[33] «AS worldwide report - Peru», *Summary for as28095*, 2016. [En línea]. Disponible en: http://www.tcpiputils.com/browse/as/28095. [Accedido: 26-jun-2016].

[34] N. RIPE, «IPv6 Statistics», *AS Overview (AS28095)*, 2016. [En línea]. Disponible en: https://stat.ripe.net/AS28095#tabId=database. [Accedido: 25-dic-2016].

[35] «AS worldwide report - Peru», *AS worldwide report*, 2016. [En línea]. Disponible en: http://www.tcpiputils.com/browse/as/pe.

[36] T. Martin, E. Vyncke, W. Nellis, J. Goh, J. Handal, y S. Simlo, «IPv6 for the Enterprise in 2015», 2015, pp. 6-18.

[37] D. Wing y A. Yourtchenko, «Happy Eyeballs: Success with Dual-Stack Hosts», en *RFC 6555*, 2012.

[38] M. Mawatari, M. Kawashima, y C. Byrne, «464XLAT: Combination of Stateful and Stateless Translation», en *RFC 6877*, 2015, pp. 1-14.

[39] M. Bagnulo, A. Sullivan, y P. Matthews, «Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers», en *RFC 6146*, 2011.

[40] X. Li, C. Bao, y F. Baker, «IP/ICMP Translation Algorithm», en *RFC 6145*, 2011, pp. 1-33.

[41] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, y X. Li, «IPv6 Addressing of IPv4/IPv6 Translators», en *RFC 6052*, 2010, pp. 1-18.

[42] «Integrated Multiprotocol Network Emulator/Simulator», 2016. [En línea]. Disponible en: http://imunes.net/. [Accedido: 21-jun-2016].

[43] B. Kneale, A. Y. De Horta, y I. Box, «VELNET: Virtual Environment for Learning Networking», en *Sixth Conf. Aus- tralasian Computing Education (ACE '04)*, 2004, pp. 161-168.

[44] «My Linux Network (MLN)», 2016. [En línea]. Disponible en: http://ostatic.com/mln. [Accedido: 21-jun-2016].

[45] «Netkit», 2016. [En línea]. Disponible en: http://wiki.netkit.org/index.php. [Accedido: 21-jun-2016].

[46] «Dynamips: Cisco 7200 Simulator», 2016. [En línea]. Disponible en: http://www.infraexpert.com/info/dynamips.html. [Accedido: 21-jun-2016].

[47] «Virtual Network User Mode Linux (VNUML)», 2016. [En línea]. Disponible en: http://www.dit.upm.es/vnumlwiki. [Accedido: 21-jun-2016].

[48] F. Galan, D. Fernandez, W. Fuertes, M. Gomez, y J. E. L. De Vergara, «Scenario-Based Virtual Network Infrastructure Manage- ment in Research and Educational Testbeds with VNUML», en *Annals of Telecomm. - Annales Des Tcommunications*, 2009, pp. 305-323.

[49] A. Ruiz-Martinez, F. Pereñiguez-Garcia, R. Marin-Lopez, P. M. Ruiz-Martinez, y A. F. Skarmeta-Gomez, «Teaching advanced concepts in

computer networks: VNUML-UM virtualization tool», en *IEEE Transactions on Learning Technologies*, 2013, vol. 6, n.º 1, pp. 85-96.

[50] D. Fernandez, A. Cordero, J. Somavilla, J. Rodriguez, A. Corchero, L. Tarrafeta, y F. Galan, «Distributed virtual scenarios over multi-host Linux environments», en *2011 5th International DMTF Academic Alliance Workshop on Systems and Virtualization Management: Standards and the Cloud, SVM 2011*, 2011.

[51] F. Galan, «VNUML Language Reference», 2005. [En línea]. Disponible en: http://www.dit.upm.es//~vnuml/doc/1.5/reference/index.html#top. [Accedido: 04-dic-2016].

[52] G. Chen, Z. Cao, C. Xie, y D. Binet, «NAT64 Deployment Options and Experience», en *RFC 7269*, 2014, pp. 1-22.

[53] IANA, «Internet Control Message Protocol version 6 (ICMPv6) Parameters», 2016. [En línea]. Disponible en: http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml. [Accedido: 12-nov-2016].

[54] UPS SCS Peru, «Registro de Ingreso de Equipos, Enero-Setiembre 2016», en *Informacion Confidencial*, 2016.

[55] LACNIC, «MANUAL DE POLÍTICAS DE LACNIC (v2.6)», 2016.

[56] R. Droms, «Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6 Status», en *RFC 3736*, 2004, pp. 1-9.

[57] «dhcpd.conf(5) - Linux man page». [En línea]. Disponible en: https://linux.die.net/man/5/dhcpd.conf. [Accedido: 26-nov-2016].

[58] T. de Monterrey, «JOOL SIIT & NAT64», 2016. [En línea]. Disponible en: https://www.jool.mx/en/index.html. [Accedido: 28-nov-2016].

[59] V. J. D. Barayuga y W. E. S. Yu, «Study of packet level UDP performance of NAT44, NAT64 and IPv6 using iperf in the context of IPv6 migration», en *2015 5th International Conference on IT Convergence and Security, ICITCS 2015 - Proceedings*, 2015, pp. 4-9.

[60] «iPerf - The ultimate speed test tool for TCP, UDP and SCTP». [En línea]. Disponible en: https://iperf.fr/. [Accedido: 17-dic-2016].

[61] J. Dugan, S. Elliott, B. Mah, J. Poskanzer, y K. Prabhu, «Public iPerf3 servers», *iPerf - The ultimate speed test tool for TCP, UDP and SCTP*, 2016. [En línea]. Disponible en: https://iperf.fr/iperf-servers.php. [Accedido: 26-dic-2016].