Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Desarrollo de un escenario para el análisis de movilidad NEMO (Network Mobile) en IPv6

Autor: Pablo Pizarro Aguilar

Tutor: Juan Antonio Ternero Muñiz

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Desarrollo de un escenario para el análisis de movilidad NEMO (Network Mobile) en IPv6

Autor:
Pablo Pizarro Aguilar

Tutor:
Juan Antonio Ternero Muñiz
Profesor colaborador

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Proyecto Fin de Carrera: Desarrollo de un escenario para el análisis de movilidad NEMO (Network Mobile) en IPv6

Autor: Pablo Pizarro Aguilar

Tutor: Juan Antonio Ternero Muñiz

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

La realización de este trabajo supone el final de mis estudios de Grado. Simplemente quiero dar las gracias a todas las personas que han depositado su confianza en mí. Especialmente a mis padres.

Gracias también a mi tutor, Juan Antonio Ternero Muñiz, por guiarme en la realización de este trabajo. En general, gracias a todos mis profesores por haberme formado como ingeniero.

Pablo Pizarro Aguilar

Sevilla, 2016

Resumen

En este proyecto se analizan dos protocolos básicos en la movilidad sobre IPv6, como son MIPv6 y NEMO. El objetivo es montar un escenario virtual donde puedan implementarse estos protocolos. Partiendo del conocimiento teórico de ambas tecnologías, realizaremos un escenario que sirva como laboratorio de pruebas. En él, medimos y comparamos el rendimiento en diferentes implementaciones de estos protocolos. Se ha buscado crear un escenario fácilmente reproducible, usando software abierto, como GNS3 y Virtualbox, y que pueda ser usado para fines académicos, o de investigación.

Abstract

This project analyzes MIPv6 and NEMO, which are two of the most important protocols in mobility over IPv6. The main goal is to create a virtual laboratory for implementing these protocols. We started describing how both technologies works, in order to design the scenario. This virtual lab will allow us to compare performance measurements between different implementations of the given protocols. With this project, we tried to create a lab that can be easily deployed by everyone. We avoid big software restrictions by using open source technologies like GNS3 and Virtualbox. This virtual lab is design to be used for researching and academic purpose.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGO

Notación

AH	Authentication Header
ACL	Access Control List
ARP	Address Resolution Protocol
BA	Binding Acknowledgement
BC	Binding Caché
BU	Binding Update
BUL	Binding Update List
CN	Correspondant Node
CoA	Care-of Address
DAD	Duplicate Address Detection
DHCPv6	Dynamic Host Configuration Protocol versión 6
ESP	Encapsulation Security Payload
FN	Foreign Network
HA	Home Agent
HoA	Home of Address
HN	Home Network
IKE	Internet Key Exchange
IPsec	Internet Protocol Security
IPv4	Internet Protocol versión 4
IPv6	Internet Protocol versión 6
MAC	Media Access Control
MIPv6	Mobile IP versión 6
MN	Mobile Node
MNET	Mobile NETwork
MNN	Mobile Network Node
MNP	Mobile Network Prefix
MR	Mobile Router
NA	Neighbor Advertisement
NAT	Network Address Translation
ND	Neighbor Discovery
NEMO	Network MObile
NS	Neighbor Solicitacion
PAN	Personal Area Network
PT	Packet Tracer
RA	Router Advertisement
RFC	Request For Comments

RS	Router Solicitation
RTT	Round-Trip delay Time
SA	Security Association
SLAAC	Stateless Address Autoconfiguration
TAP	Network tap
TCP	Transmission Control Protocol
TUN	Network TUNel
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMM	Virtual Machine Monitor

INTRODUCCIÓN

La formulación de un problema es más importante que la solución.

- Albert Einstein -

Este proyecto pretende abarcar el conocimiento de la movilidad en IP versión 6 (MIPv6). Para ello, empezaremos estudiando los principios básicos que permiten la movilidad de un nodo a través de diferentes redes sin perder la conectividad. Continuaremos ampliando este desarrollo con el protocolo *Network Mobility Basic Support* (NEMO) que permite la movilidad de redes. Una vez comprendamos esto, procederemos a implementar y configurar un escenario virtual que nos permita comprender mejor la movilidad y hacer un análisis de su rendimiento.

1.1 Motivación

La movilidad IP no es algo específico de IPv6. La propuesta inicial para el soporte de movilidad en IPv4 fue presentada en 1993. En esa época, no había ordenadores que fueran realmente "móviles". Existían los ordenadores portátiles pero eran relativamente grandes y muy caros comparados con los de sobremesa. Por su parte, los teléfonos móviles no tenían la conectividad y la capacidad de computo de ahora. Aún así, en 1996 surge *IP Mobility Support for IPv4* que fue definido por primera vez en el RFC 2002[1]. Cabe destacar que el protocolo IPv4 fue desarrollado en los años 70, por lo que no fue contemplada la posibilidad de dispositivos móviles en su diseño.

Una dirección IP consta de dos partes: una parte que identifica a la subred y otra parte que identifica al terminal dentro de la misma. Los diferentes dispositivos de encaminamiento usan el prefijo de red para entregar los paquetes en la red adecuada. Una vez en la red, utilizan el identificador del terminal para determinar el nodo al que van destinados los paquetes. Esto implica que si el nodo se desplaza a otra subred, el prefijo de red cambia. Esto no supone un problema para los equipos para los que en su día fue pensado IPv4. Sin embargo, en los últimos años, los avances de la tecnología han provocado que cada vez haya más dispositivos conectados a Internet. Se estima que con el reciente auge del *Internet de las cosas* en 5 años existirán más de 50.000 millones de dispositivos conectados, estamos hablando de smartphones, tablets, relojes, sensores, etc., que por su tamaño, nos permiten movernos pasando de una red a otra.

El problema es el siguiente, supongamos que estoy en una red y que se me asigna una dirección IP con el prefijo propio de esta red. Ahora establezco una conexión con un equipo externo, y posteriormente cambio de red. Al cambiar, se me asigna una nueva IP, por lo que las conexiones TCP/IP que pudiera tener abiertas se romperían. Además, si el equipo con el que me estaba comunicando quiere volver a

[1] Su versión más reciente es el RFC 5944

hacerlo, deberá conocer mi nueva dirección.

IP no está preparado para que un dispositivo cambie de red por lo que es necesario diseñar un protocolo que de soporte a la movilidad. Es aquí donde se encuentran *IP Mobility Support for IPv4* y, posteriormente *Mobility Support for IPv6*, definido en el RFC 6275[2]. IPv6 fue creado pensando, entre otras cosas, en la movilidad y la seguridad, esto supone una gran ventaja para MIPv6 con respecto a su predecesor, provocando que sea esta la que se imponga. IPv6 permite solucionar algunos problemas que presentaba MIPv4 como, por ejemplo, la ausencia de un mecanismo de optimización de ruta entre el nodo móvil y el nodo con el que se comunica.

1.2 Estado del arte

El estudio de MIPv6 comenzó en 1996. Los primeros pasos del proceso de estandarización de la movilidad en IPv6 fueron muy rápidos si consideramos que el primer borrador del protocolo IPv6 fue presentado en 1995. Sin embargo, el proceso de estandarización fue un proceso lento y tortuoso que finalizó 9 años más tarde con la publicación de la RFC 3775.

Durante esos años encontramos algunos proyectos que estudian, o colaboran, en el desarrollo de MIPv6. En primer lugar encontramos el proyecto KAME, centrado en la creación de escenarios donde implementar MIPv6 en BSD[3] y que cerró en 2006. El proyecto USAGI centraba sus esfuerzos en la integración de IPv6 en sistemas Linux. USAGI fue abandonado hace algunos años, y ha sido relevado por UMIP, que trabaja en la creación de escenarios controlados de movilidad sobre Linux. Este proyecto ha adquirido gran importancia y es sobre e que nos vamos a apoyar en la realización de este proyecto.

KAME, USAGI y UMIP trabajan en estrecha colaboración con otros proyectos de mayor envergadura como son TAHI y WIDE Proyect [1]. El proyecto TAHI, fundado en 1998, se creó con el objetivo de desarrollar y realizar test de verificación sobre diferentes tecnologías en IPv6, entre las que se encuentra MIPv6. Por su parte, WIDE a ayudado a identificar y resolver algunos problemas relacionados con la migración a IPv6 y, junto con el grupo Nautilus6, han trabajado en la investigación y desarrollo de tecnologías fundamentales para las comunicaciones móviles, como MIPv6 y NEMO.

1.3 Objetivos y Alcance del proyecto

Este trabajo trata de estudiar los protocolos MIPv6 y NEMO. Comenzará realizando un estudio teórico de ambos que nos permitan conocer los principios básicos de la movilidad. El objetivo final de este proyecto es la realización de un escenario virtual donde poder poner a prueba el rendimiento de los citados protocolos, y que nos ayude a tener una mayor comprensión de su funcionamiento. Es por esto, que dedicaremos un capítulo a conocer las tecnologías que están detrás del software que nos va a permitir implementar dicho escenario. Finalizaremos con un análisis de los datos obtenidos de las diferentes pruebas de rendimiento.

[2] Se definió por primera vez en 2004 en el RFC 3775

[3] *Berkeley Software Distribution* fue un sistema operativo derivado de Unix nacido a partir de los aportes realizados a este sistema por la Universidad de California en Berkeley

2 MOBILE IPV6

En este capítulo vamos a explicar los conceptos básicos para entender el funcionamiento de la movilidad en IPv6. Para ayudarnos, nos apoyaremos en algunos protocolos propios de IPv6 íntimamente ligados a la movilidad y que son, en parte, responsables de su superioridad sobre MIPv4. Estos protocolos son: *Stateless Address Autoconfiguration* (SLAAC), *Neighbor Discovery* (ND), e *Internet Protocol Security* (IPsec).

La movilidad en IPv4, como ya mencionamos en el apartado 1.1, fue añadida en una extensión posterior al diseño del protocolo IPv4. En cambio en IPv6, la movilidad está totalmente integrada, pues se tuvo en cuenta desde el diseño inicial. Esto provoca que MIPv6 sea más eficiente y evite algunos problemas de MIPv4. Por ejemplo, en MIPv4, se requiere de la presencia de un *Foreign Agent* (FA), un router que asiste al nodo móvil cuando se mueve a una red externa asignándole una nueva IP, e indicándole que ha cambiado de red. La autoconfiguración de direcciones de IPv6 proporciona la funcionalidad requerida para asignarle a un nodo una dirección IP de forma dinámica. En IPv6 se incorporan dos protocolos de autoconfiguración [1]

- *Dynamic Host Configuration Protocol* (DHCPv6) en el que el nodo se comunica con un servidor de direcciones.
- *Stateless Address Autoconfiguration* (SLAAC) [2] con el que el nodo obtiene su dirección comunicándose con el router local, y que funciona de la siguiente manera:
    - Cuando se conecta la interfaz a la red, se le asigna una dirección de enlace-local[4] (*Link-local*)
    - A través de esta dirección, la interfaz envía un *Router Solicitation* (RS) , mensaje propio del protocolo ND que explicaremos a continuación. En este mensaje se pregunta si existe algún router en el enlace.
    - Como respuesta, se recibe un *Router Advertisement* (RA) por el que el router indica su dirección dentro de la subred.
    - La interfaz del nodo solicitante coge el prefijo de red que contiene el mensaje RA recibido y lo utiliza para crear su propia dirección IPv6 Global mediante el algoritmo EUI-64.

2.1 Neighbor Discovery Protocol (NDP o ND)

Antes de explicar el funcionamiento básico de MIPv6 es necesario comprender el funcionamiento de este protocolo, ya que juega un papel fundamental. El ND surge para reemplazar al protocolo ARP (*Address Resolution Protocol*) implementado en IPv4 [4]. Para proveer dicha funcionalidad, ND se apoya en el protocolo ICMPv6.

El protocolo ND no sólo sirve para el descubrimiento estricto de vecinos, sino que capacita al nodo para realizar las siguientes acciones:

- *Address resolution*. Resolver direcciones IPv6 a MAC, como hiciera ARP.

[4] Dirección no enrutable creada únicamente para comunicaciones dentro de la subred local. Se forman a partir del prefijo FE80 : :/64 y el identificador de la interfaz

- *Router discovery.* Descubrir un router en el mismo enlace.
- *Prefix discovery.* Descubrir que prefijos de red hay en el enlace y cuáles son alcanzables a través del Gateway.
- *Address autoconfiguration.* Autoconfiguración de su dirección IPv6 sin necesidad de un servidor DHCP (SLAAC).
- *Parameter discovery.* Descubrir parámetros de red propios del enlace.
- *Neighbor unreachability detection.* Detectar cuando un nodo deja de ser accesible en el enlace.
- *Duplicate address detection.* Detectar si una si una IPv6 ya está siendo usada en el enlace.
- *Next-hop determination.* Determinar cuál es el vecino que se encargara para reenviar un paquete con destino fuera de la subred.
- *Redirect.* Un nodo recibe información de un router sobre un mejor salto para alcanzar un destino.

Para dar soporte a estas funcionalidades, ND se apoya en ICMPv6 (*Internet Control Message Protocol version 6*). No entraremos a ver en detalle este protocolo, pero si detallaremos los mensajes que dan soporte a ND[3]:

Tabla 2–1. Principales mensajes de ICMPv6 utilizados en ND.

Mensaje	Tipo	Descripción
Router Solicitation (RS)	133	Cuando una interfaz se activa, los hosts mandan este mensaje para pedir a los router que se anuncien.
Router Advertisement (RA)	134	Este mensaje es originado por un router en respuesta a un RS o periódicamente pasado un cierto intervalo. Aporta información de la red.
Neighbor Solicitation (NS)	135	Mensaje enviado por un nodo para detectar la dirección de la capa de enlace de sus vecinos, verificar si un vecino sigue siendo alcanzable o detectar direcciones duplicadas.
Neighbor Advertisement (NA)	136	Enviado en respuesta a un NS. Un nodo debe enviar este mensaje para anunciar un cambio en su dirección de enlace.
Redirect	137	Enviado por los routers para indicar a un nodo una mejor forma de alcanzar un destino.

Con *Tipo* nos referimos al campo de la cabecera ICMPv6 que indica qué tipo de mensaje es.

Tabla 2-2. Cabecera ICMPv6.

0 – 7	8 – 15	16 – 31
Tipo	Código	Checksum

Para poder dar soporte a MIPv6 este protocolo a tenido que sufrir algunas modificaciones. Estos cambios se han centrado principalmente en el mensaje RA. Se le ha añadido un bit H que nos permite indicar si el router

que envía el RA está actuando como *Home Agent* (lo veremos a continuación) o no, en función de si está a 1 o 0, respectivamente. También se ha añadido otro bit R, que si está a 1, indica que está anunciando la dirección global del router y no la dirección local al enlace como suele ser normal en ND.

Por último, se añaden dos opciones al RA. *Advertisement Interval*, para indicar el intervalo que usa el router para el envío de RA no solicitados, y *Home Agent Information*.

2.2 Entidades principales

En este apartado, vamos a presentar las diferentes entidades funcionales que participan en MIPv6[5]:

- *Mobile Node* (MN). Un nodo móvil puede moverse de una red a otra sin dejar en ningún momento de estar disponible, ni perder la conexión, manteniendo activas las comunicaciones de niveles superiores. Es la entidad protagonista del esquema de movilidad.
- *Home Agent* (HA). Router de la red origen (*Home Network*) que gestiona la localización del MN. Contiene una lista de MNs registrados y es responsable de hacer llegar al MN aquellos paquetes que son dirigidos a él mientras se encuentra fuera de la red origen.
- *Correspondant Node* (CN). Nodo fijo o móvil que se comunica con el MN.
- *Home network* o *red origen* (HN). Red de la que parte el nodo móvil y en la cual se encuentra el HA que gestiona su movilidad.
- *Foreign network* o *red visitada* (FN). Cualquier red distinta de la red origen en la que se encuentra el MN y en la que adquiere una dirección IP auxiliar.
- *Home Address* (HoA). Dirección IP permanente que se le asigna al MN en su red origen.
- *Care-of Address* (CoA). Dirección IP local que identifica a un nodo móvil en su ubicación actual. Si está en la *Home Network*, esta dirección coincidirá con la HoA.
- *Binding Cache* (BC). Conjunto de entradas las cuales disponen de una asociación entre la HoA y la CoA de un nodo móvil. Estas tablas se pueden encontrar en el HA, e incluso en algunos CN.

2.3 Funcionamiento general

Una vez vistos, los elementos que intervienen, pasaremos a explicar como interaccionan entre ellos para que la movilidad sea efectiva [6].

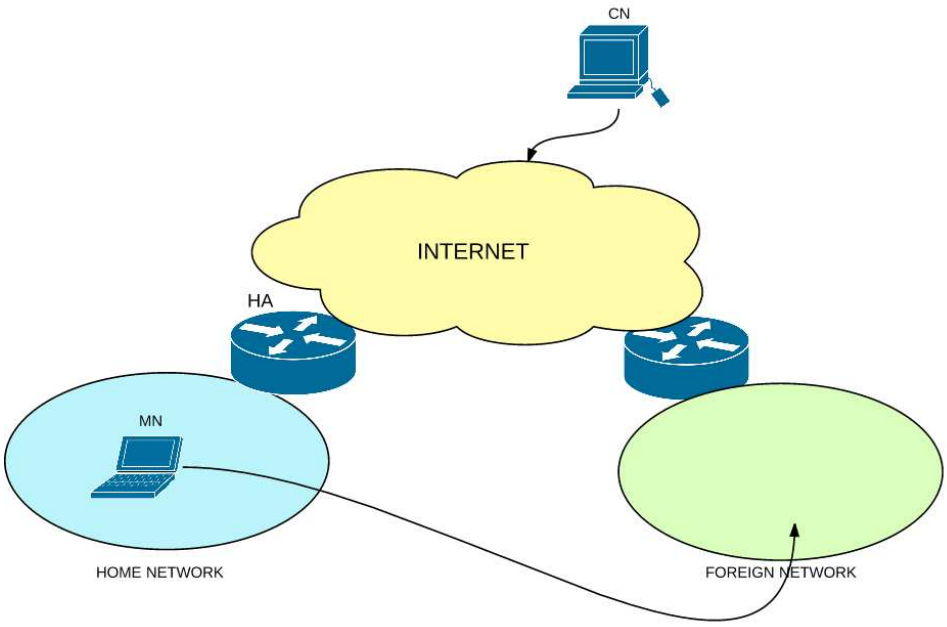


Figura 2-1. Esquema de un escenario de movilidad

Comenzamos con el MN situado en la red origen o *Home Network*. En esta red se le asigna una dirección IPv6, ya sea por SLAAC o por DHCPv6, que conformará la HoA. Esta dirección servirá para identificar de manera unívoca a este nodo en todo Internet. Cuando el MN se encuentra en la HN, todos los paquetes con destino la HoA serán encaminados de la manera tradicional, sin necesidad que intervenga el HA ni ningún otro elemento de movilidad.

La cosa cambia cuando el MN decide desplazarse a otra red (*Foreign Network*). El proceso por el cual un nodo pasa de una red a otra se conoce como *Handover* o *traspaso*. Una vez alcanzada la nueva red el MN necesita obtener una nueva dirección IP, nos referimos a la CoA. El MN detecta que se ha movido a una nueva red por medio del análisis del mensaje RA que periódicamente envía el router de la nueva red. El nodo móvil también puede forzar el envío de un RA mediante la emisión de un RS.

El MN podrá obtener una nueva dirección IP para la nueva subred (CoA), a partir de la información contenida en el RA. En primer lugar, el MN necesita comprobar que su dirección de enlace es única en el mismo, por lo que realiza el proceso de *Detección de Direcciones Duplicadas* (DAD) en su enlace local. Luego se realiza la autoconfiguración (SLAAC o DHCPv6) para formar la nueva CoA. Con esta dirección ya configurada puede realizar el proceso DAD. Este proceso funciona enviando varios NS con destino su nueva IP y esperando a ver si recibe respuesta durante, al menos, un segundo. Esto hace que el tiempo de latencia del *handover* sea mayor. Por este motivo, el MN debería realizar el DAD en paralelo con sus comunicaciones, o directamente, no realizarlo.

En la situación actual, si un CN decide comunicarse con el MN usando como dirección destino la HoA, la conexión no se hará efectiva, pues el MN está usando su nueva dirección en la *Foreign Network*. Es aquí donde interviene el HA. Este deberá recoger los paquetes destinados a la HoA y reenviarlos a la CoA de modo que la comunicación sea posible. Para ello, el HA debe registrar donde se encuentra el MN en cada momento actualizando su *Binding Cache*. La BC tiene una entrada por cada nodo móvil propio de la HN. El MN debe registrar esta asociación en su *Binding Update List* (BUL), y posteriormente enviar al HA un *Binding Update* (BU). El HA irá actualizando su BC a medida que recibe nuevos BU o expira el tiempo de vida de alguna de sus entradas. Cuando una asociación se registra correctamente el HA envía un *Binding Acknowledgement* (BA) como respuesta.

Una vez se ha realizado el registro y las BC están actualizadas con la nueva CoA, comenzará la transferencia de datos. Esta puede funcionar de dos modos distintos:

- El primero, y más básico (*triangle routing*), consiste en usar un túnel bidireccional entre el HA y el MN. De esta forma, los paquetes del CN son encaminados al HA y desde allí se envían a través del túnel al MN. Este tipo de encaminamiento es totalmente transparente al CN que no necesita ningún

tipo de soporte de movilidad para comunicarse con el MN. Para que esto funcione correctamente el HA tiene que utilizar *Proxy Neighbor Discovery* para interceptar los paquetes dirigidos a la HoA del MN, y posteriormente, enviarlos en un túnel IPv6-in-IPv6 con destino la CoA. El MN responde usando el mismo túnel en sentido inverso. Es importante remarcar que, la dirección origen de este mensaje de respuesta es CoA, para evitar problemas con las políticas ACL en la FN.

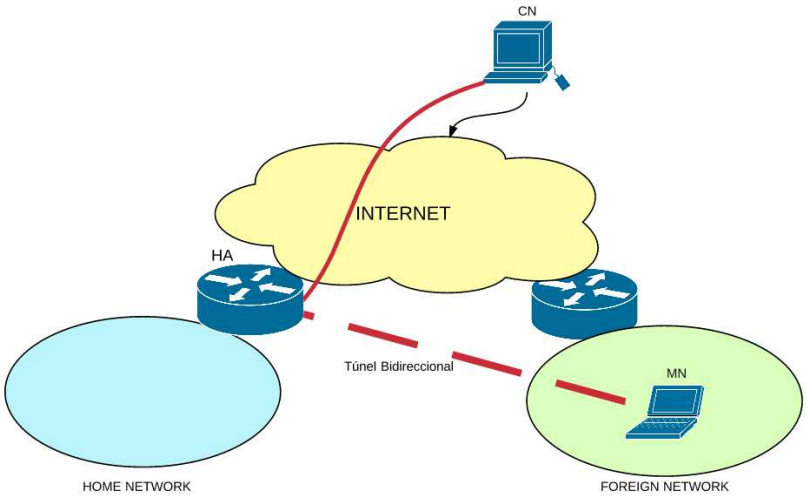


Figura 2-2. Funcionamiento sin optimización de rutas

- El segundo se conoce como *Route Optimization*, o con optimización de rutas. En este método, el CN debe tener soporte para movilidad, ya que debe tener su propia BC. El CN envía los paquetes directamente a la CoA del MN. Cuando se envía un paquete IPv6, el CN comprueba su BC. Si hubiera una correspondencia, el nodo utiliza una nueva cabecera de encaminamiento (*Routing Header*) para hacer llegar el paquete a la CoA del MN.

  Pero ¿cómo sabe el CN dónde se encuentra el MN? Cuando el MN cambia de red este le envía un BU al CN para que actualice su tabla de asociaciones. Cuando el MN manda un mensaje de respuesta, este se envía, igual que antes, con dirección origen CoA. Para que esto no suponga un problema para el CN, el MN añade en el mensaje de respuesta una cabecera de extensión en la que incluye la HoA. Al recibir el paquete el CN sustituye la dirección origen por la incluida en la cabecera de extensión, de forma que no suponga ningún problema para las capas superiores.

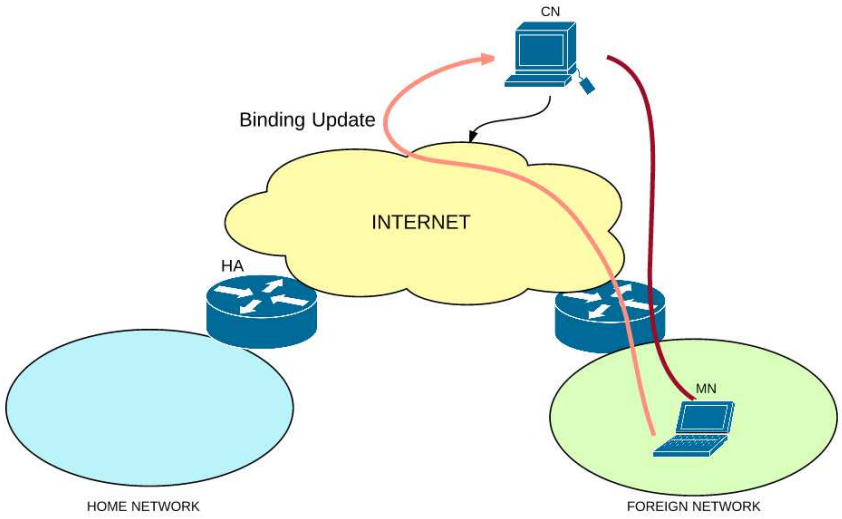


Figura 2-3. Funcionamiento con optimización de rutas.

Para que los *Binding Updates* que se usan, entre MN y CN, en este modo sean seguros MIPv6 define un nuevo protocolo IPv6, usando la cabecera de movilidad, llamado *Return Routability Procedure*[7]. A través de la cabecera porta los siguientes mensajes:

- *Home Test Init*
- *Home Test*
- *Care-of Test Init*
- *Care-of Test*

No entraremos a detallar estos mensajes, pero a continuación mostraremos un pequeño esquema de su funcionamiento.

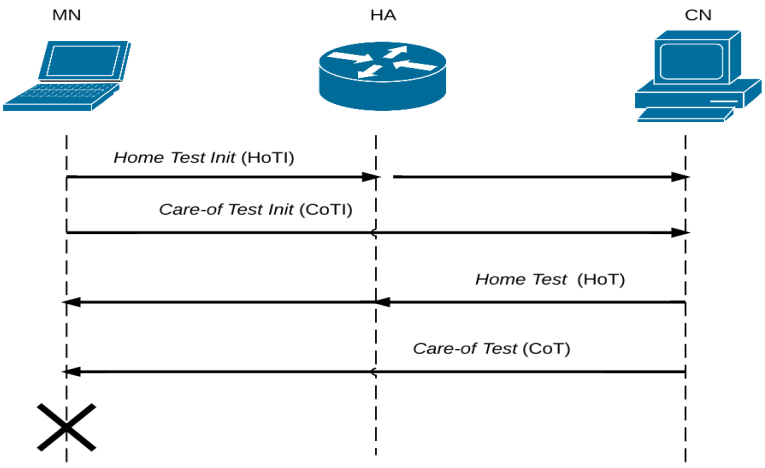


Figura 2-4. Paso de mensajes *Return Routability Procedure.*

Estos no son los únicos mensajes que ICMPv6 a tenido que implementar para dar soporte a IPv6. A continuación mostramos otros mensajes importantes para el buen desarrollo de la movilidad[1]:

Tabla 2-3. Mensajes ICMPv6 para dar soporte a MIPv6.

Mensaje	Tipo	Descripción
ICMP Home Agent Discovery Request	144	Enviado por el MN para iniciar el descubrimiento dinámico de la dirección del HA
ICMP Home Agent Address Discovery Reply	145	Mensaje generado por el HA en respuesta al anterior.
ICMP Mobile Prefix Solicitation	146	Mensaje enviado por el MN para obtener información de su red origen, cuando se encuentra fuera.
ICMP Mobile Prefix Advertisement	147	Mensaje generado por el HA en respuesta al anterior.

2.4 Seguridad e IPsec

Hasta ahora no habíamos hablado del tema de la seguridad en torno a MIPv6. Esto es algo que, quizás, no sea muy importante en nuestro pequeño escenario de pruebas, pero resulta vital cuando nos encontramos ante una implementación real.

Por el momento, estamos mandando todos los mensajes sin protección ninguna. Esto es peligroso, especialmente, con los paquetes de señalización, que contienen información sensible de movilidad. Es muy importante proteger el tráfico de señalización entre el MN y el HA, pues si no se hiciera, MNs y CNs serían vulnerables a ataques *man-in-the-middle*, *passive wiretapping* o *denial-of-service*, entre otros.

Por ejemplo, si algún elemento malicioso capturara un BU entre el MN y el HA, este podría replicar este mensaje y enviar un BU al HA haciéndose pasar por el MN y cambiando la CoA. Por lo tanto, el HA interpretaría que el MN a cambiado su ubicación y actualizaría su BC con datos falsos. Esto podría usarse para derivar tráfico a un tercer nodo para, por ejemplo, intentar provocarle una denegación de servicio.

Es por eso, que el IETF definió el uso de IPsec [8], para proteger la señalización entre MN y HA, recogiéndolo en la norma RFC 3776. Pero ¿qué es IPsec? Es un conjunto de protocolos, que actúa en la capa 3[5], cuyo objetivo es asegurar el flujo de paquetes, garantizar la autenticación mutua y establecer parámetros criptográficos. IPsec gira entorno a las *Asociaciones de Seguridad* (SA), que es un tipo de conexión que permite definir el paquete de algoritmos y parámetros que se están usando para asegurar un flujo de particular. En cada SA se define que protocolo de seguridad se usa, puede ser *Authentication Header* (AH) o *Encapsulation Security Payload* (ESP), en el caso de MIPv6 se usa el segundo.

Otro aspecto importante es la compartición de claves. Ipsec permite dos formas: configurando de forma manual ambos extremos, o emplear un protocolo de intercambio dinámico de claves como IKE[9]. Actualmente, está recomendado el uso de su siguiente versión, IKEv2, junto con IPsec, como forma de asegurar la movilidad en IPv6[6].

2.5 Network Mobile Basic Suport (NEMO)

Como parte importante de este proyecto, es necesario dedicarle un apartado a NEMO antes de cerrar el capitulo de la movilidad en IPv6. En este apartado vamos a explicar brevemente, las características de este protocolo y su funcionamiento. Puede resultar extraño, que siendo parte tan importante de este proyecto, se le dedique un apartado, y no un capítulo entero. Hemos preferido hacerlo así, pues este protocolo se entiende mejor si partimos del conocimiento de MIPv6, pues NEMO es una extensión de este. Su funcionamiento es, prácticamente, el mismo, por lo que vamos a dedicar a este apartado a explicar el porqué de su uso y qué lo diferencia de MIPv6.

2.5.1 Motivación

Impulsado por el gran éxito de la telefonía móvil, la movilidad ha cambiado el modo en que las personas se comunican. El acceso a Internet se ha convertido en algo totalmente ubícuo y las necesidades de movilidad no están restringidas únicamente a nodos individuales. Es necesario soporte para el desplazamiento de redes completas que vayan cambiando su punto de acceso a la infraestructura fija, manteniendo las conexiones de los dispositivos de la red. Para ello, necesitamos incluir un *router móvil* o *Mobile Router* (MR) que conecte los dispositivos de la red móvil con Internet y que sea quién gestione la movilidad.

Algunos ejemplos de escenarios en los que podemos encontrar redes móviles son: los transportes públicos, como barcos, aviones, autobuses o trenes. Estos permiten a los pasajeros conectar sus dispositivos a Internet a medida que se desplazan, a través de un router móvil situado en el vehículo. O las *Personal Área Networks* (PAN) o *redes personales* que permiten a una persona conectar diferentes dispositivos a Internet utilizando el teléfono móvil como router móvil[10].

MIPv6 no soporta el desplazamiento de una red, entendiéndola como un todo. Con MIPv6 podría gestionar

[5] Por lo que puede dar soporte a protocolos de la capa 4 como TCP/UDP. Otros como SSL, TSL o SSH operan de la paca 4 hacia arriba.
[6] Se recoge en el RFC 4877

individualmente la movilidad de cada nodo, pero esto tiene varias desventajas:

- Es necesario que todos los dispositivos tengan soporte para movilidad. Algunos, por la antigüedad del dispositivo o su capacidad limitada, pueden no soportarlo. Con NEMO, tenemos un único elemento gestionando la movilidad de la red. Sólo el MR debe tener soporte para movilidad, mientras el resto de aparatos no necesitan ningún software especial.
- Si estamos, por ejemplo, en un tren con 200 personas, cada dispositivo manda sus propios mensajes de señalización sobrecargando la red cada vez que cambia su punto de acceso a la infraestructura fija. Con NEMO, la señalización esta limitada a un solo elemento, el MR.

Estos problemas fueron los que llevaron al IETF a crear el *NEMO Working Group* para crear una solución estandarizada que quedaría recogida en la RFC 3963.

2.5.2 Funcionamiento general de NEMO

La forma de operar de NEMO es, prácticamente, igual que en MIPv6, pero con algunos matices e incluyendo algunos conceptos nuevos [11]. Como hemos nombrado antes, dentro de la red móvil tenemos un router (MR) que conecta el resto de dispositivos a Internet. Este router esta asignado inicialmente a una *red origen* o *Home Network* (HN). Como parte de esta red, la red móvil tiene asignadas bloques de direcciones dentro del rango de la HN, a estos bloques los conocemos como *Mobile Network Prefixes* (MNPs). Estas direcciones no cambian cuando el MR se encuentra fuera de la HN. Cuando la NEMO se ha desplazado otra red, los paquetes destinados a los nodos de la NEMO, llamados *Mobile Network Nodes* (MNNs), son enrutados a la HN, y allí el HA los envía al MR.

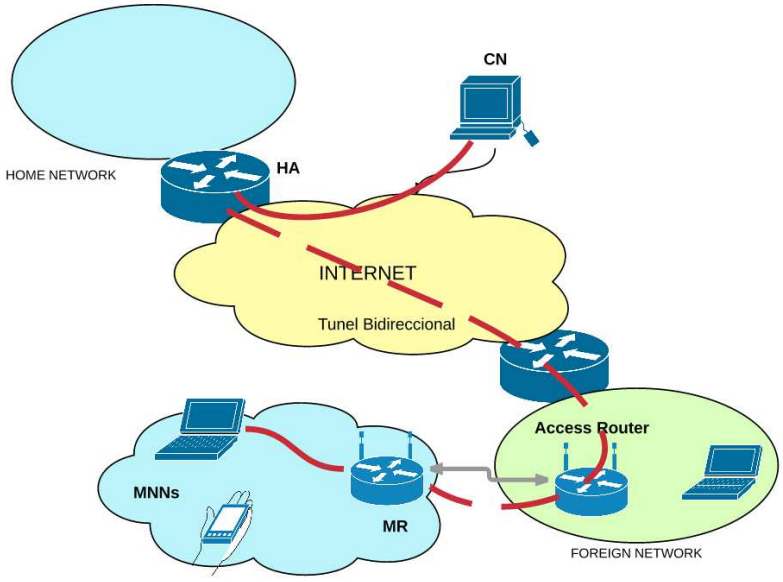


Figura 2-5. Esquema general de una arquitectura NEMO

La figura muestra un ejemplo de una posible arquitectura de un escenario NEMO. En él, observamos que el modo de enrutamiento se corresponde con el de modo sin optimización de rutas que vimos en MIPv6. Esto es así porque el soporte para optimización de rutas en NEMO es mucho más complejo y no tiene aún una solución estable. El IETF está trabajando en esta problemática que queda recogida en las RFC 4888 y RFC 4889.

Para comprender mejor los cambios introducidos por NEMO, vamos a compararlo con MIPv6. En MIPv6 identificamos tres mecanismos principales que dan soporte a la movilidad: detección del movimiento, registro de la localización y tunelado del tráfico. Así pues, vamos a organizar nuestra comparativa en base a estos mecanismos.

- *Detección del Movimiento*. Funciona sirviéndose del RA del protocolo ND por lo que, en este caso, NEMO no introduce ningún cambio en este mecanismo.

- *Registro de la Localización*. En MIPv6, el nodo móvil informa de su CoA actual, a través del mensaje *Binding Update*. Este mensaje es encapsulado en un datagrama IPv6 usando una extensión de cabecera especial, llamada *Mobility Header*. NEMO modifica el BU para introducir la siguiente información:
    - *Mobile Router Flag* (R). Este flag se pone a 1 para indicar al HA que el BU es enviado por un MR. Si estuviera a 0, el HA interpretaría que lo está recibiendo de un MN y no reenviaría hacia él ningún paquete destinado a la red móvil.
    - *Mobile Network Prefix Option*. Esta opción se añade para indicar información del prefijo de la red móvil al HA. Usado, por ejemplo, cuando la red móvil tiene varios prefijos.
- *Tunelado de Tráfico*. En NEMO, el HA no tiene que enrutar hacia el MR sólo los paquetes destinados a la HoA del MR, también debe enviar todo el tráfico destinado a cualquiera de los prefijos gestionados por la *Mobile Network*. Para determinar qué prefijos pertenecen a esta red, el HA tiene tres métodos diferentes:
    - *Explicit Mode*. El MR incluye en los BU información sobre los prefijos configurados en la red móvil.
    - *Implicit Mode*. El MR no incluye esta información y el HA determina los prefijos de la red móvil usando otro mecanismo(NEMO no especifica ninguno en concreto).
    - *Intra-domain Dynamic Routing Protocol through the bidireccional tunnel*. Entre HA y MR, también se puede usar un protocolo de enrutamiento intradominio, como RIPng o OSPF, a través del túnel bidireccional.

En resumen, NEMO extiende las funciones de MIPv6 para dar soporte a la movilidad de redes. Aunque las especificaciones actuales solo contemplan el soporte básico de movilidad, desde el IETF se trabaja para desarrollar nuevas extensiones que proporcionen soporte para, entre otras cosas, optimización de rutas y capacidad de *multihoming*.

Existen algunas implementaciones de código libre (*Open-Source*), como SHISA para sistemas BSD, o NEPL (*NEMO Platform for Linux*. Este último fue abandonado hace varios años y, su relevo lo tomó UMIP, como ya hiciera con MIPv6 y el proyecto USAGI. Y es esta la implementación que usaremos en nuestras pruebas.

3 SOFTWARE DE VIRTUALIZACIÓN

En este capítulo vamos a presentar las principales herramientas que nos han servido para implementar el escenario de movilidad. Para su desarrollo nos hemos servido de un emulador, como es GNS3, para gestionar y configurar la red, y un software de virtualización, como VirtualBox, para crear los nodos que participan en la movilidad. Otro software que hemos usado en el montaje del escenario ha sido TunTap. Es un programa sencillo, cuyo papel ha resultado vital, pues nos ha permitido conectar las máquinas virtuales a GNS3 a través de interfaces virtuales. Pero ¿por qué hemos usado estos y no otros? A continuación, haremos un pequeño repaso por los diferentes virtualizadores y simuladores de red, de forma que justifiquemos nuestra elección.

3.1 Máquinas Virtuales (VMs)

Una máquina virtual, por definición, es un contenedor de software aislado que puede ejecutar su propio sistema operativo como si fuera un ordenador físico. Estas máquinas virtuales utilizan recursos físicos como la CPU, la memoria ram, tarjeta gráfica o el disco duro del equipo donde corren. A través de la configuración, podemos establecer la cantidad de memoria, capacidad de procesamiento, etc. que puede utilizar el sistema operativo que corre dentro de la misma, llamado *sistema operativo huésped*. Este sistema operativo no es consciente de que corre en una VM por lo que podrá ejecutar cualquier programa o aplicación como si de un sistema operativo normal se tratase.

Una de las piezas centrales de la tecnologías de virtualización son los *hipervisores*, también llamado *Monitor de Máquina Virtual*(VMM). Estos son aplicaciones que presentan a los sistemas operativos huéspedes una plataforma virtual (hardware virtual), a la vez que le ocultan las características físicas reales del equipo sobre el que operan. Actualmente, podemos encontrar tres tipos de hipervisores[1][2]:

- *Hipervisores de tipo 1* o *nativos*. En ellos el hipervisor opera, directamente, sobre el hardware físico. Este hipervisor se carga antes que ninguno de los sistemas operativos huéspedes, y controla todos los accesos al hardware. Algunos de los más conocidos son: VMware ESX, Xen o Microsoft Hyper-V Server.

- *Hipervisores de tipo 2* o *hosted*. En ello el hipervisor opera sobre un sistema operativo completo, como si de una aplicación se tratase. Las máquinas virtuales se ejecutan en un tercer nivel por encima del hipervisor. Este es el modelo que vamos a necesitar para desarrollar el escenario, pues necesitamos tener varias VM con Sistema operativo Linux, funcionando sobre MAC OS X. Entre los hipervisores de tipo 2 más utilizados encontramos: VirtualBox, VMware en sus versiones Player, Fusion y Workstation, y QEMU entre otras.

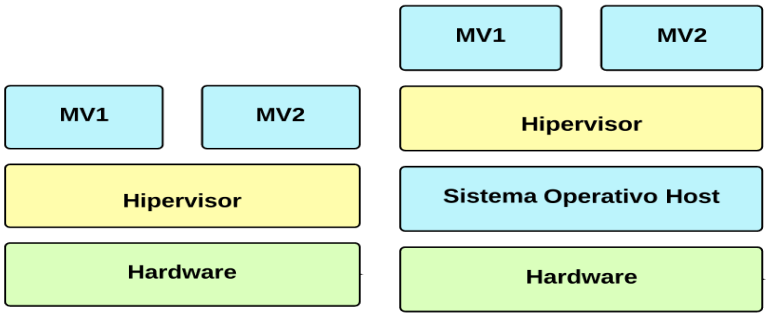


Figura 3-1. Arquitectura de Máquinas virtuales con Hipervisores de tipo 1 y 2

- *Hipervisores híbridos*: Este es una mezcla de los dos anteriores. Las VM se ejecutan por encima del hipervisor, pero también interactúan con el sistema operativo anfitrión. Aquí encontramos algunos ejemplos como Microsoft Virtual PC, Parallels o VMware Server.

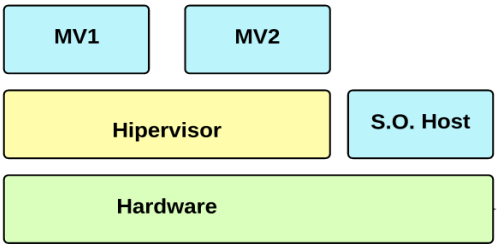


Figura 3-2. Arquitectura de Máquinas virtuales con Hipervisores híbridos

A continuación, vamos a presentar algunos de los entornos de virtualización más extendidos, con el objetivo de encontrar el que mejor se adapta a nuestras necesidades.

3.1.1 Parallels

Parallels Workstation es un software privado, de pago, disponible para Windows, MAC y Linux. Hay una versión más barata, llamada Parallels Desktop, sobre unos 70 euros. Actualmente es uno de los grandes competidores de VMware en MAC, pues ambos ofrecen una solución muy completa. Parallels destaca sobre VMware por su potencia de procesamiento y su rendimiento con los gráficos.

3.1.2 Q (QEMU)

Q puede usarse como emulador o como plataforma de virtualización y es una adaptación de QEMU para MAC OS X. Ambos son proyectos de código abiertos, y mientras que QEMU es una aplicación de línea de comandos, Q a desarrollado su propia interfaz gráfica para la configuración de las VM. Ofrece la capacidad de virtualizar una gran variedad de sistemas, aunque comparándolo con otros como Parallels o VMware, está más limitado en cuanto a características. QEMU es una gran solución para trabajar sobre Linux, e incluso Windows, pero con MAC aún tiene que mejorar.

3.1.3 VMware

VMware dispone de diferentes plataformas de virtualización, pero aquí, concretamente, nos referimos a VMware Fusion, que es la implementación para MAC OS X. VMware nos permite virtualizar sistemas con arquitecturas x86, como Windows, Linux o Solaris. Fusion fue el primer sistema de virtualización en ordenadores con procesadores Intel en salir al mercado, y actualmente es una de las opciones más extendidas, a la hora de usar Windows en MAC. Esta versión también es de pago, aunque su coste es ligeramente inferior al de Parallels Desktop.

En un principio nos decantamos por VMware Fusion 7 PRO, ya que, debido a un convenio con el departamento de ingeniería telemática, hemos podido tener acceso a un paquete de software de VMware de uso profesional de forma gratuita.

En este paquete encontramos otros programas como VMware Player 7 y VMware Workstation 11. Estos programas están ideados para operar sobre Linux y Windows. VMware Player es un producto gratuito de uso personal el que podemos virtualizar una gran variedad de sistemas. Si necesitas de un uso más profesional, con opciones más avanzadas encontramos VMware Workstation que incluye todas las características de VMware Player con la facilidad en la creación de VMs, y añade otras como la posibilidad de clonar VMs. Esta tecnología nos permite: ejecutar múltiples Sistemas operativos, simultáneamente, en el mismo ordenador, proporciona aislamiento de fallos y seguridad a nivel de hardware, flexibilidad y facilidad en los cambios, etc.

El problema es que la integración de las VMs con GNS3 no es trivial. Requiere una configuración complicada involucrando interfaces virtuales, redes virtuales e interfaces puente. Esto provocaba que algunos protocolos como SLAAC no funcionaran correctamente. El equipo de desarrollo de GNS3 se encuentra actualmente trabajando en la integración de VMware, pero hasta que llegue ese momento, decidímos que la mejor opción era trabajar con VirtualBox, el cual está perfectamente integrado.

3.1.4 Virtualbox

Es un software de virtualización para arquitecturas x86/amd64 desarrollado por Oracle. Se caracteriza por su versatilidad, pues está disponible para sistemas Linux, Windows y OS X. La aplicación, aunque sus inicios fue ofrecida bajo licencia de software privativo, se distribuye bajo licencia GPL 2, puesto que es un proyecto de código libre.

Virtualbox tiene una interfaz gráfica muy intuitiva, que hace que su manejo sea simple, por lo que requiere una curva de aprendizaje menor que otras de las opciones. Por tanto, puede ser una buena opción si no estas familiarizado con el mundo de la virtualización. Aunque tiene soporte para OS X, funciona especialmente bien en Linux y Windows. Actualmente se encuentra en la versión 5.0.

Nos decantamos por esta opción por dos motivos principales: es código libre y esta disponible de forma gratuita, y se integra de forma sencilla con GNS3

3.2 Simuladores de red

Tanto en el ámbito profesional, como en entornos académicos y de investigación los simuladores de red son herramientas ampliamente utilizadas. Los simuladores, en este caso, nos permiten estudiar el comportamiento de la red y aplicar los conocimientos teóricos adquiridos. Hoy en día, podemos encontrar una gran variedad de simuladores de red, como: NS-2, Omnet, NetSim, Packet Tracer, GNS3, etc. Entre ellos destacan Packet Tracer y GNS3 por tener interfaces más amigables, por su potencial y por ser unas de las principales herramientas para la preparación de los certificados oficiales de Cisco[3].

3.2.1 Cisco Packet Tracer

Packet Tracer (PT) es una herramienta que permite a los usuarios crear diversas topologías de red mediante una interfaz muy intuitiva y con una curva de aprendizaje inferior a la mayoría de simuladores. PT soporta un gran número de equipos (no necesariamente de Cisco) como Hosts, routers, switches, firewalls, puntos de acceso, etc.

Al ser un simulador, y no un emulador, soporta escenarios con un gran número de dispositivos simultáneos, ya que el consumo de los mismos es menor. Los dispositivos son altamente configurables, ya sea por interfaz gráfica, o por la interfaz de línea de comandos (CLI). Permite trabajar con productos comerciales, comportándose de manera muy parecida a como lo harían los dispositivos reales. Además permite hacer diferentes pruebas de conectividad para probar la correcta configuración de la red, y nos permite generar ciertos tipos de tráfico para comprobar como funciona la misma.

Puedes elegir entre varios modos de funcionamiento, como *tiempo real* o *simulación*. El modo simulación nos permite seguir paso por paso, el flujo de paquetes entre los nodos de la topología, por lo que tiene un gran valor como herramienta didáctica. Por tanto, PT nos permite ir desde las topologías más simples, hasta redes muy complejas, convirtiéndose así en una herramienta fundamental en el estudio de protocolos y arquitecturas de redes a diferentes niveles[4].

Pero con PT no todo son ventajas. Su licencia de uso está restringida a aquellas personas registradas en la *Cisco Networking Academy Program*. Además, PT simula los dispositivos de Cisco con gran fidelidad en la mayoría de los casos, pero en algunos casos no tiene soporte para los comandos y funcionalidades más avanzados. En nuestro caso necesitamos, o bien, que los hosts virtuales tengan soporte para movilidad (no sería necesario VirtualBox), cosa que no es posible, o que PT pudiera interactuar con elementos externos al simulador. Es por ello que encontramos una posible alternativa en el simulador GNS3.

3.2.2 GNS3

Es un potente simulador y emulador gráfico de redes disponible de forma gratuita, ya que es de código libre. Es un software multiplataforma, disponible en MAC OS X, Windows y Linux, que puede utilizarse para trabajar con redes complejas. GNS3 tiene varias ventajas con respecto a PT. Una de las principales, y que más valor le aporta, es que permite emular equipos reales de fabricantes, como Cisco o Juniper, mediante la carga de imágenes de sus IOS. De esta forma, tienes acceso a todas las funcionalidades y opciones de configuración de los dispositivos, llegando a esas características que PT no cubre. En este sentido hay que remarcar que, aunque el software es libre y esta abiertamente disponible, este no incorpora las imágenes de los IOS de los dispositivos que se quieren emular[5].

Otra de las principales características de GNS3, y que ha hecho que nos decantemos, finalmente, por él, es que permite comunicar las redes simuladas con redes físicas, otros PC o máquinas virtuales. Está integrado con el capturador de paquetes Wireshark, lo que nos capacita para capturar los paquetes que pasan por los enlaces virtuales y nos da un mayor conocimiento de lo que ocurre en nuestra topología[6].

GNS3 tiene algunas desventajas notables:

- Sólo puede emular routers. No es posible emular otros dispositivos como *switches*, por lo que no es apropiado para el estudio de conceptos como *Spanning-tree*.
- Es muy pobre en términos de consumo de recursos. Al estar emulando las imágenes de los IOS de routers reales, tiende a aumentar su consumo de recursos notablemente. Por tanto, no debemos tener en nuestra topología un número elevado de routers.

Esto no nos supone un problema en nuestro caso, pues el escenario no contiene más de 3 routers emulados. Hay que tener en cuenta que el software del emulador, llamado *Dynamips*, necesita una pequeña configuración, pues en un primer momento tiende a abarcar todos los recursos posibles del PC. Esto lo veremos en el siguiente capítulo.

3.3 TunTap para OS X

Este proyecto de código libre proporciona una extensión del kernel para Mac OS X que permite crear interfaces de red virtuales. Desde el punto de vista del S.O. el comportamiento de estas interfaces es similar al de un adaptador de red físico. Estas interfaces son utilizadas por los programas que corren en el S.O.. Este software permite crear dos tipos de interfaces diferentes[7]:

- *Network TUNnel* (TUN). Simulan un dispositivo de nivel de red y operan en capa 3.
- *Network TAP* (TAP). Simulan un dispositivo de nivel de enlace y operan en capa 2.

Funciona creando un conjunto de dispositivos */dev/tunX* y */dev/tapX*, siendo X un número entre 0 y 15 que es el número máximo de interfaces virtuales soportadas en nuestro caso. Cuando una aplicación abre una interfaz virtual *(/dev/tap7*) esta se activa automáticamente en el sistema con el nombre correspondiente (*tap7*). Una vez configuradas, estas interfaces actúan como una interfaz real.

4 DESPLIEGUE DEL ESCENARIO PARA MIPV6

En este apartado explicaremos los pasos a seguir para montar un escenario con los elementos necesarios para realizar pruebas de movilidad IPv6. En primer lugar, mostraremos la configuración y distribución de la red, cuyo núcleo principal residirá en el emulador GNS3. Crearemos, con VirtualBox, varias máquinas virtuales que nos permitirán simular el comportamiento de los diferentes elementos que intervienen en el proceso. Y mostraremos como conectar estas máquinas con GNS3 a través de interfaces virtuales de nivel tres con el software TunTap. El objetivo es establecer el laboratorio de pruebas que nos permitirá realizar un análisis de rendimiento, en el próximo capítulo.

4.1 Configuración de GNS3

Como hemos visto en el capítulo anterior, GNS3 no es un simulador, sino un emulador. Nos permite instalar un router CISCO, por ejemplo, haciendo que el comportamiento de este sea más realista, y dándonos acceso a todas sus funcionalidades. Por tanto, necesitaremos una imagen de router para poder emularlo. En este caso hemos recurrido al c3725, ya que dispone de los requisitos técnicos que necesitamos, que tenga soporte para movilidad y IPv6.

Para instalar la imagen, descomprimimos la imagen IOS dentro del fichero correspondiente de GNS3.

```
unzip –p c3725-adventerprisek9-mz.124-15.T14.bin > $HOME/GNS3/Images/
c3725-adventerprisek9-mz.124-15.T14.image
```

A continuación, entramos en GNS3 creando un proyecto nuevo. Nos vamos al apartado de preferencias y en la pestaña Dynamips -> IOS routers indicamos que queremos emular un nuevo dispositivo, indicando la ruta a la imagen que hemos descomprimido anteriormente, y algunas características físicas del dispositivo.

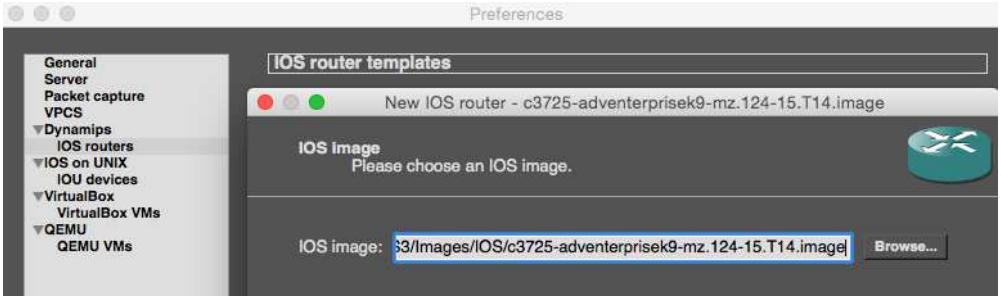


Figura 4-1. Declaración de una nueva imagen en GNS3

Ahora podemos encontrar nuestro router en la sección de dispositivos disponibles, por lo que podemos proceder al montaje del escenario.

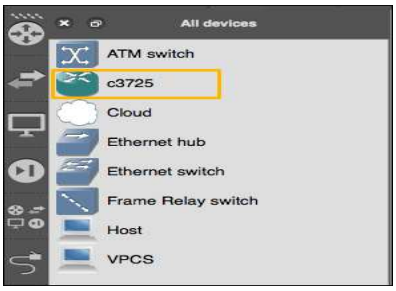


Figura 4-2. Cisco IOS instalada y disponible

Para desplegar el escenario necesitamos que estén presentes, como mínimo cada una de las entidades funcionales fundamentales para MIPv6. Aunque, GNS3 dispone de "Hosts virtuales" que nos permiten simular los equipos finales, como pueden ser el CN o el MN, estos no tiene soporte para movilidad y solo disponen de algunas funcionalidades básicas, como envió de PINGs. Es por esto, que los elementos principales que intervienen en el proceso de movilidad son simulados con máquinas virtuales y conectados a GNS3, como explicaremos posteriormente.

Además de estos dispositivos, necesitamos routers y conmutadores de red (switch), para dar forma a la topología de red que buscamos y que se muestra a continuación:

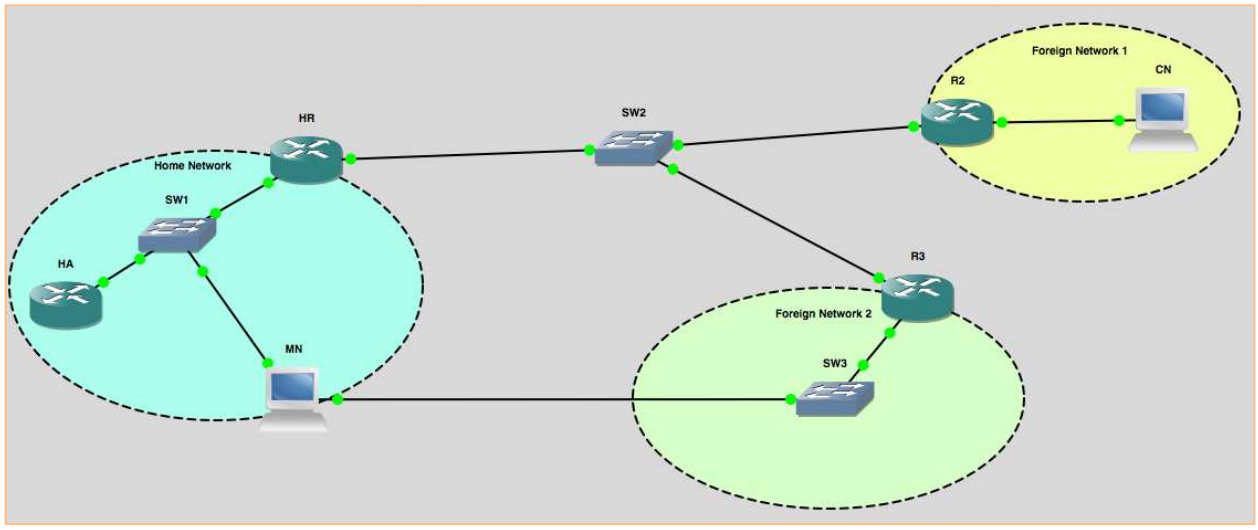


Figura 4-3. Escenario MIPv6 en GNS3

En este escenario, hemos optado por usar un router por cada red. Si estuviéramos montando un escenario físico, seguramente, habríamos optado por montar un único router que creara una *Virtual Local Area* network (VLAN) distinta por cada interfaz, para ahorrar recursos. Pero aprovechando que contamos con un emulador creemos más conveniente montar una red más parecida a lo que pueda ser un escenario real.

En este caso, necesitamos cuatro redes:

- Una red central, que sirva de conexión entre los tres routers de las redes periféricas sirviéndose de un switch.
- Una *Home Network*, como red más importante en este escenario, pues es la red de la cual parte el *Mobile Node* y dónde se encuentra el *Home Agent*.
- Dos *Foreign Networks*, una de ellas será la red en la que se encuentre el *Correspondant Node*, y la otra, será a la que migrará el MN una vez producido el traspaso. A efectos prácticos, podríamos a ver usado una única FN, pero de esta forma se asemeja más a una situación real.

La distribución de los rangos de IP se recogen en la siguiente tabla.

Tabla 4-1. Resumen de las IPv6 de las redes del escenario

Red	Rango de IPv6
Central Network	2001:db8:1111:4: :/64
Home Network	2001:db8:1111:1: :/64
Foreign Network 1 (CN)	2001:db8:1111:2: :/64
Foreign Network 2	2001:db8:1111:3: :/64

Ahora veremos como llevar esta configuración a los routers del escenario. Pero antes tenemos que hacer un pequeño ajuste. Cuando estamos simulando una red en GNS3, es muy probable que este tienda a consumir el 100% del uso de la CPU. Esto es debido a que Dynamips (el emulador) no sabe cuando el router virtual está ocioso o cuando está activo. Para solucionar este problema necesitamos asignar un valor a *Idle-PC*, con este valor se pretende calcular el tiempo en el que la imagen IOS del router se encuentra ociosa y pone el router en *sleep mode*. Antes había que calcular este valor de forma manual, lo que provocaba muchos dolores de cabeza. Ahora GNS3 incluye la opción *Auto Idle-PC* que permite calcularlo de forma automática. A continuación, mostramos la configuración del router de la *Home Network*.

Código 4-1. Configuración del Router.

```
HR# configure terminal

// Configuramos la interfaz de la Home Network
HR(config)# interfaz f0/0

// Activamos IPv6 y le asignamos una IP
HR(config-if)# ipv6 enable
HR(config-if)# ipv6 address 2001:db8:1111:1::1/64

// Configuramos los datos necesarios para el protocolo ND
// Definimos el prefijo que es anunciado y configuramos el intervalo
// de envío de Router Advertisement
HR(config-if)# ipv6 nd prefix 2001:DB8:1111:1::/64
HR(config-if)# ipv6 nd advertisement-interval
HR(config-if)# ipv6 nd ra interval msec 1000

// Activamos la interfaz
HR(config-if)# no shut
HR(config-if)# exit

// Realizamos el mismo proceso con la interfaz de la red central

HR(config)# interfaz f0/1
```

```
// Activamos IPv6 y le asignamos una IP
HR(config-if)# ipv6 enable
HR(config-if)# ipv6 address 2001:db8:1111:4::1/64

// Configuramos los datos necesarios para el protocolo ND
// Definimos el prefijo que es anunciado y configuramos el intervalo
// de envío de Router Advertisement
HR(config-if)# ipv6 nd prefix 2001:DB8:1111:4::/64
HR(config-if)# ipv6 nd advertisement-interval
HR(config-if)# ipv6 nd ra interval msec 5000

// Activamos la interfaz
HR(config-if)# no shut
HR(config-if)# exit
```

Una de las formas que tiene un MN de darse cuenta que ha cambiado de red, es la recepción de un RA de un router con prefijo diferente al de su subred. Con el fin de acelerar este proceso, los intervalos de envío de RA en la HN y en la FN son inferiores al de la red central, donde es indiferente.

Al estar las rutas definidas de forma estática y no haber ningún protocolo de enrutamiento implementado, debemos incluir rutas por defecto a las subredes a las que no está directamente conectado el router.

```
HR(config)# ipv6 route 2001:DB8:1111:2::/64 2001:DB8:1111:4::2
HR(config)# ipv6 route 2001:DB8:1111:3::/64 2001:DB8:1111:4::3
```

El reenvío de datagramas IPv6 está desactivado por defecto, por lo que necesitamos activarlos con el siguiente comando.

```
HR(config)# ipv6 unicast-routing
HR(config)# exit

// Guardamos la configuración actual
HR# write
```
[7]

Repetimos el mismo proceso en los routers R2 y R3 cambiando las IPv6 que correspondan en cada caso. Para comprobar que la configuración se ha llevado a cabo realizamos diferentes pings a las distintas interfaces. También podemos correr el siguiente comando y analizar la configuración de cada interfaz.

```
HR# show ipv6 interface f0/0
FastEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::C001:5FF:FE48:0
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8:1111:1::1, subnet is 2001:DB8:1111:1::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF00:1
    FF02::1:FF48:0
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
```

[7] Aunque GNS3 tiene una opción de guardar, esta solo guarda la disposición de los elementos en la topología y sus conexiones. La configuración de los routers hay que guardarla individualmente de la forma indicada.

```
ICMP redirects are enabled
ICMP unreachables are sent
ND DAD is enabled, number of DAD attempts: 1
ND reachable time is 30000 milliseconds
ND advertised reachable time is 0 milliseconds
ND advertised retransmit interval is 0 milliseconds
ND router advertisements are sent every 1000 milliseconds
ND router advertisements live for 1800 seconds
ND advertised default router preference is Medium
Hosts use stateless autoconfig for addresses.
```

Aquí podemos observar que la configuración se ajusta a la deseada. Podemos ver alguna información útil como la dirección IPv6 local al enlace, los grupos multicast a los que está subscrito, y que permite la autoconfiguración de la IPv6 de los hosts.

4.2 Instalación del software de UMIP en las Máquinas virtuales

El software de movilidad debe estar presente en las tres entidades principales: HA, MN y CN. En un principio, al lector le puede resultar extraño que el CN necesita soporte para movilidad. En efecto, no es necesario para un escenario de movilidad básico, pero si queremos implementar *Optimización de rutas* necesitamos configurar UMIP también el CN. De esta forma el CN puede intercambiar mensajes de movilidad con el MN.

4.2.1 Crear Máquinas virtuales

El proceso es muy sencillo, pero cabe destacar lo siguiente. Para instalar el software de movilidad, necesitamos recompilar el kernel para adaptarlo a las necesidades de UMIP. Como este es un proceso muy tedioso, vamos a realizarlo una única vez sobre la primera máquina virtual. Una vez completado el proceso, crearemos las otras dos VM a partir de la imagen de la primera, mediante una clonación.

Figura 4-4. Crear una máquina virtual con VitualBox

Indicamos que queremos crear una nueva máquina y rellenamos los datos principales, como el nombre, el tipo o la capacidad asignada. Una vez creada, seleccionamos la imagen del sistema operativo que queremos instalar y que previamente nos hemos descargado. En este caso '*debian-7.8.0-i386-netinst.iso*'. Hemos seleccionado Debian como sistema operativo debido a que UMIP solo contiene paquetes compatibles con este.

4.2.2 Preparación del kernel

Como hemos comentado anteriormente, necesitamos una recompilación del kernel para poder correr el software de movilidad. Vamos a proceder a instalar una versión del kernel superior a las 3.8.1, ya que, a partir

de esta, los kernels tienen soporte para movilidad[1].

Para ello debemos hemos seguido los pasos que se detallan a continuación. En primer lugar, nos descargamos el nuevo kernel de kernel.org[8]:

```
# cd /usr/src/
# wget http:/www.kernel.org/pub/linux/kernel/v3.x/linux-3.8.2.tar.bz2
```

Descomprimimos la fuente y creamos un enlace simbólico a las fuentes del kernel

```
# tar xf linux-3.8.2.tar
# ln -s /usr/src/linux-3.8.2/ /usr/src/linux
# cd linux-3.8.2/
```

A continuación, debemos editar la configuración del nuevo kernel para activar el soporte de movilidad. Podemos reusar la antigua configuración del kernel y luego añadir las opciones necesarias. Para hacer esto necesitamos instalar la librería *libncurses5-dev*.

```
# apt-get install libncurses5-dev
# make oldconfig
# make menuconfig
```

Para activar el soporte de movilidad, debemos activar en la configuración del kernel, las siguientes opciones:

Código 4-2. Opciones del Kernel.

```
General setup
--> Prompt for development and/or incomplete code/drivers
[CONFIG_EXPERIMENTAL]
--> System V IPC [CONFIG_SYSVIPC]

Networking support [CONFIG_NET]
--> Networking options
    --> Transformation user configuration interface[CONFIG_XFRM_USER]
    --> Transformation sub policy support [CONFIG_XFRM_SUB_POLICY]
    --> Transformation migrate database [CONFIG_XFRM_MIGRATE]
    --> PF_KEY sockets [CONFIG_NET_KEY]
    --> PF_KEY MIGRATE [CONFIG_NET_KEY_MIGRATE]
    --> TCP/IP networking [CONFIG_INET]
    --> The IPv6 protocol [CONFIG_IPV6]
        --> IPv6: AH transformation [CONFIG_INET6_AH]
        --> IPv6: ESP transformation [CONFIG_INET6_ESP]
        --> IPv6: IPComp transformation [CONFIG_INET6_IPCOMP]
        --> IPv6: Mobility [CONFIG_IPV6_MIP6]
        --> IPv6: IPsec transport mode
 [CONFIG_INET6_XFRM_MODE_TRANSPORT]
        --> IPv6: IPsec tunnel mode [CONFIG_INET6_XFRM_MODE_TUNNEL]
        --> IPv6: MIPv6 route optimization mode
 [CONFIG_INET6_XFRM_MODE_ROUTEOPTIMIZATION]
        --> IPv6: IP-in-IPv6 tunnel (RFC2473) [CONFIG_IPV6_TUNNEL]
        --> IPv6: Multiple Routing Tables
```

[8] Se ha optado por la versión del kernel 3.8.2 porque, aunque ya es algo antigua, esta recomendada desde el proyecto UMIP, pues asegura su compatibilidad.

```
  [CONFIG_IPV6_MULTIPLE_TABLES]
         --> IPv6: source address based routing [CONFIG_IPV6_SUBTREES]

 File systems
 --> Pseudo filesystems
     --> /proc file system support [CONFIG_PROC_FS]
```

Una vez hemos terminado con este proceso procedemos a finalizar con la compilación e instalación del nuevo kernel

```
 # make
 # make modules_install
 # make install
 # make headers_install
```

Este proceso puede ser bastante largo. Una vez completado, reiniciamos la VM y en el menú de arranque seleccionamos el nuevo kernel.

4.2.3 Instalación de UMIP

Si hemos seguido correctamente los pasos anteriores, estamos en situación de instalar el software de UMIP. Para ello necesitamos contar con los siguientes paquetes instalados en nuestro equipo.

```
 # apt-get install autoconf automake bison flex libssl-dev indent
ipsec-tools radvd
```

En principio, *radvd* solo será necesario en el HA. Ahora, bajamos el código fuente del repositorio oficial de UMIP en GitHub.

```
 # cd /usr/src/
 # git clone git//git.umip.org/umip.git
```

Terminamos con la compilación e instalación.

```
 # cd umip/
 # autoreconf -i
 # CPPFLAGS='-isystem /usr/src/linux/usr/include/' ./configure
--enable-vt
 # make
 # make install
```

CPPFLAGS se usa para indicar el lugar donde se encuentran las cabeceras del kernel. La opción *--enable-vt* nos permite disponer de un terminal virtual donde podemos ver en tiempo real la información de la *binding cache* en el HA y la *binding update list*.

4.3 Configuración del HA

En este apartado vamos a proceder a configurar el HA, para lo cual, nos vamos a guiar por la documentación oficial de UMIP. Para comenzar, vamos crear el fichero de configuración */usr/local/etc/mip6d.*conf con el siguiente contenido[2].

Código 4-3. Fichero de configuración del HA

```
# Sample UMIP configuration file for a MIPv6 Mobile Node
NodeConfig HA;

# Set DebugLevel to 0 if you do not want debug messages
DebugLevel 10;

# Replace eth0 with the interface connected to the home link
Interface "eth0";


# Binding information
BindingAclPolicy 2001:db8:1111:1::2 allow;
DefaultBindingAclPolicy deny;

# Enable IPsec static keying
UseMnHaIPsec disabled;
KeyMngMobCapability disabled;

# IPsec Security Policies information
IPsecPolicySet {
    HomeAgentAddress 2001:db8:1111:1::3;
    HomeAddress 2001:db8:1111:1::2/64;
    # All MH packets (BU/BA/BERR)
    IPsecPolicy Mh UseESP 11 12;
    # All tunneled packets (HoTI/HoT, payload)
    IPsecPolicy TunnelPayload UseESP 13 14;
    # All ICMP packets (MPS/MPA, ICMPv6)
    IPsecPolicy ICMP UseESP 15 16;
}
```

Vamos a explicar qué pretendemos hacer con esta configuración. Lo primero que tenemos que indicar es el nodo que estamos configurando con *NodeConfig* y el nivel de detalle que vamos a requerir en los logs, que en principio, vamos a poner siempre al máximo.

Debemos indicar expresamente cual es la interfaz que está conectada a la HN (eth0) y en la que se aplicará esta configuración. UMIP se sirve de *Access Control List* (ACL) para mejorar la seguridad del proceso. Para que sólo el MN pueda intercambiar BU y BA con el HA, necesitamos que, por defecto, niegue todas las peticiones usando el comando *DefaultBindingAclPolicy deny* e indicarle expresamente que acepte los Bindings que provengan de la IPv6 del MN con *BindingAclPolicy 2001:db8:1111:1: :2 allow*.

El ultimo bloque se usa para configurar las opciones de IPsec. La opción *UseMnHaIPsec* nos permite activar o desactivar la protección de los paquetes de señalización y datos intercambiados entre HA y MN. En esta primera prueba, esta opción estará desactivada para poder analizar en detalle el intercambio de mensajes en MIPv6.

El bloque de *IPsecPolicySet* se encarga de proteger el tráfico IPsec entre el MN y el HA. Las direcciones IPv6 del HA (*HomeAgentAddress*) y la HoA del MN (*HomeAddress*) es lo primero que debemos especificar. A continuación, pedimos protección IPsec usando ESP para

- Señalización entre MN y HA, como por ejemplo mensajes BU, BA y BE (*IPsecPolicy Mh UseESP 11 12*).
- Tráfico de datos a través de un túnel entre el MN y HA, y mensajes *Home Address Test Init* y

Home Address Test que se usan en el proceso de optimización de rutas (*IPsecPolicy TunnelPayload UseESP 13 14*).

- Tráfico ICMP entre Ha y MN (*IPsecPolicy ICMP UseESP 15 16*).

Con estas reglas cubrimos todo el tráfico, de datos y señalización, entre el MN y el HA. UMIP usará estas opciones para desplegar un conjunto de políticas de seguridad para lo que necesita de la presencia de *Security Associations* (SA).

Estas asociaciones son definidas en el fichero */usr/local/etc/setkey.conf* el cual tiene el siguiente contenido:

Código 4-4. Configuración del fichero setkey.conf

```
# IPsec Security Associations
# HA address: 2001:db8:1111:1::3;
# MR HoAs: 2001:db8:1111:1::2/64;

# Flush the SAD and SPD
flush;
spdflush;

# MN1 -> HA transport SA for BU
add 2001:db8:ffff:0::2 2001:db8:ffff:0::1000 esp 0x11
    -u 11
    -m transport
    -E 3des-cbc "MIP6-011--12345678901234"
    -A hmac-sha1 "MIP6-011--1234567890" ;

# HA -> MN1 transport SA for BA
add 2001:db8:1111:1::3 2001:db8:1111:1::2 esp 0x12
    -u 12
    -m transport
    -E 3des-cbc "MIP6-012--12345678901234"
    -A hmac-sha1 "MIP6-012--1234567890" ;

# MN1 -> HA tunnel SA for any traffic
add 2001:db8:1111:1::2 2001:db8:1111:1::3 esp 0x13
    -u 13
    -m tunnel
    -E 3des-cbc "MIP6-013--12345678901234"
    -A hmac-sha1 "MIP6-013--1234567890" ;

# HA -> MN1 tunnel SA for any traffic
add 2001:db8:1111:1::3 2001:db8:1111:1::2 esp 0x14
    -u 14
    -m tunnel
    -E 3des-cbc "MIP6-014--12345678901234"
    -A hmac-sha1 "MIP6-014--1234567890" ;

# MN1 -> HA transport SA for ICMP (including MPS/MPA)
add 2001:db8:1111:1::2 2001:db8:1111:1::3 esp 0x15
    -u 15
    -m transport
    -E 3des-cbc "MIP6-015--12345678901234"
    -A hmac-sha1 "MIP6-015--1234567890" ;
```

```
# HA -> MN1 transport SA for ICMP (including MPS/MPA)
add 2001:db8:1111:1::3 2001:db8:1111:1::2 esp 0x16
    -u 16
    -m transport
    -E 3des-cbc "MIP6-016--12345678901234"
    -A hmac-sha1 "MIP6-016--1234567890" ;
```

El HA debe anunciar en la HN que él es el *Home Agent* de esa red. Por tanto, el HA debe actuar como un router y mandar RA periódicos indicándolo. Esto se hace poniendo el bit H a 1.

Como router, el HA necesita poder reenviar información. Para ello, podemos cambiar el contenido del fichero */proc/sys/net/ipv6/conf/all/forwarding* a 0. Si queremos activar esta opción directamente en el inicio, debemos descomentar la línea *net.ipv6.conf.all.forwarding=1* del fichero */etc/sysctl.conf*.

Con el fin de que el HA pueda mandar RAs con la información necesaria, debemos configurara el software *radvd* que instalamos anteriormente. Este programa nos permite mandar RAs periódicos con información como el prefijo de red.

El fichero */etc/radvd.conf*, que contiene la configuración de *radvd* tendrá el siguiente cuerpo:

Código 4-5. Configuración del fichero radvd.conf

```
# Home Agent radvd configuration file
# Replace eth0 with the interface connected to the home link
interface eth0
{
    AdvSendAdvert on;
    MaxRtrAdvInterval 3;
    MinRtrAdvInterval 1;
    AdvIntervalOpt on;
    AdvHomeAgentFlag on;
    AdvHomeAgentInfo on;
    HomeAgentLifetime 1800;
    HomeAgentPreference 10;

    # Home Agent address
    prefix 2001:db8:ffff:0::1000/64
    {
        AdvRouterAddr on;
        AdvOnLink on;
        AdvAutonomous on;
    };
};
```

Observamos como, a parte del prefijo de la red, también contiene la información necesaria para la gestión de la movilidad (*AdvHomeAgentFlag* y *AdvHomeAgentInfo*). Para arrancar este servicio ejecutamos: *radvd –C /etc/radvd.conf*. De igual forma, este servicio se ejecutará al inicio debido al script que se encuentra en */etc/init.d/*.

El lector se habrá podido percatar de que en la HN, tanto el router que da salida al exterior de la red, como el HA, están emitiendo RA periódicamente anunciando el prefijo de red. A primera vista, puede parecer que esto va a crear algún tipo de conflicto. Sin embargo, todo funciona con normalidad y protocolos como SLAAC no se ven afectados.

4.4 Configuración del MN

En esta sección detallaremos la configuración del Nodo Móvil. La máquina virtual que actúa como MN estará conectada a dos redes diferentes, entre las que conmutará en el proceso de traspaso. Por la interfaz *eth0* se conecta a la HN y por la *eth1* se conecta a una FN. Esta máquina es un clon de la anterior por lo que ya cuenta con el software necesario para soportar MIPv6 solo necesitamos introducir la configuración adecuada[2]. Para empezar, crearemos el fichero */usr/local/etc/mip6d.conf* con el siguiente código:

Código 4-6. Fichero de configuración del MN

```
# Sample UMIP configuration file for a MIPv6 Mobile Node
NodeConfig MN;

# Set DebugLevel to 0 if you do not want debug messages
DebugLevel 10;

# Enable the optimistic handovers
OptimisticHandoff enabled;

# Disable RO with other MNs (it is not compatible
# with IPsec Tunnel Payload)
DoRouteOptimizationMN disabled;

# The Binding Lifetime (in sec.)
MnMaxHaBindingLife 60;

# List here the interfaces that you will use
# on your mobile node. The available one with
# the smallest preference number will be used.
Interface "eth0" {
    MnIfPreference 1;
}
Interface "eth1" {
    MnIfPreference 2;
}

# Replace eth0 with one of your interface used on
# your mobile node
MnHomeLink "eth0" {
    HomeAgentAddress 2001:db8:1111:1::3;
    HomeAddress 2001:db8:1111:1::2/64;
}

# Enable IPsec static keying
UseMnHaIPsec disabled;
KeyMngMobCapability disabled;

# IPsec Security Policies information
IPsecPolicySet {
    HomeAgentAddress 2001:db8:1111:1::3;
    HomeAddress 2001:db8:1111:1::2/64;

    # All MH packets (BU/BA/BERR)
    IPsecPolicy Mh UseESP 11 12;
```

```
    # All tunneled packets (HoTI/HoT, payload)
    IPsecPolicy TunnelPayload UseESP 13 14;
    # All ICMP packets (MPS/MPA, ICMPv6)
    IPsecPolicy ICMP UseESP 15 16;
}
```

Este fichero de configuración es muy similar al anterior, a excepción de algunos campos que comentaremos a continuación. El parámetro *OptimisticHandoff* se encuentra activado, haciendo que el traspaso sea más rápido. Esto es así, porque permite al MN empezar a enviar datos por el túnel en cuanto manda el primer BU, sin necesidad de esperar al BA.

En un principio, el parámetro *DoRouteOptimizationMN* va a permanecer desactivado. Este sirve para activar y desactivar el modo de optimización de rutas, que se explicó en el apartado 2.3. Lo activaremos posteriormente para hacer pruebas de rendimiento, pero al ser incompatible con IPsec, lo mantendremos desactivado por el momento. A continuación, le asignamos un valor de preferencia a las interfaces del MN, siendo la preferencia más alta cuanto más bajo es el número asignado. Esto es útil para, en caso de llegar a una nueva red, el MN intenta comunicarse por la interfaz preferida. Sino estuviera disponible, lo haría por la siguiente en la lista de preferencias, que en nuestro caso es *eth1*.

Por último, indicamos la dirección HoA del MN y la dirección del HA, y configuramos las opciones relacionadas con IPsec, que en principio estará desactivado. Los parámetros son exactamente los mismos que se explicaron al analizar la configuración del HA. Para completar la configuración de IPsec, al igual que pasaba en el HA, debemos declarar las *Security Associations* (SA). Estas, son exactamente iguales en ambos nodos, por lo que copiamos el mismo fichero *setkey.conf* del HA en la ruta */usr/local/etc/setkey.conf* del MN.

4.5 Configuración del CN

La configuración de este equipo es mínima. Para poder ejecutar el proceso de optimización de rutas necesitamos que este nodo tenga soporte para movilidad. Es por ello, que está máquina ha sido clonada de la primera, y contiene el nuevo kernel con soporte para MIPv6. En los primeros escenarios no requeriremos ninguna configuración especial de este equipo, más que el software necesario para medir el rendimiento del tráfico TCP/UDP.

Cuando realicemos las pruebas con optimización de rutas necesitaremos activar el software de movilidad usando un pequeño archivo de configuración como este:

Código 4-7. Fichero de configuración del CN

```
# Sample UMIP configuration file for a MIPv6 Mobile Node
NodeConfig CN;

# Disable RO with other MNs (it is not compatible
# with IPsec Tunnel Payload)
DoRouteOptimizationCN enabled;
```

Con esto damos por acabado la configuración del software de movilidad UMIP en los nodos esenciales del escenario.

4.6 Conexión entre las VMs y GNS3

Por un lado tenemos una red creada en GNS3 con todos los elementos necesarios, y por otro, tenemos tres máquinas virtuales configuradas para actuar como MN, HA y CN. Ahora vamos a explicar los pasos a seguir

para conseguir que estos nodos se puedan comunicar a través de la red creada en el emulador. Para ello, nos serviremos del software TunTap para Mac OS X que detallamos en el capítulo 3.

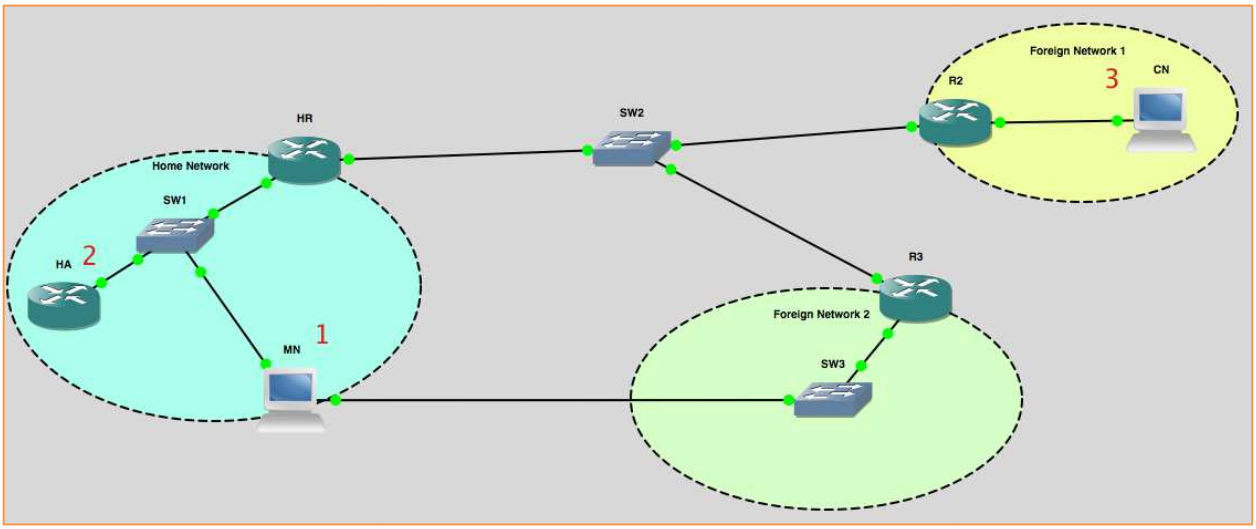


Figura 4-5. VMs en escenario MIPv6 en GNS3

Los elementos señalados con números, se corresponden con nuestras máquinas virtuales. Siendo, lo mostrado en la imagen, el resultado final que buscamos en el montaje de nuestro laboratorio de pruebas.

Para conectar, por ejemplo, el CN al emulador debemos seguir los siguientes pasos:

1. Activamos en la configuración del Host de GNS3 que actuará como CN, una interfaz TAP. Como ya explicamos, estas son interfaces virtuales de nivel de enlace (capa 2), no necesitan configuración IP y actuarán como puente entre la VM y GNS3.

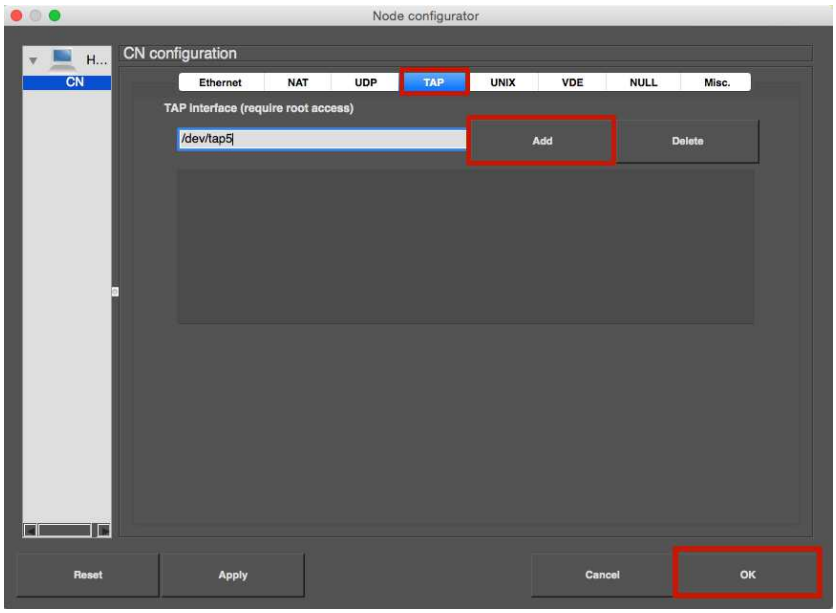


Figura 4-6. Activación de la interfaz virtual

Para activarla, la añadimos a la lista de interfaces del Host y, automáticamente, esta interfaz se activa en nuestro ordenador. Si acudimos al comando ifconfig, podemos comprobar que la interfaz tap5 se encuentra disponible.

```
tap5: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        ether d6:0d:2b:60:a0:ab
        media: autoselect
        status: active
        open (pid 30195)
```

Figura 4-7. Interfaz tap5 activa en el S.O. host

2. Conectamos la figura del CN en GNS3 con el router de su red. Podemos comprobar en la imagen que el host virtual se conecta a través de la interfaz tap5 creada en el paso anterior.

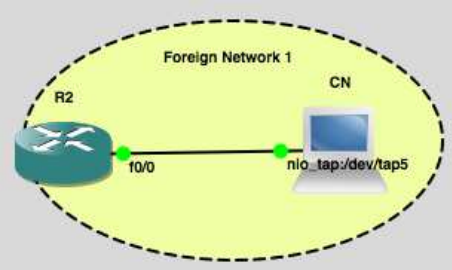


Figura 4-8. Conexión del host virtual a tap5

3. Ahora que tenemos la interfaz activa, y que hemos conectado el host virtual a la red mediante la misma, solo nos falta conectar la máquina virtual a la interfaz creada. Así pues, nos vamos a la configuración del CN en VirtualBox y en el apartado de red, ponemos el adaptador de red en modo puente y lo asociamos a la interfaz dada.

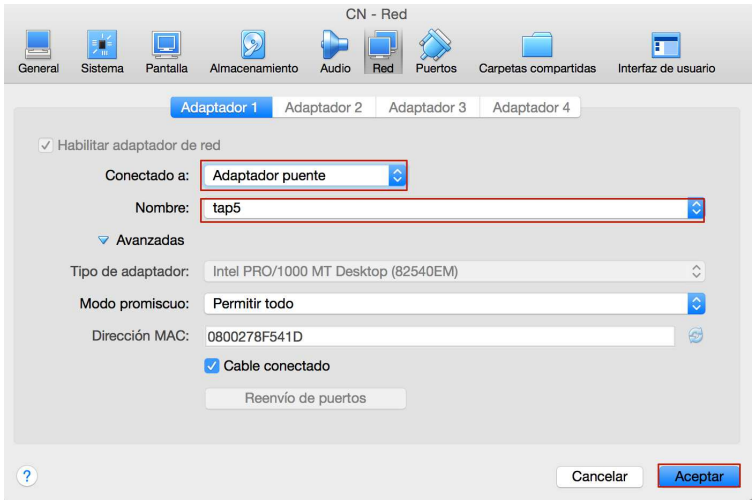


Figura 4-9. Conexión de la VM a tap5

Con esto, nuestro CN puede comunicarse con cualquier nodo dentro e la red emulada en GNS3. Actuamos de igual forma con el MN y el HA eligiendo diferentes interfaces virtuales y con la particularidad de que el MN necesita configurar dos adaptadores de red distintos.

El MN tiene un adaptador conectado a la HN y otro a la FN, esto es debido a la forma en la que ejecuta el traspaso. En un primer momento, la interfaz *eth0* (HN) se encuentra activa y la *eth1* (FN) apagada. Así, el MN se encuentra la HN. Cuando se realiza el traspaso, se enciende *eth1* y se apaga *eth0* pasando el MN a estar dentro de la FN. Aclarado este aspecto, ya podemos pasar a la sección de análisis.

5 ANÁLISIS DE MIPV6

Nos encontramos ante uno de los capítulos principales de este trabajo. En este apartado vamos a poner en práctica todos los conocimientos teóricos que hemos ido desarrollando a lo largo de esta memoria. Para ello, después de seguir los pasos del capítulo anterior, contamos con un laboratorio apto para pruebas de MIPv6. Vamos a implementar tres configuraciones diferentes:

1. Escenario básico
2. Escenario con IPsec
3. Escenario con optimización de rutas y sin IPsec [9] [10]

En primer lugar, vamos a realizar unas pruebas simples con el fin de mostrar el funcionamiento y las peculiaridades de los citados escenarios. Usaremos estas pruebas para aplicar los conocimientos teóricos que hemos tratado anteriormente sobre MIPv6.

Posteriormente probaremos el rendimiento de los escenarios, realizando una comparativa entre los tres. Aplicaremos distintos tipos de tráfico y estudiaremos la respuesta de los mismos, en cuanto a tiempos de transmisión, perdidas de paquetes, etc. En definitiva, compararemos el rendimiento de las diferentes configuraciones.

5.1 Estudio y Comparación del funcionamiento

En este apartado, buscamos estudiar el funcionamiento de los diferentes escenarios propuestos para MIPv6, para comprobar si, efectivamente, el protocolo funciona como se expuso en la teoría. Con este fin, vamos a realizar pruebas de *ping* sencillas, en las cuales, observaremos el intercambio de mensajes entre los diferentes agentes.

La prueba consistirá en lo siguiente, sirviéndonos del comando de Linux *ping6* realizaremos una serie de pings consecutivos que irán desde el CN al MN. En un instante dado de la prueba provocaremos el traspaso del MN. Este pasará de estar en la HN a estar en una FN. En ese punto, estudiaremos el intercambio de mensajes que se produce y como se transmiten los datos, una vez el proceso se ha completado con éxito.

5.1.1 Escenario básico

Hemos optado por empezar por el escenario básico, pues al no tener activado IPsec, nos va a permitir observar el contenido de todos los paquetes. Por tanto, en este escenario, la opción *UseMnHaIPsec* de la configuración del MN y el HA deben estar desactivadas. En este caso, no vamos a usar optimización de rutas, por lo que, *DoRouteOptimizationMN* y *DoRouteOptimizationCN* deben estar desactivados en MN y CN, respectivamente.

5.1.1.1 Situación incial

Antes de empezar activamos el envío de RA en el HA mediante el comando *radvd –C /etc/radvd.conf*. Ahora este nodo está anunciando que es el HA de la red.

[9] La optimización de rutas e IPsec son procesos incompatibles por razones obvias. IPsec protege los paquetes transmitidos entre el HA y el MN. Con optimización de rutas la comunicación entre el CN y el MN es directa, no pasa por el HA, por lo que no podemos aplicar IPsec.
[10] Los problemas de seguridad que implica la optimización de rutas vienen detallados en RFC 4225

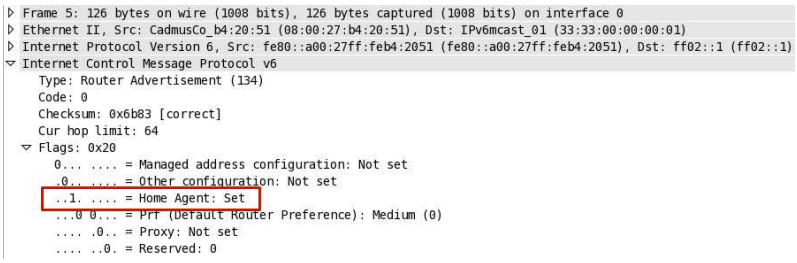


Figura 5-1. Router Advertisement del HA[11]

Iniciamos el software de movilidad en HA y MN. Tenemos la opción de iniciarlo directamente en el arranque mediante un script de servicio en */etc/init.d/*, pero hemos preferido iniciarlo de forma manual para tener un mayor control, por lo que usamos el siguiente comando para iniciarlo con nuestro fichero de configuración: *mip6d –c /usr/local/etc/mip6d.conf*. En este punto, podemos comenzar a realizar los pings. El comando *ping6* nos sirve para mandar pings a direcciones IPv6. Este comando envía paquetes constantemente. Para poder ver mejor que pasa entre un ping y otro, y que no se nos llenen las capturas de Wireshark de pings, vamos a realizar una secuencia de 10 pings espaciados con 3 segundos entre cada uno. En un terminal del CN escribimos lo siguiente:

```
ping6 2001:db8:1111:1::2 (IPv6 de MN) –i 3 –c 10
```

En los momentos previos al traspaso la situación es la siguiente. Al estar el MN en la HN, no se produce ningún intercambio de BU/BA y las tablas de *Binding Caché* de HA y *Binding Update List* de MN se encuentran vacías.

Los primeros paquetes enviados por el CN realizan el siguiente recorrido:

1. El *echo (ping) request* llega al router de la HN (HR).
2. Este pregunta por la dirección del MN. Como esta en la red, lo envía directamente al MN sin pasar por el HA.
3. El mensaje *echo reply* se envía a través del HA, esto es porque en la tabla de encaminamiento, aparece como puerta por defecto al exterior, la IP local al enlace del HA

```
::/0      fe80::a00:27ff:feb4:2051   UG   996   0   eth0
```

4. Del HA lo envía al CN a través del HR.

5.1.1.2 Traspaso

Después de recibir varios *echo (ping) reply* haremos el traspaso del MN de la HN a una FN. ¿Cómo se ejecuta este proceso? Como recordará, el MN tenía dos adaptadores de red. Por la interfaz *eth0* el MN está conectado a la HN con rango IPv6 *2001:db8:1111:1::/64*, por la interfaz *eth1* enlaza con una FN con rango IPv6 *2001:db8:1111:3::/64*. Esto, que en un principio puede resultar extraño, se hizo así para realizar el traspaso de forma rápida. En primer lugar, sólo se encuentra activa la interfaz que pertenece a la HN (*eth0*). En el momento de traspaso, se activa la interfaz de la FN (*eth1*) y se desactiva *eth0*. De esta forma, el MN pasa a pertenecer a la FN, produciéndose el traspaso. Para que este proceso se ejecute rápido utilizamos el siguiente comando:

```
ifup eth1; ifdown eth0
```

Como observará, se levanta la nueva interfaz, antes de apagar la antigua. Esto se hace para que el tiempo del

[11] Esta, y el resto de imágenes del apartado 5.1.1 son extraidas del fichero .pcapng que se encuentra incluido en el CD en Wireshark > básico > basicoMN.pcapng

handover sea menor, aunque, a efectos prácticos, las diferencias no son significativas.

Ahora, para que el MN genere un BU, este debe darse cuenta de que ha cambiado de red. Esto lo puede hacer de dos formas diferentes, pero ambas a través del *Neighbor Discovery Protocol*. Puede detectar el cambio, porque envía un NS o un RS y en la respuesta obtiene información de un medio diferente al de la HN, o bien, porque recibe un RA con el prefijo de la nueva red. En nuestro caso, se produce el segundo caso. Por este motivo fue por el que pusimos un intervalo de envío de RAs bajo a la hora de configurar los routers.

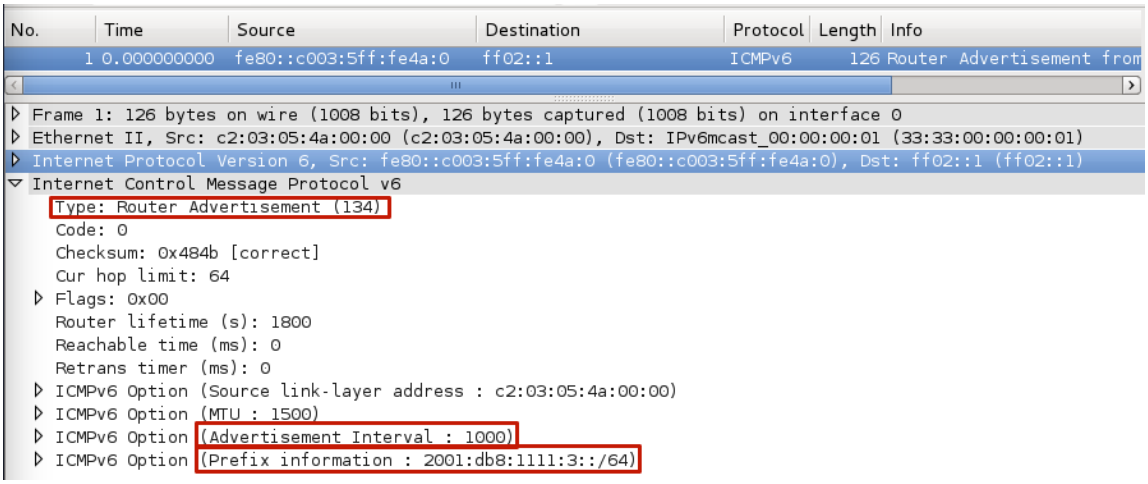


Figura 5-2. RA recibido por el MN en la FN

Cuando nos cambiamos de red, recibimos un RA cuya dirección origen es la IPv6 local al enlace del router de la FN (R3, en nuestra topología). Esto ya nos deja claro que se ha producido un cambio. Sin embargo, si miramos dentro del RA, vemos que el prefijo anunciado no se corresponde con el de la HN. Además observamos que el envío de RAs por parte del router se produce cada 1000ms como habíamos configurado.

Una vez configurada su nueva dirección se produce la asociación entre la HoA y la CoA, para ello MN manda un *Binding Update*.

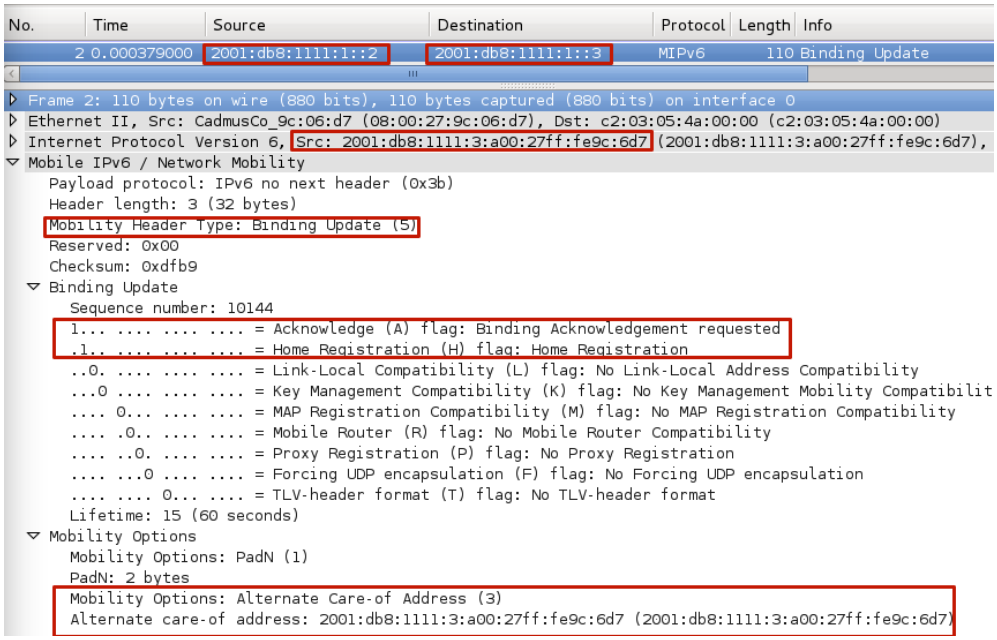


Figura 5-3. Binding Update enviado por el MN

Observamos que la dirección origen que se muestra es diferente a la que aparece en la cabecera IPv6. Esto es porque en el campo *Destination Option* de esta cabecera se incluye la HoA, y Wireshark la sustituye en la dirección de origen del paquete. Así, desde niveles superiores, no se ve que haya cambiado de red.

Entre las opciones que contiene el BU destacamos, el hecho de que requiere un BA como respuesta. Sin embargo, este BA no es necesario para que el MN empiece a enviar datos, pues para eso activamos la opción

de *OptimisticHandoff* en la configuración de UMIP. Activando el bit H indicamos que queremos registrar una dirección en el HA. Por último, podemos ver que la nueva CoA no es la que viene en el origen del paquete, sino que viene expresada explícitamente dentro de las opciones de movilidad. En respuesta al BU, el HA manda un BA como el siguiente:

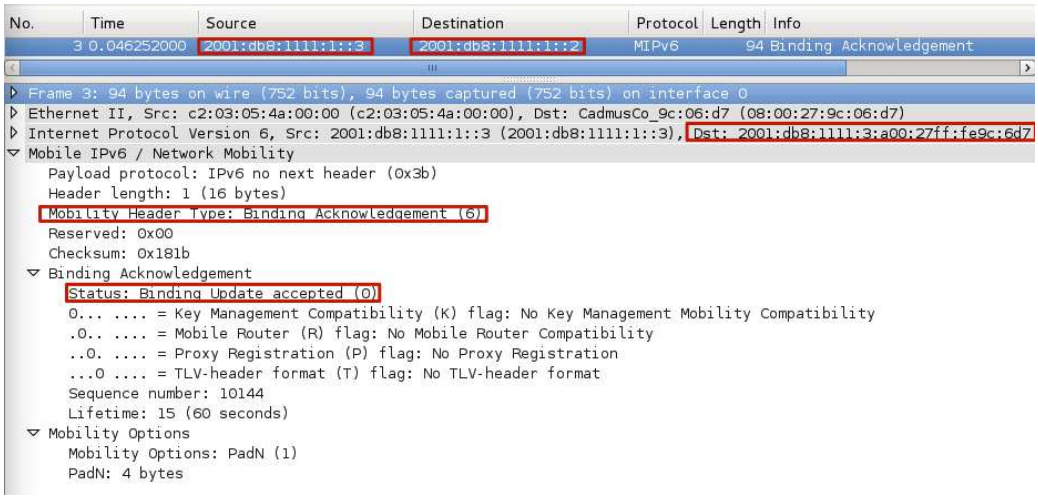


Figura 5-4. Binding Acknowledge enviado por el HA

Podemos ver que, al igual que pasaba con el BU, Wireshark sustituye la verdadera dirección destino (2001:db8:1111:3:a00:27ff:fe9c:6d7) por la HoA (2001:db8:1111:1::2), de forma que parece que ambos están en la misma red. Vemos, también, que se indica que el mensaje BU anterior a sido aceptado por el HA. Con esto se completa el traspaso.

5.1.1.3 Situación final

Para comprobar que el traspaso se a producido de forma satisfactoria vamos a mirar en la *Binding Caché* del HA, para ver si se ha registrado la nueva CoA del MN. Nos conectamos al terminal virtual de UMIP, e introducimos como root *telnet localhost 7777*. Podemos introducir el comando *verbose yes* para obtener información más detallada.

```
mip6d> bc
hoa 2001:db8:1111:1:0:0:0:2 nonce 0 status registered
 coa 2001:db8:1111:3:a00:27ff:fe9c:6d7 nonce 0 flags AH--
 local 2001:db8:1111:1:0:0:0:3 tunnel ip6tnl1 link eth0
 lifetime 59 / 60 seq 15744 unreach 0 mpa -6759 / 601 retry 0
```

Figura 5-5. Binding caché en el HA

Efectivamente, comprobamos que, a la HoA inicial del MN, le tiene asignado la CoA de la nueva red. Algo parecido debemos encontrar si miramos la *Binding Update List* en el MN.

```
mip6d> bul
== BUL_ENTRY ==
Home address    2001:db8:1111:1:0:0:0:2
Care-of address 2001:db8:1111:3:a00:27ff:fe9c:6d7
CN address      2001:db8:1111:1:0:0:0:3
 lifetime = 60,  delay = 57000
 flags: IP6_MH_BU_HOME IP6_MH_BU_ACK
 ack ready
 dev eth1 last_coa 2001:db8:1111:3:a00:27ff:fe9c:6d7
 lifetime 59 / 60 seq 15809 resend 0 delay 57(after 57s) expires 59
 mps 70313 / 77756
```

Figura 5-6. Binding Update List en el MN

Destaca en esta captura que *CN address* no se corresponde con la dirección real del CN, sino que es la del equipo a la que se le envía la respuesta (en este escenario el HA). Cuando usemos optimización de rutas, veremos que esa dirección si coincidirá con la del CN.

Ahora que está registrado el camino que siguen los paquetes es el siguiente:

1. El *echo (ping) request* llega al router de la HN (HR) y este pregunta por la HoA del MN.
2. El HA recibe los paquetes con destino HoA y se los envía a la CoA a través de un túnel IPv6-in-IPv6.
3. El MN envía su respuesta por el mismo túnel en sentido contrario.
4. El HA reenvía la respuesta del MN al CN, completando el proceso.

Podemos observar la existencia del túnel IPv6-in-IPv6 en la siguientes capturas.

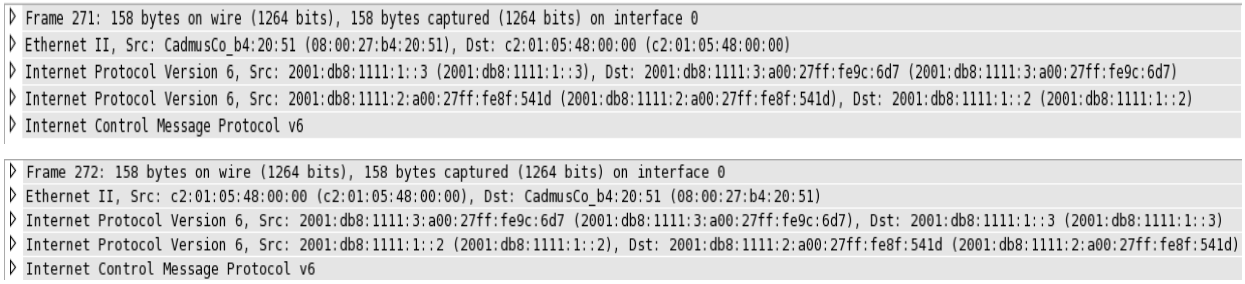


Figura 5-7. Mensajes de ping a través del túnel

Observamos que los paquetes tienen dos cabeceras IPv6. La primera, es la que forma el túnel que comunica el HA con la CoA. La segunda contiene las direcciones globales del paquete, desde el CN hasta la HoA y viceversa. Son las que aparecen en el paquete de Wireshark y las que ven los protocolos de capas superiores.

De la realización de esta prueba sacamos varias conclusiones. Para empezar, demostramos que el proyecto UMIP funciona y reproduce de forma fidedigna el funcionamiento e intercambio de mensajes de MIPv6, permitiéndonos mover un nodo de una red a otra sin perder la comunicación. Por otro lado, se hace patente la necesidad de una herramienta para medir el rendimiento, pues con ping no es suficiente.

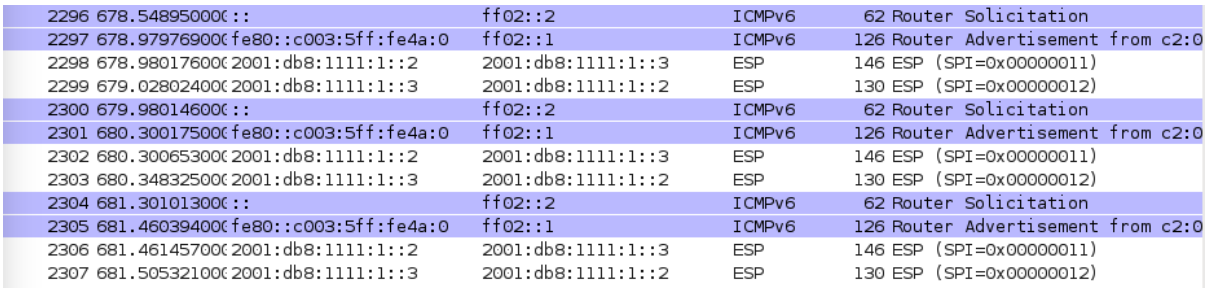
5.1.2 Escenario con IPsec

En esta prueba pretendemos demostrar que podemos añadir cierto nivel de seguridad a las comunicaciones en MIPv6. Vamos a servirnos de IPsec para asegurar con ESP el tráfico entre el HA y el MN. Lo primero que tenemos que hacer en ambos nodos es activar la opción *UseMnHaIPsec* en el fichero de configuración de UMIP. Antes de empezar el proceso debemos activar las SAs que, si recuerda, configuramos en el capítulo 4. Para activarlas introducimos en HA y MN lo siguiente:

```
setkey –f /usr/local/etc/setkey.conf
```

El procedimiento a seguir en el transcurso de la prueba es el mismo que el visto en la subsección anterior. Con la única diferencia que esta vez el tráfico entre el MN y el HA estará encriptado y no podremos ver el contenido de los paquetes.

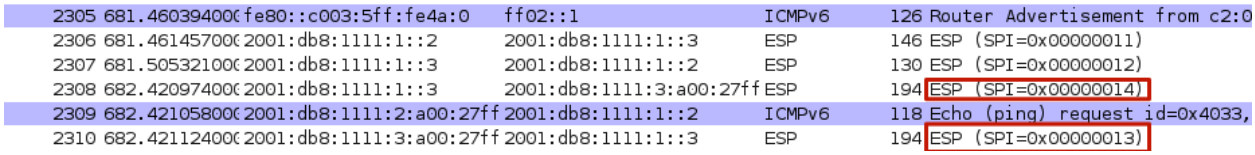
Podemos observar el tráfico de señalización entre el MN y el HA que se envía de forma periódica y corresponde con los BU y BA que intercambian constantemente. Sabemos que son estos paquetes porque están identificados con el SPI (*Security Parameter Index*) 11 y 12 respectivamente, pero no podemos ver su contenido.



2296	678.54895000(	::	ff02::2	ICMPv6	62 Router Solicitation
2297	678.97976900(	fe80::c003:5ff:fe4a:0	ff02::1	ICMPv6	126 Router Advertisement from c2:0
2298	678.98017600(	2001:db8:1111:1::2	2001:db8:1111:1::3	ESP	146 ESP (SPI=0x00000011)
2299	679.02802400(	2001:db8:1111:1::3	2001:db8:1111:1::2	ESP	130 ESP (SPI=0x00000012)
2300	679.98014600(	::	ff02::2	ICMPv6	62 Router Solicitation
2301	680.30017500(	fe80::c003:5ff:fe4a:0	ff02::1	ICMPv6	126 Router Advertisement from c2:0
2302	680.30065300(	2001:db8:1111:1::2	2001:db8:1111:1::3	ESP	146 ESP (SPI=0x00000011)
2303	680.34832500(	2001:db8:1111:1::3	2001:db8:1111:1::2	ESP	130 ESP (SPI=0x00000012)
2304	681.30101300(	::	ff02::2	ICMPv6	62 Router Solicitation
2305	681.46039400(	fe80::c003:5ff:fe4a:0	ff02::1	ICMPv6	126 Router Advertisement from c2:0
2306	681.46145700(	2001:db8:1111:1::2	2001:db8:1111:1::3	ESP	146 ESP (SPI=0x00000011)
2307	681.50532100(	2001:db8:1111:1::3	2001:db8:1111:1::2	ESP	130 ESP (SPI=0x00000012)

Figura 5-8. Intercambio de BU/BA encriptados con ESP[12]

Observamos que las direcciones origen y destino son las del HA y la HoA. En cambio, con el tráfico de datos, el que es tunelado, se sustituye la HoA por la CoA, es decir, los dos extremos del túnel. Los paquetes de datos se identifican con los SPI 13 y 14.



2305	681.46039400(	fe80::c003:5ff:fe4a:0	ff02::1	ICMPv6	126 Router Advertisement from c2:0
2306	681.46145700(	2001:db8:1111:1::2	2001:db8:1111:1::3	ESP	146 ESP (SPI=0x00000011)
2307	681.50532100(	2001:db8:1111:1::3	2001:db8:1111:1::2	ESP	130 ESP (SPI=0x00000012)
2308	682.42097400(	2001:db8:1111:1::3	2001:db8:1111:3:a00:27ff	ESP	194 ESP (SPI=0x00000014)
2309	682.42105800(	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	ICMPv6	118 Echo (ping) request id=0x4033,
2310	682.42112400(	2001:db8:1111:3:a00:27ff	2001:db8:1111:1::3	ESP	194 ESP (SPI=0x00000013)

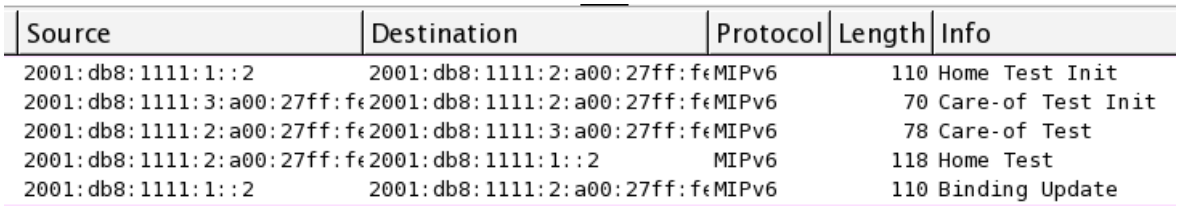
Figura 5-9. Intercambio de datos encriptados a través del túnel.

Podemos suponer que estos mensajes son los *echo request* y *echo reply* pues son los únicos datos generados, pero, a priori, no podríamos saber el contenido de estos paquetes, por lo que IPsec cumple su cometido.

5.1.3 Escenario con optimización de rutas y sin IPSec

En este escenario hemos descartado el uso de IPsec, pues como comentamos al principio del capítulo 5, no son compatibles. Por tanto,para que este escenario sea efectivo, necesitamos desactivar la opción *UseMnHaIPsec* y permitir la optimización de rutas con la opción *DoRouteOptimizationMN enable*. Para terminar, necesitamos iniciar el software de movilidad en el CN para que pueda registrar la nueva dirección del MN. Recordamos que la única opción que había que habilitar en la configuración del CN era *DoRouteOptimizationCN*.

La situación de partida y el modo de traspaso no varía con respecto a las pruebas anteriores. El cambio viene una vez que el MN llega a la FN. En este momento el MN inicia el *Return Routability Procedure* un protocolo que describimos en el apartado 2.3 y que sirve para securizar el intercambio de BUs entre el MN y el CN. Como ya explicamos, este proceso consta de 4 mensajes que se muestran a continuación.



Source	Destination	Protocol	Length	Info
2001:db8:1111:1::2	2001:db8:1111:2:a00:27ff:fe	MIPv6	110	Home Test Init
2001:db8:1111:3:a00:27ff:fe	2001:db8:1111:2:a00:27ff:fe	MIPv6	70	Care-of Test Init
2001:db8:1111:2:a00:27ff:fe	2001:db8:1111:3:a00:27ff:fe	MIPv6	78	Care-of Test
2001:db8:1111:2:a00:27ff:fe	2001:db8:1111:1::2	MIPv6	118	Home Test
2001:db8:1111:1::2	2001:db8:1111:2:a00:27ff:fe	MIPv6	110	Binding Update

Figura 5-10. Mensajes en Return Routability Procedure.[13]

Estos mensajes son originados en el MN con el fin de asegurar el envío de Bus al CN. Los mensajes *Home Test Init* y *Home Test* son intercambiados a través del HA. Los mensajes *Care-of Test Init* y *Care-of Test* van directamente de la CoA al CN y viceversa.

Una vez termina este proceso, el MN envía periódicamente BUs tanto al HA, como al CN. El CN envía los paquetes a la CoA, sin necesidad de pasar por la HN, el MN responde del mismo modo como podemos observar en la siguiente captura.

[12] Esta, y el resto de imágenes del apartado 5.1.2 son extraidas del fichero .pcapng que se encuentra incluido en el CD en Wireshark > IPsec > IPsecMN.pcapng

[13] Esta, y el resto de imágenes del apartado 5.1.3 son extraidas del fichero .pcapng que se encuentra incluido en el CD en Wireshark > Opt-rutas > optrutasMN.pcapng

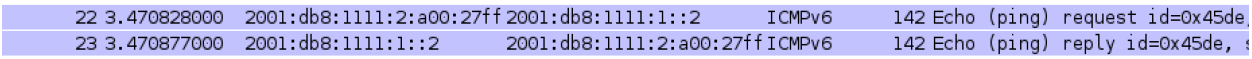


Figura 5-11. Pings con optimización de rutas

5.2 Análisis del rendimiento

Una vez hemos analizado el funcionamiento, vamos a estudiar como responde MIPv6 ante un flujo de tráfico constante. Para ello realizaremos diferentes pruebas sobre nuestros tres escenarios, con el objetivo de medir el rendimiento. Implementaremos distintos flujos de tráfico en la capa de transporte, para ver como responde. Observando el tiempo de indisponibilidad que provoca el traspaso, el número de paquetes perdidos, o el aumento de transmisiones que se producen (en caso de que haya retrasmisión), podemos hacernos una idea del rendimiento.

En concreto, las pruebas consistirán en la generación de tráfico UDP y TCP desde el CN al MN. Podríamos realizar las pruebas en sentido contrario (del MN al CN), pero creemos que así aportan más información. Si realizamos la transmisión del MN al CN ocurre que, durante el traspaso, el MN deja de enviar datagramas. Por tanto, solo podemos medir el tiempo de indisponibilidad del nodo.

Para realizar las pruebas necesitamos un software que nos permita generar el tráfico y medirlo. Usaremos Iperf y Jperf. Iperf es un programa que deberemos instalar en ambos nodos y que nos permite crear un flujo constante de datos. Iperf se maneja mediante línea de comandos, por lo que, para hacer más sencillo su uso, instalamos Jperf que no es más que una interfaz gráfica para Iperf. Desde esta interfaz podemos configurar las características del tráfico que buscamos y nos muestra información, como los paquetes perdidos y totales, y el ancho de banda.

Configuramos el MN como un servidor escuchando en el puerto 5001. El CN actuará como cliente siendo la fuente del tráfico. Con este software y ayudándonos de Wireshark, podemos recolectar los datos que necesitamos para completar nuestro análisis

5.2.1 Tráfico TCP

En esta prueba vamos a trabajar con el protocolo de capa 4 TCP. Crearemos un flujo de tráfico TCP entre el CN y el MN ayudándonos de las herramientas Iperf y Jperf, instaladas en ambos nodos. Para crear el flujo de datos Iperf nos ofrece dos opciones.

- Crear un flujo constante durante un periodo de tiempo determinado (10 segundos, por ejemplo).
- Transmitir un fichero de el cliente al servidor.

La primera opción nos permite medir el tiempo de indisponibilidad del nodo MN en el traspaso, o el número de paquetes que se han entregado correctamente. Sin embargo, con la segunda opción podemos extraer datos como el tiempo total que tarda en transferir el archivo, el número de paquetes necesarios, y por supuesto, el tiempo de indisponibilidad. Es por eso, que hemos optado por la segunda opción.

Para esta prueba realizaremos test sobre 4 escenarios, un primer escenario sin traspaso que nos sirva de referencia, y los tres escenarios analizados al comienzo de este capítulo. Para realizar el test deberemos apoyarnos en un en la capa de aplicación. Configuramos el CN para que envíe al puerto 5001 un fichero que pesa 4,2 Mb con un tiempo máximo de transmisión de 60 segundos.

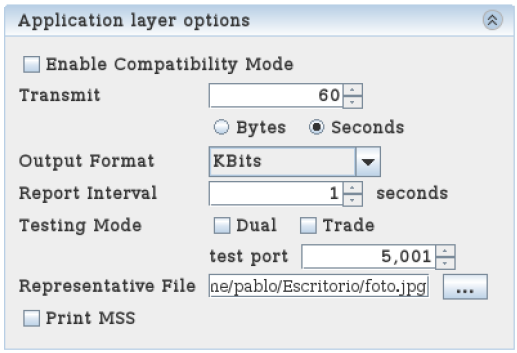


Figura 5-12. Configuración Iperf en el CN

La siguiente tabla recoge el resultado de los experimentos. En ella se recogen la media y la desviación típica de diversos parámetros sacada a partir de la realización de 10 experimentos por escenario. Se entiende, por paquetes necesarios, el número total de paquetes, entre datos y señalización, recogidos en el MN. El tiempo de transferencia abarca desde que se crea el flujo TCP con los mensajes de sincronización, hasta que se recibe el último dato.

Tabla 5-1. Resultados experimentos TCP

TCP		Referencia	Básico	IPsec	Opt. rutas
Paquetes necesarios	media	5125,5	5408,25	5313,13	5394,72
	Des. típica (σ)	69,99	92,48	124,19	133,84
Tiempo de transferencia (segundos)	media	37,47	66,69	66,25	64,71
	Des. típica (σ)	1,12	1,25	1,84	1,02

Si observamos la desviación típica en el escenario de referencia vemos que el número de paquetes totales es varía de una prueba a otra. Esto es debido a que el CN envía datos a una tasa superior a la que MN está recibiendo por lo que se producen retrasmisiones.

Los tres escenarios tienen un rendimiento muy similar, pues vemos que el tiempo aumenta entre un 70 y un 80 por ciento en los tres escenarios, con respecto a la referencia. A la vista de los resultados resaltar los siguientes comentarios.

A priori, podría pensarse que el Escenario con IPsec debería tener un rendimiento más pobre debido a la mayor carga de procesamiento que requiere la implementación de este protocolo. Observando la tabla vemos que el rendimiento es bastante similar al escenario sin seguridad, por lo que IPsec no ha lastrado el funcionamiento de MIPv6.

Se ha de destacar que medir el número de paquetes totales con IPsec, usando Wireshark no es algo directo. Los paquetes están cifrados por lo que no sabemos que es lo que contienen. Para averiguarlo podemos declarar en Wireshark las SAs definidas entre HA y MN de forma que este pueda descifrar los paquetes. Para ello nos dirigimos a *Edit* → *Preferences* → *Protocols* → *ESP*, y configuramos las SAs con la información contenida en los ficheros *setkey.conf*.

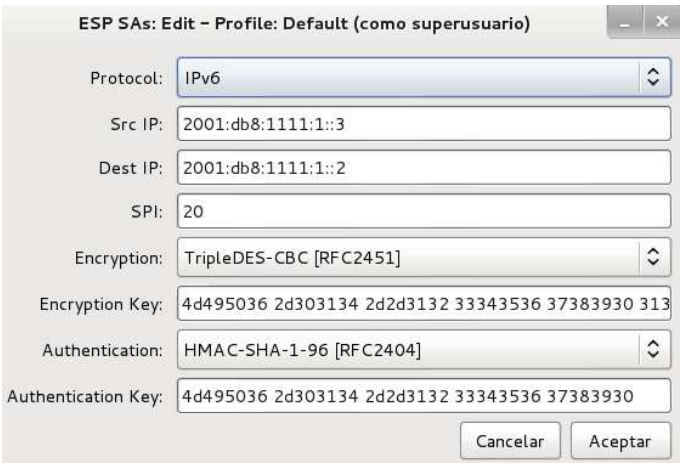


Figura 5-13. SAs declaradas en Wireshark

Por otro lado, vemos que el escenario con optimización de rutas no supone una gran ventaja con respecto al resto. Esto se debe, principalmente a dos puntos.

- El escenario no es lo suficientemente grande como para que la optimización de rutas implique una mejora sustancial. Los datagramas con optimización de rutas dan sólo un salto menos que en el escenario básico.
- El segundo y más importante, es que hemos observado en las capturas, que el proceso de registro del MN en el CN, no se produce hasta que no cesa el flujo TCP en curso. Por tanto, a efectos prácticos el escenario se comporta como el escenario básico. Si creáramos otro flujo TCP una vez el nodo esta en la FN, este si utilizaría la optimización de rutas.

5.2.2 Tráfico UDP

En esta segunda tanda de pruebas, vamos a analizar como afecta el traspaso a un flujo de datagramas en UDP. Hemos usado, de nuevo Iperf y Jperf para crear tráfico UDP del CN al MN. En este caso el experimento consiste en generar un flujo constante de paquetes de 1300 Bytes durante 10 segundos. Al igual que en TCP, realizaremos 10 experimentos por escenario, para luego mostrar la media y la desviación típica, a modo de resumen.

A diferencia de TCP, UDP no implementa mecanismos para asegurar la trasmisión satisfactoria de los datagramas. No existen retrasmisiones, si un nodo no está disponible, los paquetes se pierden. Por tanto, en esta prueba vamos a medir, el tiempo en el que el nodo no está disponible durante el traspaso y el número de paquetes que se pierden. Un resumen de los datos se recoge en la siguiente tabla.

Tabla 5-2. Resultados experimentos UDP

UDP		Referencia	Básico	IPsec	Opt. rutas
Paquetes perdidos (%)	media	0	31,09	33,91	29,11
	Des. típica (σ)	0	4,3	3,99	3,16
Tiempo de Indisponibilidad (segundos)	media	0	3,598	3,76	3,45
	Des. típica (σ)	0	0,46	0,21	0,37

Como era de esperar, en el escenario de referencia, en el cual no se produce traspaso, el MN está siempre disponible y no se pierden paquetes. Esto nos indica que nuestra red funciona sin problemas.

Al igual que pasaba con TCP, obtenemos resultados muy similares en los tres escenarios. Los datos del

número de paquetes perdidos los obtenemos de Iperf, ya que este es capaz de hacer una estimación. En total se trasmiten 1063 datagramas, pero hemos creído mejor mostrar los resultados en porcentajes, pues resulta más representativo.

Si vemos el tiempo de indisponibilidad vemos que oscila entre los 3,4 y los 3,8 segundos y que los resultados de los diferentes experimentos suelen ser muy parecidos pues la desviación típica es pequeña. Para medir el tiempo de indisponibilidad nos vamos a la captura de Wireshark en el MN y buscamos el último paquete UDP recibido antes del traspaso. Le asignamos una referencia de tiempo y buscamos el siguiente datagrama UDP recibido, el cual, estará después de que el MN mande un BU.

En el escenario de optimización de rutas se repite el mismo caso que detectamos en TCP y es que, como se ve en la imagen el *Return Routability Procedure* se produce una vez e flujo UDP se ha completado.

2004	5.463958000	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	UDP	1404	Source port: 58185 Destination port: 5001
2005	5.464037000	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	UDP	1364	Source port: 58185 Destination port: 5001
2006	5.476008000	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	UDP	1404	Source port: 58185 Destination port: 5001
2007	5.476057000	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	UDP	1364	Source port: 58185 Destination port: 5001
2008	5.487841000	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	UDP	1404	Source port: 58185 Destination port: 5001
2009	5.487899000	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	UDP	1364	Source port: 58185 Destination port: 5001
2010	5.499091000	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	UDP	1404	Source port: 58185 Destination port: 5001
2011	5.499150000	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	UDP	1364	Source port: 58185 Destination port: 5001
2012	5.511298000	2001:db8:1111:3:a00:27ff	ff02::fb	MDNS	325	Standard query response 0x0000 PTR _udisks-s
2013	5.511438000	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	UDP	1404	Source port: 58185 Destination port: 5001
2014	5.511496000	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	UDP	1364	Source port: 58185 Destination port: 5001
2015	5.523525000	2001:db8:1111:1::3	2001:db8:1111:1::2	MIPv6	96	Binding Acknowledgement
2016	5.529588000	2001:db8:1111:1::2	2001:db8:1111:2:a00:27ff	MIPv6	72	Home Test Init
2017	5.529596000	2001:db8:1111:1::2	2001:db8:1111:2:a00:27ff	MIPv6	112	Home Test Init
2018	5.529825000	2001:db8:1111:3:a00:27ff	2001:db8:1111:2:a00:27ff	MIPv6	72	Care-of Test Init
2019	5.530021000	2001:db8:1111:1::2	2001:db8:1111:2:a00:27ff	UDP	1364	Source port: 5001 Destination port: 58185
2020	5.530029000	2001:db8:1111:1::2	2001:db8:1111:2:a00:27ff	UDP	1404	Source port: 5001 Destination port: 58185
2021	5.580461000	2001:db8:1111:2:a00:27ff	2001:db8:1111:3:a00:27ff	MIPv6	80	Care-of Test
2022	5.615861000	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	MIPv6	120	Home Test
2023	5.615919000	2001:db8:1111:2:a00:27ff	2001:db8:1111:1::2	MIPv6	80	Home Test
2024	5.616267000	2001:db8:1111:1::2	2001:db8:1111:2:a00:27ff	MIPv6	112	Binding Update

Figura 5-14. Return Routability Procedure después del flujo UDP

Otra cosa que se ha detectado al ver los resultados de todos los experimentos, es que existe una relación, prácticamente lineal entre el tiempo de indisponibilidad y el número de paquetes perdidos. Esto parece obvio, pues el CN envía los datos a una tasa constante, por lo que, cuanto más tiempo este el MN fuera de alcance, más paquetes se perderán.

A la vista de los resultados, podemos concluir que UDP no es el mejor protocolo de trasporte para servicios que requieran una alta fidelidad de trasmisión. En el traspaso la comunicación se corta totalmente durante al menos 3 segundos, lo cual está lejos de ser óptimo.

6 DESPLIEGUE DEL ESCENARIO PARA NEMO

En este capítulo vamos a explicar brevemente los pasos a seguir para transformar nuestro escenario para pruebas de MIPv6, en un escenario en el que realizar pruebas de movilidad para redes móviles (NEMO). La configuración de este laboratorio de pruebas es algo más extensa, pues entre otras cosas, implica a más elementos. Pero partiendo del escenario anterior, podemos tener un escenario para experimentar con NEMO en unos pocos pasos.

6.1 Configuración de GNS3

En primer lugar necesitamos un nodo que haga de *Mobile Network Node* (MNN). Este nodo podría ser un simple Host virtual de GNS3 que nos permite hacer pruebas con PING. Pero como queremos analizar el rendimiento del tráfico UDP y TCP, esto no nos es suficiente. Creamos otra máquina virtual llamada MNN que será un clon del CN[14].

En esta nueva disposición el MN pasa de ser un equipo final a ser un *Mobile Router* (MR), por lo que debemos cambiar su configuración para que se comporte como tal. El nuevo MR crea una red que será la *Mobile Network* (la llamaremos MNET, para diferenciarlo de MN). Esta nueva red tendrá el siguiente rango de IPs:

Tabla 6-1. Rango IPv6 de la red móvil

Red	Rango de IPv6
Mobile Network	2001:db8:1111:ff01: :/64

[14] En este escenario el CN no necesitan tener soporte para movilidad, pues optimización de rutas y NEMO no son compatibles dentro del proyecto UMIP.

El escenario representado en GNS3 de la siguiente manera:

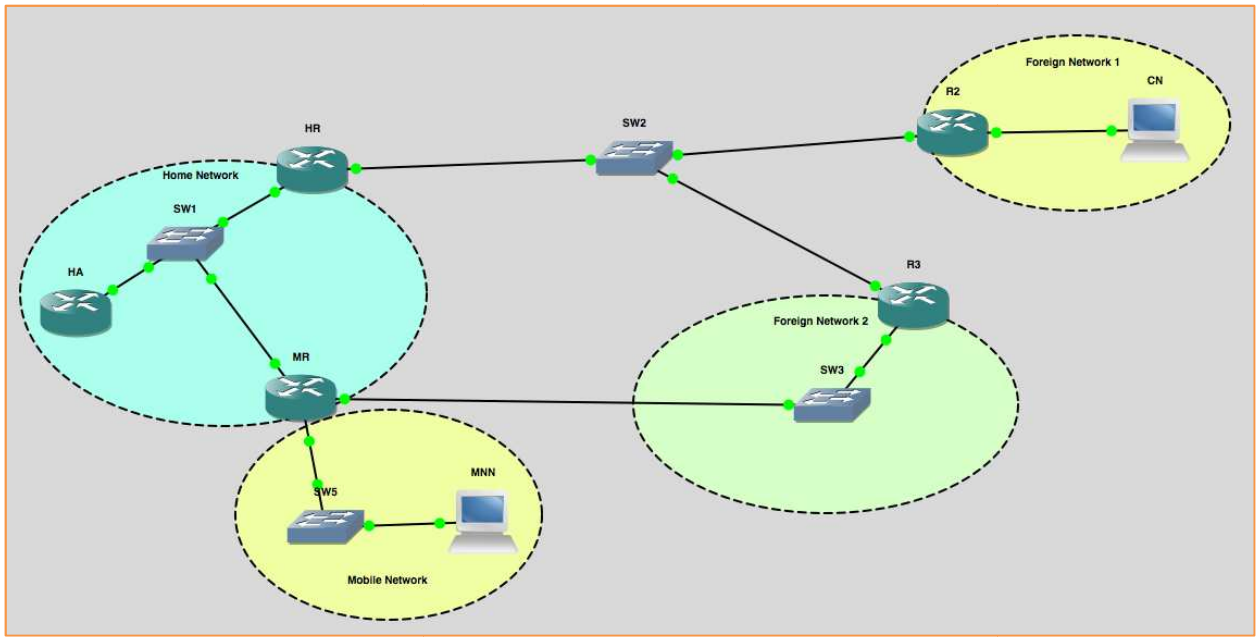


Figura 6-1. Escenario NEMO en GNS3

Podemos observar como el MR tiene un nuevo adaptador de red activo que lo une a la MNET. Aunque sigue siendo un VM hemos cambiado el icono del MR para dejar claro que ahora es un router.

6.2 Configuración del MNN

El recién incorporado MNN no requiere ninguna configuración especial. Simplemente, tiene activada su interfaz *eth0* para que se configure automáticamente por SLAAC a partir del prefijo anunciado por el MR. El MNN llevará instalado los software Iperf, Jperf y Wireshark, al igual que el CN, ya que estos son necesarios para implementar los test y recoger los datos[1].

6.3 Configuración del HA

Para adaptar el HA a NEMO debemos añadir algunas líneas al fichero de configuración de UMIP */usr/local/etc/mip6d.conf*.

Código 6-1. Nuevo fichero de configuración del HA

```
# Sample UMIP configuration file for a MIPv6 Mobile Node
NodeConfig HA;

# Set DebugLevel to 0 if you do not want debug messages
DebugLevel 10;

# Replace eth0 with the interface connected to the home link
Interface "eth0";

# Accept registrations from Mobile Routers
HaAcceptMobRtr enabled;
HaServedPrefix 2001:db8:1111:1::/64;
```

```
# Binding information
BindingAclPolicy 2001:db8:1111:1::2 (2001:db8:1111:ff01::/64) allow;
DefaultBindingAclPolicy deny;

# Enable IPsec static keying
UseMnHaIPsec disabled;
KeyMngMobCapability disabled;

# IPsec Security Policies information
IPsecPolicySet {
    HomeAgentAddress 2001:db8:1111:1::3;
    HomeAddress 2001:db8:1111:1::2/64;
    # All MH packets (BU/BA/BERR)
    IPsecPolicy Mh UseESP 11 12;
    # All tunneled packets (HoTI/HoT, payload)
    IPsecPolicy TunnelPayload UseESP 13 14;
    # All ICMP packets (MPS/MPA, ICMPv6)
    IPsecPolicy ICMP UseESP 15 16;
}
```

Los cambios son aquellas líneas remarcadas en negrita. El primer parámetro *HaAcceptMobRtr* permite al HA aceptar Bindings del MR. Con *HaServedPrefix* definimos cual es el prefijo de red de la HoA del MR. Con NEMO es posible definir una HoA diferente para el *Mobile Network Prefix* (MNP) asociado al MR. En nuestro caso, simplemente introducimos el prefijo de la HN. El HA necesita declarar cual es el MNP asociado al MR. Para ello, ponemos entre paréntesis los prefijos necesarios en el parámetro *BindingAclPolicy*, que ya existía antes.

Con respecto a la configuración de IPsec y el fichero *setkey.conf* que definía las asociaciones, no es necesario ningún ajuste y podemos continuar usando el mismo. Donde si debemos tocar es en el fichero de configuración de *radvd* en el cual debemos añadir una línea habilitando el flag que indica que el soporte para NEMO esta activado.

```
AdvMobRtrSupportFlag on;
```

Para completar la configuración del HA, nos dirigimos al fichero */etc/network/interfaces* desde donde se configuran las interfaces de red y añadimos una ruta estática indicando que la comunicación con la red móvil debe hacerse a través de la IP del MN.

```
up ip -6 route add 2001:db8:1111:ff01::/64 via 2001:db8:1111:1::2
```

6.4 Configuración del MR

En este equipo es el que requiere un mayor cambio en la configuración, pues pasa de ser un nodo final a un router. En primer lugar, vamos a activar el *forwarding* de forma que pueda enrutar los paquetes entre la MNET y el resto. Al igual que hicimos en su momento con el HA descomentamos en el fichero */etc/sysctl.conf* la línea *net.ipv6.conf.all.forwarding=1*. Esta acción es incompatible con la autoconfiguración de las interfaces de redes, es decir, ya no podemos configurar las interfaces del MR mediante el protocolo SLAAC y por tanto debemos configurar */etc/network/interfaces* como sigue:

Código 6-2. Configuración de las interfaces del MR

```
# This file describes the network interfaces availables on your
# system and how to actívate them. For more information, see
```

```
# interfaces(5)

# The loopback network interface
auto lo eth0 eth2
iface lo inet loopback

iface eth0 inet6 static
       address 2001:db8:1111:1::2
       netmask 64
       gateway FE80::C001:5FF:FE48:0
iface eth1 inet6 static
       address 2001:db8:1111:3::2
       netmask 64
       gateway FE80::C003:5FF:FE4A:1
iface eth2 inet6 static
       address 2001:db8:1111:ff01::1
       netmask 64
```

Observamos que las interfaces eth0 y eth2 se activan en el inicio. Estas son las que se comunican con la HN y la MNET, respectivamente. La e*th1* se mantendrá apagada hasta el momento del traspaso. A continuación, detallaremos los cambios a realizar en el fichero de configuración de UMIP */usr/local/etc/mip6d.conf* que queda así:

Código 6-3. Nuevo fichero de configuración del MR

```
# Sample UMIP configuration file for a MIPv6 Mobile Node
NodeConfig MN;

# Set DebugLevel to 0 if you do not want debug messages
DebugLevel 10;

# Enable the optimistic handovers
OptimisticHandoff enabled;

# Disable RO with other MNs (it is not compatible
# with IPsec Tunnel Payload)
DoRouteOptimizationMN disabled;

# The Binding Lifetime (in sec.)
MnMaxHaBindingLife 60;

# Use NEMO Explicit Mode
MobRtrUseExplicitMode enabled;

# List here the interfaces that you will use
# on your mobile node. The available one with
# the smallest preference number will be used.
Interface "eth0" {
    MnIfPreference 1;
}
Interface "eth1" {
    MnIfPreference 2;
}
```

```
# Replace eth0 with one of your interface used on
# your mobile node
MnHomeLink "eth0" {
    IsMobRtr enabled;
    HomeAgentAddress 2001:db8:1111:1::3;
    HomeAddress 2001:db8:1111:1::2/64 (2001:db8:1111:ff01::/64);
}

# Enable IPsec static keying
UseMnHaIPsec disabled;
KeyMngMobCapability disabled;

# IPsec Security Policies information
IPsecPolicySet {
    HomeAgentAddress 2001:db8:1111:1::3;
    HomeAddress 2001:db8:1111:1::2/64;

    # All MH packets (BU/BA/BERR)
    IPsecPolicy Mh UseESP 11 12;
    # All tunneled packets (HoTI/HoT, payload)
    IPsecPolicy TunnelPayload UseESP 13 14;
    # All ICMP packets (MPS/MPA, ICMPv6)
    IPsecPolicy ICMP UseESP 15 16;
}
```

Activamos el modo de registro explicito de NEMO que por defecto está activado, por lo que no sería necesario. El resto de cambios se centran en el *MnHomeLink* donde permitimos al MR actuar como un router móvil mediante el parámetro *IsMobRtr*. Además debemos añadir el prefijo de la red móvil como ya hiciéramos en el HA. IPsec no necesita cambios por lo que, tanto el bloque dedicado a IPsec en *mip6d.conf* y el fichero *setkey.conf*, no sufren ninguna variación.

Por otro lado, el MR necesita mandar Ras en la red móvil anunciando su MNP. Al igual que pasaba en el HA, usaremos el software radvd con este fin. Para ello configuramos */etc/radvd.conf* de la siguiente forma:

Código 6-4. Configuración del fichero radvd.conf en el MR

```
# Mobile Router radvd configuration file
# Replace eth1 with your ingress interface name
interface eth2
{
    AdvSendAdvert on;
    MaxRtrAdvInterval 3;
    MinRtrAdvInterval 1;
    AdvIntervalOpt on;
    IgnoreIfMissing on;

    # Mobile Router address on the ingress interface
    prefix 2001:db8:ffff:ff01::1/64
    {
        AdvRouterAddr on;
        AdvOnLink on;
        AdvAutonomous on;
        AdvPreferredLifetime 60;
        AdvValidLifetime 120;
```

```
        AdvLinkMTU 1280;
    };
};
```

Este fichero es muy sencillo, pues no contiene opciones especificas de movilidad. Podemos arrancar este fichero en el arranque, o de forma manual con *radvd –C /etc/radvd.conf*. Con este paso, tenemos todo lo necesario para empezar a experimentar con NEMO.

7 ANÁLISIS DE NEMO

Llegamos al principal capítulo de este proyecto. En él, vamos a analizar el funcionamiento del protocolo de movilidad para redes NEMO. Partiendo de lo ya visto en MIPv6, estudiaremos los aspectos diferenciales de ambos protocolos. Al ser NEMO un protocolo más complejo que MIPv6, decidimos en la teoría empezar por este último, a la vez que explicamos los conceptos principales de la movilidad y otros protocolos implicados, como SLAAC o ND. Una vez conocíamos esto, el salto a NEMO era más sencillo.

Esta misma forma de proceder es la que queremos usar en la parte práctica. En el capítulo 5 estudiamos el comportamiento de MIPv6 sobre un escenario virtual, analizando el intercambio de mensajes y el rendimiento ante flujos de datos de clases diferentes. Esto mismo es lo que haremos en este capítulo. Empezaremos analizando los elementos que diferencian MIPv6 de NEMO en una implementación práctica. Luego probaremos el rendimiento de este protocolo ante diferentes flujos de datos.

7.1 Estudio y Comparación del funcionamiento

En esta sección vamos a realizar unas pruebas básicas que nos permitan estudiar las particularidades de NEMO en el intercambio de mensajes. Para ello realizaremos varios pings con el comando *ping6* que irán desde el CN al MNN. En un momento dado se realizará el traspaso de la red móvil y podremos analizar el intercambio de paquetes que se produce.

7.1.1 Situación inicial

Al igual que en los escenarios de MIPv6 necesitamos activar el reenvío de RAs en el HA con el comando *radvd –C /etc/radvd.conf*. La diferencia es que ahora también tenemos que activarlo en el MR, pues al ser este el gateway de la red móvil, debe anunciar el prefijo para que el MNN se pueda configurar por SLAAC.

Hecho esto podemos iniciar los software de movilidad en el HA y el MN[15]. Iniciamos UMIP en ambos nodos con *mip6d –c /usr/local/etc/mip6d.conf*, siendo estos los nuevos ficheros de configuración que comentamos en el capítulo anterior. Comenzamos el envío de pings desde el CN:

```
ping6 2001:db8:1111:ff01:a00:27ff:fe8f:1
```

[15] Al no haber Optimización de rutas el CN no necesita activar UMIP

El camino que siguen los mensajes de ping en este instante se recoge en la siguiente imagen:

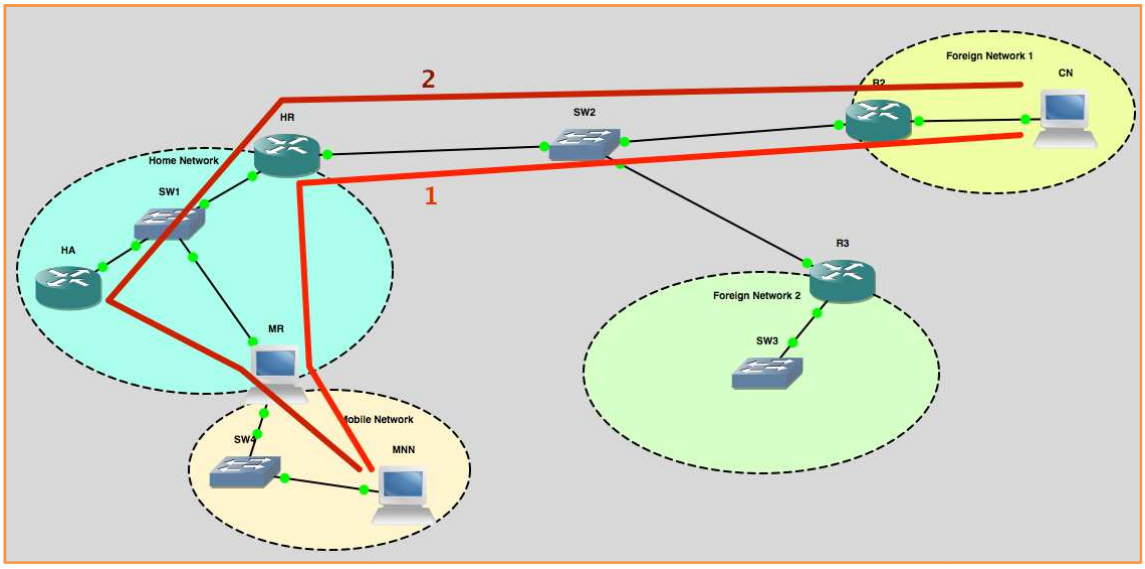


Figura 7-1. Escenario NEMO antes del traspaso

Al igual que ocurría en MIPv6, los mensajes *echo (ping) request* (1) van directamente a la red móvil sin pasar por el MN. En el router HR habíamos declarado que el acceso a la red móvil (2001:db8:1111:ff01::/64) se hacía a través de la IPv6 2001:db8:1111:1::2 (MR). Como el HR ve en su red al MR le envía directamente los paquetes sin necesidad de pasar por el HA. Los *echo (ping) reply* provenientes de la red móvil si que pasan por el HA. Hasta aquí no observamos diferencias con MIPv6.

7.1.2 Traspaso

Tras varios pings realizaremos el traspaso de la MNET desde la HN a una FN. El proceso es el mismo, en el MR encendemos la interfaz *eth1* (que pertenece a la FN), y apagamos la *eth0.* El MR pasa a pertenecer a la FN produciéndose el traspaso.

A través del protocolo ND, el MR se percatará de que ha cambiado de red, y en ese momento iniciara el proceso de actualización de *Binding* con el HA. Aquí, si encontramos diferencias con respecto a MIPv6, pues el BU contiene información adicional.

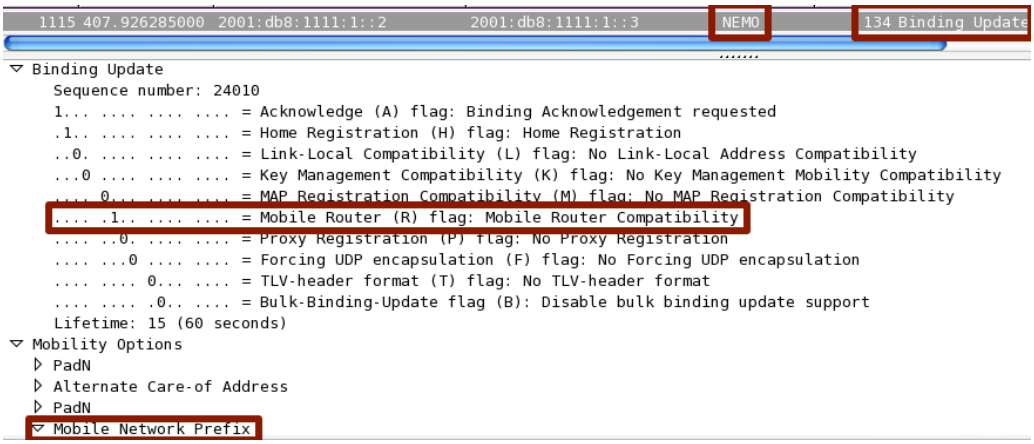


Figura 7-2. Binding Update en NEMO[16]

A primera vista, observamos que el nombre del protocolo ahora es NEMO. Pero si miramos dentro del paquete, encontramos dos campos que no existían antes. El bit R (*Mobile Router Flag*) que indica que se está

[16] Esta imagen es extraida del fichero .pcapng que se encuentra incluido en el CD en Wireshark > NEMO > nemoMR.pcapng

comunicando con un MR y es a él al que tiene que mandar los paquetes que van dirigidos a la MNET, cuyo prefijo se especifica en el campo *Mobile Network Prefix*, que es la otra novedad de este mensaje. El mensaje BA también contiene el nuevo bit R.

7.1.3 Situación final

Una vez se ha producido el intercambio de BUs/BAs se actualizan las *Binding Caché* y *Binding Update List*, del HA y MR respectivamente. Entonces podemos dar el traspaso por concluido. A partir de ese momento, la comunicación sigue el siguiente camino.

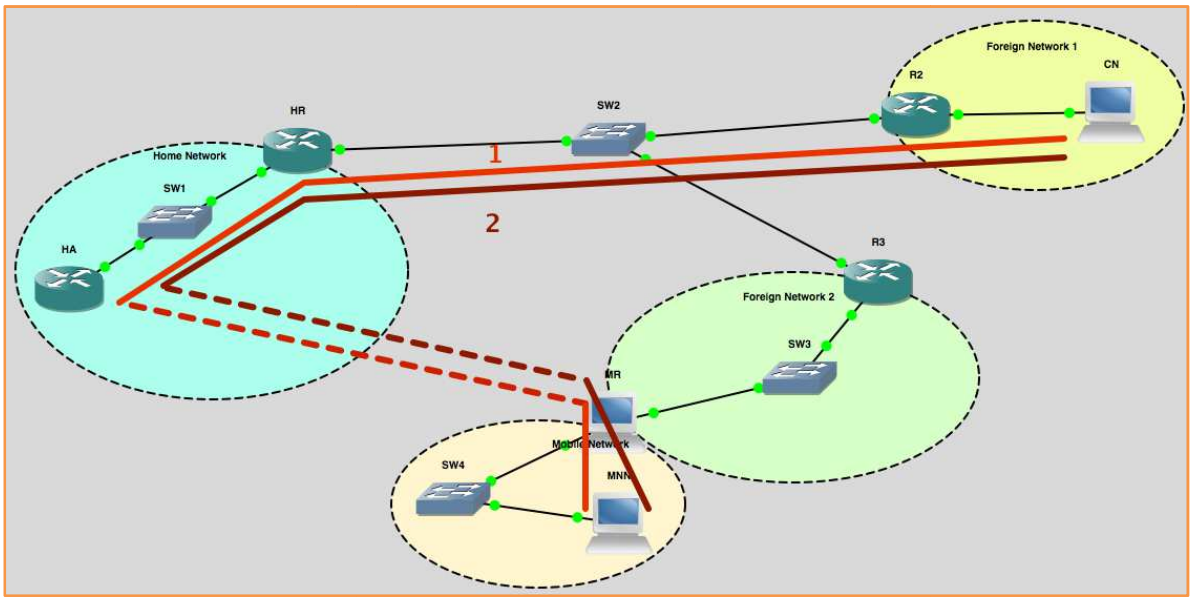


Figura 7-3. Escenario NEMO tras el traspaso.

Como era de esperar, los datos pasan siempre por el HA, ya que no se implementa optimización de rutas. La comunicación entre el HA y el MR se realiza mediante un túnel IPv6-in-IPv6. Una vez que los datagramas llegan a la MNET, el MR se encarga de enrutarlos al MNN.

7.2 Análisis del rendimiento

Por último, trataremos de medir el rendimiento de este protocolo comparándolo con MIPv6. Para lo cual, repetiremos las pruebas del capítulo 5. Ayudándonos de las herramientas Iperf y Jperf, vamos a crear flujos constantes de tráfico TCP y UDP entre el CN y el MNN. El objetivo es ver como responde NEMO ante un traspaso, y como afecta este a la comunicación ante diferentes protocolos de capa de transporte.

7.2.1 Tráfico TCP

Para esta prueba vamos a simular la trasmisión de una fotografía (.jpg) desde el CN al MN. Configuramos el CN como cliente y el MNN en el servidor escuchando en el puerto 5001, como ya hiciéramos anteriormente. En esta prueba se pretende medir la sobrecarga de paquetes que provoca el traspaso debido a las retrasmisiones y el tiempo total que tarda en producirse el envío.

En primer lugar vamos a tomar un experimento aleatorio y vamos a analizar los datos que obtenemos del flujo TCP. En este primera gráfica podemos observar la evolución del ancho de banda durante el traspaso. Esta gráfica es obtenida a través de Wireshark en *Statistics→TCP StreamGraph*. Si la comparáramos con la obtenida mediante Jperf en el MNN el resultado es similar.

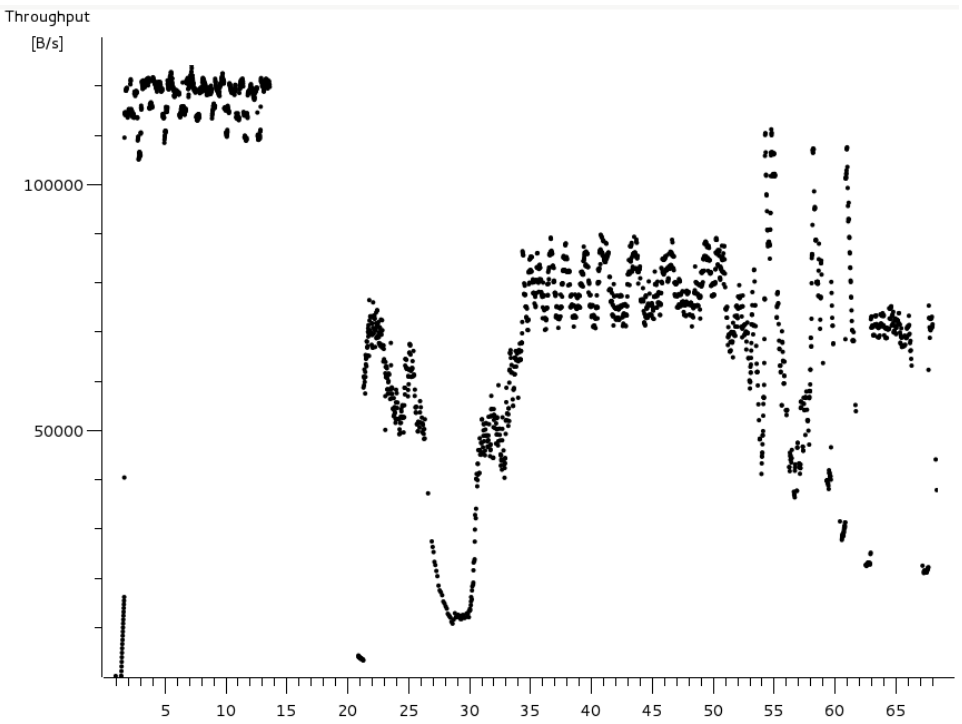


Figura 7-4. Ancho de banda con tráfico TCP

Observamos la evolución del ancho de banda durante la transmisión y identificamos, claramente el salto en la gráfica provocado en el momento del cambio de red. Este se origina entorno a las 15 segundos y se prolonga durante 6 segundos. Este periodo corresponde al *handover latency*, que en TCP no coincide con el tiempo de indisponibilidad del MR, que como veremos es ligeramente superior a 3 segundos. El *handover latency* es el retardo temporal que registra un paquete debido al traspaso. El hecho de que no coincida se debe al sistema de control de congestión que implementa TCP, cosa que no ocurre en UDP.

Hemos recogido el resumen de los resultados de 10 experimentos en la siguiente tabla, en la cual encontramos un test de referencia en el que no se produce traspaso, el test con traspaso con la implementación de NEMO y hemos tomado, a modo de comparativa, los datos recogidos en el experimento en MIPv6.

Tabla 7-1. Resultados experimentos TCP

TCP		Referencia	NEMO	MIPV6
Paquetes necesarios	media	5079,6	5380,4	5408,25
	Des. típica (σ)	82,38	140,03	92,48
Tiempo de transferencia (segundos)	media	40,02	66,52	66,69
	Des. típica (σ)	2,55	2,37	1,25

Como vemos, los resultados son muy similares, pues el funcionamiento es, a diferencia de unos bits, es el mismo. El hecho de que los datagramas necesiten dar un salto más para llegar al MNN en NEMO, no afecta al rendimiento. Si estuviéramos trabajando con equipos físicos seguro notaríamos alguna leve diferencia, pero en la simulación, el comportamiento de los enlaces es ideal.

7.2.2 Tráfico UDP

En esta prueba probaremos el comportamiento de NEMO ante un flujo de tráfico UDP. Apoyándonos en Iperf y Jperf, creamos un envío de datos constantes desde el CN al MNN durante 10 segundos. En este periodo se producirá el traspaso de la red móvil a una FN. Entre Jperf y Wireshark, tendremos las herramientas necesarias

para medir el número de paquetes perdidos en el proceso y el tiempo que la red móvil permanece inalcanzable.

Hemos tomado un experimento al azar para mostrar una gráfica de la entrada de paquetes UDP en el MNN, que nos permite remarcar de forma visual, el tiempo de indisponibilidad.

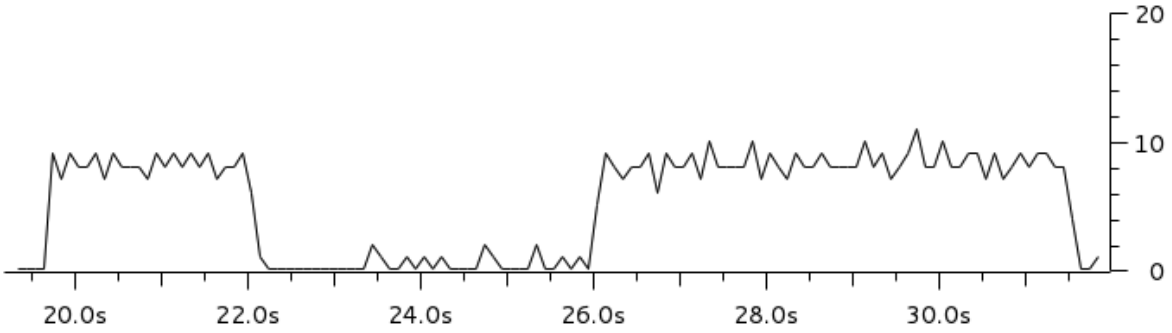


Figura 7-5. Gráfica IO de UDP en Wireshark

En la gráfica vemos como entorno a los 22 segundos comienza el cambio de red y se corta la recepción de datagramas UDP. Esta se restaura pasados, aproximadamente, 4 segundos. Esta continúa hasta los 31,5 segundos cuando el CN cesa el envío, después de 10 segundos.

Si recogiéramos esta misma gráfica, pero analizando la salida de tráfico del CN, el resultado sería igual al que se produce aquí entre el segundo 26 y el 30. El CN no para nunca de enviar datos, aunque el MR no esté disponible. Los datos que se envían en el periodo de indisponibilidad se pierden y no pueden recuperarse.

En referencia a la prueba de TCP, cabe destacar que en este caso el tiempo de indisponibilidad si coincide con el *handover latency* pues no existe en UDP mecanismos de control de congestión que frenen la reanudación del transmisión.

En la siguiente gráfica recogemos el resumen de los resultados de los diferentes test. Al igual que en casos anteriores, se han realizado 10 experimentos por cada prueba.

Tabla 7-2. Resultados experimentos UDP

UDP		Referencia	NEMO	MIPV6
Paquetes perdidos (%)	media	0	28,58	31,09
	Des. típica (σ)	0	2,50	4,3
Tiempo de Indisponibilidad (segundos)	media	0	3,22	3,598
	Des. típica (σ)	0	0,28	0,46

Como pasó en TCP, el resultado es muy similar entre NEMO y MIPv6. Nos encontramos con un tiempo de indisponibilidad superior a 3 segundos y una perdida cercana al 30 %. Esto es lógico, pues si bien la tasa de transmisión es constante, y el experimento dura 10 segundos, de forma teórica se calculan unas perdidas del 10% por cada segundo que dure el cambio de red.

Por tanto, UDP no es buena solución para situaciones en las que un flujo de datos constantes sea crítico o que se necesite asegurar la recepción de los paquetes. Habría que trabajar en reducir el tiempo de indisponibilidad por el traspaso, para reducir las perdidas.

8 CONCLUSIONES Y TRABAJOS FUTUROS

Hemos estudiado en profundidad los mecanismos que permiten gestionar la movilidad de un nodo a través de diferentes redes. MIPv6 nos permiten movernos entre diferentes redes sin perder conectividad y manteniendo la misma sesión. Esta tecnología es de gran aplicación, por ejemplo, en el campo de los transportes. Imagine viajar en un tren de Sevilla a Madrid. Por el camino iría pasando bajo la influencia de diferentes redes, pero gracias a MIPv6, usted no perdería la conectividad en ningún momento, evitando una experiencia de usuario negativa.

Esto supone un gran avance, pero todavía podemos ir un poco más lejos. Piense ahora, que al igual que usted, hay mas personas en el tren conectadas a internet. En este caso, se gestionaría individualmente cada conexión provocando un aumento del número de paquetes de señalización empleados. Este proceso se vuelve mucho más eficiente con la aparición de NEMO, pues permite transportar una red completa. En nuestro ejemplo del tren, necesitaríamos un router que haga de puerta al exterior. Este sería el único nodo que registraría su desplazamiento, mejorando el rendimiento de la comunicación y reduciendo la saturación de la red.

Con nuestros escenarios y los test realizados hemos puesto en evidencia la utilidad de estos protocolos, y hemos sentado las bases para que alguien pueda experimentar con la movilidad sin necesidad de un laboratorio hardware.

Hemos probado el rendimiento de los principales protocolos de transporte en nuestros escenarios. A la vista de los resultados encontramos a TCP como el protocolo idóneo para lidiar con la movilidad, pues asegura la recepción de los datos a pesar del momentáneo corte de la comunicación. Como punto negativo, podemos destacar que debido al control de congestión que implementa, el *handover latency* es sensiblemente superior al de UDP. Este por su parte, no es una alternativa fiable pues la perdida de paquetes es total durante el periodo de traspaso. Por tanto, me gustaría proponer como trabajo futuro y partiendo de los escenarios aquí desplegados, la implementación de protocolos como *Mobile IPv6 Fast Handovers* (FMIPv6)[17].

Otra propuesta sería, realizar estos mismos experimentos pero estableciendo comunicación entre varias máquinas. Medir el rendimiento de varios MNs moviéndose a la vez y compararlo con el traspaso de una red móvil en la que haya el mismo número de modos trasmitiendo o recibiendo datos. De esta forma, podremos observar más diferencias en el rendimiento y en la cantidad de tráfico de señalización intercambiado.

Por otro lado, sería algo muy interesante, poder modelar los cables que unen los nodos en GNS3 de forma que no se comporten de forma ideal y se puedan configurar retrasos por propagación o algunas perdidas, para hacerlo más realista. Así podríamos simular el comportamiento con redes que están verdaderamente lejos las unas de las otras.

A titulo personal, me gustaría valorar este proyecto que para mí a supuesto todo un reto. Me ha permitido conocer los entresijos de una tecnología de que la no tenía mucha información, a priori. He aprendido a manejar herramientas de gran utilidad como GNS3. Aunque pueda parecer un proyecto sencillo, no lo ha sido tanto. El hecho de tener varias máquinas, configurarlas, recompilar el kernel, etc. es una labor tediosa. El principal escollo que encontré, es la comunicación de GNS3 con las máquinas virtuales. En un principio empecé usando VMware como virtualizador, pero la integración no era óptima y algunos paquetes se perdían. Estos problemas parece que se han solucionado en la última actualización de GNS3 ahora en Noviembre, que me gustaría probar. Por el camino, he aprendido a lidiar con los problemas que iban surgiendo y a buscar la manera de superarlos, que creo es una de las cosas de las que trata esto de la ingeniería.

[17] Recogido en la RFC 7411 de Noviembre de 2014

ANEXO A: CARACTERÍSTICAS DEL EQUIPO

En este anexo mostraremos las características más relevantes del equipo donde se ha simulado el escenario de movilidad.

- **Hardware:**
  - Macbook Pro con pantalla Retina de 13 pulgadas
  - Intel Core i5 de doble núcleo a 2,7 GHz
  - Turbo Boost de hasta 3,1 GHz
  - 8GB de memoria LPDDR3 a 1.866 MHz
  - 128 GB de almacenamiento flash PCIe
  - Intel Iris Graphics 6100
  - Batería integrada
  - Trackpad Force Touch
- **Sistema Operativo:**
  - Yosemite OS X

REFERENCIAS

Capítulo 1. Introduction

[1] R. Kuntz, J. Lorchat y T. Ernst, «Real-life demonstrations using IPv6 and mobility support mechanisms». Noviembre 2005

R. Kuntz, F. Leiber y T. Ernst, «A Live Light-Weight IPv6 Demostration Platform for ITS Usages». Junio 2005

Enable Project, «Enabling Efficient and Operational Mobility in Large Heterogeneous IP Networks». Marzo 2006

Capitulo 2. Mobile IPv6

[1] Antti Järvinen, «Comparing IPv4 and IPv6 Mobility and autoconfiguration for residential networks». 2002.

[2] S. Thomson, T. Narten y T.Jinmei, «IPv6 Stateless Address Autoconfiguration».RFC 4862. Septiembre 2007.

[3] Autor desconocido, «IPv6 Neighbor Discovery»,
http://www.see-my-ip.com/tutoriales/protocolos/ipv6_neighbor%20discovery.php

[4] E. Nordmark, T. Narten, H. Soliman y W.Simpson, «Neighbor Discovery for IP versión 6», RFC 4861. Septiembre 2007.

[5] C. Perkins, D. Johnsony, J. Arkko «Mobility Support in IPv6», RFC 6275. Julio 2011.

[6] Carlos Dalia Pacheco «Diseño y desarrollo de un simulador de movilidad IPv6 para redes de próxima generación», Trabajo fin de grado. Junio 2015

[7] I. Kandirakis, «Route Optimization for Mobile IPv6 using the Return Routability Procedure. Testbed implementation and security analysis».Tesis. Marzo 2007.

[8] J. Arkko, V. Devarapalli y F. Dupont «Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents», RFC 3776. Junio 2004.

[9] V. Devarapalli y F. Dupont «Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture». RFC 4887. Abril 2007.

[10] Carlos J. Bernardos, Antonio de la Oliva y Maria Calderón , «NEMO. Bringing ubiquity to the Internet access». 2012

[11] Vijay Devarapalli, Ryuji Wakikawa, Alexandru Petrescu, Pascal Thubert, «Network Mobility (NEMO) Basic Support Protocol», RFC 3963. January 2005.

Capitulo 3. Software de virtualización

[1] Lucas D. Ordóñez Pacheco, «La Tecnología de Virtualizacion en las computadoras» Junio 2009.

[2] Autor desconocido, «Maquina Virtual,» https://es.wikipedia.org/wiki/M%C3%A1quina_virtual

[3] Mario Montagud, «Aprendizaje Mediante Simulación de Redes: Análisis, Implantación y Evaluación». 2013

[4] Herbert, Bradley Mark «The Role of Cisco Virtual Internet Routing Lab in network training environments». Julio 2015

[5] «Documentación de GNS3,» disponible en
https://community.gns3.com/community/software/documentation

[6] James Mahon, «GNS3 AS A FEASIBLE TEACHING, TESTING AND DESIGNING TOOL FOR NETWORK DESIGN». Julio 2012

[7] «Documentación de TUNTAP MAC OS X,» disponible en http://tuntaposx.sourceforge.net/

Capitulo 4. Despliegue del Escenario para MIPv6

[1] UMIP «How to install UMIP». disponible en http://umip.org/docs/umip-install.html

[2] UMIP «How to setup a Mobile IPv6 testbed with IPsec static keying». disponible en http://umip.org/docs/umip-mip6.html

Capitulo 6. Despliegue del Escenario para NEMO

[1] UMIP «How to setup a NEMO basic support testbed with IPsec static keying». disponible en http://www.umip.org/docs/umip-nemo.html

GLOSARIO

Estos conceptos han sido extraídos de la web http://www.ipv6.es/es-es/glosario del Gobierno de España o añadidos por el autor de este texto.

Binding. Asociación que se establece entre la dirección HoA y CoA del nodo móvil.

Binding Cache (BC). Conjunto de entradas las cuales disponen de una asociación entre la HoA y la CoA de un nodo móvil. Estas tablas se pueden encontrar en el HA, e incluso en algunos CN.

Binding Update List. Lista mantenida en cada nodo móvil. Contiene un elemento por cada binding que el nodo móvil tiene o está intentando establecer. Ambos registros, direcciones HoA y CoA, están incluidos en esta lista. Los elementos de la lista son eliminados cuando expira el tiempo de vida del binding correspondiente.

Cabecera de extensión. Cabecera que se sitúa entre la cabecera IPv6 y las cabeceras de protocolos de nivel superior, y que permiten agregar funcionalidades adicionales a IPv6.

Care-of Address (CoA). Dirección IP local que identifica a un nodo móvil en su ubicación actual. Si está en la *Home Network*, esta dirección coincidirá con la HoA.

CN. Nodo fijo o móvil que se comunica con el MN.

DHCPv6. Protocolo de configuración con estado (en inglés stateful) que proporciona direcciones IP, direcciones de los servidores DNS y otros parámetros de configuración.

Dirección. Identificador único asignado a nivel de la capa de red a una interfaz o conjunto de ellas, que puede ser empleado como campo de origen o destino en datagramas IPv6.

Dirección MAC. Dirección de nivel de enlace, conocidas comúnmente como dirección física, dirección de hardware o dirección de la interfaz de red.

Encaminador. Nodo que retransmite datagramas que no van destinados a él. En las redes IPv6 los encaminadores envían también anuncios relativos a su presencia y la información de configuración.

Enlace: Segmento o segmentos de una red de área local limitados por encaminadores.

Enlace habitual (home link). Término utilizado en movilidad IP para indicar el enlace en el que el nodo móvil reside cuando está en su red.

Foreign Network (FN). Cualquier red distinta de la red origen en la que se encuentra el MN y en la que adquiere una dirección IP auxiliar.

Host. Véase nodo.

HA. Router de la red origen (*Home Network*) que gestiona la localización del MN. Contiene una lista de MNs registrados y es responsable de hacer llegar al MN aquellos paquetes que son dirigidos a él mientras se encuentra fuera de la red origen.

Home Address (HoA). Dirección IP permanente que se le asigna al MN en su red origen.

Home Network (HN). Red de la que parte el nodo móvil y en la cual se encuentra el HA que gestiona su movilidad.

ICMPv6. Protocolo que proporciona en IPv6 mensajes de error, control, información, diagnóstico, descubrimiento de vecindario, etc.

Interfaz. Nexo físico o lógico de un nodo a un enlace.

IPsec. Conjunto de estándares que proporciona comunicaciones privadas y autenticadas a nivel de red, por medio de servicios criptográficos. IPSEC soporta autenticación a nivel de entidades de red, autenticación del

origen de datos, integridad y cifrado de datos y protección anti-repeticiones.

Neighbour Discovery. Conjunto de mensajes y procesos ICMPv6 que determinan las relaciones entre nodos vecinos. Sustituye a ARP, al descubrimiento de rutas ICMP y al mensaje de redirección ICMP empleados en IPv4. También proporciona detección de vecino inalcanzable.

Nodo. Dispositivo que no puede reenviar datagramas no originados por él mismo. Habitualmente un nodo es el origen y/o el destino del tráfico y por tanto descartará aquello que no esté dirigido específicamente a sí mismo.

Nodo móvil. Un nodo móvil puede moverse de una red a otra sin dejar en ningún momento de estar disponible, ni perder la conexión, manteniendo activas las comunicaciones de niveles superiores. Es la entidad protagonista del esquema de movilidad.

Paquete. Unidad de datos del protocolo (PDU, Protocol Data Unit), que en el caso de IPv6, consta de una cabecera IPv6 y la carga útil.

Prefijo de red. Parte fija de una dirección IPv6 que permite determinar la ruta, rango de direcciones o subred.

Prefijo de sitio. Prefijo de 48 bits que identifica el bloque de direcciones de dicho sitio. La tabla de prefijos almacena dicho prefijo de tal manera que se evita que el tráfico de dicho sitio pueda ser filtrado a Internet.

Protocolo de nivel superior. Protocolo que utiliza IPv6 como transporte y se sitúa en la capa inmediatamente superior a IPv6, por ejemplo ICMPv6, TCP o UDP.

RFC. Paso previo de un documento estándar de Internet (STD), aunque en la actualidad, los fabricantes implementan en sus productos RFCs, sin esperar a que sean STD.

Router. Véase encaminador.

Round-trip delay time (RTT). Tiempo que tarda un paquete de datos enviado desde un emisor en volver a éste, habiendo pasado por el receptor de destino.

Subred. Uno o más enlaces que utilizan el mismo prefijo de 64 bits.

Traspaso. Cuando un nodo móvil cambia de punto de unión a Internet, es decir, que ya no está conectada al mismo link que antes. Si un nodo móvil no está conectado a su home link, se dice que está “fuera de casa”.

Tiempo de indisponibilidad. Intervalo de tiempo que comienza cuando un equipo que estaba recibiendo/enviando paquetes deja de estar disponible, hasta que vuelve a estar operativo.