Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Escenario de movilidad IPv6 basado en una sola interfaz de red

Autor: Daniel Díaz López

Tutor: Juan Antonio Ternero Muniz

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Escenario de movilidad IPv6 basado en una sola interfaz de red

Autor:
Daniel Díaz López

Tutor:
Juan Antonio Ternero Muñiz
Profesor Colaborador

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Proyecto Fin de Carrera: Escenario de movilidad IPv6 basado en una sola interfaz de red

Autor: Daniel Díaz López

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, 2018

El Secretario del Tribunal

Agradecimientos

Me gustaría darle las gracias a mi tutor, Juan Antonio, por su ayuda, conocimientos y paciencia a lo largo de todo el tiempo que ha durado la elaboración de este proyecto, el cual se ha alargado mucho debido sobre todo a mis circunstancias laborales.

También agradecer a mis padres y mi hermano, por apoyarme y ayudarme en todo lo que ha estado en su mano.

Por último, dar las gracias a Mari Carmen por darme las fuerzas para seguir y esforzarme cuando me faltaban la energía o la motivación necesaria, y a mis amigos, por estar siempre a mi lado para lo que haga falta.

Resumen

La aparición de los dispositivos móviles (teléfonos móviles, portátiles, etc.) es una de las principales causas del agotamiento de las direcciones IPv4. A su vez, también da sentido a la necesidad de los protocolos de movilidad marcando así el contexto en el que se ubica este proyecto. La conjunción de IPv6 y un servicio de movilidad como Mobile IPv6 tratan de cubrir las necesidades que ambas problemáticas plantean.

Por tanto, en este proyecto se pretende comprender y evaluar el funcionamiento del protocolo de movilidad MIPv6, a través de un escenario de pruebas donde simular su funcionamiento y medir su rendimiento. En concreto, y como principal particularidad respecto a otros proyectos de movilidad, este escenario contará con un nodo móvil con una sola interfaz de red, lo cual nos aportará una visión algo más próxima a un escenario real.

Con este objetivo en mente, el primer paso será realizar un estudio de los protocolos implicados, como IPv6, MIPv6, OSPFv3, etc.

Una vez sentadas las bases teóricas, se construirá un escenario de pruebas con equipos que simularán el comportamiento de los distintos elementos que intervienen en el protocolo MIPv6. Estos son, el Mobile Node, o nodo móvil, que será el equipo que se moverá de su red origen a una red foránea, el Home Agent, que ejercerá de vínculo entre el Mobile Node y su red de origen y el Correspondent Node, que será el equipo que intentará comunicarse con el Mobile Node desde una red externa.

El objetivo será que la comunicación entre el Mobile Node y el Correspondent Node se vea afectada lo menos posible por los cambios de red que sufrá el Mobile Node, utilizando para ello las herramientas que proporciona el protocolo MIPv6.

Para comprobar si este objetivo se cumple en mayor o menor medida se llevarán a cabo varias pruebas, tanto de funcionamiento a través del protocolo ICMPv6, como de rendimiento simulando comunicaciones UDP y TCP entre los equipos.

A partir de estas pruebas, aportaremos nuestras conclusiones y propuestas de trabajo futuro.

Abstract

The unexpected grow of Internet users, due to the appearance of mobile devices (among other reasons), caused the exhaustion of IPv4 addresses. Together with mobile devices also came the mobility protocols and that is the context of this project. In that context, IPv6 and MIPv6 provide a solution to both problems.

Therefore, in this project we want to get to understand the MIPv6 protocol keys using a test scenario where we can simulate its behaviour and evaluate its performance. Specifically, and making a difference from others mobility projects, our scenario will count with a mobile node with just a network interface, which will bring us closer to the real life.

Aiming for this purpose, first step will be studying the different protocols involved like IPv6, MIPv6 or OSPF3.

Once we have a good knowledge of the theory, we will build our test scenario with different computers that will perform the role of the different elements involved in MIPv6 protocol. These elements are the Mobile Node, which is the device that will leave his home network and move to a foreign network, the Home Agent, which will be the link between the Mobile Node and his home network and, finally, the Correspondent Node, which will try to establish a communication with the Mobile Node from an external network.

The main purpose will be that the communication between Mobile Node and Correspondent Node is disturbed as less as posible by the network movements that the Mobile Node could suffer, using the tools that MIPv6 provides.

To check if we fulfill this purpose, to a greater or lesser extent, we will make some operational tests using ICMPv6 tools, and performance tests establishing different UDP and TCP communication between the devices.

Based on these tests we will come to some conclusion and we will provide some future work proposal.

Índice

Índice de Tablas

Índice de Figuras

Notación

AH	Autentication Header
BA	Binding Acknowledgement
BE	Binding Error
BU	Binding Update
BUL	Binding Update List
CICA	Centro Informático Científico de Andalucía
CIDR	Classless Inter-Domain Routing
CN	Correspondent Node
ESP	Encapsulation Security Payload
FN	Foreign Network
HA	Home Agent
HoA	Home Address
HN	Home Network
IANA	Internet Assigned Number Authority
ICMPv6	Internet Control Message Protocol version 6
IEEE	Institute of Electrical and Electronic Engineers
IKE	Internet Key Exchange
IPsec	Internet Protocol security
IPv4	Internet Protocol versión 4
IPv6	Internet Protocol version 6
MAC	Media Access Control
MIPv6	Mobile IP version 6
MN	Mobile Node
NA	Neighbor Advertisement
NAT	Network Address Translation
ND	Neighbor Discovery
NS	Neighbor Solicitation
OSI	Open System Interconnection
OSPFv3	Open Shortest Path First version 3
RA	Router Advertisement
RFC	Request For Comments
RIR	Regional Internet Registry
RS	Router Solicitation
RTT	Round-Trip delay Time
SA	Security Association
SLAAC	Stateless Address Autoconfiguration
SO	Sistema Operativo
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network

1 INTRODUCCIÓN Y ESTADO DEL ARTE

En este proyecto se pretende construir un escenario de movilidad IPv6 basándose en el creado por mi compañero Miguel Miralles [1], en el que utilizaba el software de movilidad UMIP [2], pero ampliando dicho escenario con un segundo router Cisco y utilizando además, de forma externa, los servicios de conexión IPv6 que nos ofrece la red CICA [3], para obtener de esta forma una red más compleja donde poder estudiar el rendimiento de la movilidad IPv6 [4] con el software UMIP.

Otra novedad que introduciremos es la utilización de una sola interfaz del equipo que ejerce de nodo móvil. Para simular la movilidad, en este caso, jugaremos con los puertos de los routers y con las distintas VLAN [5] que crearemos y a las que asignaremos dichos puertos.

Para hacer todo esto posible, primero necesitamos conocer los aspectos más importantes del protocolo IPv6 por lo que daremos una descripción lo más completa posible de dicho protocolo, así como de otros protocolos (ICMP, OSPF, ND, etc.), en los que se apoya para su correcto funcionamiento. Después, analizaremos las características más importantes de movilidad en IPv6, explicando cada uno de los equipos que intervienen en el proceso, así como protocolos auxiliares que se utilizan y las limitaciones que presenta.

Seguiremos con una descripción del escenario montado, así como de los equipos utilizados para ello, junto con la configuración necesaria en todos ellos para que nuestra red quede correctamente preparada.

Por último, mostraremos capturas y datos de Wireshark [6] obtenidos a partir de varias pruebas que nos permitirán analizar en profundidad el correcto funcionamiento y el rendimiento del servicio de movilidad en nuestro escenario. Acompañaremos estos datos con nuestras conclusiones y sugerencias de trabajo futuro que amplíen y complementen este proyecto.

1.1 Estado del Arte

La situación actual de movilidad en IPv6 va de la mano con la implantación de dicho protocolo a nivel de usuario, ya que es en dispositivos móviles donde cobra sentido la intervención de protocolos de movilidad.

Aunque a lo largo de los últimos años ha sido frecuente leer noticias sobre el agotamiento de las direcciones IPv4, el despliegue de IPv6 varía bastante entre los distintos países. Por ejemplo, en España es prácticamente nulo a nivel de usuario (0,11% en Febrero de 2017) y, sin embargo, hay países donde el despliegue sí está bastante avanzado como Estados Unidos o Bélgica.

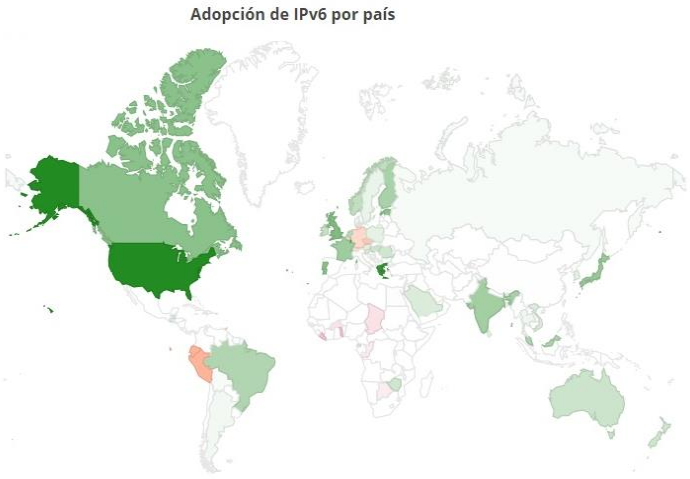


Figura 1.1. Adopción de IPv6 según el pais

Aunque los datos del gráfico anterior son de 2016, son un indicativo claro de que aún queda tiempo para que IPv6 sustituya por completo a IPv4 a nivel de usuario. Sin embargo, el agotamiento de las direcciones IPv4 debido, entre otras causas, a la irrupción de los dispositivos móviles y la asignación ineficiente de direcciones en los primeros años de internet, es inevitable.

A pesar de que se ha conseguido retrasar este agotamiento con medidas como el uso de redes privadas, la aparición de protocolos como NAT o DHCP [7] [8], o la renumeración de redes que han permitido recuperar amplios bloques de direcciones que estaban asignados y sin usar, el último bloque libre de direcciones fue entregado por la IANA (Internet Assigned Number Authority) a las distintas RIRs (Regional Internet Registry) el 3 de Febrero de 2011 [9] [10]. En esta situación, IPv6 se presenta como la única solución a largo plazo.

No obstante, para que IPv6 se establezca definitivamente, todos los elementos que intervienen en Internet, desde los ISP [11] hasta los equipos finales, deben estar preparados para soportar dicho protocolo. Por ejemplo, todos los equipos con menos de 5 años de antigüedad ya incorporan soporte a IPv6, quedando el resto obsoletos en caso de una transición brusca a IPv6.

Por su parte las operadoras también deberán dar un paso adelante para facilitar la llegada de IPv6, adecuando sus infraestructuras y equipos al nuevo protocolo.

Centrándonos propiamente en MIPv6, apenas hay bibliografía y, la que hay, se centra en explicar la solución que propone MIPv6 al problema de movilidad [12]. De hecho, no hay muchos proyectos específicos sobre este tema (KAME, USAGI, UMIP), y la mayoría están cerrados o abandonados [13] [14]. De los citados, sólo UMIP presenta algo de actividad y actualizaciones, que abarcan no sólo MIPv6, sino otros proyectos de movilidad en IPv6 como Proxy MIPv6 o MCoA / DSMIPv6 implementation (Multiple Care-of Addresses Registration y Dual Stack Mobile IPv6 implementation) [15] [16] [17].

En este proyecto utilizaremos la información, instrucciones y archivos de configuración disponibles en el sitio web de UMIP, que nos facilitarán la tarea de simular el comportamiento de los distintos elementos que intervienen en MIPv6 y que veremos en el capítulo 3.

2 INTERNET PROTOCOL VERSION 6 (IPV6)

Para empezar, y antes de explicar el concepto de movilidad, daremos una visión general sobre IPv6, ya que es el pilar fundamental sobre el que trabajaremos y cuyo conocimiento es imprescindible para entender el proyecto.

Para ello explicaremos el porqué de la aparición de IPv6, así como sus aspectos más destacados, de forma que cuando acabe el capítulo podamos centrarnos en el resto del proyecto con una base sólida que ayude a entender los demás apartados.

2.1. Introducción

Cuando, en los años 70, se creó IPv4, era difícil imaginar el imparable crecimiento que tendría el uso de internet en la sociedad actual y, por tanto, la importancia que tendría este protocolo. Con la evolución de la tecnología cada vez son más los dispositivos que necesitan usar IPv4, lo que ha provocado que su rango de direcciones (2^{32}), se haya quedado corto rápidamente a pesar de la aparición de mecanismos para explotar de forma más eficiente dicho rango (CIDR o NAT) [18].

Esta falta de direcciones IP evidenciaba un cuello de botella insalvable en el crecimiento de internet, por lo que la necesidad de un nuevo protocolo que sustituyera y mejorara a IPv4 era innegable. Así, en 1996 fue publicada la RFC 1883 que recogía el protocolo IPv6. Esta RFC quedó obsoleta con la publicación en 1998 de la RFC 2460 que, a su vez, ha sufrido varias actualizaciones (RFCs 5095, 5722, 5871, etc.) [19].

Aprovechando la creación de este nuevo protocolo, se introdujeron mejoras con respecto a IPv4, más allá del mencionado aumento en el rango de direcciones.

Como principales características de este nuevo protocolo podríamos mencionar [20]:

- Mayor espacio de direcciones, pasando de direcciones de 32 bits a direcciones de 128 bits lo que nos da un rango de 2^128 direcciones que soluciona, a priori y por mucho tiempo, la problemática de la falta de direcciones.
- Seguridad en el núcleo del protocolo mediante la utilización de IPSec [21].
- Autoconfiguración de direcciones (SLAAC) [22].
- Nuevo etiquetado de flujo para mejorar servicios de tiempo real
- Movilidad de equipos entre diferentes redes
- Ha sido diseñado para poder añadir nuevas funcionalidades en el futuro sin rediseñar el protocolo.
- Cabecera de tamaño fijo

A continuación, daremos una descripción del formato de la cabecera y sus distintos campos en IPv6, así como del funcionamiento del direccionamiento en este protocolo.

2.1.1 Cabecera IPv6

Un paquete de datos IPv6 está compuesto por dos partes: la cabecera y la carga útil. Los primeros 40 bytes de cada paquete IPv6 componen la cabecera con la siguiente estructura [23]:

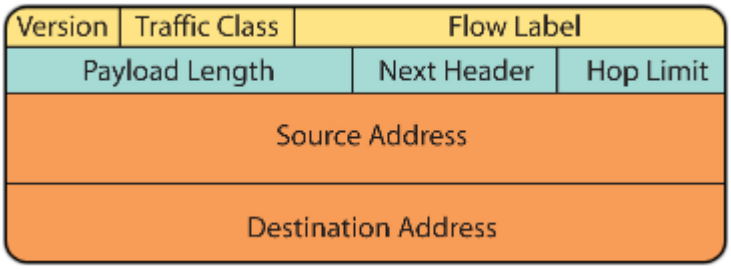


Figura 2.1. Formato cabecera IPv6

Comentemos brevemente la función de cada uno de estos campos:

- *Version*: Este campo de 4 bits contiene la versión del protocolo IP (Internet Protocol) que estamos usando. Al tratarse de IPv6 su valor siempre será 6, mientras que en IPv4 su valor será 4. Cabe destacar que la utilidad de este campo es limitada ya que para distinguir paquetes IPv4 de paquetes IPv6 no se usa este campo sino en el tipo de protocolo que aparece en la cabecera de capa 2.
- *Traffic Class*: También conocido como campo de prioridad de paquetes, este campo de 8 bits se utiliza para distinguir entre las diferentes clases o prioridades de paquete IPv6.
- *Flow Label*: Este campo de 20 bits identifica a conjuntos de paquetes que tienen que ser manejados por los routers intermedios de alguna forma particular. Esto permite, en conjunto con el campo Traffic Class, aplicar mecanismos de calidad de servicio (QoS), clase de servicio (CoS) o servicios en "tiempo real".
- *Payload Length*: Es un entero sin signo de 16 bits que, tal como indica su nombre, nos informa del tamaño de la carga útil del paquete IPv6 en octetos. Es decir, nos informa de la longitud del resto del paquete que sigue a la cabecera, incluidas las cabeceras de extensión.
- *Next Header*: En este campo de 8 bits se identifica el tipo de cabecera que sigue inmediatamente a la cabecera IPv6, normalmente la del protocolo de capa de transporte que contiene la carga útil del paquete. Utiliza los mismos valores que el campo protocolo de la cabecera IPv4, por ejemplo, 6 para TCP o 17 para UDP.
- *Hop Limit*: Valor límite de saltos (8 bits), que puede dar un paquete. El valor de este campo se decrementa en 1 por cada nodo que reenvía el paquete y al llegar a 0, este se descarta.
- *Source Address*: Dirección de origen del paquete IPv6 (128 bits).
- *Destination Address*: Dirección de destino del paquete IPv6 (128 bits).

Como principales ventajas de la cabecera IPv6, respecto a la de IPv4, podríamos destacar que la cabecera de IPv6 tiene tamaño fijo, lo que facilita su tratamiento en los encaminadores, consumiendo estos menos recursos de su procesador. Por otro lado, los paquetes IPv6 pueden ser tratados salto a salto (hop-by-hop) o solo en el nodo destinatario, lo que acelera el paso de los paquetes por los nodos intermedios.

2.1.2 Direccionamiento IPv6

Una de las principales ventajas de IPv6 frente a IPv4 es el enorme rango de direcciones que nos proporcionan los 128 bits que ocupan las direcciones en IPv6 frente a los 32 bits de IPv4, lo cual nos permite dar una dirección única a cada dispositivo o nodo conectado a internet. Esto permite obviar técnicas aplicadas en IPv4 para extender su rango de direcciones, como NAT que, aunque efectivas, también limitaron la capacidad de conexión punto a punto.

2.1.2.1 Formato de direcciones IPv6

La RFC 4291 establece 3 formatos para representar una dirección IPv6 [24]. El primer formato establece que una dirección está compuesta por 8 grupos de cuatro dígitos hexadecimales separados por dos puntos.

A continuación, mostramos un ejemplo de una dirección IPv6 valida:

2001:720:c18:b1:0:0:0:1

Como puede observarse, cualquier grupo de 4 ceros puede ser reducido y expresado como un solo 0, e incluso, si son varios grupos de ceros consecutivos, pueden ser omitidos y expresados por un doble signo ":". Este sería el segundo formato definido en la RFC. Por tanto, la dirección anterior también puede ser expresada de la siguiente manera:

2001:720:c18:b1::1

Es importante señalar que, en este formato, sólo puede utilizarse una vez la agrupación de ceros en una misma dirección. Por ejemplo, la dirección 2001:0:0:b1:0:0:0:1 debe abreviarse así 2001:0:0:b1::1.

Por último, el tercer formato sería el utilizado para expresar direcciones IPv4 y tendría este aspecto X:X:X:X:X:X:d.d.d.d siendo las "d" los campos de una dirección IPv4 estándar. Por ejemplo, como parte final de una dirección IPv6 la dirección 13.1.14.1 se expresaría como ::13.1.14.1

2.1.2.2 Tipos de direcciones

Dentro del formato descrito en el apartado anterior, podemos encontrar varios tipos de direcciones IPv6:

- *Unicast*: Identifica a una interfaz concreta de un nodo IPv6.
- *Multicast*: Identifica a un grupo de interfaces IPv6 y, por tanto, un mensaje multicast será procesado por todas ellas.
- *Anycast*: Identifica a un grupo de interfaces, pero sólo será procesado por una de ellas (la más cercana al nodo emisor).

Dentro de las direcciones *Unicast* pueden distinguirse varios tipos. A continuación, mostramos algunos de ellos:

- *Globales*: Direcciones genéricas en IPv6. Contienen un prefijo de enrutamiento global, asignado a las distintas organizaciones, un identificador de subred, que permite para organizar la red internamente como dicha organización considere oportuno y, por último, un identificador de interfaz para identificar a cada uno de los host.
- *Link-Local*: Se usa en mecanismos de autoconfiguración, descubrimiento de vecinos, etc. y su ámbito se reduce a la red local. Su formato es el siguiente: FE80::<Id. Interfaz>/10.
- *Loopback*: Cada dispositivo la usa para referirse a sí mismo. Su formato es ::1.
- *Unique-Local*: Direcciones con alta probabilidad de ser únicas, pero pensadas para comunicaciones locales, por lo que no son enrutables globalmente, sino que su ámbito se reduce a áreas reducidas a las que pertenecen, por ejemplo, redes corporativas.

2.1.2.3 Redes IPv6

Las redes en IPv6 se expresan con notación CIDR (Classless Inter Domain Routing). Una red o subred IPv6 debe estar compuesta por un conjunto de direcciones contiguas con un tamaño potencia de 2.

Los bits iniciales de una dirección IP (iguales para todos los miembros de la red o subred), forman el prefijo de red. El tamaño del prefijo de red en bits, aparece, en formato decimal, justo detrás de la dirección IPv6 separado por el símbolo /.

Por ejemplo, si escribimos 2001:720:c18:b0::/64, estamos diciendo que la dirección de red es 2001:720:c18:b0 y que comprende todas las direcciones que van de 2001:720:c18:b0:: a 2001:720:c18:b0:ffff:ffff:ffff:ffff.

Otra novedad interesante es que, a diferencia de IPv4, no se reserva una dirección para identificar a la subred

ni para difusión.

2.2 ICMPv6

ICMPv6 (Internet Control Message Protocol Version 6) [25], es una nueva versión del protocolo ICMP y tiene un papel fundamental en el funcionamiento de IPv6. ICMPv6 da soporte a protocolos como ND (Neighbor Discovery), MLD (Multicast Listener Discovery) o MIPv6 (Mobile IPv6), el cual es el objeto de nuestro proyecto y que analizaremos más en profundidad en el capítulo 3 [26] [27].

Los mensajes ICMPv6 van encapsulados como carga útil dentro de los paquetes IPv6, se identifican con el valor 58 en el campo "Next Header" y tienen el siguiente formato:

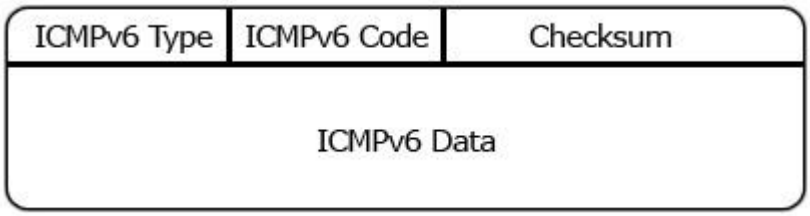


Figura 2.1. Formato mensaje ICMPv6

El campo "Tipo" de 8 bits sirve para distinguir si el mensaje es de error o de información. Concretamente, el bit más significativo es el que los diferencia. Si el bit más significativo vale 0 (Valores de 0 a 127) se tratará de mensajes de error. Por el contrario, si vale 1 (128-255) indicará que se trata de un mensaje de información.

El campo "Código" dependerá del tipo de mensaje y se usa para crear un nuevo subnivel de clasificación en los mensajes. Por último, el campo "Checksum" sirve para detectar errores en el mensaje ICMP y en parte del mensaje IPv6.

Como ya hemos dicho, los mensajes ICMPv6 pueden ser de información o de error. A continuación, mostramos una lista de los principales mensajes de ambos tipos.

Type	Meaning
1	Destination Unreachable
2	Packet Too Big
3	Time Exceeded
4	Parameter Problem
128	Echo Request
129	Echo Reply
130	Group Membership Query
131	Group Membership Report
132	Group Membership Reduction
133	Router Solicitation
134	Router Advertisement
135	Neighbor Solicitation
136	Neighbor Advertisement
137	Redirect
138	Router Renumbering

Figura 2.2 Mensajes ICMPv6

De todos ellos, algunos los veremos en los siguientes apartados ya que se usan para dar soporte a las nuevas funcionalidades del protocolo ICMPv6. Por ahora vamos a dar una breve descripción de los mensajes estándar de ICMP:

- *Destination Unreachable (1)*: Mensaje de error de destino inalcanzable que informa al host remitente que no es posible entregar un paquete
- *Packet Too Big (2)*: Mensaje de error que informa al host remitente de que el paquete es demasiado grande para ser reenviado.
- *Time exceeded (3)*: Mensaje de error que informa al host remitente de que el límite de saltos de un paquete IPv6 ha caducado (su valor ha llegado a 0).
- *Parameter Problem (4)*: Mensaje de error que informa al host remitente de un error al procesar el encabezado IPv6 o un encabezado de extensión IPv6.
- *Echo Request (128)*: Mensaje informativo que se utiliza para determinar si un nodo IPv6 está disponible en la red
- *Echo Reply (129)*: Mensaje informativo que se utiliza como respuesta a un mensaje Echo Request.

Como ya mencionamos antes, ICMPv6 da soporte a otros protocolos como Neighbor Discovery, el cual pasamos a analizar en el siguiente apartado.

2.2.1 Neighbor Discovery (ND)

Neighbor Discovery es un protocolo definido en la RFC 4861 que sustituye, y completa con nuevas funcionalidades, al protocolo ARP (Address Resolution Protocol) en IPv6 [28]. Así, ND permite a un nodo descubrir a otros nodos vecinos con los que comparte red, obteniendo sus direcciones y detectar si dicha dirección cambia o si dicho nodo deja de ser alcanzable.

Además de esto, permite a un equipo descubrir routers vecinos, autoconfigurar su dirección IP, conocer el prefijo de la red a la que pertenece o conocer parámetros de dicha red.

Por último, los routers hacen uso de este protocolo para anunciar su presencia, configurar parámetros de otros equipos o determinar el siguiente salto para un paquete que se deba reenviar.

Para ello, como ya hemos dicho, utiliza los siguientes mensajes de ICMPv6:

- *Router Solicitation (RS)*: Un equipo envía el mensaje RS para pedir a los routers vecinos que anuncien su presencia lo antes posible. El campo *tipo* de la cabecera ICMPv6 será 133.
- *Router Advertisement (RA)*: Mensaje generado periódicamente por un router cada cierto tiempo y generado también como respuesta a un mensaje RS. Contiene información como el prefijo de la red, la puerta de enlace, el límite de saltos, orden de usar o no la autoconfiguración de direcciones, así como el tiempo de vida de dichas direcciones. El campo *tipo* de la cabecera ICMPv6 será 134.
- *Neighbor Solicitation (NS)*: Este tipo de mensaje se envía con el objetivo de conocer la dirección de capa de enlace de un nodo vecino, comprobar una dirección duplicada o determinar si un nodo vecino sigue siendo alcanzable. El campo *tipo* de la cabecera ICMPv6 será 135.
- *Neighbor Advertisement (NA)*: Este mensaje se genera como respuesta a un NS o para anunciar un cambio en la dirección de capa de enlace. El campo *tipo* de la cabecera ICMPv6 será 136.
- *Redirect*: Mensaje generado por un router para informar a un nodo si descubre un mejor siguiente salto (next hop), para un destino específico. El campo *tipo* de la cabecera ICMPv6 será 137.

2.2.2 Stateless Address Autoconfiguration (SLAAC)

Como comentamos al inicio del capítulo, una de las ventajas de IPv6 es que permite la autoconfiguración de direcciones. Para ello, hace uso de varios mensajes de ICMPv6 comentados en el apartado anterior como son *Router Solicitation, Router Advertisement y Neighbor Solicitation*.

Primero, al conectarse la interfaz de un nodo a una red, esta se asigna un dirección link-local, con la que envía un mensaje RS. El router vecino contestará con su correspondiente RA en el que irá incluido el prefijo de red. A partir de dicho prefijo y de su dirección MAC el nodo obtendrá su dirección IPv6.

Este mecanismo tiene ciertos problemas de privacidad, ya que una dirección así generada lleva implícita la dirección MAC, por lo que, aunque cambie el prefijo de red, el identificador del equipo será siempre el mismo, lo que permitiría rastrear el dispositivo y su actividad. Para paliar este defecto, en la RFC 4941 se implementó una extensión de privacidad que utilizaba números aleatorios para generar la dirección IPv6. Es por eso que al utilizar SLAAC normalmente se generan dos direcciones IPv6.

Por otra parte, Una vez el nodo obtiene su dirección IPv6, debe comprobar que dicha dirección no está duplicada enviando un mensaje NS con su dirección como dirección de destino y dirección de origen sin especificar[1] (::/128).

Además de SLAAC, una dirección IPv6 puede asignarse de forma manual o utilizando mecanismos dinámicos como DHCPv6, según convenga en cada caso.

2.3 IPSec

Como dijimos al principio de este capítulo, una de las novedades que incorpora IPv6 es la inclusión por defecto de IPSec (Internet Protocol Security). Como su propio nombre indica, este protocolo se encarga de la seguridad en IPv6. Además, y como principal característica, se trata de un protocolo de nivel de red (capa 3), lo que le permite dar soporte a protocolos de capa 4 como TCP o UDP.

IPSec ofrece, entre otros, los siguientes servicios de seguridad:

- Integridad y autenticación del origen de los datos
- Confidencialidad
- Detección de repeticiones
- Control de acceso: autenticación y autorización
- No repudio

Para ello IPSec cuenta con una serie de componentes que analizaremos brevemente a continuación:

- *Protocolos de seguridad*. IPSec cuenta con 2 protocolos que proporcionan mecanismos de seguridad para proteger tráfico IP. Estos son AH (Authentication Header) y ESP (Encapsulating Security Payload) [29] [30].

  Authentication Header permite garantizar la autenticación e integridad de los datagramas IP, así como el no repudio si se eligen los algoritmos criptográficos adecuados.

[1] Utilizará la dirección multicast ff02::1, que identifica a todos los nodos de la red local, para recibir la posible respuesta.

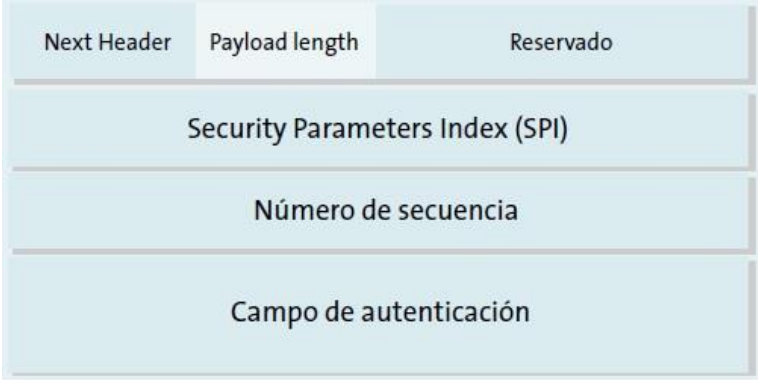


Figura 2.3 Authentication Header

Por otro lado, ESP proporciona confidencialidad y, opcionalmente, autenticación y protección de integridad.

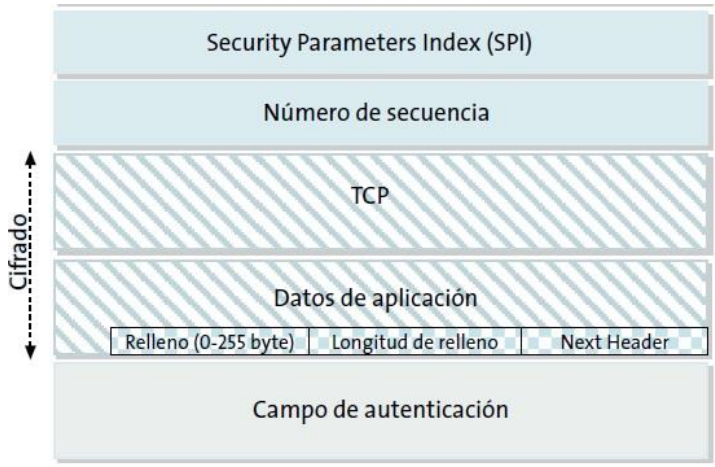


Figura 2.4 Formato ESP

- *Asociaciones de Seguridad (SA)*. Para establecer un servicio de seguridad (AH, ESP), primero debe establecerse una asociación entre las partes implicadas de forma que ambos extremos de la comunicación puedan ponerse de acuerdo, tanto en las claves criptográficas como en otros parámetros de la comunicación. Para ello existen las asociaciones de seguridad, canales de comunicación unidireccional que conecta dos nodos. Por tanto, como la comunicación debe ser bidireccional, cada conexión IPSec consta de dos SAs, una por cada sentido de la comunicación.
- *Administración de Claves*. Como hemos comentado, para establecer un servicio de seguridad, ambos extremos de la comunicación deben ponerse de acuerdo en distintos parámetros. Esta configuración puede llevarse a cabo de forma manual, o de forma automática, mediante el protocolo IKE (Internet Key Exchange), cuya última versión viene definida en la RFC 7296 (IKEv2) [31] [32].
- *Algoritmos de Autenticación y Encriptado*. En el RFC 7321 se establecen una serie de algoritmos de autenticación y encriptado que deben usarse, o no, para AH o ESP.
    - Para AH se recomienda utilizar algoritmos de autenticación como AES-GCM, AES-GMAC o AES-XCBC-MAC-96, y evitar usar el algoritmo DES-CBC.
    - Para ESP se recomienda utilizar algoritmos de autenticación como HMAC-SHA1-96, AES-GMAC o AES-XCBC-MAC-96. También se recomienda utilizar algoritmos de encriptado como AES-CBC, AES-CTR o TripleDES-CBC, evitando utilizar el algoritmo DES-CBC
    - Por último, en caso de querer utilizar ESP para proporcionar autenticación y confidencialidad, se recomienda utilizar los algoritmos AES-GCM o AES-CCM.

2.4 OSPFv3

Para finalizar este capítulo, vamos a describir de forma concisa las características del protocolo de encaminamiento OSPF (Open Shortest Path First). Aunque en nuestro proyecto usaremos OSPFv3 (apta para IPv6), primero vamos a analizar las características generales del protocolo para después nombrar alguna de las novedades que aporta esta última versión.

Como hemos dicho, OSPF es un protocolo IGP (Interior Gateway Protocol), de encaminamiento para IP (encapsula sus mensajes directamente en paquetes IP con número de protocolo 89). Se basa en el algoritmo de Dijkstra de estado del enlace para calcular la ruta óptima entre dos nodos cualesquiera de un sistema autónomo (AS). Su versión para IPv4, OSPFv2, viene definida en la RFC 2328 y la versión para IPv6, OSPFv3, en la RFC 5340 [33] [34].

2.4.1 Mensajes Importantes

OSPF hace uso de varios mensajes que intercambian los distintos routers y que le permite recoger la información del estado de los enlaces que componen el sistema autónomo y compartir dicha información con los routers vecinos. Esta información se almacena en la llamada LSDB (Link-State Database) para cada router, de forma que cada enlace que compone la red se corresponde con una entrada en la base de datos y lleva asociado un coste. A continuación, describimos los mensajes más importantes:

- *Hello*: Mensaje de saludo que se envía periódicamente. Permite a un router descubrir otros routers adyacentes. Además de establecer un primer contacto entre dispositivos vecinos, permite intercambiar diversos parámetros OSPF.
- *Database Description*: Contienen descripciones de la topología de la red o de un área. Sirven para inicializar la LSDB de un router
- *Link-State Request*: Los envía un router para obtener información actualizada de un enlace concreto.
- *Link-State Update*: Mensaje con información actualizada sobre el estado de un enlace que se envía como respuesta a un Link-State Request o en caso de que el estado de un enlace cambie. Cuando un router recibe este mensaje, actualiza su LSDB. Dentro de este tipo de mensaje, hay varios tipos de LSA (Link-State Advertisement), según quien lo envíe, su contenido y el alcance de este.
- *Link-State Acknowledgment*: Mensaje para aportar fiabilidad al proceso de intercambio de mensajes y se envía como confirmación tras recibir un Link-State Update.

2.4.2 Funcionamiento

En este apartado daremos una descripción muy resumida del proceso que se sigue entre los routers que componen un sistema autónomo con OSPF.

Cuando un router se inicia por primera vez mandara un mensaje *Hello* para comprobar si hay routers vecinos que usen OSPF y seguirá mandándolos periódicamente para descubrir posibles nuevos routers vecinos que se añadan a la red.

Cuando se determina que hay un nuevo router adyacente, se envían los mensajes *Database Description* quedando así las LSDB inicializadas. A partir de este momento, dicho router estará listo para inundar la red con sus mensajes *Link-State Update* informando del estado de sus enlaces, así como de cualquier cambio que detecten en la topología de la red.

A su vez recibirá los *Link-State Update* enviados por otros routers y les responderá con sus correspondientes *Link-State Acknowledgment*. Por otro lado, un router podrá solicitar información sobre un enlace concreto a través de un *Link-State Request*.

Como dijimos al principio de este capítulo, toda esta información se almacena en la LSDB, teniendo cada router una copia de la misma. De esta forma obtienen una visión neutral de las distintas conexiones que componen el sistema autónomo. A partir del conocimiento de la red que le ofrece la LSDB, cada router debe

formarse su propia perspectiva, lo que le permitirá determinar el camino más corto entre él y los demás. Para ello, cada router usa la LSDB para crear su *Shortest Path First Tree (SPF Tree)*.

Este "árbol" SPF cuenta con el router que lo crea como punto más alto y, a partir de él, nace una rama por cada router o red directamente conectado a él y, a su vez, de cada uno de estos router o redes, nacen nuevas ramas correspondientes a sus respectivas conexiones y así hasta cubrir el sistema autónomo completo.

De esta forma el router ya podrá calcular el coste mínimo que le supone alcanzar cualquier router o red del sistema autónomo o del área a la que pertenece, por lo que el router creará una tabla de enrutamiento con una entrada por cada red, con el coste que le supone alcanzarla y el router al que debe mandar el mensaje como primer salto para alcanzarla. Por último, como es de esperar, si la LSDB cambia el SPF Tree será recalculado.

2.4.3 Áreas

Para terminar con la explicación de OSPF, y antes de enumerar las novedades impuestas por su versión OSPFv3, vamos a analizar brevemente el concepto de área en OSPF.

Si el sistema autónomo en el que se está utilizando OSPF es relativamente pequeño, todos los routers tendrán el mismo rol, tal como hemos descrito hasta ahora, ya que serán iguales los unos respecto a los otros, en lo que al protocolo OSPF se refiere. Sin embargo, si el tamaño del sistema autónomo aumenta, llegará un punto en el que esta estructura básica resultará ineficiente, ya que el número de LSAs que recorrerán la red será excesivo y las LSDB alcanzaran un tamaño superior al que puedan almacenar y procesar los routers.

En esos casos, OSPF permite establecer una estructura jerárquica más compleja. En esta estructura, el sistema autónomo deja de considerarse una sola estructura unitaria, sino que se divide en distintas áreas, cada una de las cuales contiene un número de routers adyacentes.

Estas áreas estarán numeradas, identificadas por un área ID de 32 bits con el mismo formato que una dirección IPv4 (sin serlo), y serán gestionadas por los routers en su interior casi como si se trataran de sistemas autónomos. Sin embargo, las distintas áreas estarán interconectadas de forma que la información de enrutamiento pueda ser difundida por el sistema autónomo completo.

Aunque hay varios tipos de áreas, según el tipo de tráfico que procesan y otros parámetros, no vamos a entrar mucho en detalle en este aspecto. Simplemente vamos a destacar el *Backbone Area* o *Área 0*, con área ID 0.0.0.0, que será el área central de un sistema autónomo y a la que estarán directamente conectadas las demás áreas. Ocupa por tanto el lugar predominante en esta estructura jerárquica.

Como es lógico, en esta nueva estructura, no todos los routers son iguales, sino que según su ubicación en las distintas áreas cumplirán diferentes roles. Estos son algunos ejemplos:

- *Internal Router*: Están conectados únicamente a otros routers dentro de su misma área. Sólo mantienen una LSDB de su área y no saben nada del estado del enlace de otras áreas.
- *Area Border Router (ABR)*: Estos routers estarán conectados a routers que pertenecen a distintas áreas, por lo que deberán mantener una LSDB por cada área a la que estén conectados.
- *Backbone Router*: Forman parte del Backbone Area. Por definición, todos los ABR son, a su vez, Backbone Routers. Sin embargo, puede haber también Backbone Routers que sólo estén conectados a otros Backbone Routers y, por tanto, sólo pertenecerían al área 0.
- *Autonomous System Boundary Router (ASBR)*: Conectan el sistema autónomo con un dominio externo

2.4.4 OSPFv3

Para terminar con este apartado, y con este capítulo, nombraremos alguna de las novedades que incluyó la versión OSPFv3 respecto a su versión anterior (OSPFv2):

- Anuncia rutas IPv6, en lugar de IPv4.
- En OSPFv3 se utilizan direcciones link-local para la comunicación entre routers.
- En OSPFv3 se utiliza IPSec para la autenticación de los mensajes.
- Se añaden nuevos tipos de LSA.
- Incluye además diferencias de configuración en los routers que lo utilizan, en nuestro caso, Cisco.

3 MOVILIDAD IPV6

La movilidad en IPv6 constituye el núcleo del proyecto, y es su estudio y puesta a prueba el principal objetivo de este. Por tanto, primero debemos explicar en qué consiste el concepto de movilidad y cuál es su importancia en el futuro de IPv6. Además, explicaremos en detalle su funcionamiento, los protocolos en los que se apoya y sus limitaciones, para poder adentrarnos con seguridad en el desarrollo práctico de un escenario de pruebas.

3.1 Introducción

El concepto de movilidad no tiene mucho sentido si pensamos en equipos grandes, como los PCs clásicos ya que lo normal es que no se muevan de su red por lo que su dirección IP no cambiará. Sin embargo, con la aparición de los portátiles, dispositivos móviles o tablets el panorama es completamente distinto. Estos equipos, a priori, deben ir cambiando de red, conforme su usuario se vaya moviendo, por lo que su dirección IP irá cambiando. Esta situación da lugar a una problemática. Si cualquiera de estos equipos tiene establecida una comunicación con otro equipo, al cambiar de red y, por tanto, de IP, cuando el otro equipo intente comunicarse con él, no podrá ya que ya no podrá localizarlo con su antigua dirección. Si, además, teníamos establecida, por ejemplo, una conexión TCP/IP, está se rompería al cambiar de dirección.

Ante este problema, una posible solución sería que cada equipo se identificara con una dirección IP global única, fácilmente aplicable en IPv6 con su nuevo rango de direcciones. Sin embargo, esto obligaría a los routers a mantener tablas de enrutamiento de un tamaño excesivo con una entrada por equipo, lo que daría lugar a un problema aún mayor.

Descartada esta opción, sólo queda introducir los protocolos de movilidad como son IP Mobility Support for IPv4 (RFC 5944) o Mobility Support in IPv6 (RFC 6275). Mobile IPv6 presenta una serie de ventajas con respecto a su versión en IPv4 como son [35] [36]:

- La característica de optimización de ruta, que explicaremos más adelante en este capítulo, está diseñada para IPv6. En IPv4 puede implementarse con una serie de extensiones opcionales que no están disponibles en todos los equipos.
- Características inherentes a IPv6 como el protocolo Neighbor Discovery o la autoconfiguración de direcciones, permiten a la movilidad IPv6 prescindir de una entidad obligatoria en movilidad IPv4, como es el Foreign Agent

3.2 Fundamentos

En este apartado presentaremos el funcionamiento estándar de la movilidad en IPv6, así como las entidades que intervienen en él, el formato de la cabecera y los mensajes más importantes del protocolo.

El elemento central de la movilidad en IPv6 es el *Mobile Node (MN)* o *Nodo Móvil*. Este nodo se irá moviendo por las distintas redes y el objetivo de todo el protocolo es que no deje de estar accesible, manteniendo en todo momento la misma dirección IP.

Inicialmente el nodo móvil se encontrará en su red origen o *Home Network (HN),* donde obtendrá su dirección IP, que deberá mantener incluso cuando se mueva a otra red. Esta dirección es la que llamaremos *Home Address (HoA)*[1]. Mientras el MN se mantenga en su HN, los paquetes que lleguen con dirección de destino la

[1] Usaremos las siglas HoA para diferenciar entre Home Address y Home Agent (HA)

HoA, serán encaminados de forma tradicional por el router de acceso a esta red.

Supongamos ahora que el MN comienza a moverse y cambia de red, a la que llamaremos *Foreign Network (FN)*. Este cambio o migración de red es lo que se conoce como *Traspaso o Handover*. En esta nueva red, el nodo móvil solicitará mediante un RS una nueva dirección IP, que obtendrá cuando sea contestado mediante un mensaje RA. Esta nueva dirección es la que se conoce como *Care of Address (CoA)*.

Al principio del apartado dijimos que el nodo móvil debería mantener la misma dirección IP durante todo el proceso. Esta dirección va a ser la HoA, sin embargo, en este momento el MN tiene como dirección la CoA, por lo que si un equipo externo al que se conoce como *Nodo Corresponsal o Correspondent Node (CN)* intenta comunicarse con el MN, lo hará utilizando su HoA como dirección de destino, por lo que no al MN no le llegarían esos mensajes.

En esta situación, interviene el *Home Agent (HA)*, otra entidad funcional que tiene un papel clave en la movilidad. Ubicado en la HN, el HA será el encargado de recoger los mensajes que tengan la HoA como dirección de destino y reenviarlos a la dirección CoA actual del nodo móvil.

Para que esto sea posible, el HA debe conocer dicha dirección CoA. Esto se consigue estableciendo una asociación o *binding* entre las direcciones HoA y CoA en el HA y en el MN. Para ello, una vez que el MN obtiene su CoA registra esta asociación en la *Binding Update List (BUL)* e informa al HA con un mensaje *Binding Update (BU)*. Al recibirlo, el HA registra esta asociación en una tabla llamada *Binding Cache*, que se irá actualizando conforme vaya recibiendo nuevos BU del MN o si expira el tiempo de vida de alguna de las entradas existentes. Por último, como respuesta el BU, el HA envía un mensaje *Binding Acknowledgment (BA)*.

Una vez registrada la asociación entre HoA y CoA, el HA ya reenviará los paquetes que lleguen al MN, sin importar su ubicación. Por otro lado, queda explicar cómo contesta el MN a los mensajes que le llegan reenviados por el HA. Puede hacerlo de dos formas, usando al HA como intermediario o directamente al CN.

La primera de estas opciones se lleva a cabo estableciendo un túnel bidireccional entre el MN y el HA. El HA interceptará todos los paquetes con dirección de destino HoA y los reenviará a través de un túnel IPv6-in-IPv6 a la CoA. El MN utilizará este mismo túnel para enviar su respuesta y, por último, el HA reenviará esta respuesta al CN utilizando como dirección de origen la HoA para evitar problemas con las ACL en el FN.

En caso de querer responder a los mensajes directamente al CN, lo que se conoce como *Route Optimization*, el CN debe tener soporte a MIPv6. En este caso, el MN, tras recibir el paquete del CN, a través del HA, enviará un BU directamente al CN, usando esta vez la CoA como dirección de origen.

Llegados a este punto, podríamos ver un posible error ya que el CN envió un paquete a la HoA y recibe respuesta de la CoA. Esto podría dar problemas en las capas superiores del CN. Para evitar esta situación, el MN añade una cabecera de extensión llamada *Destination Option* donde se incluye la HoA. El CN, al recibir este paquete sustituye la CoA por la HoA, de forma que las capas superiores sólo ven la HoA[1].

Utilizar *Route* Optimization tiene ciertas ventajas, como que aprovecha la ruta más corta entre el CN y el MN, o la independencia respecto al HA aunque, como ya hemos dicho, obliga al CN a soportar MIPv6.

[1] Algo similar ocurre con el mensaje de respuesta del CN. Lo veremos en el apartado 3.3.3

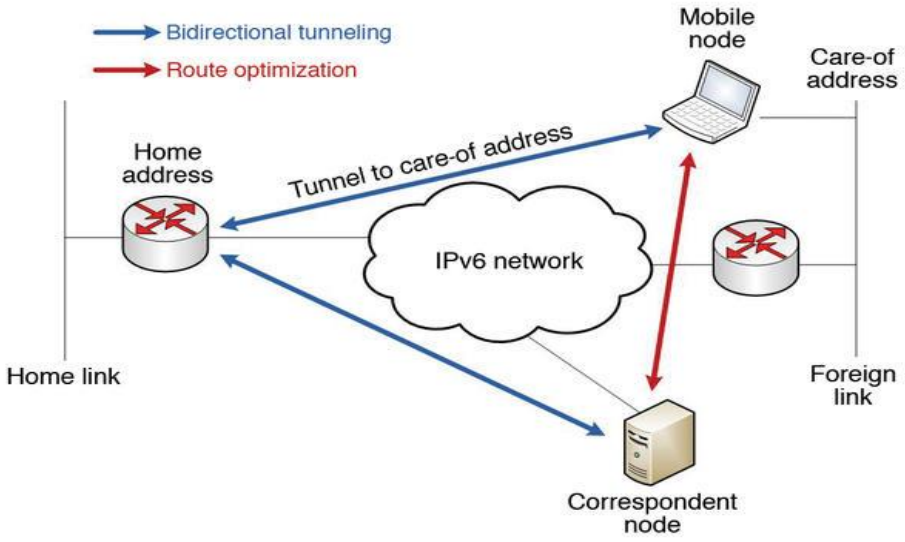


Figura 3.1 Esquema de comunicación MIPv6

3.2.1 Cabecera MIPv6

Ya tenemos una visión general de los elementos principales que intervienen en el proceso de movilidad y la función que cumplen. A continuación, vamos a analizar el formato de la cabecera que usa el protocolo MIPv6, como paso previo a analizar más en profundidad los principales mensajes (Binding Update, Binding Ack, etc.).

Para empezar, diremos que esta cabecera se identifica en el campo *Next Header* de la cabecera previa con el valor 135. En la siguiente figura podemos observar su formato:

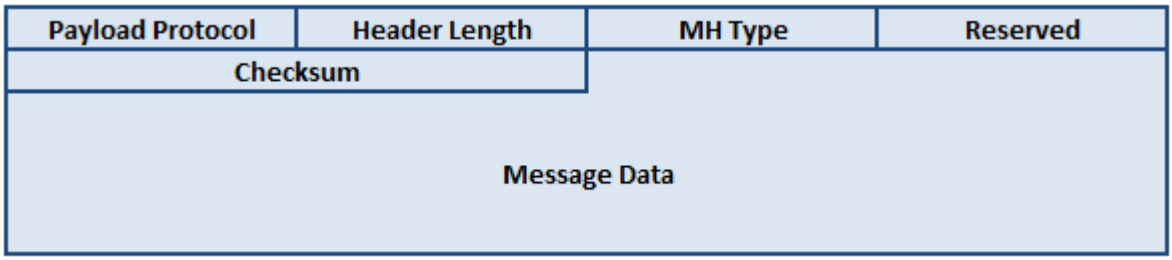


Figura 3.2 Cabecera MIPv6

Veamos una breve descripción de cada campo.

- *Payload Protocol*: El Protocolo de Carga Útil (8 bits), identifica el tipo de cabecera que sigue a la de movilidad, tomando los mismos valores que para el campo *Next Header* de la cabecera de IPv6.
- *Header Length*: Entero sin signo (8 bits) que indica la longitud de la cabecera de movilidad en unidades de 8 octetos, excluyendo los primeros 8. Debe ser múltiplo de 8 octetos.
- *MH Type*: selector de 8 bits para identificar el tipo de mensaje. Algunos de estos tipos los veremos en el siguiente apartado.
- *Reserved*: Campo de 8 bits reservado para futuros usos. Debe ser puesto a 0 por el que envía el mensaje e ignorado por el que lo recibe.
- *Checksum*: Entero sin signo de 16 bits para verificar que la cabecera de movilidad sea correcta.
- *Message Data*: Campo de longitud variable con los datos del mensaje

3.2.2 Tipos de Mensaje

El abanico de mensajes de movilidad IPv6 es amplio. Sin embargo, en este apartado vamos a centrarnos sólo en algunos de ellos, concretamente en los que van a aparecer a lo largo de nuestras pruebas.

Binding Update (BU): Como dijimos en la descripción del proceso de movilidad, el nodo móvil emplea este mensaje para notificar a otros nodos su nueva dirección CoA. El campo *MH Type* toma el valor 5 para este mensaje y su formato es el siguiente:

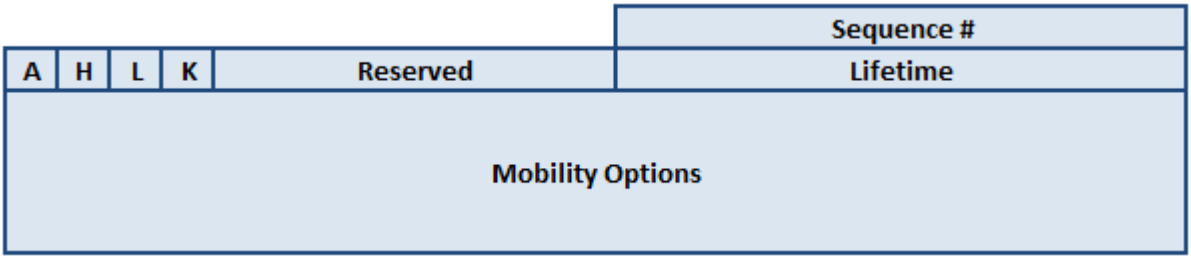


Figura 3.3 Formato Binding Update

Veamos una breve descripción de cada campo:

- *Sequence #*: Entero sin signo de 16 bits. Se usa en el nodo receptor para ordenar los BU recibidos y por el nodo emisor para identificar a que BU se corresponde cada BA recibido.
- *Acknowledge (A)*: El nodo móvil pone este bit a 1, si quiere recibir un BA como respuesta.
- *Home Registration (H)*: El nodo móvil pone este bit a 1 para solicitar que el nodo receptor actúe como su HA. La dirección de destino del paquete debe ser la de un router que comparte el mismo prefijo de subred que la HoA del nodo móvil.
- *Link-Local Address Compatibility (L)*: Este bit se pone a 1 cuando la HoA enviada por el nodo móvil tiene el mismo identificador de interfaz que la dirección *Link-Local* del nodo móvil.
- *Key Management Mobility Capability (K)*: Si este bit está a 0, el protocolo usado para establecer las asociaciones de seguridad IPSec no soportará traspasos, por lo que tendría que volver a ser ejecutado. Si se configura manualmente IPSec, este bit debe ponerse a 0. Este bit sólo es válido en BU enviados al HA y debe ser puesto a 0 en cualquier otro BU. Por último, los CN deben ignorar este bit.
- *Reserved*: 12 bits sin usar, deben ser puestos a 0 por el emisor e ignorados por el receptor.
- *Lifetime*: Entero sin signo de 16 bits. Es el número de unidades de tiempo (4 segundos), restantes para que la asociación (*binding*) se considere caducada. Si llega a 0, la entrada en la *Binding Cache* correspondiente a esa asociación debe ser borrada.
- *Mobility Options*: Campo de longitud variable de forma que la longitud total de la cabecera de movilidad sea múltiplo de 8 octetos. Las opciones que se pueden configurar son diversas, pero vamos a destacar la *Alternate Care-of Address Option* que nos permite utilizar una dirección CoA para la asociación, diferente a la que contiene la cabecera IPv6, siempre que la dirección alternativa sea correcta (unicast y encaminable). Por otra parte, si no se incluye ninguna opción, deben incluirse 4 octetos de relleno en este campo.

Binding Acknowledge (BA): Este mensaje se usa para confirmar la recepción de un mensaje BU. El campo MH Type tomará el valor 6. En la siguiente figura podemos ver el formato del mensaje y a continuación describiremos sus campos.

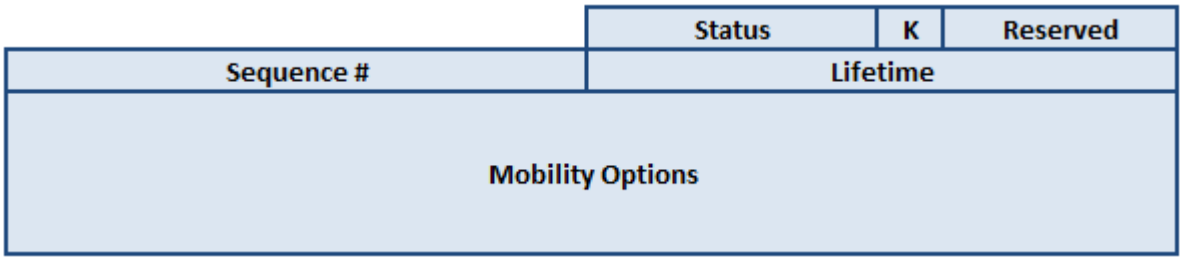


Figura 3.4 Formato Binding Acknowledgment

- *Status*: Entero sin signo de 8 bits. Se enviará un valor menor de 128 para aceptar el BU o un valor mayor para indicar que el BU ha sido rechazado por el nodo receptor. A continuación, mostramos alguno de los valores que puede tomar este campo:
    - '0': *Binding Update accepted*
    - '1': *Accepted but prefix discovery necessary*
    - '133': *Not Home Agent for this Mobile Node*
    - '174': *Invalid Care-of Address*
- *Key Management Mobility Capability (K)*: Si este bit está a 0, el protocolo usado por el HA para establecer las asociaciones de seguridad IPSec no soportará traspasos, por lo que tendría que volver a ser ejecutado. El CN debe poner este bit a 0.
- *Reserved*: 7 bits sin usar, deben ser puestos a 0 por el emisor e ignorados por el receptor.
- *Sequence #*: Entero sin signo de 16 bits copiado del BU al que está respondiendo.
- *Lifetime*: Entero de 16 bits sin signo que informa de las unidades de tiempo durante las que este nodo debe mantener la entrada para este nodo móvil en su *Binding Cache*. Si el valor de *status* indica que la asociación es rechazada, este campo queda sin definir.
- *Mobility Options*: Campo de longitud variable de forma que la longitud total de la cabecera de movilidad sea múltiplo de 8 octetos. Las siguientes opciones son válidas en un BA:
    - *Binding Authorization Data option*
    - *Binding Refresh Advice option*

Binding Error (BE): Mensaje usado por el CN para informar un error relacionado con la movilidad. Aunque no se había comentado este mensaje en la descripción general del protocolo, es conveniente conocerlo por si llegara a aparecer durante las pruebas. El campo *MH Type* tomará el valor 8. Veamos el formato del mensaje:

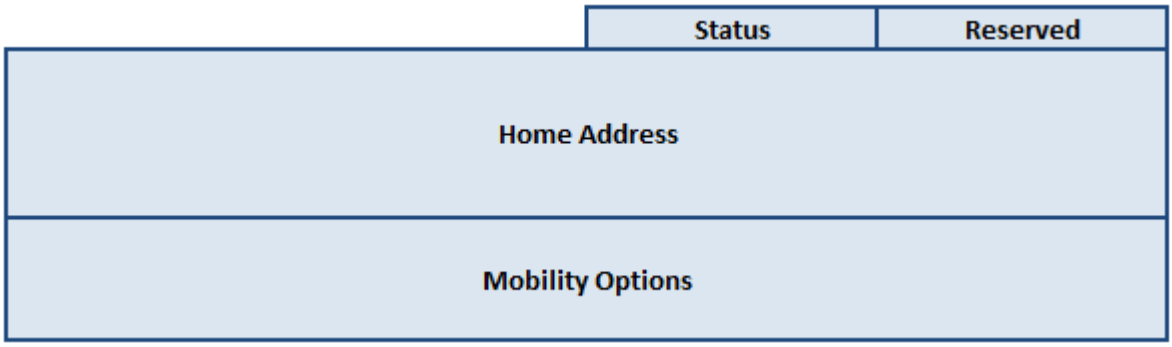


Figura 3.5 Formato Binding Error

Vamos a comentar brevemente los campos que aparecen en este mensaje BE:

- *Status*: Entero sin signo de 8 bits que indica la razón del mensaje de error. Puede tomar los siguientes valores:
    - '1': Asociación (*binding*) desconocida para la opción *Home Address* especificada
    - '2': Valor de *MH Type* desconocido
- *Reserved*: Campo de 8 bits reservado para usos futuros. Debe ser puesto a 0 por el emisor e ignorado por el receptor.
- *Home Address*: La HoA indicada en la *Home Address destination option*. El MN usará este campo para saber cuál es la asociación que no existe (en caso de tener más de una HoA).
- *Mobility Options*: Campo de longitud variable de forma que la longitud total de la cabecera de movilidad sea múltiplo de 8 octetos.

3.3 Protocolos de soporte

Aparte de los conceptos comentados hasta ahora de movilidad, hay diversos protocolos que ven modificado o ampliado su comportamiento básico con nuevas funcionalidades para dar soporte al servicio de movilidad. A continuación, describiremos algunos de ellos [37].

3.3.1 ICMPv6

Tal como se comentó en apartados anteriores, el protocolo ICMPv6 da soporte a diversos protocolos, entre ellos el protocolo de movilidad. Para ello se implementan los siguientes mensajes:

- *Home Agent Address Discovery Request*: El MN envía este mensaje a su red origen (*Home Network, HN*), para iniciar el mecanismo de descubrimiento dinámico de dirección de HA. Este mecanismo se da cuando el MN no conoce la dirección de su HA (por ejemplo, porque éste haya cambiado de dirección). El campo *ICMP type* para este mensaje tomará el valor 144.
- *Home Agent Address Discovery Reply*: Mensaje que un HA envía como respuesta al *Home Agent Address Discovery Request*. El valor del campo *ICMP type* será 145.
- *Mobile Prefix Solicitation*: Mensaje enviado por el MN a su HA para descubrir si se han producido cambios en el prefijo de su HN y ajustar, en caso afirmativo, su HoA. El *ICMP type* será 146.
- *Mobile Prefix Advertisement*: Mensaje enviado por el HA de forma periódica o como respuesta una solicitud (mensaje anterior), donde se indica el prefijo de la HN. El campo *ICMP type* valdrá 147.

3.3.2 Neighbor Discovery

Dentro de este protocolo el mensaje RA sufre una serie de modificaciones para adaptarlo a las necesidades de MIPv6. La primera de ellas es el añadido del bit H, que toma valor 1 para indicar que el equipo que lo envía es un Home Agent.

Por otro lado, se añaden 2 opciones al mensaje RA:

- *Advertisement Interval*: Intervalo de tiempo entre los envíos periódicos del mensaje RA.
- *Home Agent Information*: Se usa para anunciar campos cuando estos cambian su valor por defecto. Alguno de estos campos son *Home Agent Preference* o *Home Agent Lifetime*.

3.4 IPv6 y MIPv6

El protocolo IPv6 como tal también sufre algunos cambios. El primero de ellos en la nueva cabecera de extensión *Destination Option* (tomando el campo *Next Header* el valor 60), que será usada por el MN cuando quiera incluir su HoA en los paquetes enviados, por ejemplo, al utilizar Route Optimization.

Por último, el protocolo MIPv6 incluye la cabecera de encaminamiento tipo 2, que usara el CN para incluir la HoA en un campo adicional, dejando la CoA como dirección de destino del paquete. Esto permitirá que el paquete enviado por el CN pueda llegar al MN, que al recibirlo recuperará su HoA y la dejará como dirección de destino final.

3.5 Seguridad en MIPv6

Como hemos podido observar, a lo largo de todo el proceso de movilidad se intercambian un gran número de mensajes, muchos de los cuales contienen información que podría ser utilizada de forma dañina por una tercera parte que los interceptara. Por ejemplo, podrían capturar un mensaje BU y modificar el campo CoA, haciéndose pasar por nuestro MN, o utilizar la información contenida en alguno de los mensajes, para hacerse pasar por nosotros. Para evitarlo, es necesario que los mensajes MIPv6 que mandamos cumplan una serie de requisitos de seguridad como son la integridad y la confidencialidad.

Como ya vimos en el capítulo 2, la seguridad en IPv6 es gestionada mediante el protocolo IPSec y así se hace también en MIPv6. Concretamente se utiliza ESP ya que así lo establece el IETF. Para ello las claves pueden intercambiarse de forma manual, tal como haremos en nuestro escenario de pruebas, o mediante IKEv2 (RFC 4877), ya que el protocolo IKE en su versión original es demasiado complejo.

En el capítulo 4 veremos cómo deben configurarse las entidades que intervienen en la movilidad para garantizar la seguridad en el intercambio de mensajes.

3.6 Limitaciones de MIPv6

Como hemos podido observar en este capítulo, el IETF define las entidades que intervienen en el proceso de movilidad, así como las interacciones que deben darse entre ellas para que todo funcione correctamente. Sin embargo, hay ciertos aspectos que no quedan cubiertos y que limitan la aplicación de MIPv6 a escalas mayores que un escenario de pruebas experimental.

Quizás la limitación más significativa es que no está preparado para un servicio a gran escala ya que se necesitarían incluir nuevas entidades en el proceso que implementaran una infraestructura AAA (Authentication, Authorization and Accounting), de las que los distintos proveedores de movilidad hacen uso.

Además, los nodos móviles requieren de una serie de parámetros de configuración (Home Address, Home Agent Address, etc.), que junto a la necesidad del intercambio de claves entre el HA y el MN para securizar las comunicaciones, dan lugar lo que el IETF llama *bootstraping problems* o problemas de arranque en el RFC 5637.

Por otro lado, pueden aparecer problemas con firewalls que no reconozcan los mensajes de movilidad o que no autoricen el paso a mensajes con seguridad IPSec, o simplemente el hecho de que MIPv6 no es compatible con redes IPv4, lo que limitaría mucho su ámbito de aplicación.

No obstante, existe un proyecto IST financiado por la Unión Europea llamado proyecto ENABLE cuyos objetivos principales son paliar estos defectos en movilidad IPv6 para permitir su despliegue a gran escala, enriquecer el servicio básico de movilidad con características adicionales y, a largo plazo, establecer una serie de objetivos y diseños que vayan más allá de los actuales de MIPv6 [38].

4 ESCENARIO PRACTICO MIPV6

En este capítulo describiremos todo el proceso de montaje y configuración de nuestro escenario de pruebas para MIPv6 comenzando con el esquema lógico de la red montada, para continuar con una breve descripción de los equipos utilizados.

Para terminar, mostraremos la configuración necesaria en cada uno de los equipos dejándolos preparados para realizar las pruebas que analizaremos en el capítulo 5 y de las que podremos sacar nuestras conclusiones finales en el capítulo 6.

4.1 Esquema de la Red

Para empezar, mostraremos un esquema con los equipos que van a intervenir en nuestro escenario de pruebas donde, como es de esperar, podremos identificar todas las entidades funcionales que vimos en el capítulo anterior.

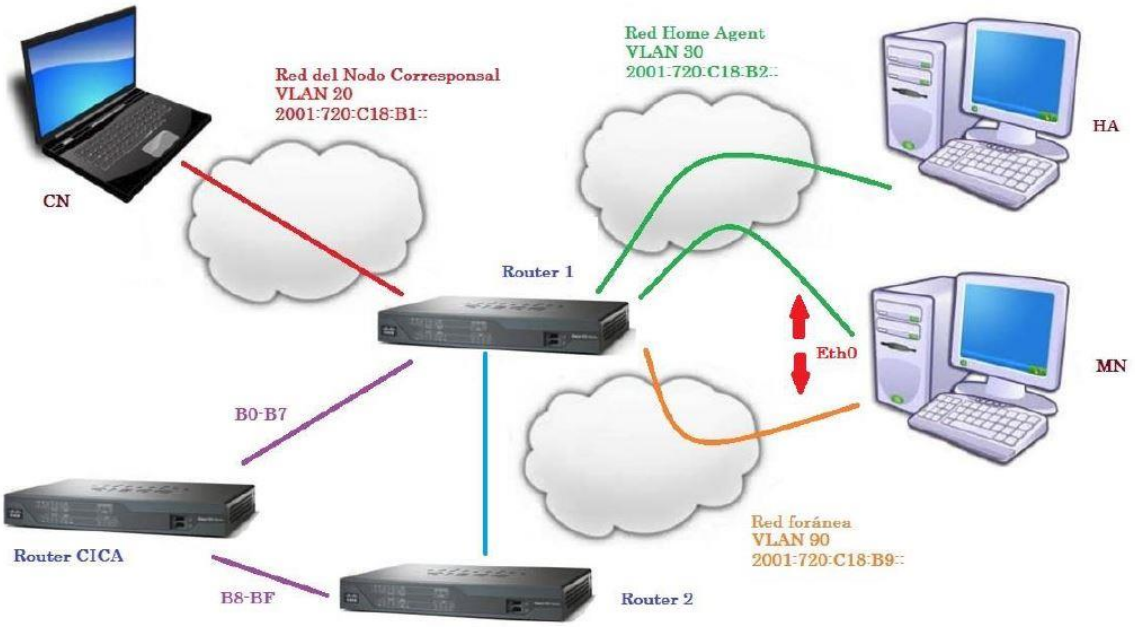


Figura 4.1 Esquema de conexiones del escenario de pruebas

Como podemos observar tenemos 3 redes virtuales diferentes:

- VLAN 20. Creada en nuestro Router 1[1]. Será la red externa desde la que el CN intentará comunicarse con nuestro MN.
- VLAN 30. Creada también en el Router 1. Será la red origen del MN y, por tanto, la red que también contendrá al HA.

[1] Veremos la configuración de los routers en el apartado 4.3

- VLAN 90. Creada en ambos routers, aunque aislada de las otras 2 redes virtuales en el Router 1, de forma que para comunicarse con ellas deba utilizar el camino alternativo a través del router 2 y del router CICA. Será la red destino a la que migrará nuestro MN y desde la cual deberá poder seguir comunicándose con el CN.

El router CICA (Centro Informático Científico de Andalucía) que aparece en el esquema provee de direcciones IPv6 al departamento de telemática con un rango de direcciones desde 2001:720:C18:B0:: a 2001:720:C18:BF:: mediante dos puertos diferentes (B0-B7 y B8-BF), y usa el protocolo de encaminamiento OSPFv3.

En este punto conviene señalar por qué hemos usado un segundo router, si con uno solo nos habría bastado para simular las 3 redes virtuales. Uno de los motivos ha sido complicar algo el escenario incluyendo un salto más en las comunicaciones, intentando de esta forma aproximarnos más a un escenario real, quedándonos, en cualquier caso, lejos de este objetivo. Por otro lado, al incluir en nuestro escenario el router CICA de la universidad de Sevilla encontramos problemas a la hora de conectarlo a nuestros router locales a través de puertos de capa 2 y utilizar encaminamiento OSPFv3, por lo que acabamos utilizando el único puerto de capa 3 que incluyen cada uno de nuestros routers locales para la conexión con el router CICA.

En relación con esto, también llamará la atención del lector el hecho de que el MN no esté directamente conectado al Router 2. Esto es debido al objetivo de este proyecto de utilizar sólo una interfaz de red del MN para conectarlo al router y realizar la migración de una red a otra directamente en el router, en lugar de conectar un segundo cable de red e ir alternando entre las 2 interfaces de red con las que cuenta el MN. Por tanto, cómo solo disponíamos de un cable de red conectado al Router 1, se pensó en interconectar ambos routers usando un puerto trunk y compartiendo una misma VLAN. De esta forma, el MN, tras "moverse" a su red foránea, está conectado al router 2 de forma indirecta y aislado[1] del resto de redes virtuales existentes en el Router 1.

En la siguiente figura podremos observar cómo es la comunicación entre el Nodo Corresponsal y el Nodo Móvil, antes y después del cambio de red.

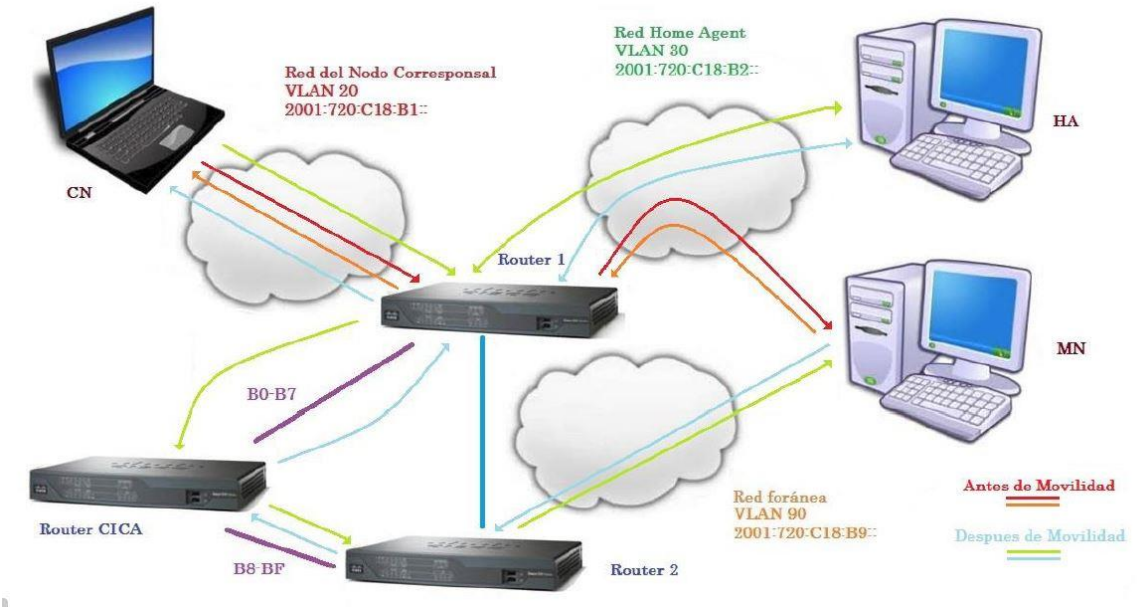


Figura 4.2 Esquema de comunicaciones entre CN y MN

[1] Aislado de forma directa. Podrá comunicarse indirectamente con dichas redes a través del Router 2 y el Router CICA.

Vamos a explicar esta figura:

- Antes del cambio de red del MN
    - Con las flechas rojas mostramos el camino de ida de un mensaje enviado por el CN. Al encontrarse el MN en su red origen, el mensaje le llega directamente a través del Router 1.
    - Con las flechas naranjas mostramos el mensaje de respuesta al anterior. Por la misma lógica, este mensaje irá directo del MN al CN a través del Router 1.
- Después del cambio de red del MN
    - Con las flechas verdes podemos seguir el recorrido de un mensaje enviado por el CN al MN. Al llegar a la red origen del nodo móvil a través del Router 1, este mensaje es capturado por el HA que lo reenvía a la dirección CoA del MN. Para ello, como hemos visto en este mismo apartado deberá atravesar los 3 routers que componen nuestro escenario para llegar al MN.
    - Con las flechas de color azul claro queda marcado el camino de un mensaje de respuesta del MN al CN. Al igual que en el caso anterior, el mensaje atravesará los 3 routers de nuestro escenario y será capturado por el HA, que sustituirá la CoA por la HoA y reenviará el mensaje al CN.

4.2 Equipos utilizados

En este capítulo daremos una descripción básica de los equipos que componen nuestro escenario de pruebas. Para una descripción más detallada puede consultarse el Anexo A.

Para empezar tanto el MN como el HA son equipos pertenecientes al departamento de telemática. El primero de ellos es un Pentium 4 con 512 MB de memoria RAM y que dispone de 2 tarjetas de red, aunque nosotros sólo utilizaremos una de ellas. Por otro lado, el HA es también un Pentium 4 pero con 256 MB de RAM y una sola tarjeta de red. En ambos equipos se utiliza el sistema operativo Debian 7.4 debido a que el software de movilidad que vamos a utilizar (UMIP) sólo cuenta con paquetes de este SO.

El papel de CN lo desempeñara mi portátil personal, un HP Pavilion i5 con 4 GB de RAM donde se instaló una máquina virtual con el sistema operativo Ubuntu 10.04 LTS que nos permitía trabajar con algún software necesario para realizar las pruebas.

Para interconectar estos equipos y crear las redes virtuales se han utilizado dos routers Cisco 892. Por último, no contamos con detalles técnicos sobre el router CICA.

4.3 Configuración de los Routers

En este capítulo describiremos la configuración de ambos routers locales, de forma que contengan las redes virtuales descritas en el esquema de red.

Para configurar nuestros dos routers, conviene partir en ambos desde cero. Para ello, vamos a resetearlos. Tenemos la opción de hacerlo manualmente o por software. Para un reset manual, el procedimiento dependerá del modelo de router. En nuestro caso, según el manual del fabricante [39], se debe pulsar el botón de reset que puede encontrarse en la carcasa del equipo durante 5 segundos.

Para el reset mediante software debemos conectar el puerto de consola del router con un cable serial a un PC donde tendremos instalados algún software que nos permita enviar órdenes al router. En nuestro caso, hemos optado por el software Minicom para Linux. El proceso de instalación y configuración de este software puede encontrarse en el Anexo B [40].

Una vez conectado el router a nuestro PC y con el software Minicom en ejecución, debemos encender nuestro router [41]. Durante el proceso de arranque del router debemos interrumpirlo pulsando "*CTRL - A + Z*". Tras esto pulsamos la tecla F. Así conseguimos entrar en modo *Rommon*, donde debemos introducir los siguientes comandos:

```
rommon 1 > confreg 0x2142
rommon 2 > reset
```

Con la instrucción *confreg 0x2142* estamos estableciendo la posición de memoria de arranque a una posición vacía. Tras esto reseteamos el router. Cuando termine de arrancar introduciremos:

```
Router>enable
Router#copy running-config startup-config
```

Como hemos arrancado desde una posición vacía de memoria, la configuración inicial del router será la configuración por defecto. Ahora copiamos esta configuración por defecto al registro de configuración de arranque (posición 0x2102 de la memoria) y volvemos a establecer la posición de arranque correcta y reiniciamos:

```
Router#configure terminal
Router (config) #config-register 0x2102
Router (config) #end
Router#reload
```

Tras este reset ya podemos configurar nuestro router y empezar a trabajar desde cero. Por ejemplo, podemos establecer varios usuarios de acceso y sus contraseñas, mostrar un mensaje de bienvenida o asignar roles a los distintos usuarios [42].

En nuestro caso, tras realizar este proceso de reset en ambos routers, se ha optado por dejar la configuración básica por defecto, sin añadir usuarios ni contraseña de acceso dado que nadie más iba a utilizar estos equipos, agilizando así el acceso al router.

Tras el proceso de configuración básica, debemos continuar con la configuración de las distintas VLAN que necesitamos para montar nuestro escenario de pruebas de movilidad IPv6 [43].

4.3.1 Router 1

El router 1 será el que contenga las VLANS de la red origen del nodo móvil y de la red externa del nodo corresponsal. Además, estará conectado al router de la red CICA de la universidad a través de la pata que ofrece direcciones con rango B0/B7. Por último, también tendrá la VLAN de la nueva red a la que migrará el nodo móvil (VLAN 90), pero no tendrá dirección IPv6 en dicha VLAN por lo que para comunicarse con equipos en dicha red deberá hacerlo a través del router CICA.

Recordar también que el router CICA utiliza el protocolo de encaminamiento OSPFv3 por lo que deberemos incluir nuestros routers en la misma área del router CICA (0.0.2.98) de forma que puedan establecer comunicación entre ellos a través de este.

Veamos ahora una tabla con los detalles más relevantes de la configuración del Router 1 así como una representación de las conexiones de sus puertos:

Configuración Router 1						
ID Vlan	Nombre	Puertos asignados	Dirección	Link-local	OSPF	Otros
VLAN 20	B1	fa3	2001:720:c18:b1::20	FE80::20	id: 1 área: 0.0.2.98	
VLAN 30	B2	fa6, fa7	2001:720:c18:b2::30	FE80::30	id: 1 área: 0.0.2.98	RA Lifetime: 5 segundos RA Interval: 0,5 segundos fa7 spanning tree portfast
VLAN 90	B9	(fa7)				fa7 spanning-tree portfast
Otros	Conexión Router Cica	fa8	autoconfig			
Otros	Puerto Trunk de conexión al Router 2	fa1				

Tabla 4-1 VLANs y otras configuraciones del Router 1

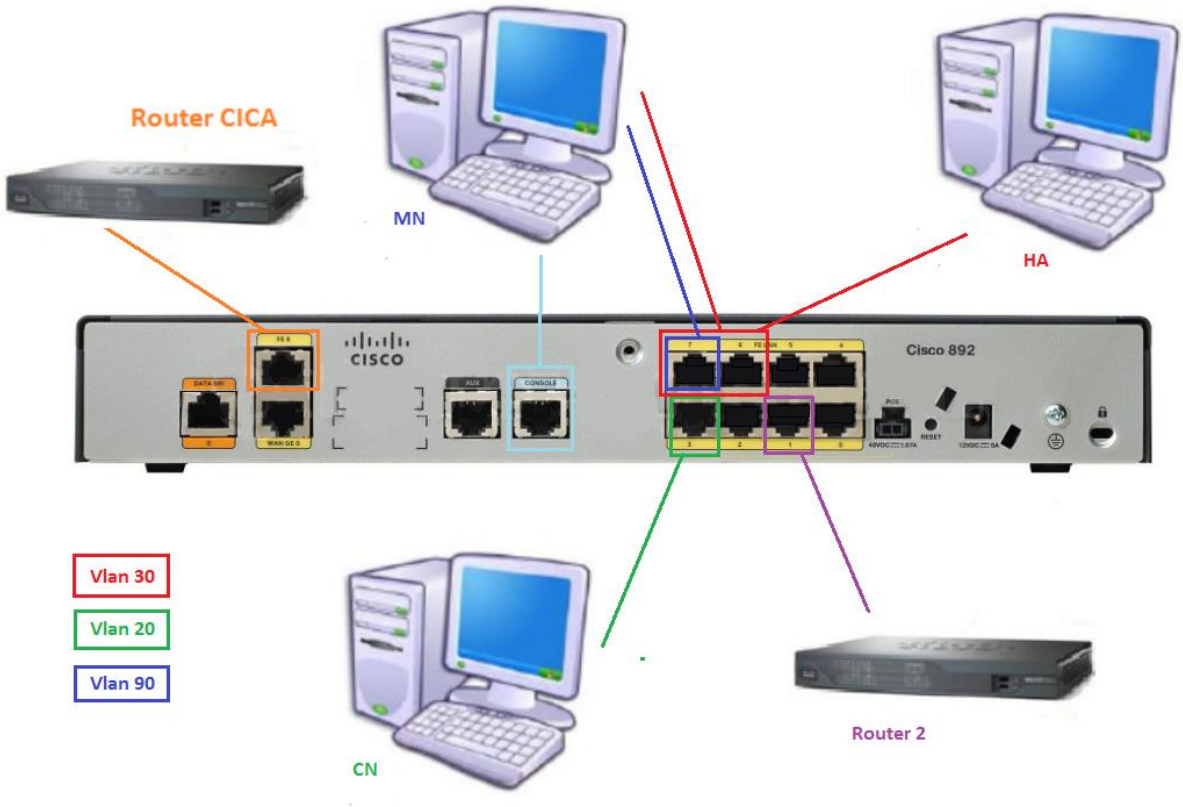


Figura 4.3 Esquema de conexiones del Router 1

Podemos observar como la Vlan 30 será la que contenga al HA y, inicialmente, al MN. La Vlan 20 será la red del CN y la Vlan 90 la red a la que migrará el MN.

Además, usaremos el puerto fe8 para conectarnos al Router CICA ya que es el único al que se le puede asignar una dirección IP. Por otro lado, el puerto fe1 se usará como puerto trunk para conectarlo al Router 2.

Vemos además que el puerto fe7 se configura como spanning-tree portfast [44] [45]. Con esto conseguimos que el puerto se salta las fases de Listening and Learning del protocolo STP y pase directamente a la fase de Forwarding, reduciendo notablemente el tiempo que el puerto tarda en estar completamente operativo tras un cambio de red. Es importante señalar que esta configuración se aplicó tras las primeras pruebas, por lo que el apartado inicial de pruebas se realizó sin esta opción activada, y los resultados de aplicar el spanning-tree portfast se estudiará en el apartado de pruebas optimizadas.

Toda esta configuración se realizará introduciendo las órdenes contenidas en el [Anexo E], en un equipo conectado al router por el puerto de consola y usando el software Minicom.

4.3.2 Router 2

El router 2 será el que contenga la red de destino del MN tras la migración (VLAN 90). Estará conectado al router 1 a través de un puerto trunk (fa1) y al router CICA a través del puerto 8 que nos permite configurarlo con OSPFv3.

Veamos ahora una tabla como resumen de su configuración y una representación de las conexiones en sus puertos, al igual que hicimos con el Router 1.

VLANs Router2						
ID	Nombre	Puertos Asignados	Dirección	Link-local	OSPF	Otros
VLAN 90	B9		2001:720:c18:B9::90	FE80::90	id: 1 área: 0.0.2.98	RA Interval: 1 segundo
Otros	Conexión Router Cica	fa8	autoconfig			
Otros	Puerto Trunk de conexión al Router 1	fa1				

Tabla 4-2 VLANS y otras configuraciones del Router 2

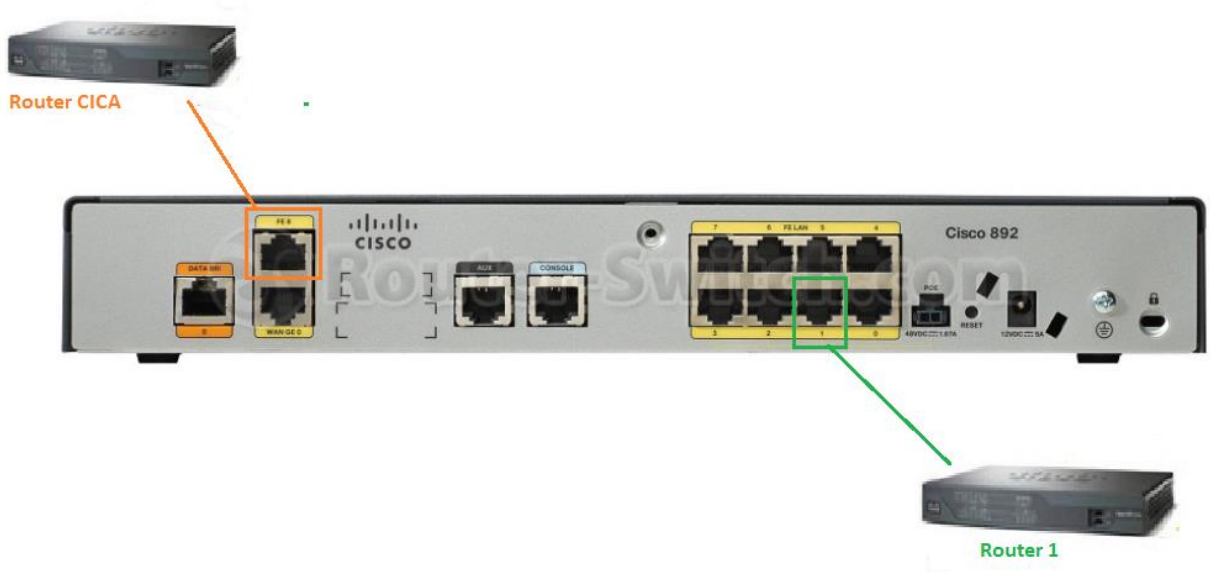


Figura 4.4 Esquema de conexiones del Router 2

Esta configuración es mucho más simple. Se crea la VLAN 90 pero no se le asigna ningún puerto ya que el MN irá conectado al Router 1. Además, al igual que en el Router 1, se utiliza el puerto fe8 para la conexión con el Router CICA y el puerto fe1 como puerto trunk para conectar los Routers 1 y 2.

Al igual que en el Router 1, las instrucciones necesarias para esta configuración pueden encontrarse en el [Anexo E].

Tras configurar el router 2 ya tenemos nuestros routers listos para interconectar todos los equipos que formarán nuestro escenario de pruebas. A continuación, veremos la configuración necesaria en cada uno de los PCs que utilizaremos, empezando por el nodo móvil.

4.4 Instalación de UMIP

El primer paso antes de configurar los distintos equipos para simular nuestro escenario de movilidad es instalar el software de movilidad UMIP siguiendo los pasos indicados en su página web. Estos pasos serán comunes para el MN y el HA [46].

4.4.1 Recompilación del Kernel

Antes de instalar el software de movilidad es necesario contar con un kernel apto para movilidad. En concreto para kernels versión 3.*, sólo versiones superiores a la 3.8.1 incorporan soporte a movilidad. Por tanto, primero debemos comprobar que versión de kernel tenemos en nuestro equipo. Para ello introducimos en un terminal (como usuario root) el comando *uname –a*.

Si la versión de nuestro kernel es inferior a la 3.8.1 deberemos instalar una versión superior. En nuestro caso optamos por la versión estable 3.8.2. Para descargar esta versión ejecutaremos en un terminal como usuario root:

cd /usr/src/

wget http://www.kernel.org/pub/linux/kernel/v3.x/linux-3.8.2.tar.bz2

A continuación, descomprimimos e instalamos las librerías necesarias:

bunzip2 linux-3.8.2.tar.bz2

tar xf linux-3.8.2.tar

apt-get install fakeroot kernel package

apt-get install libncurses5-dev

Para evitar complicaciones partiremos de la configuración de nuestro kernel actual. Para ello copiaremos dicha configuración:

cp /boot/config/-3.2.0-4-686-pae /usr/src/linux-3.8.2/.config

Para establecer la nueva configuración ejecutamos los siguientes comandos:

make oldconfig (pulsamos “enter” hasta que acabe).

make menuconfig (necesaria la librería *libncurses5-dev* para ejecutar este comando).

Tras ejecutar este comando se mostrará un menú donde debemos activar los módulos de movilidad en el 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
[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 activados los módulos necesarios, continuamos con la recompilación del kernel.

make –kpkg clean

export CONCURRENCY_LEVEL=5

fakeroot make-kpkg –append-to-version "-mipv6" –revision "1" –initrd kernel_image kernel_headers; shutdown –h now

Con estos comandos limpiamos el árbol del código fuente en caso de que se haya intentado recompilar el kernel previamente, agilizamos el proceso de recompilado y finalmente compilamos el nuevo kernel. Este proceso puede llevar varias horas, por lo que se ejecuta el comando shutdown para apagar el equipo al terminar.

Tras esto, deben existir dos paquetes *.deb* en nuestro directorio de trabajo. Si ese el caso, finalmente ejecutamos:

*dpkg –i *.deb*

A continuación, reiniciamos nuestro equipo y el nuevo kernel debería aparecer en el menú de arranque, así que procederemos a ejecutarlo.

4.4.2 UMIP

Para compilar y utilizar UMIP hay ciertas herramientas que necesitaremos y que descargaremos primero introduciendo los siguientes comandos en un terminal como usuario root:

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

Una vez descargadas e instaladas estas aplicaciones, es el turno de descargar el código fuente de UMIP.

cd /usr/src/

git clone http://git.umip.org/umip/umip.git

cd umip/

Para terminar, compilamos el código fuente:

autoreconf –i

CPPFLAGS='-isystem /usr/src/linux-3.8.2/' ./configure –enable-vt

make

make install

Con CPPFLAGS le indicamos la ubicación de las cabeceras del núcleo y con la opción enable-vt habilitamos un terminal virtual que nos permitirá ejecutar ciertos comandos de UMIP que serán de gran utilidad en nuestras pruebas.

4.5 Configuración de los Equipos

El siguiente paso, una vez instalado UMIP, es crear los archivos de configuración, tanto en el nodo móvil como en el home agent que determinen el comportamiento de cada uno de ellos dentro de nuestro escenario de movilidad. El contenido de estos archivos puede consultarse en la web oficial de UMIP [47].

4.5.1 Nodo Móvil

Para empezar, crearemos el archivo *mip6d.conf* en la ruta */usr/local/etc/* que contendrá el código necesario para configurar su rol como nodo móvil:

Configuración /usr/local/etc/mip6d.conf (Nodo Móvil)

```
# 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 2001:720:c18:b2:24f:4eff:fe01:ad52;
    HomeAddress 2001:720:c18:b2:24f:4eff:fe0f:77ff/64;
}

# Enable IPsec static keying
UseMnHaIPsec enabled;
KeyMngMobCapability disabled;

# IPsec Security Policies information
IPsecPolicySet {
    HomeAgentAddress 2001:720:c18:b2:24f:4eff:fe01:ad52;
    HomeAddress 2001:720:c18:b2:24f:4eff:fe0f:77ff/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;
}
```

Si revisamos este código podemos observar detalles como la activación de la opción *Optimistic Handover* que habilita la transmisión de paquetes una vez enviado el BU, sin necesidad de esperar la recepción del BA, reduciéndose el tiempo de traspaso.

Por otro lado, se desactiva la opción *DoRouteOptimizationMN* ya que impediría el uso de IPsec. Además, se establece una jerarquía entre las interfaces disponibles, dándole preferencia a la "eth1" que es la que usaremos para conectarnos a la red.

También se establece la dirección HoA y la del Home Agent. Este es una de las limitaciones de nuestro escenario de pruebas, ya que debemos conocer a priori estas direcciones.

A continuación, tenemos las opciones *UseMnHaIPsec* y *KeyMngMobCapability*. Con la primera activamos o desactivamos el protocolo IPsec (probaremos nuestro escenario en ambos casos), y la segunda nos permite cambiar el bit K de los mensajes BU y BA, para indicar si se usará o no un protocolo de intercambio de claves. En nuestro caso, las claves se establecerán manualmente, por lo que este parámetro estará siempre desactivado.

Por último, el bloque de configuración *IPsecPolicySet* es en el encargado de regular las condiciones de la protección por IPsec, tanto para el tráfico de señalización como para el de datos. Concretamente, estamos estableciendo protección ESP para el tráfico de señalización de movilidad, el tráfico de datos y el tráfico ICMP.

El siguiente paso es crear un archivo de configuración de las SA (Security Associations) con las claves necesarias para que MN y HA se comuniquen. Situaremos este archivo *setkey.conf* en la misma ruta */usr/local/etc/*.

Configuración de /usr/local/etc/setkey.conf (Mobile Node y Home Agent)

```
# IPsec Security Associations
# HA address: 2001:720:c18:b2:24f:4eff:fe01:ad52;
# MR HoAs:    2001:720:c18:b2:24f:4eff:fe0f:77ff/64;

# Flush the SAD and SPD
flush;
spfflush;

# MN1 -> HA transport SA for BU
add 2001:720:c18:b2:24f:4eff:fe0f:77ff 2001:720:c18:b2:24f:4eff:fe01:ad52
esp 0x11
     -u ll
     -m transport
     -E 3des-cbc "MIP6-011--12345678901234"
     -A hmac-sha1 "MIP6-011--1234567890" ;
# HA -> MN1 transport SA for BA
add 2001:720:c18:b2:24f:4eff:fe01:ad52 2001:720:c18:b2:24f:4eff:fe0f:77ff
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 2001:720:c18:b2:24f:4eff:fe0f:77ff 2001:720:c18:b2:24f:4eff:fe01:ad52
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 2001:720:c18:b2:24f:4eff:fe01:ad52 2001:720:c18:b2:24f:4eff:fe0f:77ff
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 2001:720:c18:b2:24f:4eff:fe0f:77ff 2001:720:c18:b2:24f:4eff:fe01:ad52
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 2001:720:c18:b2:24f:4eff:fe01:ad52 2001:720:c18:b2:24f:4eff:fe0f:77ff
esp 0x16
      -u 16
      -m transport
      -E 3des-cbc "MIP6-016--12345678901234"
      -A hmac-sha1 "MIP6-016—1234567890" ;
```

Con estos dos archivos ya tenemos lista la configuración del nodo móvil.

4.5.2 Home Agent

Para empezar, al igual que con el MN, tenemos el archivo de configuración *mip6d.conf* que ubicaremos en la ruta */usr/local/etc/* con el siguiente contenido:

Configuración de /usr/local/etc/mip6d.conf (Home Agent)

```
# Sample UMIP configuration file for a MIPv6 Home Agent
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 2001:720:c18:bb:24f:4eff:fe01:ad52 allow;
DefaultBindingAclPolicy deny;

# Enable IPsec static keying
UseMnHaIPsec disabled;
KeyMngMobCapability disabled;

# IPsec Security Policies information
IPsecPolicySet {
      HomeAgentAddress 2001:720:c18:b2:24f:4eff:fe01:ad52;
      HomeAddress 2001:720:c18:b2:24f:4eff:fe0f:77ff/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;
```

En este código podemos ver que se establece la interfaz "eth0" como interfaz de conexión a la HN. Por otro lado, se establece una política ACL de autenticación en el servidor para que sólo se permitan binding a la dirección IP de nuestro MN, rechazando cualquier otro binding. Esta política podría suponer un problema de seguridad si alguien conociera la IP de nuestro nodo móvil, pero al tratarse de un escenario de pruebas, obviaremos esta posibilidad.

Por último, se establecen las mismas directivas de protección IPsec que en el nodo móvil. Es de entender, por tanto, que las SAs serán iguales, por lo que el código del fichero *setkey.conf* será el mismo que el del nodo móvil, por lo que solo tendremos que copiar este fichero a la ruta */usr/src/etc/* de nuestro HA.

Además, este equipo necesitará enviar RA periódicos para que el MN detecte quien es su Home Agent. Para ello se instaló en el apartado 4.3.2.2 el software *radvd*. Un daemon que envía RA periódicos con el prefijo de red, indicando el MTU máximo y la dirección de la puerta de enlace, que será la de nuestro router 1. Para configurar esta información se utiliza el archivo *radvd.conf* situado en */etc/* con el siguiente contenido [48]:

Configuración de /etc/radvd.conf (Home Agent)

```
# 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 2001:720:c18:b2::/64
      {
            AdvRouterAddr on;
            AdvOnLink on;
            AdvAutonomous on;
      };
};
```

En este código se está indicando, entre otros parámetros, la interfaz por la que deben enviarse los mensajes RA, el tiempo de vida del *Home Agent* o el tiempo periódico entre los RA enviados.

Para activar el software *radvd* deberemos ejecutar por línea de comandos *radvd –C /etc/radvd.conf*.

De esta forma ya tendremos configurado nuestro *Home Agent*.

4.5.3 Nodo Corresponsal

Para terminar este apartado solo queda señalar que el equipo que ejercerá de CN no requiere mucha configuración salvo la instalación de algunas herramientas que necesitaremos para realizar nuestras pruebas, como el capturador de tráfico Wireshark o el cliente FTP Filezilla. La instalación de este último software puede consultarse en el Anexo C.

4.6 Arrancar software de movilidad

Llegados a este punto sólo nos queda arrancar en los equipos el software UMIP y todo aquel programa que necesitemos para realizar nuestras pruebas. Podemos ejecutar UMIP automáticamente desde un script de arranque tal como se muestra en el anexo D o de forma manual ejecutando en un terminal como usuario root el comando *mip6d –c /usr/local/etc/mip6d.conf*.

En nuestro caso se ha optado por esta segunda opción para poder ejecutar el servicio cuando nos convenga dejándonos más libertad para preparar los demás programas necesarios para realizar nuestras pruebas, como por ejemplo Wireshark.

5 PRUEBAS REALIZADAS

Finalmente llegamos al capítulo que justifica toda la base teórica y todo el trabajo de construcción del escenario de pruebas. Tras explicar el protocolo IPv6 junto a algunos protocolos auxiliares necesarios, desgranar las características principales de MIPv6 y describir la estructura de nuestra red y la configuración de los equipos implicados, ha llegado el momento de realizar las pruebas que nos darán una visión aproximada del rendimiento del protocolo de movilidad en IPv6.

Para ello dividiremos las pruebas en 2 partes. Durante la primera parte realizaremos pruebas de funcionamiento que nos ayudaran a entender el protocolo de movilidad pero que no nos aportará dato alguno en cuanto a su rendimiento.

En la segunda parte sí entraremos a fondo a evaluar el rendimiento del protocolo MIPv6, siempre teniendo en cuenta las limitaciones de nuestro escenario, pero con la idea de obtener una visión aproximada y suficientemente fiable de dicho rendimiento.

5.1 Pruebas de funcionamiento

El objetivo de estas pruebas es comprobar que nuestro escenario de pruebas se comporta como esperamos según lo descrito en la parte teórica de MIPv6.

Las pruebas de funcionamiento que hemos realizado son las siguientes:

- Ping sin traspaso. Existe comunicación entre el CN y el MN sin que el MN salga de su HN.
- Ping con traspaso. Durante la comunicación entre el CN y el MN, éste cambia de red, pasando a la FN.
- Ping tras traspaso. Comunicación entre el CN y el MN una vez el CN se ha trasladado de la HN a la FN.
- Ping con regreso. El MN se encuentra en la FN comunicándose con el CN, y cambia de red volviendo a su HN.
- Ping con traspaso con protocolo ESP activado

Los resultados de estas pruebas se han recogido con capturas de tráfico de Wireshark, así como en la consola de UMIP y pueden encontrarse en el CD incluido en esta memoria. Sin embargo, en esta memoria nos limitaremos a explicar aquellas pruebas que muestran información de interés comenzando con la prueba de Ping con traspaso.

5.1.1 Ping con traspaso

En esta prueba veremos el comportamiento de nuestro escenario de pruebas en el caso básico de movilidad. Es decir, el nodo móvil está comunicándose con un CN estando en su HN y entonces se produce un cambio de red, pasando a estar en una FN.

Veremos que en el momento del cambio dejarán de recibirse Pings hasta que el HA recibe el binding update del MN y puede empezar a reenviarle los mensajes que reciba para él.

Hay que señalar además que en esta prueba no usaremos protocolo ESP, ya que sólo veríamos mensajes encriptados que no nos aportarían apenas información.

Para realizar la prueba mandaremos mensajes Ping6 desde del CN al MN durante todo el proceso de cambio de la HN a la FN. Para ello ejecutaremos en el CN la siguiente orden por línea de comandos:

ping6 2001:720:c18:b2:24f:4eff:fe0f:77ff

Así se enviarán mensajes echo request cada segundo de forma indefinida. De esta forma podremos observar con cierta precisión el tiempo de desconexión del MN en los cambios de red.

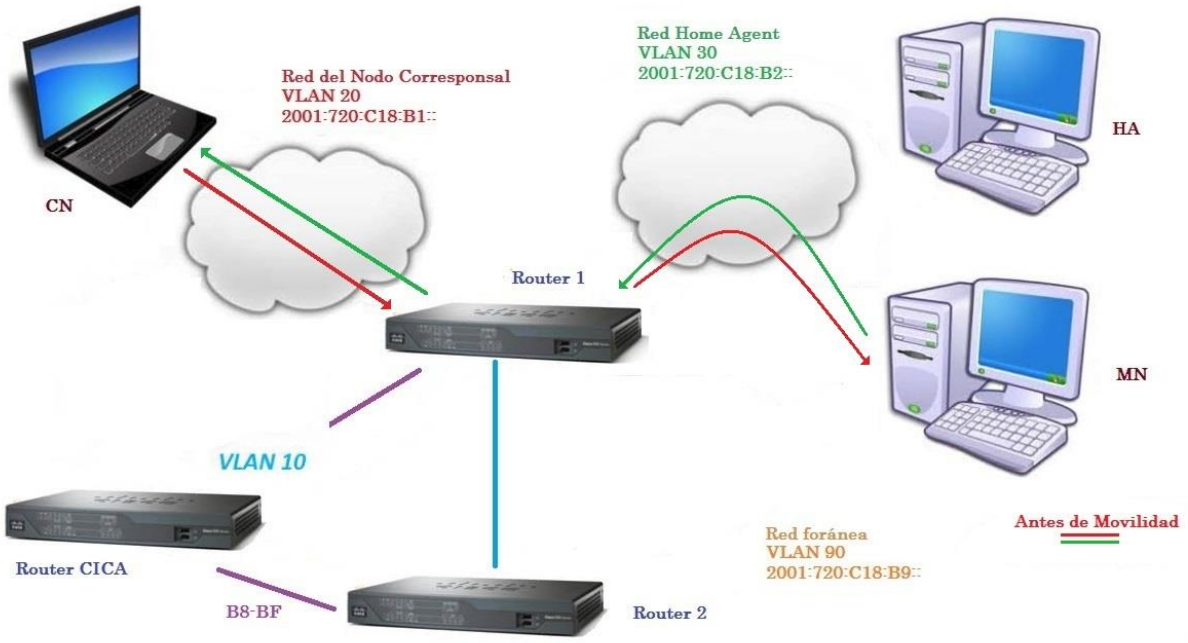


Figura 5.1 Intercambio de mensajes entre CN y MN antes del traspaso

Como puede observarse la trayectoria de los mensajes antes de que el MN salga de su red inicial es bastante simple.

Los mensajes echo request del CN llegan al Router 1 que se lo entrega al MN. Por el contrario, los mensajes echo reply del MN se envían al Router 1 que es su puerta de enlace y este lo manda al CN.

En este momento, vamos a simular el cambio de red del MN. Para ello, usando Minicom ejecutaremos en el Router 1 un conjunto de instrucciones que cambiarán la VLAN del puerto al que está conectado el MN, pasando de este modo a estar conectado a la FN. Este conjunto de instrucciones puede consultarse en el anexo E.

Tras el cambio de red la comunicación seguirá los caminos descritos en el siguiente esquema de conexiones:

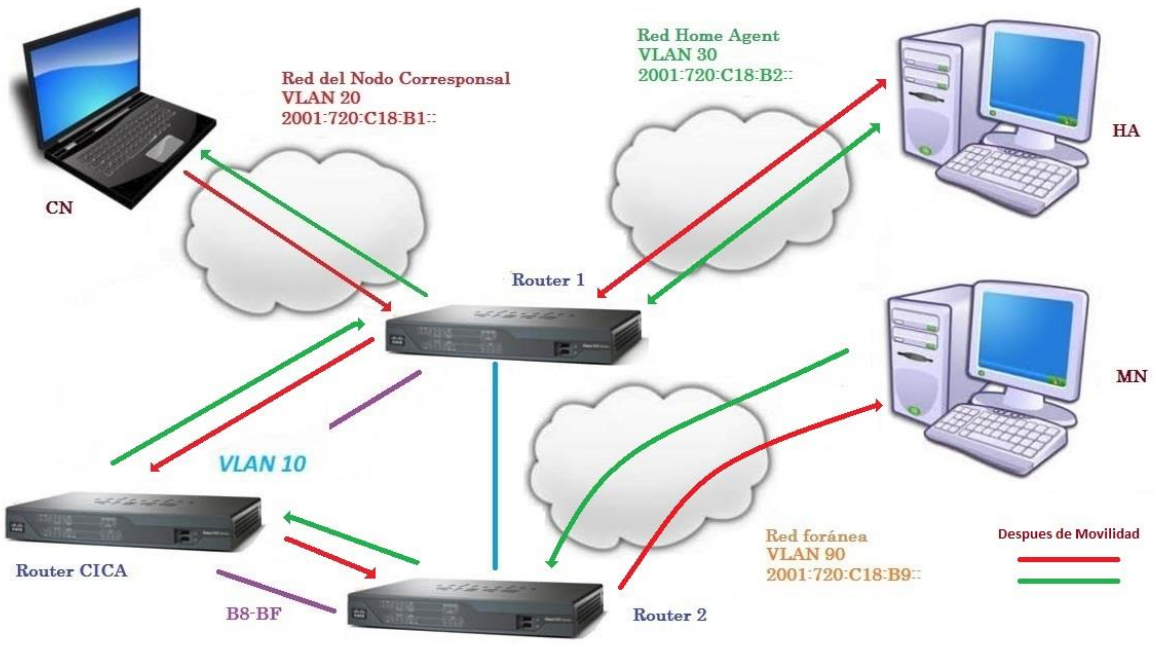


Figura 5.2 Intercambio de mensajes entre CN y MN después del traspaso

Tras el cambio de red, la complejidad del camino que siguen los mensajes entre el CN y el MN aumentan considerablemente involucrando a todos los equipos presentes en el escenario de pruebas.

El echo request enviado por el CN llega al router 1 que lo reenvía por su VLAN 30 llegándole al HA. El HA detecta que el destinatario es el MN, por lo que lo reenvía con la nueva dirección de destino del MN. El router 1 aunque está conectado directamente al router 2 tiene deshabilitado el enrutamiento en ese puerto, por lo que tiene que optar por el camino alternativo a través del Router CICA. Finalmente llega al router 2 que lo envía al MN a través del router 1 donde está conectado a la VLAN 90.

Veamos ahora algunas capturas de Wireshark para analizar en profundidad cómo ha sido todo el proceso de movilidad.

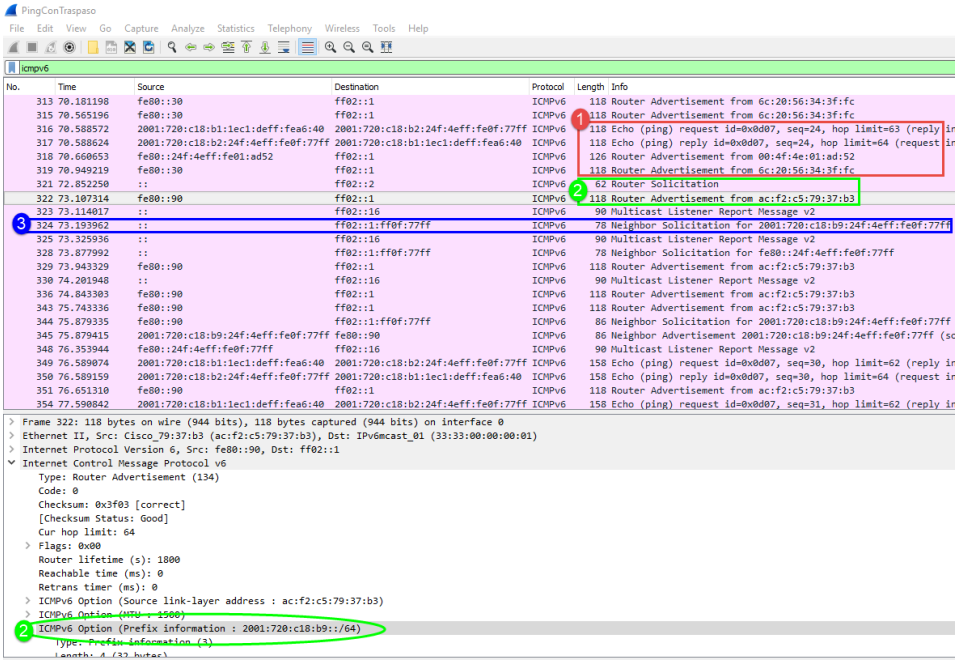


Figura 5.3 Captura mensajes Ping en Wireshark

Es esta primera imagen podemos observar el último mensaje Echo request con su correspondiente respuesta (1) justo antes de que el MN pase a la red foránea. También puede apreciarse que antes del cambio de red al MN le llegan mensajes RA del router 1 y del HA.

Una vez el MN pasa a la red foránea comienza a enviar mensajes *Router Solicitation* (2), al que responde el Router 2 con mensajes *Router Advertisement* con el prefijo de red de forma que el MN pueda autoconfigurar su nueva dirección ip con SLAAC. A su vez, el MN envía un mensaje NS para comprobar si su dirección ya existe en la red (3).

Una vez el MN tiene su nueva dirección se produce la asociación entre la HoA y la CoA. Esto puede comprobarse en el terminal en el que estemos ejecutando UMIP en el MN donde nos ira apareciendo las entradas de la BUL cada vez que se actualiza:

```
== BUL_ENTRY ==
Home address    2001:720:c18:b2:24f:4eff:fe0f:77ff
Care-of address 2001:720:c18:b9:24f:4eff:fe0f:77ff
CN address      2001:720:c18:b2:24f:4eff:fe01:ad52
 lifetime = 60,  delay = 1000
```

En esta entrada de la BUL podemos observar detalles como que la HoA y la CoA sólo se diferencian en el prefijo, lo cual tiene sentido ya que esa dirección se obtiene mediante SLAAC y estamos utilizando la misma interfaz de red, por lo que la dirección MAC sigue siendo la misma. Por otra parte, la dirección del CN es la del HA, que es el equipo al que se envían las respuestas ya que no está activado el *Route Optimization*.

Una vez establecida esta asociación de direcciones en la BUL, el MN envía el mensaje BU al HA. Veamos dicho mensaje en Wireshark.

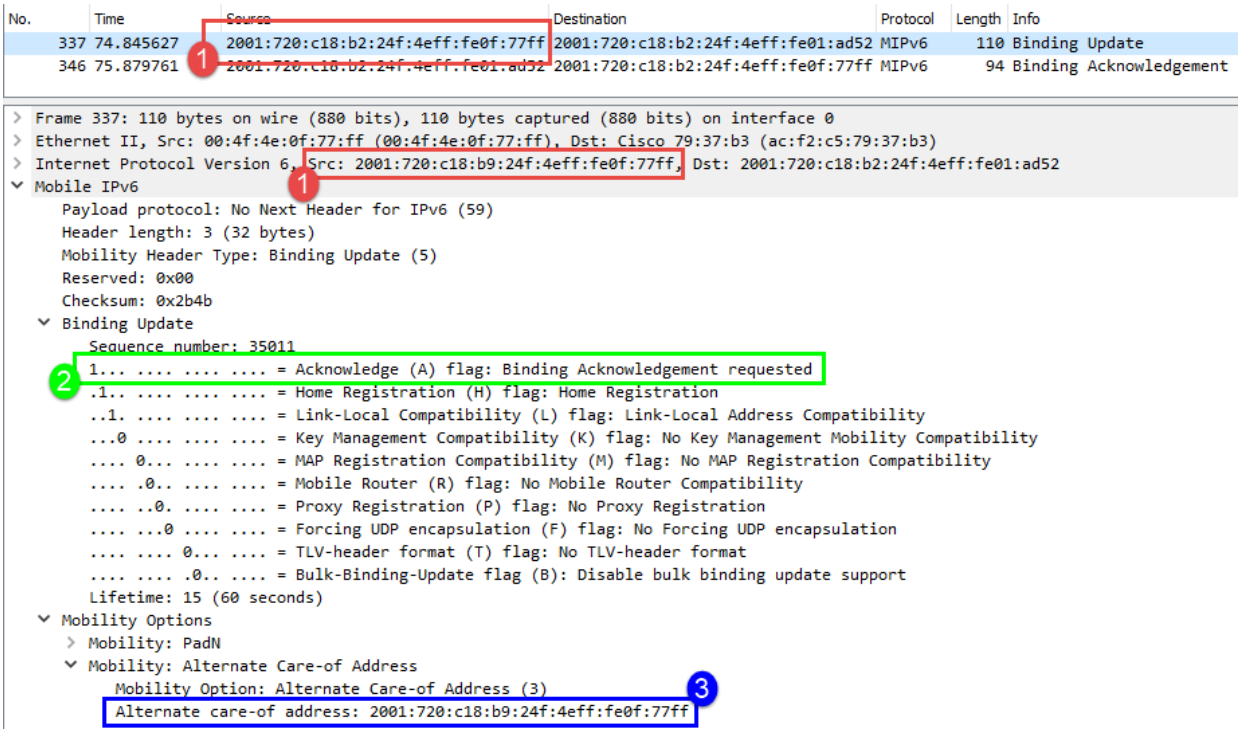


Figura 5.4 Mensaje Binding Update en Wireshark

Lo primero que llama la atención es que la dirección mostrada como origen del paquete difiere de la que aparece en el campo *source address* de la cabecera IPv6 (1). Tal como vimos en el capítulo 3, el mensaje BU contiene el campo *destination option* que contiene la dirección *HoA*. En este caso, Wireshark ha tomado este valor como dirección origen del paquete, lo cual puede llevar a confusión si no se conoce el protocolo y no se estudia el contenido del paquete detenidamente.

Una vez visto esto podemos fijarnos también en otros detalles como el bit *Aknowledge Flag* puesto a 1, que indica que se pide un mensaje *Binding Acknowledge* como respuesta al BU (2). Para terminar, destacar la presencia del campo *Alternate CoA* que indica que debe usarse esta dirección como *CoA* y no usar la de dirección de origen del paquete (3).

Tras la llegada al HA del BU, este registra la asociación en la *Binding Cache*. Si todo es correcto responde con un mensaje BA. A continuación, podemos ver parte del contenido de la consola de ejecución de UMIP en el HA:

```
Mon Nov 21 10:17:39 mh_bu_parse: Binding Update Received
Mon Nov 21 10:17:40 ndisc_do_dad: Dad success
Mon Nov 21 10:17:40 __tunnel_add: created tunnel ip6tnl1 (4)
from 2001:720:c18:b2:24f:4eff:fe01:ad52
to 2001:720:c18:b9:24f:4eff:fe0f:77ff user count 1
Mon Nov 21 10:17:40 mh_send_ba: status Binding Update accepted (0)
Mon Nov 21 10:17:40 mh_send: sending MH type 6
from 2001:720:c18:b2:24f:4eff:fe01:ad52
to 2001:720:c18:b2:24f:4eff:fe0f:77ff
Mon Nov 21 10:17:40 mh_send: remote CoA
2001:720:c18:b9:24f:4eff:fe0f:77ff
Mon Nov 21 10:18:36 mh_bu_parse: Binding Update Received
Mon Nov 21 10:18:36 tunnel_mod: modifying tunnel 4 end points with
from 2001:720:c18:b2:24f:4eff:fe01:ad52
to 2001:720:c18:b9:24f:4eff:fe0f:77ff
Mon Nov 21 10:18:36 mh_send_ba: status Binding Update accepted (0)
Mon Nov 21 10:18:36 mh_send: sending MH type 6
from 2001:720:c18:b2:24f:4eff:fe01:ad52
to 2001:720:c18:b2:24f:4eff:fe0f:77ff
```

En la *Binding Cache* se van recogiendo los distintos eventos según se van sucediendo. Por ejemplo, tras recibir el BU se comprueba si la dirección del MN esta duplicada (*Dad success*). Tras esta comprobación se crea el túnel *IPv6 in IPv6* entre HA y MN, por donde se enviarán tanto paquetes de movilidad como paquetes de datos, y se responde al MN con un mensaje BA (*MH type 6*). A partir de ese momento, al recibir nuevos BU del MN, no se crean nuevos túneles, sino que se modifica el original.

Veamos ahora en Wireshark el contenido del mensaje BA de respuesta enviado por el HA.

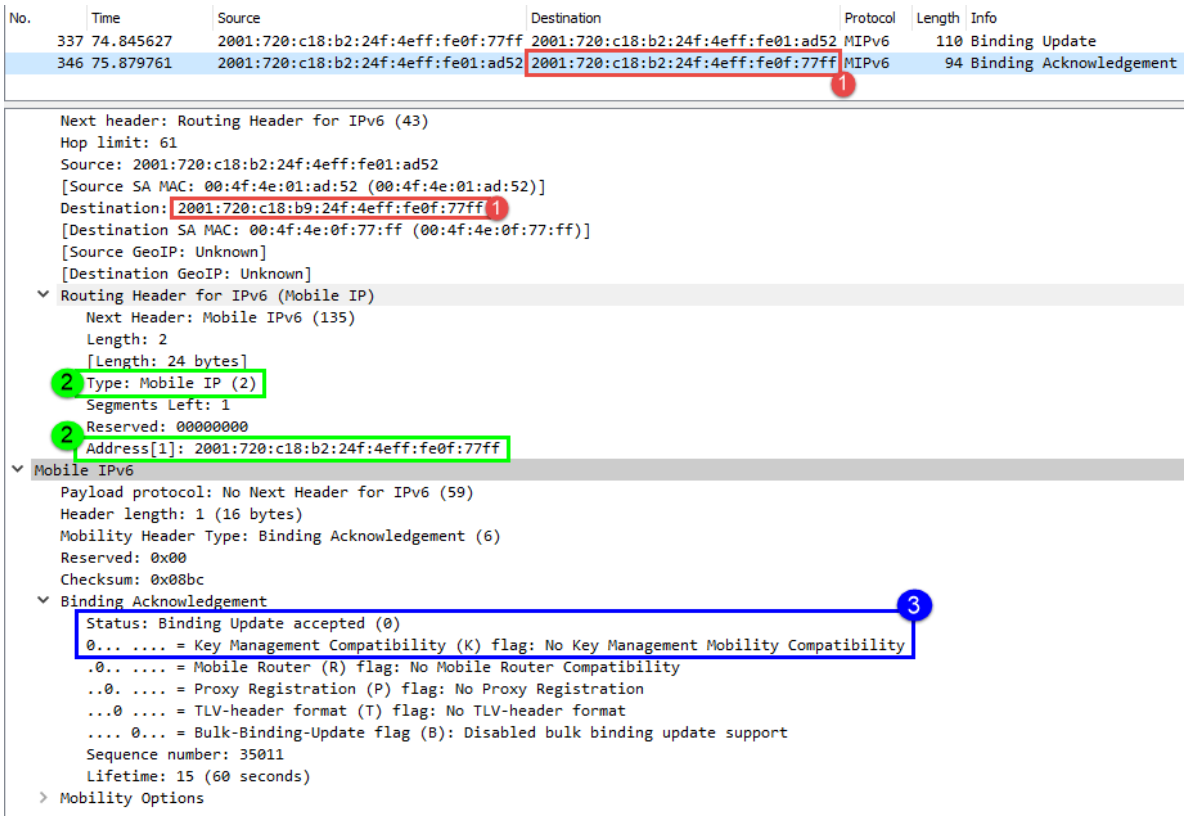


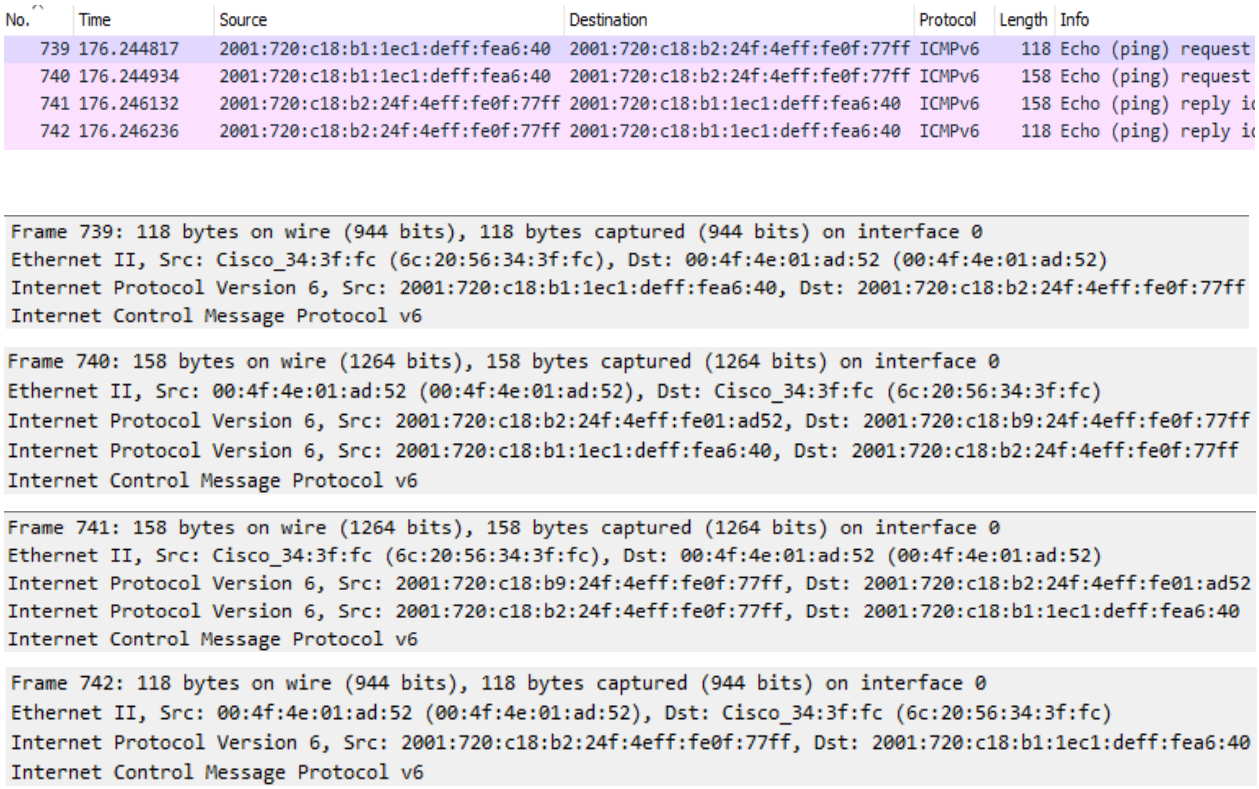
Figura 5.5 Mensaje Binding Ack en Wireshark

Al igual que ocurría en el mensaje BU, la dirección que aparece cómo de destino es distinta a la que realmente lleva el paquete (1). Además, se está enviando una cabecera de tipo 2 junto con la dirección HoA que el MN usará para comprobar que la asociación con el HA es correcta (2).

Por último, si miramos el mensaje BA podemos destacar que el bit K vale 0 indicando así la gestión manual de las claves de seguridad (3).

Llegados a este punto, donde se ha establecido correctamente la asociación entre MN y HA, este último recogerá todo mensaje con destino la *HoA* y se los enviará al MN a través del túnel establecido entre los dos. A su vez, el MN responderá a estos mensajes a través de este mismo túnel y el HA los reenviará al CN.

Podemos ver un ejemplo de este comportamiento si estudiamos en Wireshark los mensajes *echo request* y *echo reply* que se intercambian tras el traspaso del MN a la red foránea. En este caso vamos a hacerlo desde el punto de vista del HA, lo que nos va a permitir analizar las diferencias entre los mensajes *echo* que el HA recibe y los que reenvía.



No.	Time	Source	Destination	Protocol	Length	Info
739	176.244817	2001:720:c18:b1:1ec1:deff:fea6:40	2001:720:c18:b2:24f:4eff:fe0f:77ff	ICMPv6	118	Echo (ping) request
740	176.244934	2001:720:c18:b1:1ec1:deff:fea6:40	2001:720:c18:b2:24f:4eff:fe0f:77ff	ICMPv6	158	Echo (ping) request
741	176.246132	2001:720:c18:b2:24f:4eff:fe0f:77ff	2001:720:c18:b1:1ec1:deff:fea6:40	ICMPv6	158	Echo (ping) reply i
742	176.246236	2001:720:c18:b2:24f:4eff:fe0f:77ff	2001:720:c18:b1:1ec1:deff:fea6:40	ICMPv6	118	Echo (ping) reply i

```
Frame 739: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
Ethernet II, Src: Cisco_34:3f:fc (6c:20:56:34:3f:fc), Dst: 00:4f:4e:01:ad:52 (00:4f:4e:01:ad:52)
Internet Protocol Version 6, Src: 2001:720:c18:b1:1ec1:deff:fea6:40, Dst: 2001:720:c18:b2:24f:4eff:fe0f:77ff
Internet Control Message Protocol v6
```

```
Frame 740: 158 bytes on wire (1264 bits), 158 bytes captured (1264 bits) on interface 0
Ethernet II, Src: 00:4f:4e:01:ad:52 (00:4f:4e:01:ad:52), Dst: Cisco_34:3f:fc (6c:20:56:34:3f:fc)
Internet Protocol Version 6, Src: 2001:720:c18:b2:24f:4eff:fe01:ad52, Dst: 2001:720:c18:b9:24f:4eff:fe0f:77ff
Internet Protocol Version 6, Src: 2001:720:c18:b1:1ec1:deff:fea6:40, Dst: 2001:720:c18:b2:24f:4eff:fe0f:77ff
Internet Control Message Protocol v6
```

```
Frame 741: 158 bytes on wire (1264 bits), 158 bytes captured (1264 bits) on interface 0
Ethernet II, Src: Cisco_34:3f:fc (6c:20:56:34:3f:fc), Dst: 00:4f:4e:01:ad:52 (00:4f:4e:01:ad:52)
Internet Protocol Version 6, Src: 2001:720:c18:b9:24f:4eff:fe0f:77ff, Dst: 2001:720:c18:b2:24f:4eff:fe01:ad52
Internet Protocol Version 6, Src: 2001:720:c18:b2:24f:4eff:fe0f:77ff, Dst: 2001:720:c18:b1:1ec1:deff:fea6:40
Internet Control Message Protocol v6
```

```
Frame 742: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
Ethernet II, Src: 00:4f:4e:01:ad:52 (00:4f:4e:01:ad:52), Dst: Cisco_34:3f:fc (6c:20:56:34:3f:fc)
Internet Protocol Version 6, Src: 2001:720:c18:b2:24f:4eff:fe0f:77ff, Dst: 2001:720:c18:b1:1ec1:deff:fea6:40
Internet Control Message Protocol v6
```

Figura 5.6 Mensajes Ping tras el traspaso

Vamos a analizar uno a uno estos mensajes. El primero (*frame 739*), es el *echo request* que envía el CN con destino la HoA del MN. Cuando llega a la red originaria del MN, el HA captura el mensaje y lo reenvía (*Frame 740*), encapsulándolo en un paquete IPv6 con destino la dirección CoA del MN.

Una vez que el MN lo recibe, responde con un mensaje *echo reply* con origen la HoA y destino la dirección del CN (*Frame 741*). Además, lo encapsula en un paquete IPv6 con origen la CoA y destino la dirección del HA. Finalmente, el HA recibe el mensaje, lo desencapsula y lo reenvía al CN (*Frame 742*).

Viendo el comportamiento de nuestro escenario en esta primera prueba de funcionamiento, parece que cumple con la descripción teórica que dimos en los primeros capítulos del proyecto. Además, hemos podido comprobar como el CN puede seguir comunicándose con el MN a pesar de que este cambie de red, ya que sigue usando la misma dirección IPv6 que tenía antes del traspaso.

Pasamos pues a la siguiente prueba, cuyo análisis puede resultar interesante: Ping con regreso.

5.1.2 Ping con regreso

En esta prueba veremos lo que sucede si el MN regresa a su red original durante un intercambio de mensajes Ping. Partimos de una situación donde el MN se encuentra en una red foránea y comienza a recibir mensajes *echo request* del CN encapsulados por el HA como ya vimos en el apartado anterior.

Ahora simularemos que el MN regresa a su red original. Para ello copiamos en la consola de Minicom el conjunto de órdenes[1] que modificaran la VLAN a la que está conectado el MN pasando de la VLAN 9 a la VLAN 2.

Cuando el MN llega a su red original la BUL cambia, coincidiendo ahora la HoA con la CoA.

[1] Como se comentó en el apartado anterior, este conjunto de ordenes puede consultarse en el Anexo E

```
== BUL_ENTRY ==
Home address    2001:720:c18:b2:24f:4eff:fe0f:77ff
Care-of address 2001:720:c18:b2:24f:4eff:fe0f:77ff
CN address      2001:720:c18:b2:24f:4eff:fe01:ad52
 lifetime = 60,  delay = 1000
 flags: IP6_MH_BU_HOME IP6_MH_BU_ACK IP6_MH_BU_LLOCAL
```

Además, el MN envía un mensaje BU al HA informando de su nueva situación:

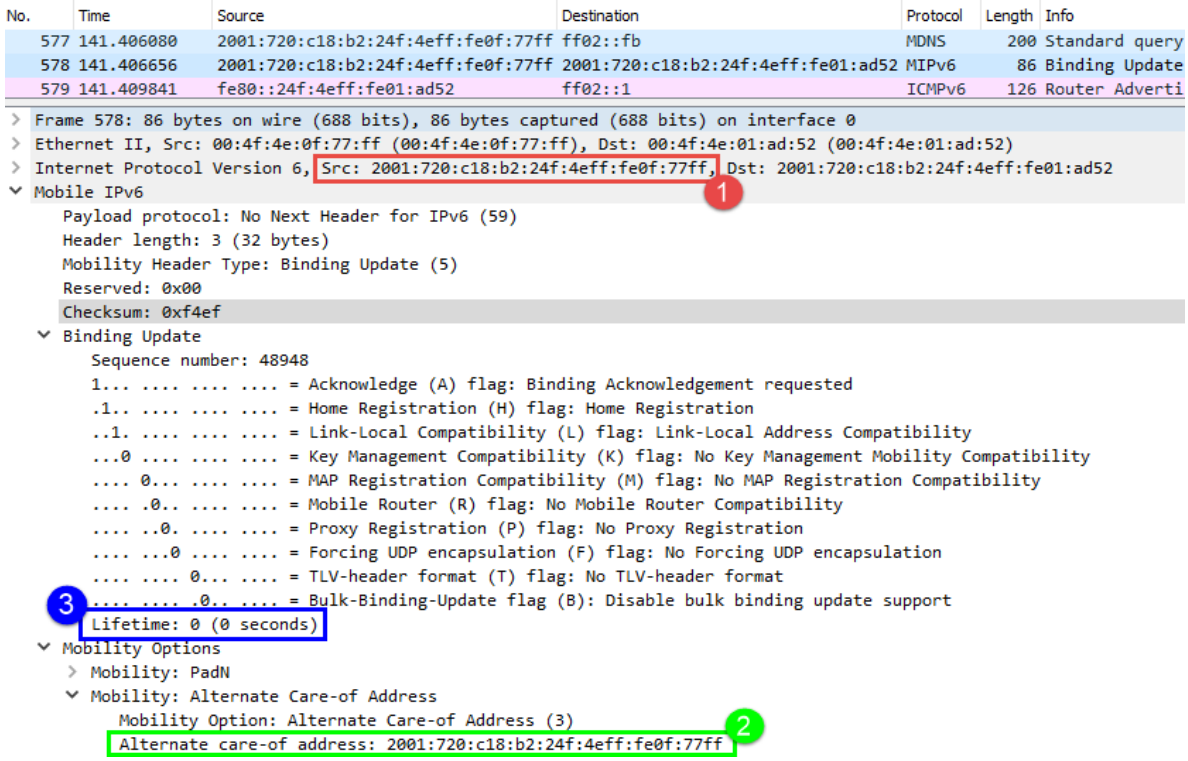


Figura 5.7 Mensaje Binding Update de regreso

Como datos a destacar de este *Binding Update* podemos señalar que tanto la dirección de origen del paquete (1), como la *Alternate CoA* (2), coinciden con la HoA, lo cual es indicador inequívoco de que el MN se encuentra en su red original. Además, al tener una *Lifetime* de 0 segundos se indica que esta asociación no se añadirá a la *Binding Cache* o, si se añade, sería una modificación en el túnel ya creado y se eliminaría seguidamente.

Al no llegar más BU que actualicen el tiempo de vida de la antigua asociación, el tiempo de vida de este termina por expirar, procediéndose a eliminar el túnel entre el HA y la CoA del MN. Este hecho puede observarse en la consola de UMIP en el HA:

```
Mon Nov 21 10:39:14 mh_send_ba: status Binding Update accepted (0)
Mon Nov 21 10:39:14 mh_send: sending MH type 6
from 2001:720:c18:b2:24f:4eff:fe01:ad52
```

```
to 2001:720:c18:b2:24f:4eff:fe0f:77ff
Mon Nov 21 10:39:14 mh_send: remote CoA
2001:720:c18:b9:24f:4eff:fe0f:77ff
Mon Nov 21 10:39:23 mh_bu_parse: Binding Update Received
Mon Nov 21 10:39:23 tunnel_mod: modifying tunnel 6 end points with
from 2001:720:c18:b2:24f:4eff:fe01:ad52
to 2001:720:c18:b2:24f:4eff:fe0f:77ff
Mon Nov 21 10:39:23 __tunnel_mod: modified tunnel iface ip6tnl1 (6)
from 2001:720:c18:b2:24f:4eff:fe01:ad52
to 2001:720:c18:b2:24f:4eff:fe0f:77ff
Mon Nov 21 10:39:23 __tunnel_del: tunnel ip6tnl1 (6)
from 2001:720:c18:b2:24f:4eff:fe01:ad52
to 2001:720:c18:b2:24f:4eff:fe0f:77ff user count decreased to 0
Mon Nov 21 10:39:23 __tunnel_del: tunnel deleted
Mon Nov 21 10:39:23 mh_send_ba: status Binding Update accepted (0)
Mon Nov 21 10:39:23 mh_send: sending MH type 6
from 2001:720:c18:b2:24f:4eff:fe01:ad52
to 2001:720:c18:b2:24f:4eff:fe0f:77ff
```

Como vemos, si se ha registrado la modificación en el túnel debido al BU recibido del MN, pero inmediatamente se elimina dicho túnel al tener un tiempo de vida de 0 segundos.

A partir de este momento, como es lógico los mensajes *echo request* del CN serán entregados directamente al MN sin pasar por el HA, y lo mismo sucederá con los *echo reply* de respuesta del MN. Como muestra, en la siguiente imagen capturada de Wireshark puede comprobarse este hecho.

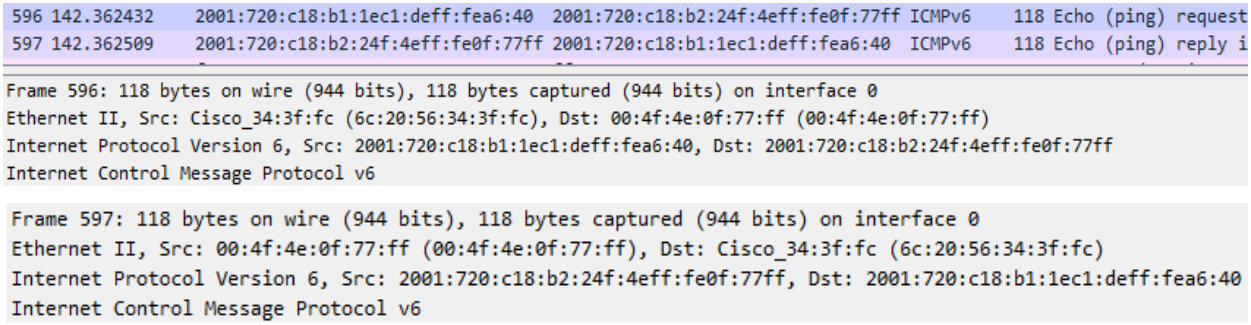


Figura 5.8 Mensajes Ping tras regreso

Para terminar con las pruebas de funcionamiento, vamos a repetir la prueba del primer apartado, pero activando la encriptación con ESP, probando así el correcto comportamiento de nuestro escenario de pruebas con IPSec, según lo explicado en el capítulo 3.

5.1.3 Ping con traspaso. ESP activado

Como paso previo a realizar esta prueba debemos activar el uso de IPSec tanto en el MN como en el HA. Para ello modificamos en el fichero de configuración *mip6d.conf* la opción *UseMnHaIPsec* pasando de *disabled* a *enable*.

Lo más interesante de esta prueba es la encriptación de los mensajes que intercambian MN y HA, por lo que vamos a partir directamente de una situación en la que el MN ya ha cambiado de red y está recibiendo mensajes *echo* del CN a través del HA. Estos mensajes llegarán normalmente al HA pero una vez que este los reenvíe irán encriptados.

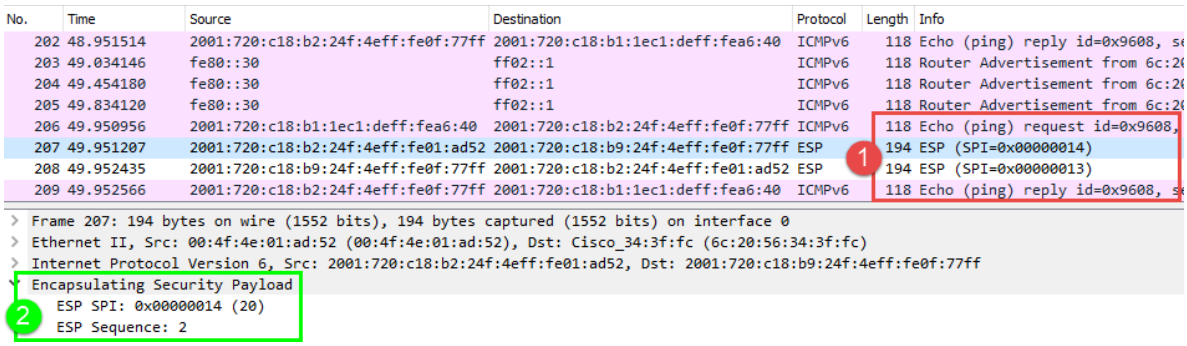


Figura 5.9 Mensajes encriptados con ESP en Wireshark

En la imagen podemos ver los 4 mensajes *echo* que llegan y se reenvían en el HA (1). Los dos primeros serían los *echo request* estando el segundo encriptado (2) ya que se envía del HA al MN. Lo mismo sucede con los *echo reply*, el primero va encriptado ya que lo envía el MN hacia el HA y el segundo lo desencripta el HA para enviarlo al CN.

A continuación, pasamos a analizar las pruebas de rendimiento donde, más allá del correcto funcionamiento del sistema de pruebas que ya ha quedado patente con las pruebas anteriores, veremos si el tiempo que pasa desde que el MN cambia de red hasta que establece la asociación con el HA y se crea el túnel entre ambos es suficientemente pequeño como para que la movilidad sea viable.

5.2 Pruebas de rendimiento

El objetivo de este capítulo es obtener una visión aproximada del rendimiento que MIPv6 puede ofrecer en situaciones lo más próximas a la realidad posible dentro de las limitaciones de nuestro escenario de prueba.

Para medir este rendimiento vamos a basarnos en el tiempo de indisponibilidad que sufre MN desde que cambia de red hasta que establece su asociación con el HA y en la sobrecarga de paquetes que soporta nuestro sistema por retransmisiones o por la propia movilidad IPv6.

Este conjunto de pruebas se realizará en el mismo escenario descrito en el apartado 5.1, por lo que en este apartado no nos detendremos a describirlo de nuevo.

Otro detalle importante a tener en cuenta es que estas pruebas se van a realizar en el nivel de transporte del modelo de capas OSI. En el apartado anterior nos limitamos a estudiar el escenario a nivel 3 ya que sólo se pretendía estudiar que el comportamiento del escenario de pruebas era el correcto, pero de cara a medir el rendimiento hemos considerado importante realizar pruebas que se asemejen al tráfico real, concretamente utilizaremos los protocolos UDP y TCP.

Las pruebas que analizaremos en esta memoria son:

1. UDP del CN al MN con y sin traspaso. El CN enviará datagramas UDP de forma constante de forma que podremos medir el retraso que se produce cuando el MN cambia de red así como los paquetes perdidos. Se utilizará la herramienta Iperf que nos permite crear flujos de datos UDP, junto con su interfaz gráfica Jperf.
2. FTP del CN al MN con y sin traspaso. Situando el servidor FTP en el MN y el cliente en el CN, este enviara un fichero y, al producirse el cambio de red, podremos estudiar las retrasmisiones que se producen, así como la sobrecarga de paquetes. Como cliente FTP se utilizará el software FileZilla y para el servidor se utilizará ProFTPd que viene incluido por defecto en el sistema operativo.

Aparte de estas pruebas, en el CD adjunto a esta memoria pueden encontrarse las mismas pruebas, pero en sentido contrario, es decir, del MN al CN.

5.2.1 UDP. Envío de datagramas UDP del CN al MN

Como hemos comentado en la introducción de este apartado, esta prueba consistirá en el envío de datagramas UDP desde el CN a un puerto predeterminado donde escuchará el MN. Para ello se utilizará el software Iperf y su interfaz gráfica Jperf.

La primera parte de la prueba se producirá en un escenario en el que el MN sigue en su red de origen por lo que no habrá pérdidas causadas por el cambio de red del MN. Para enviar los datagramas, en el CN deberemos configurar Jperf para usar IPv6, con un tamaño de datagrama (1400 bytes) y una duración del envío de 10 segundos. El procedimiento de instalación y configuración de Iperf y Jperf puede encontrarse en el anexo G.

Para esta primera parte de la prueba se llevarán a cabo 5 experimentos ya que no debería aportarnos demasiados datos de interés.

Experimento	Paquetes Enviados/Recibidos	Paquetes perdidos
E1	895	0
E2	895	0
E3	895	0
E4	895	0
E5	895	0

Tabla 5-1 Intercambio de paquetes sin traspaso

Pocas conclusiones podemos sacar de estos experimentos ya que no se pierde ningún paquete al no haber pérdidas provocadas por movilidad.

Además, hay que comentar que para obtener el número total de paquetes enviados o recibidos el procedimiento que seguimos es aplicar un filtro en Wireshark para quedarnos solo con los paquetes que nos interesan. Una vez tenemos filtrada nuestra captura de paquetes pulsamos en *File – Export Expecified Packets* y nos aparecerá una ventana donde podremos encontrar el número de paquetes.

Pasemos ahora a la segunda parte de nuestra prueba. En este caso, el MN cambiará de red durante el tiempo que el CN está enviándole datagramas UDP.

Con esta prueba podremos medir, entre otras cosas, el tiempo de indisponibilidad del MN debido al traspaso y el número de paquetes perdidos durante este proceso, lo que nos dará una idea de la viabilidad de este protocolo lo movilidad para una comunicación con UDP.

Durante esta prueba hemos detectado ciertas cosas que debemos comentar. Lo primero es que si se observan las pruebas llegan más de 895 paquetes UDP, pero los sobrantes se envían fuera de los 10 segundos de envío configurados en Jperf y consideramos que no son paquetes de datos, sino de información de control, por lo que no los tendremos en cuenta en nuestros resultados.

Además, podemos ver varios mensajes “Port Unreachable”. Este tipo de mensajes vienen provocados por

problemas en el host. Entre las posibles causas están la configuración del firewall, la saturación del receptor o la indisponibilidad el puerto destino.

Por último, Para esta prueba vamos a repetir el experimento 10 veces pudiendo así obtener una media de los valores obtenidos para los distintos parámetros lo que nos dará una visión más aproximada del desempeño real del escenario.

Experimento	Enviados	Recibidos	No Recibidos	% Perdidos	Tiempo de indisponibilidad
E1	895	458	437	48,82681564	4,907617
E2	895	500	395	44,13407821	4,429165
E3	895	487	408	45,58659218	4,582608
E5	895	454	441	49,27374302	4,952284
E6	895	443	452	50,5027933	5,075498
E7	895	504	391	43,68715084	4,395132
E8	895	472	423	47,26256983	4,769876
E9	895	401	494	55,19553073	5,545805
E10	895	505	390	43,57541899	4,386001
Media	895	467,040832	425,804112	47,57587844	4,784034379

Tabla 5-2 Paquetes UDP Intercambiados/Perdidos con traspaso

Como puede observarse en la Tabla 5-2 los resultados no son especialmente alentadores. Casi un 50% de paquetes perdidos provocados por un tiempo de indisponibilidad medio de casi 5 segundos que hacen que la aplicación de MIPv6 para un intercambio de datos con UDP sea totalmente inviable. Vamos a analizar además una gráfica generada en Wireshark para un experimento concreto donde podremos observar las 3 fases por las que pasa el experimento [49]. En el anexo H puede consultarse el procedimiento para obtener las distintas gráficas que aparecerán a lo largo de este capítulo.

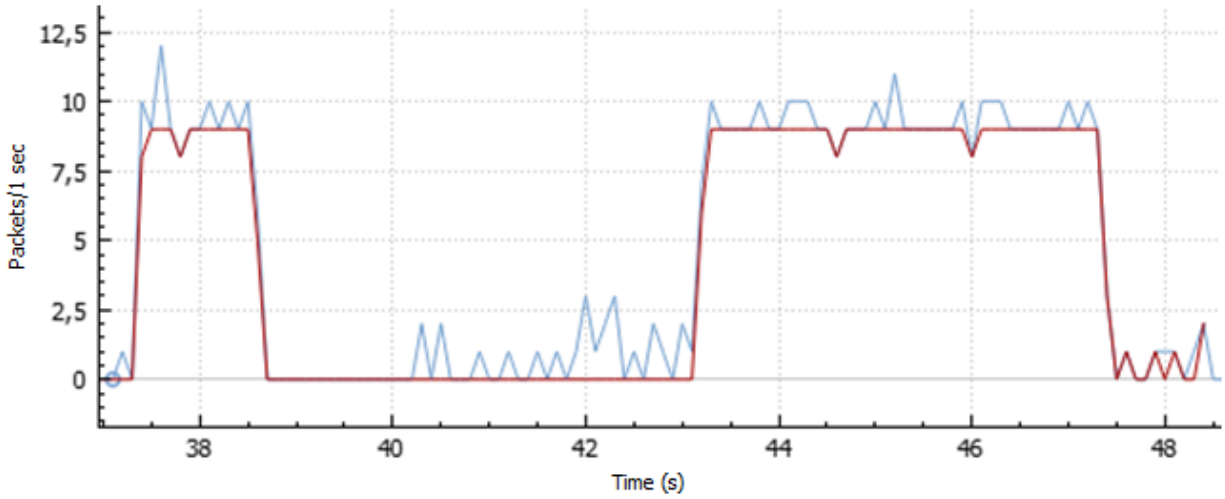


Figura 5.10 Gráfica I/O de paquetes UDP

Antes del traspaso el MN recibe datagramas UDP (color rojo) normalmente hasta un instante entre el segundo 38 y 39 donde se produce el traspaso. Dejamos entonces de recibir datagramas UDP, aunque seguimos recibiendo paquetes del protocolo ND o de MIPv6 (color azul). Finalmente, alrededor del segundo 43 volvemos a recibir datagramas UDP una vez se ha establecido el traspaso del MN a la red foránea. Además, puede observarse claramente la indisponibilidad de uno 4,5 segundos, algo inferior a la media obtenida de los 10 experimentos.

Como hemos comentado antes, la conclusión más clara que puede sacarse es que el tiempo de indisponibilidad que obtenemos aplicando nuestro protocolo es demasiado grande como para que pueda utilizarse en aplicaciones que intercambien datos vía UDP, como puede ser una transmisión en streaming o una comunicación con VoIP.

5.2.2 FTP. Transferencia de archivo del CN al MN

Una vez realizada la prueba de comunicación con el protocolo UDP y analizado el rendimiento de nuestro escenario de pruebas, pasamos ahora a utilizar otro protocolo de la capa de transporte como TCP.

Para ello usaremos un protocolo de un nivel superior como FTP que se apoya en TCP, concretamente transferiremos un archivo desde el CN al MN utilizando este protocolo.

Lo primero que necesitamos es un servidor FTP que ubicaremos en el MN. Como comentamos en la introducción de este capítulo, vamos a utilizar ProFTPd que viene instalado por defecto en Debian.

Además, el CN ejercerá de cliente FTP y, para ello, hemos instalado el software FileZilla para facilitarnos el trabajo, aunque podríamos habernos limitado a usar la línea de comandos. Recordamos que los detalles de instalación y configuración de FileZilla pueden encontrarse en el anexo C.

Tal como sucedió en la prueba de UDP, inicialmente transferiremos el fichero up.png de 2 Mb sin que el MN haya cambiado de red y solo repetiremos la prueba 5 veces. Se contabilizan los paquetes de datos y de control junto con el tiempo desde que se transmite el primer paquete de datos hasta que se recibe el mensaje de transferencia FTP completa, dejando fuera el intercambio de paquetes del comando MLSD. Señalar también que tanto el tráfico de datos como el de control compartirán el mismo camino en todo momento durante nuestra prueba (antes y después del traspaso del MN).

Prueba	Paquetes	Tiempos
E1	2475	8,0758
E2	2469	7,795
E3	2476	7,9907
E4	2445	7,8588
E5	2399	7,8646
Media	2452,8	7,91698

Tabla 5-3 Paquetes TCP intercambiados y tiempo de transferencia sin traspaso

La variación del número de paquetes es llamativa porque, presumiblemente, no deberían producirse pérdidas, y por tanto retransmisiones, al no haberse producido el traspaso del MN. Sin embargo, la aparición de mensajes *TCP ACKed unseen segment* provocan esta oscilación del número final de paquetes. Estos mensajes aparecen cuando Wireshark captura un ACK correspondiente a un número de secuencia de envío que no ha capturado. La explicación hallada para esta situación es que la velocidad del capturador no sea suficiente escapándosele algunos paquetes.

Pasamos ahora a un escenario donde el CN comienza a transmitir el archivo y, durante dicho proceso, el MN cambia de red. Esta situación nos va a permitir medir parámetros el número de paquetes transferidos, el tiempo total de la transferencia o la latencia (retardo de los paquetes). Primero veamos en 10 repeticiones de la prueba el tiempo total de la transferencia FTP así como los paquetes transmitidos.

Prueba	Paquetes	Tiempos
E1	2640	12,8838
E2	2683	12,6635
E3	2634	13,2137
E4	2696	12,5088
E5	2625	12,9847
E6	2707	12,5996
E7	2679	12,6317
E8	2690	12,6885
E9	2675	12,6804
E10	2684	12,603
Media	2671,3	12,74577

Tabla 5-4 Paquetes intercambiados y tiempo de trasferencia con traspaso

Como vemos, el tiempo de transmisión medio ha aumentado en unos 5 segundos y el número de paquetes en unos 200 debido al traspaso del MN. Es fácil determinar que este aumento en ambos parámetros, con respecto a la situación sin movilidad, es debido a la retransmisión de paquetes a causa del periodo de indisponibilidad por el traspaso del MN.

Si nos centramos ahora en un experimento cualquiera de los 10 realizados podremos observar con más detalle el tiempo de indisponibilidad o la latencia. Para ello haremos uso de dos gráficas TCP que nos aportan información útil en Wireshark: gráfica Stevens y gráfica Round Trip Time (RTT). Antes de empezar ya intuimos que la latencia, o retardo temporal, debido al traspaso no coincidirá con el tiempo de indisponibilidad de la red. Esto se debe al control de congestión que incluye TCP. Explicado a grandes rasgos, este control de congestión también conocido como slow-start, o arranque lento, va doblando el tamaño de la ventana de transmisión por cada ACK recibido del receptor, hasta que se produce un vencimiento del tiempo límite de retransmisión, momento en el que vuelve al tamaño anterior.

En nuestro caso, al producirse congestión en la red debido al traspaso, se producirán varios vencimientos del tiempo de retransmisión, por lo que se irá reduciendo la tasa de transmisión. Por lo tanto, es fácil deducir que el retardo de los paquetes retransmitidos será superior al tiempo de indisponibilidad.

Sabiendo esto, veamos ahora las gráficas nombradas antes empezando por la gráfica Stevens, que nos muestra la evolución en el tiempo de los números de secuencia de un flujo de datos TCP. Como hemos dicho, tomaremos un experimento cualquiera de los 10 realizados para esta prueba [50].

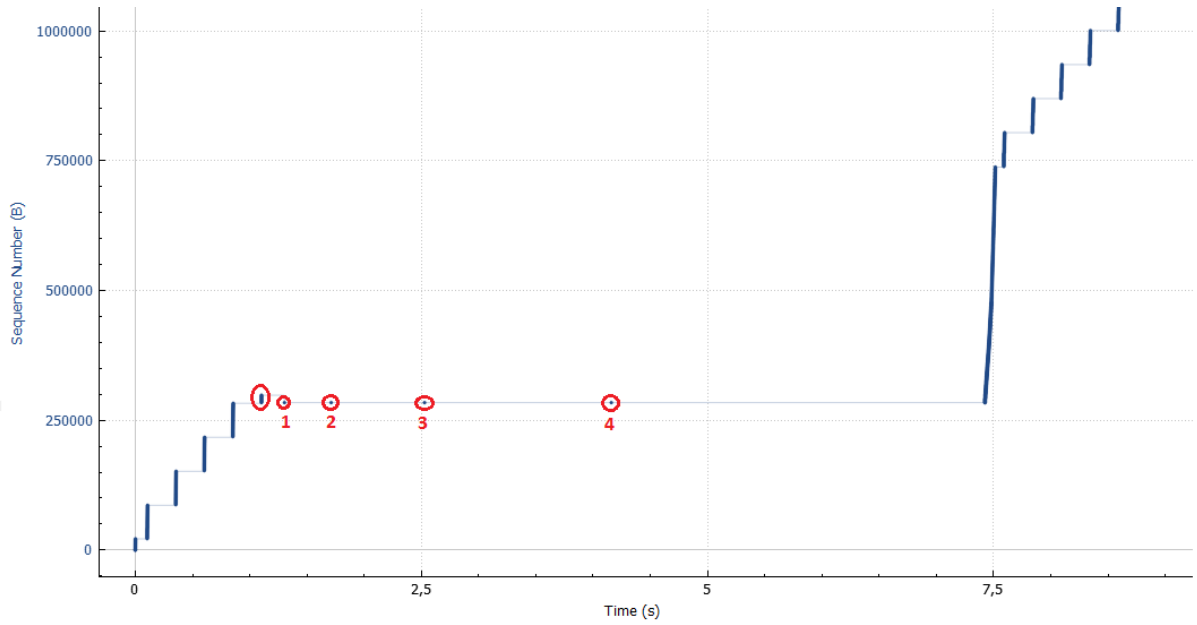


Figura 5.11 Gráfica Stevens del flujo TCP

En esta gráfica puede observarse claramente lo que hemos comentado sobre el control de congestión, ya que cada uno de los intentos de retransmisión está separado por el doble de tiempo que el anterior. Además, podemos ver que el tiempo de retransmisión es algo superior a los 6 segundos. Podemos compararlo con el tiempo de indisponibilidad[1] para este experimento que es de 4,41 segundos.

Veamos ahora la siguiente gráfica (RTT), que nos muestra el Round Trip Time de cada paquete, es decir, el tiempo de transmisión de dicho paquete más el tiempo que tardamos en recibir su correspondiente asentimiento.

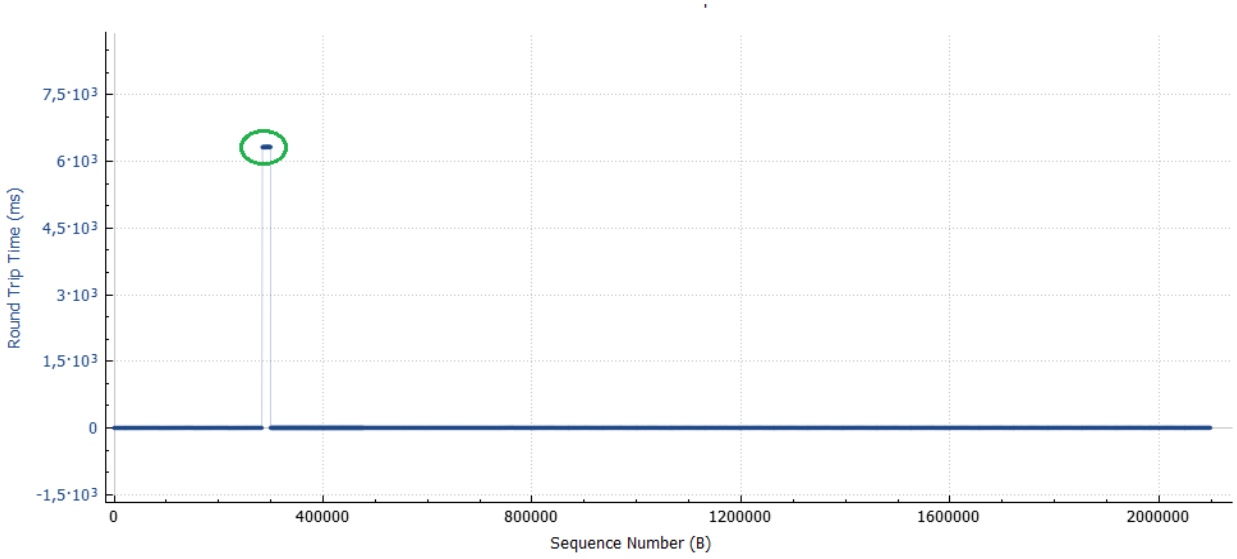


Figura 5.12 Grafica RTT del flujo TCP

Es fácil detectar cuales son los paquetes que han sufrido retransmisiones debido al traspaso. Si hacemos zoom en dichos paquetes:

[1] Tomándolo como el tiempo entre que se envía el último segmento de datos antes del traspaso y se recibe el binding ACK del HA

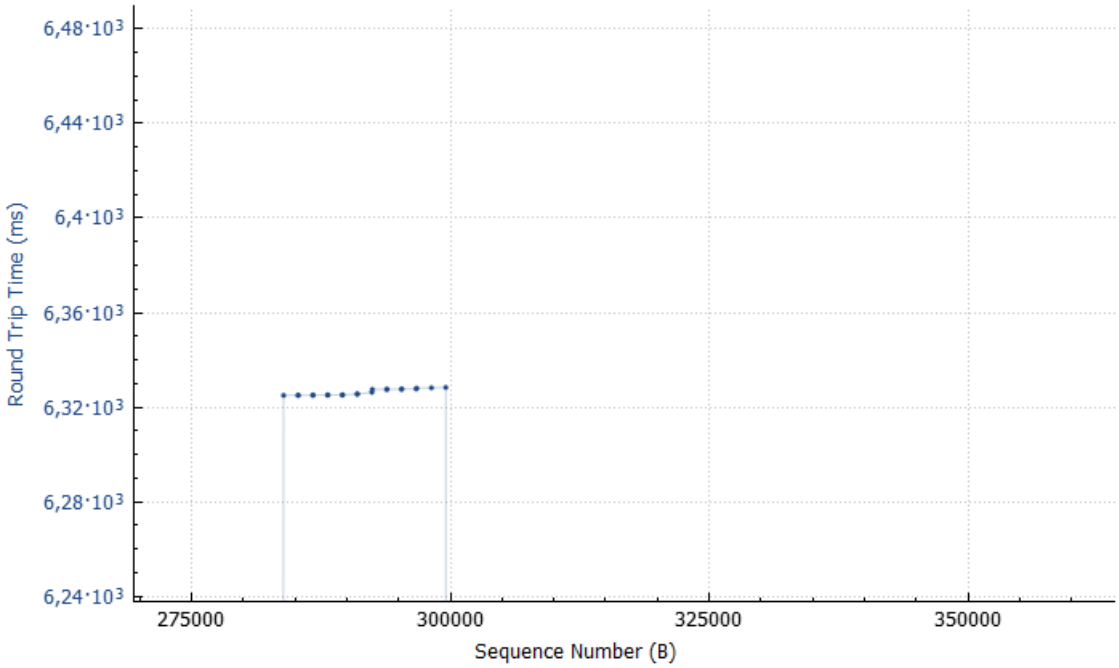


Figura 5.13 Gráfica RTT con zoom

Podemos ver que los paquetes afectados por las retransmisiones debido al traspaso alcanzan un RTT de unos 6.3 segundos, y que, tal como esperábamos los números de secuencia se corresponden con los de los paquetes retransmitidos debido al traspaso:

```
TCP      86 60884 → 56183 [ACK] Seq=1 Ack=285288 Win=120000 Len=0 TSva
TCP     126 [TCP Retransmission] 56183 → 60884 [ACK] Seq=292428 Ack=1
TCP    1474 [TCP Retransmission] 56183 → 60884 [ACK] Seq=292468 Ack=1
TCP      86 60884 → 56183 [ACK] Seq=1 Ack=285328 Win=120000 Len=0 TSva
TCP     126 [TCP Retransmission] 56183 → 60884 [ACK] Seq=293856 Ack=1
TCP      86 60884 → 56183 [ACK] Seq=1 Ack=286716 Win=122816 Len=0 TSva
TCP    1474 [TCP Retransmission] 56183 → 60884 [ACK] Seq=293896 Ack=1
TCP      86 60884 → 56183 [ACK] Seq=1 Ack=286756 Win=122816 Len=0 TSva
TCP     126 [TCP Retransmission] 56183 → 60884 [ACK] Seq=295284 Ack=1
TCP      86 60884 → 56183 [ACK] Seq=1 Ack=288144 Win=122816 Len=0 TSva
TCP    1474 [TCP Retransmission] 56183 → 60884 [ACK] Seq=295324 Ack=1
TCP      86 60884 → 56183 [ACK] Seq=1 Ack=288184 Win=122816 Len=0 TSva
TCP     126 [TCP Retransmission] 56183 → 60884 [ACK] Seq=296712 Ack=1
TCP      86 60884 → 56183 [ACK] Seq=1 Ack=289572 Win=122816 Len=0 TSva
TCP    1474 [TCP Retransmission] 56183 → 60884 [ACK] Seq=296752 Ack=1
TCP      86 60884 → 56183 [ACK] Seq=1 Ack=289612 Win=122816 Len=0 TSva
TCP     126 [TCP Retransmission] 56183 → 60884 [ACK] Seq=298140 Ack=1
TCP      86 60884 → 56183 [ACK] Seq=1 Ack=291000 Win=122816 Len=0 TSva
TCP    1474 [TCP Retransmission] 56183 → 60884 [ACK] Seq=298180 Ack=1
TCP      86 60884 → 56183 [ACK] Seq=1 Ack=291040 Win=122816 Len=0 TSva
TCP     126 [TCP Retransmission] 56183 → 60884 [ACK] Seq=299568 Ack=1
```

Figura 5.14 Paquetes TCP retransmitidos

Como conclusión, ya que en TCP se prioriza la transferencia del 100% de los datos intercambiados por encima del bajo retardo, podemos decir que el protocolo MIPv6 es válido si se usa como soporte de TCP, aun incrementándose ligeramente el tiempo total te transmisión.

No obstante, nos gustaría mejorar en lo posible el rendimiento de nuestro escenario dentro de nuestras posibilidades.

5.2.3 Pruebas Optimizadas

Para mejorar el rendimiento de nuestro escenario se identificó como principal causa del retardo en el traspaso, el tiempo que tarda el MN en detectar su nueva situación y en establecer la asociación con él HA.

Para reducir este tiempo, se localizó un parámetro de configuración en el router Cisco que nos permite saltarnos las fases de escucha y aprendizaje en el protocolo Spanning Tree. Dicho parámetro se configura en el puerto al que se conecta el MN y se activa con la orden "*spanning-tree portfast"*.

Una vez aplicado este cambio se repitieron todas las pruebas mostradas durante en los apartados anteriores de este capítulo y a continuación mostraremos los resultados de algunas de ellas, concretamente las pruebas de rendimiento.

5.2.3.1 UDP del CN al MN con traspaso optimizado.

Al igual que en la prueba original repetiremos el experimento 10 veces. Los resultados obtenidos son los siguientes:

Experimento	Enviados	Recibidos	No Recibidos	% Perdidos	Tiempo de indisponibilidad
p1	895	651	244	27,26256983	2,749492
p2	895	616	279	31,17318436	3,137883
p3	895	676	219	24,46927374	2,465811
p4	895	610	285	31,84357542	3,205125
p5	895	668	227	25,36312849	2,5555
p6	895	622	273	30,5027933	3,081858
p7	895	648	247	27,59776536	2,779428
p8	895	606	289	32,29050279	3,238561
p9	895	667	228	25,47486034	2,566608
p10	895	623	272	30,39106145	3,059396
Media	895	638,214488	255,068604	28,49928538	2,870308137

Tabla 5-5 Resultados comunicación UDP con configuración optimizada

Comparemos ahora con los resultados obtenidos en la prueba original:

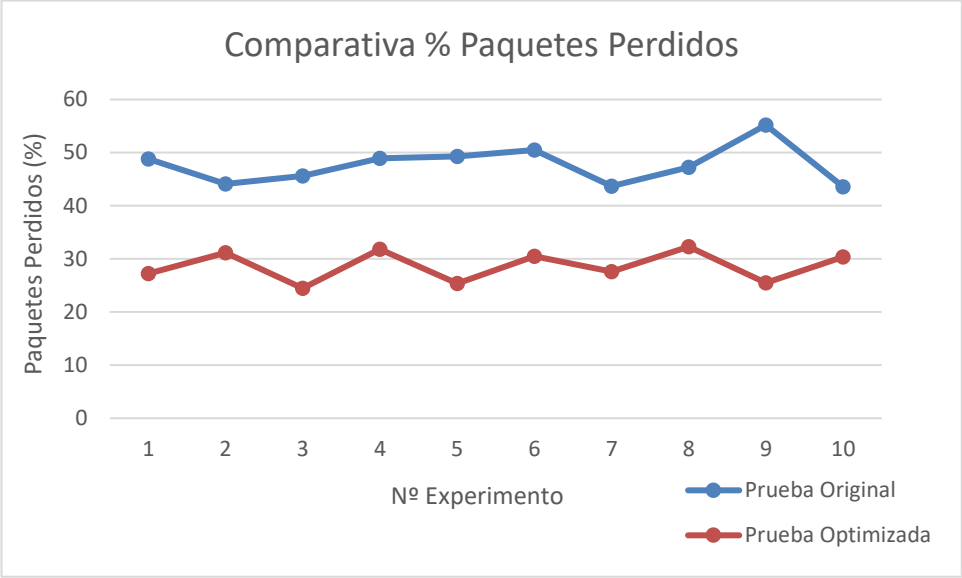


Figura 5.15 Comparativa paquetes perdidos en prueba original y optimizada

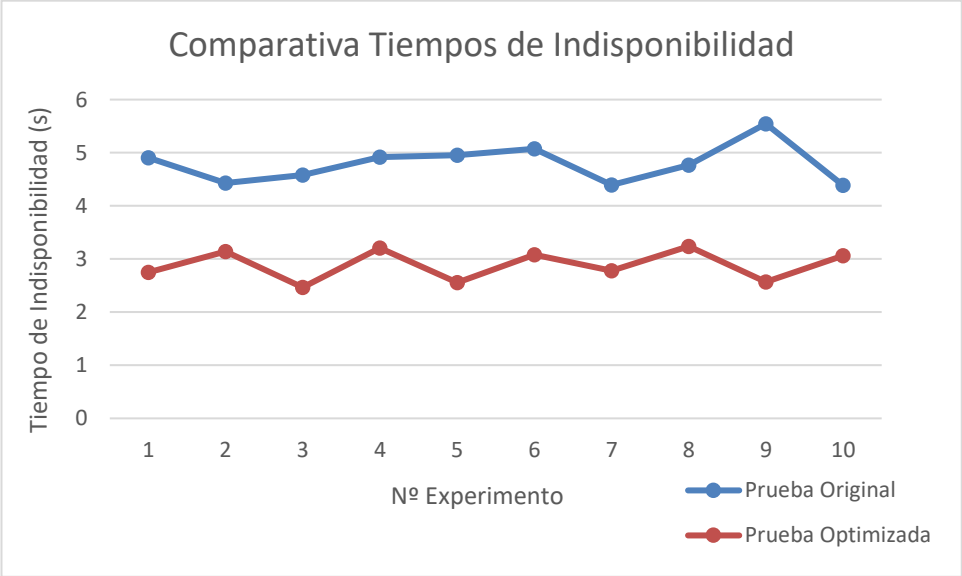


Figura 5.16 Comparativa Tiempo de Indisponibilidad en prueba original y optimizada

Es evidente que los resultados han mejorado considerablemente. La media de paquetes perdidos se ha reducido en un 20% aproximadamente, de un 48% a un 28%, y el tiempo medio de indisponibilidad ha bajado unos 2 segundos.

No obstante, seguimos teniendo casi 3 segundos de tiempo medio de indisponibilidad lo que confirma que este protocolo de movilidad no es el más apropiado para UDP, ya que los extremos de la comunicación podrían considerar que se ha perdido la conexión y detener la transmisión de datos.

5.2.3.2 FTP del CN al MN con traspaso optimizado

Veamos ahora que resultados obtenemos en una transferencia FTP igual a la realizada en el apartado 5.2.2 una vez cambiada la configuración del Router. Al igual que hemos hecho con la prueba de UDP, vamos a mostrar una tabla con los resultados obtenidos y luego haremos una comparativa con los obtenidos en la prueba original.

Prueba	Paquetes	Tiempos
E1	2687	9,4572
E2	2694	9,4939
E3	2647	9,3608
E4	2674	9,4563
E5	2647	9,4644
E6	2670	9,4944
E7	2650	9,3215
E8	2655	9,4639
E9	2691	9,3999
E10	2652	9,4943
Media	2666,7	9,44066

Tabla 5-6 Resultados Flujo TCP con configuración optimizada

Veamos ahora una comparativa con la prueba original.

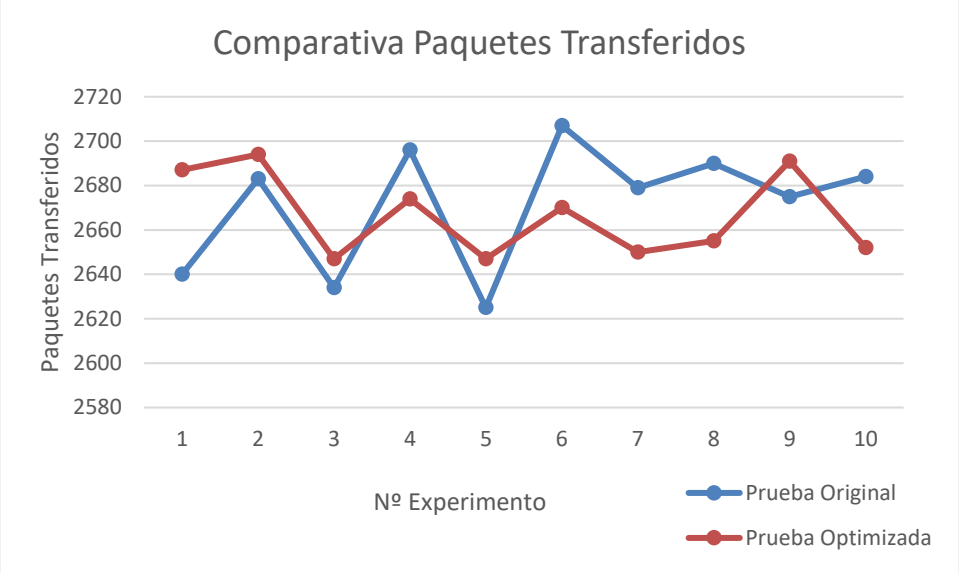


Figura 5.17 Comparativa paquetes transferidos en prueba original y optimizada

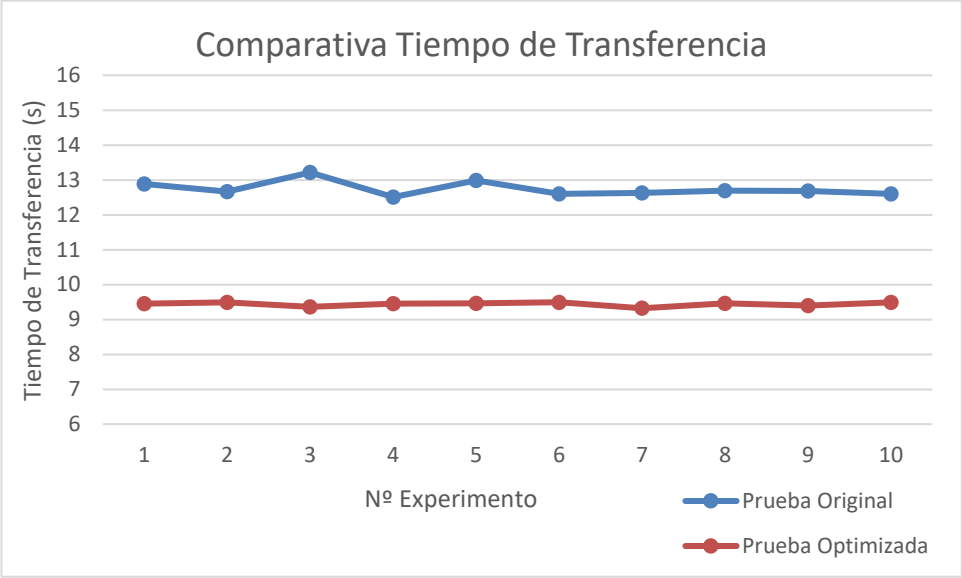


Figura 5.18 Comparativa Tiempo de transferencia en prueba original y optimizada

Como vemos en las gráficas comparativas, el tiempo de transferencia ha disminuido en unos 3 segundos, aunque el número de paquetes transferidos es similar. Entendemos que esto se debe al ya citado control de congestión de TCP que ralentiza la transmisión de datos cuando la red se satura, por lo que no se nota la diferencia entre una prueba y otra.

La mejor forma de confirmar esta teoría es consultar la gráfica Stevens de un experimento cualquiera en esta prueba optimizada.

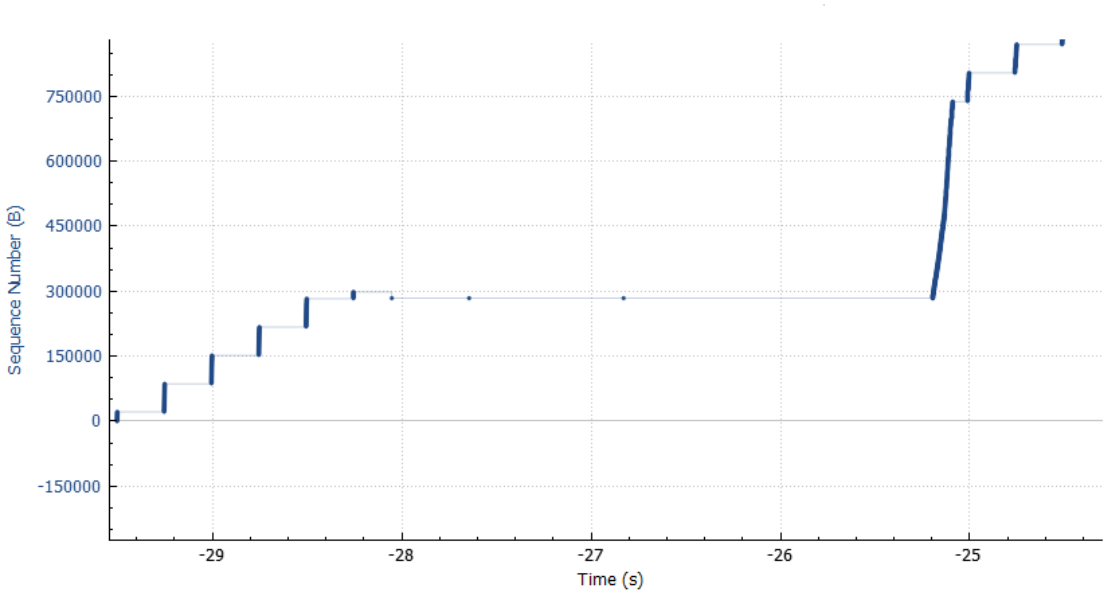


Figura 5.19 Gráfica Stevens en prueba optimizada

Como vemos, el aspecto es muy similar a la que vimos en el apartado 5.2.2, por lo que tiene sentido que el número medio de paquetes transferidos sea aproximadamente el mismo.

Como conclusión, nuevamente hemos visto cómo el rendimiento de nuestro escenario ha mejorado con la nueva configuración del router, ya que ha mejorado el rendimiento de la transferencia FTP, haciéndola más rápida e igualmente fiable.

6 CONCLUSIONES Y TRABAJOS FUTUROS

Para terminar, cerramos este proyecto fin de carrera con mis conclusiones y propuestas de trabajo futuro relacionadas con el trabajo hecho.

En primer lugar, aun siendo consciente de las limitaciones del escenario de pruebas utilizado considero que las conclusiones de este proyecto pueden ser aprovechadas como referencia para comprender qué problemas y qué ventajas presenta el protocolo de movilidad estudiado.

Como hemos comentado tras mostrar los resultados de las pruebas, el protocolo MIPv6 parece válido para una comunicación que se apoye en TCP y no sea sensible al retraso, ya que la comunicación se completa correctamente, aunque con algo de retraso debido al tiempo de indisponibilidad del MN debido al traspaso.

Este mismo tiempo de indisponibilidad es el que impide que MIPv6 sea una buena opción para una comunicación sensible al retraso (normalmente basada en UDP), ya que aplicaciones tipo streaming se ven interrumpidas por largos periodos de indisponibilidad (varios segundos), como es nuestro caso.

En cuanto a posibles trabajos futuros, y teniendo en cuenta las limitaciones con las que contamos para simular este tipo de protocolos, considero una buena opción un estudio en profundidad que compare los distintos protocolos de movilidad existentes, de forma que puedan observarse las ventajas e inconvenientes de cada uno.

En caso de buscarse un enfoque más práctico, se podría intentar construir un escenario de pruebas para el protocolo PMIPv6 (Proxy Mobile IPv6), objetivo inicial de este proyecto que tuve que abandonar al no encontrar ninguna forma de simular los distintos elementos que intervienen en dicho protocolo en nuestros equipos.

Por último, y como apunte personal, he de decir que este proyecto me ha llevado a estudiar protocolos que, o bien no conocía, como los de movilidad, o bien he tenido que repasar tras estudiarlos a lo largo de la carrera en las asignaturas propias de la rama de telemática. Además, he tenido la oportunidad de configurar desde cero un router Cisco, algo que he disfrutado mucho y que me gustaría estudiar más en profundidad en el futuro, por ejemplo, con una certificación Cisco.

ANEXO A. EQUIPOS UTILIZADOS

En este anexo se recoge las características más relevantes conocidas de cada uno de los equipos utilizados en el escenario de pruebas utilizado en este proyecto.

A.1. Mobile Node

- Sistema Operativo Debian:
  - Versión 7.4 (wheezy) de 32-bit
  - Núcleo Linux 3.8.2-mipv6
  - GNOME 3.4.2
- Hardware
  - Memoria RAM disponible: 501.9 Mb
  - Procesador: Intel Pentium® 4 CPU 3.00 Ghz
  - Dos interfaces de red Ethernet con controlador Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+
  - Tarjeta gráfica Nvidia GeForce FX 5200

Figura A.0.1 Mobile Node

A.2. Home Agent

- Sistema Operativo Debian:
  - Versión 7.4 (wheezy) de 32-bit

- Núcleo Linux 3.8.2-mipv6
- GNOME 3.4.2

- Hardware
  - Memoria RAM disponible: 501.9 Mb
  - Procesador: Intel Pentium® 4 CPU 1.50 Ghz
  - Una interfaz de red Ethernet con controlador Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+
  - Tarjeta gráfica Nvidia RIVA TNT2 model 64

Figura A.0.2 Home Agent

A.3. Routers Cisco 892

Figura A.0.3 Cisco 892

La siguiente tabla muestra las características de los dos routers utilizados localmente en nuestro escenario de pruebas. No poseemos información acerca del Router CICA.

General	
Tipo de dispositivo	*Router - conmutador de 8 puertos (integrado)*
Tipo incluido	*Sobremesa*
Tecnología de conectividad	*Cableado*
Protocolo de interconexión de datos	*Ethernet, Fast Ethernet*
Capacidad	*Túneles VPN IPSec : 50*
Red / Protocolo de transporte	*L2TP, IPSec, FTP, DHCP, DNS, L2TPv3, DDNS*
Protocolo de direccionamiento	*OSPF, RIP-1, RIP-2, BGP, EIGRP, HSRP, VRRP, NHRP, PIM-SM, GRE*
Protocolo de gestión remota	*Telnet, SNMP 3, HTTP, HTTPS, SSH*
Algoritmo de cifrado	*LEAP, DES, Triple DES, SSL, PEAP, TKIP, PKI, AES de 128 bits, AES de 192 bits, AES de 256 bits*
Método de autentificación	*RADIUS, TACACS+*
Características	*Soporte de NAT, asistencia técnica VPN, equilibrio de carga, soporte VLAN, señal ascendente automática (MDI/MDI-X automático), snooping IGMP, limitación de tráfico, Stateful Packet Inspection (SPI), filtrado de contenido, soporte DiffServ, filtrado de dirección MAC, soporte IPv6, Alta disponibilidad, Sistema de prevención de intrusiones (IPS), filtrado de URL, Stateful Failover, Class-Based Weighted Fair Queuing (CBWFQ), Weighted Fair Queuing (WFQ), admite Spanning Tree Protocol (STP), soporte de Access Control List (ACL), Quality of Service (QoS), Link Fragmentation and Interleaving (LFI), Dynamic Multipoint VPN (DMVPN), recuperación de errores WAN, Servidor DHCP, Virtual Route Forwarding-Lite (VRF-Lite), DNS proxy, Bidirectional Forwarding Detection (BFD)*
Cumplimiento de normas	*IEEE 802.1D, IEEE 802.1Q, IEEE 802.1x , CISPR 24, EN 61000-3-2, EN55022, ICES-003, EN 61000-3-3, EN55024, CISPR 22, EN50082-1, EN55022 Class B, ICES-003 Class B, EN 61000-6-1, AS/NZ 3548 Class B, EN 60555-2, FCC CFR47 Part 15, EN300-386, FCC CFR47 Part 15 B, VCCI V-3*
Memoria RAM	*512 MB (instalados) / 768 MB (máx.)*
Memoria Flash	*256 MB (instalados) / 256 MB (máx.)*
Indicadores de estado	*Estado puerto, alimentación*
Expansión/Conectividad	
Interfaces	*LAN : 8 x 10Base-T/100Base-TX - RJ-45* *Administración : 1 x consola - RJ-45* *WAN : 1 x 10Base-T/100Base-TX - RJ-45* *WAN : 1 x 10Base-T/100Base-TX/1000Base-T - RJ-45* *Administración : 1 x auxiliar - RJ-45* *USB 2.0 : 2 x 4 PIN USB tipo A* *WAN : 1 x BRI ST*

Tabla A-1 Características CISCO 892

A.4 Correspondent Node

- Sistema Operativo Ubuntu:
  - Versión 10.04 LTS
  - Núcleo Linux 2.6.32-38
- Hardware
  - Memoria RAM disponible: 3.8 Gb
  - Procesador: Intel® Core™ i5 CPU procesador de 2.53 GHz
  - Una interfaz de red Atheros AR9285 802.11b/g/n Wifi Adapter
  - Tarjeta gráfica Nvidia GeForce FX 5200

ANEXO B: INSTALACIÓN MINICOM

Minicom es un software de emulación de terminal para Sistemas Operativos Linux. En nuestro proyecto será usado tanto para poder resetear el router como para posteriores configuraciones de éste.

La instalación es muy sencilla, solo tenemos que abrir un terminal y teclear: apt-get install minicom

Una vez completada, procedemos a la configuración. En primer lugar, escribimos en el terminal, como root, minicom –s, y nos aparecerá una ventana como la siguiente:

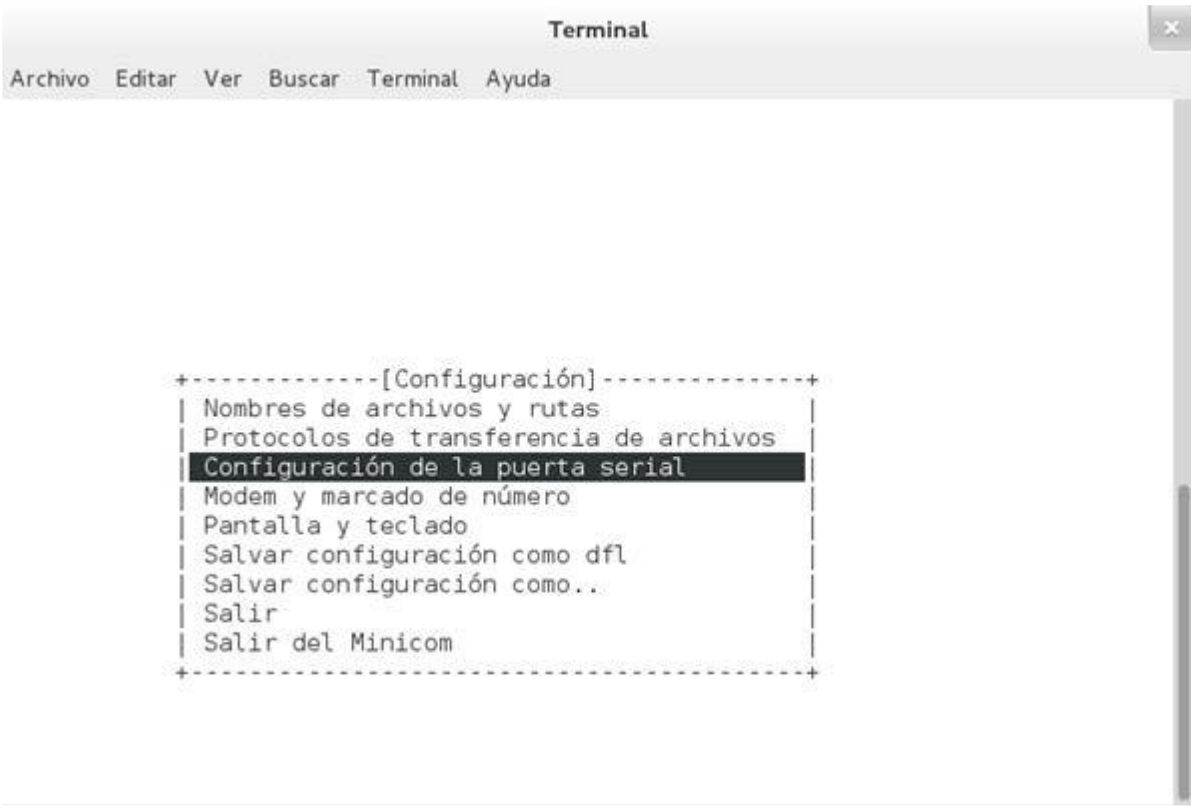


Figura B.0.1 Menú de configuración de Minicom

Nos vamos a configuración de la puerta serial, donde con la tecla F le tenemos que decir al programa que no queremos establecer ningún control de flujo por hardware. Posteriormente, pulsando en A, cambiamos la ruta donde se encuentra el puerto serial, prestando especial atención, ya que el que usamos en nuestro caso se encuentra en /dev/ttyS0, pero la primera vez que se salva la configuración, el programa puede cambiar automáticamente la ruta a /dev/tty0, lo que impediría poder conectarnos al dispositivo deseado.

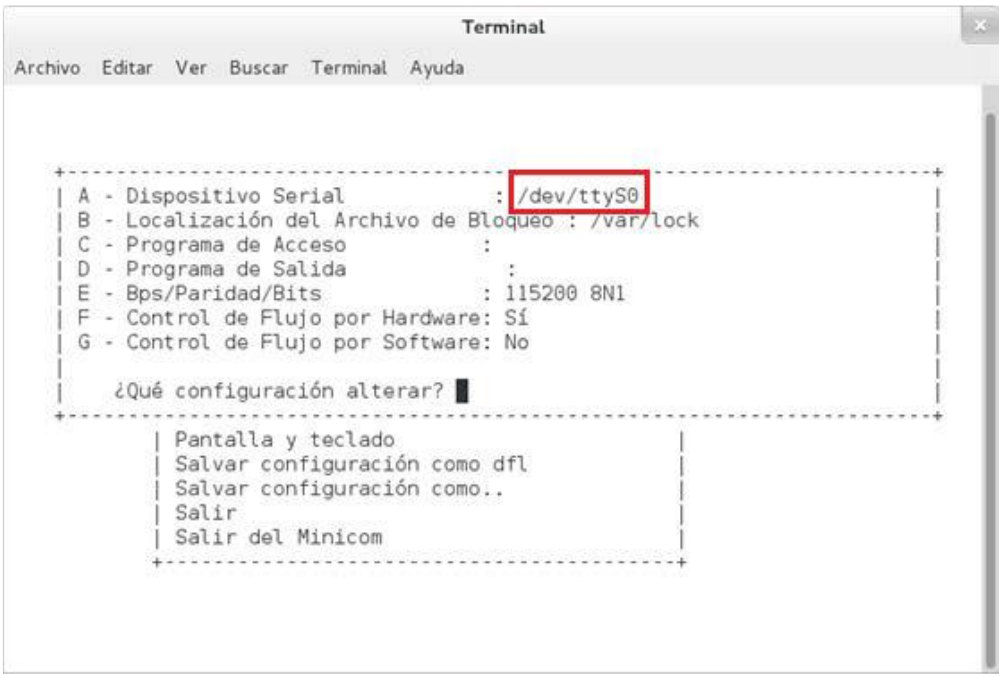


Figura B.0.2 Ruta del puerto serial

Continuemos con nuestra configuración pulsando la tecla E. En el menú que nos aparece, cambiamos los baudios por segundo (hay que dejarlo en 9600) la paridad (8 bits) y bits de parada (1):

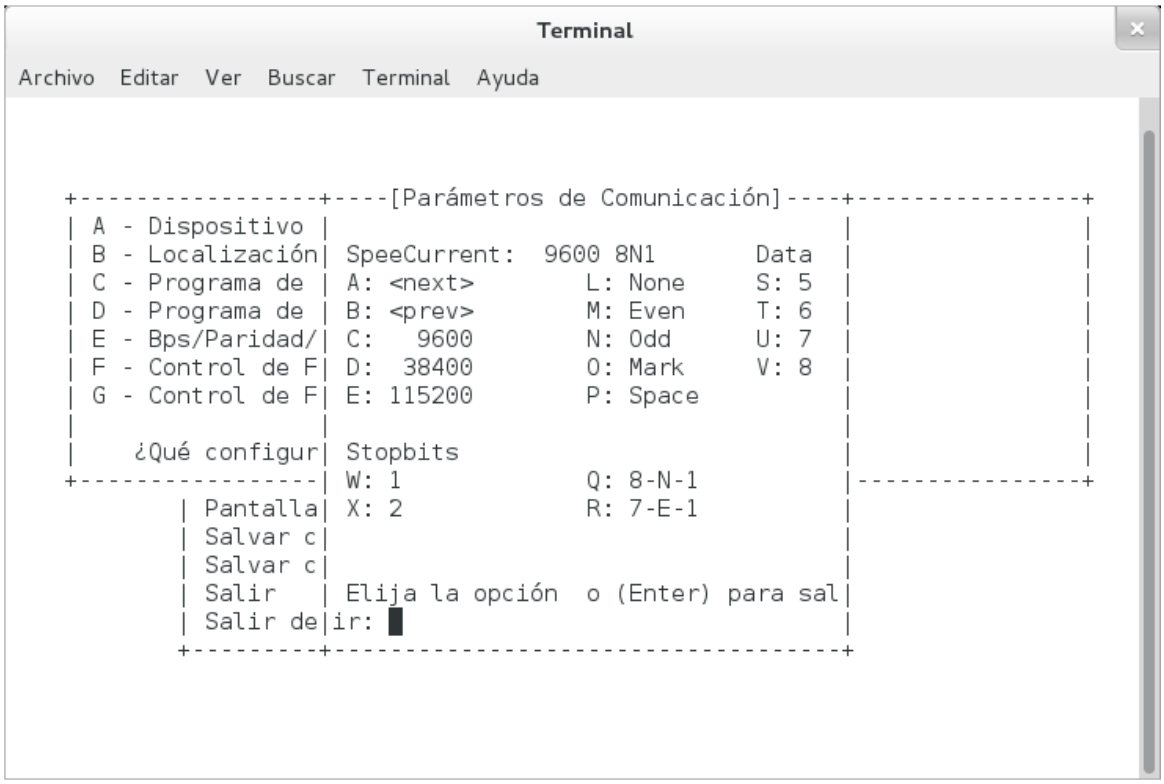


Figura B.0.3 Parámetros de comunicación.

Finalmente, guardamos nuestra configuración indicando que queremos que sea la de por defecto:

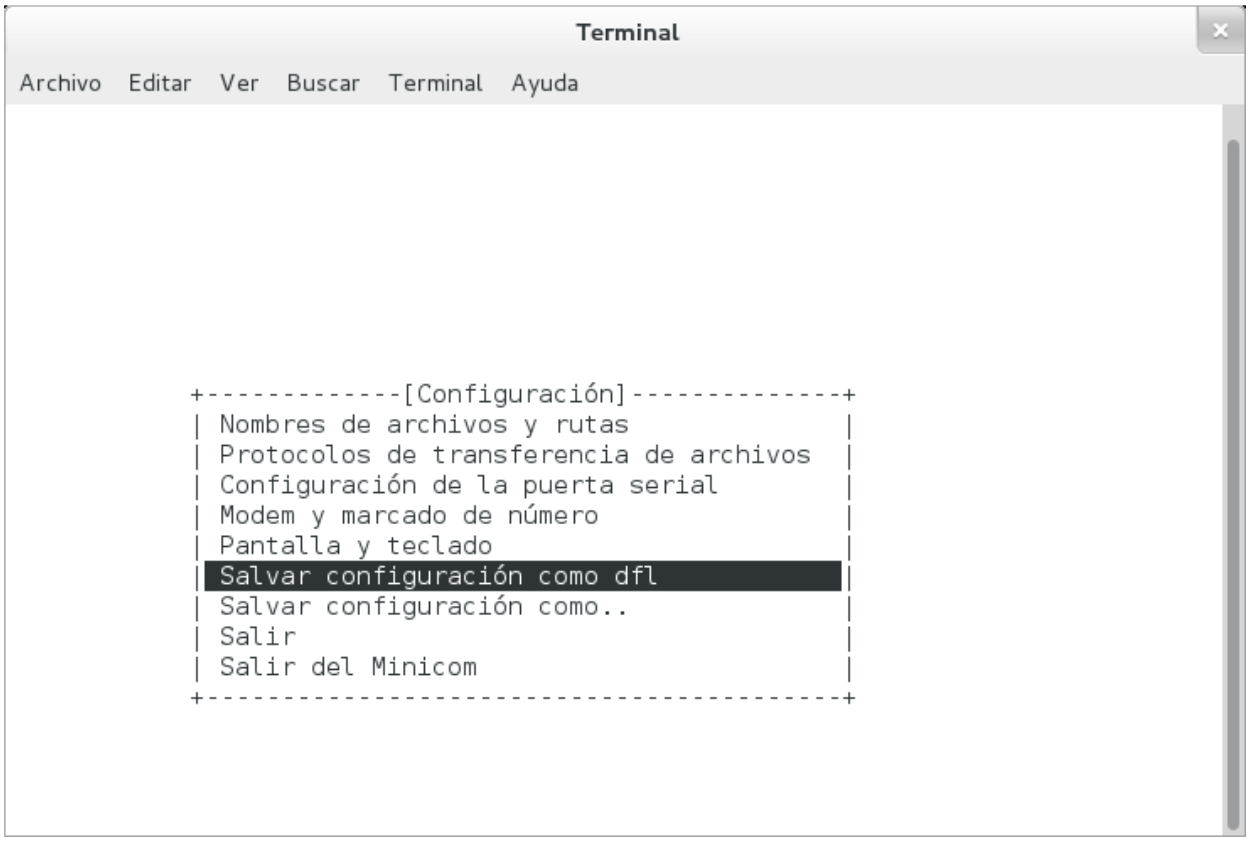


Figura B.0.4 Guardar configuración.

Una vez terminado estos pasos, podemos optar por pulsar el botón *Salir* para poder enviar los comandos al dispositivo al que estemos conectados, *o Salir del Minicom*, si no queremos utilizar aún este programa.

ANEXO C. INSTALACIÓN FILEZILLA

FileZilla es un cliente FTP multiplataforma de código abierto y software libre. Soporta los protocolos FTP, SFTP y FTP sobre SSL/TLS (FTPS).

La instalación de FileZilla es bastante simple. Solo hay que abrir un terminal como root y ejecutar:

apt-get install filezilla

Una vez se ha completado la instalación, procedemos a la configuración según nuestras necesidades.

Lo primero que vamos a hacer es limitar la velocidad tanto de bajada como de subida a 256 kB/s. Para ello pulse en Transferencia → Límites de Velocidad → Configurar. Y nos saldrá una ventana como ésta donde introduciremos la velocidad deseada y activaremos este límite de velocidad:

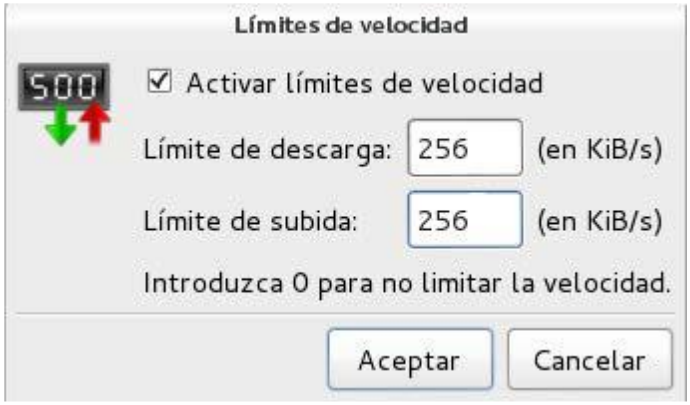


Figura C.0.1 Límites de velocidad de transferencia

Debido a la alta velocidad de nuestra red, la descarga del fichero que usamos en nuestra prueba era muy corta y no daba tiempo a ejecutar el script de traspaso. Había por tanto dos opciones, o usar un fichero más grande o cambiar la velocidad de transferencia. La primera da lugar a archivos de capturas de Wireshark demasiado grandes y difíciles de manejar, es por ello que se optó por la opción de limitar la velocidad.

Lo segundo que hacemos es pulsar en Edición → Opciones y en la ventana que se abre buscamos Transferencia → Tipos de archivos y añadimos la extensión del tipo de archivo que queremos transferir, en nuestro caso jpg.

El último paso es configurar al cliente para que se conecte al servidor deseado. Para llevarlo a cabo, pulsamos en Archivo → Gestor de sitios y se nos muestra la siguiente ventana donde escribiremos la dirección IPv6 de nuestro servidor entre "[ ]" (el número de puerto no hace falta indicarlo porque se usará el de por defecto), que utilizaremos FTP simple, y por último seleccionamos modo de acceso "Normal" y escribimos el nombre de usuario y contraseña de un usuario del servidor, preferentemente uno con privilegios de root, para evitar problemas de permisos con ficheros y carpetas.

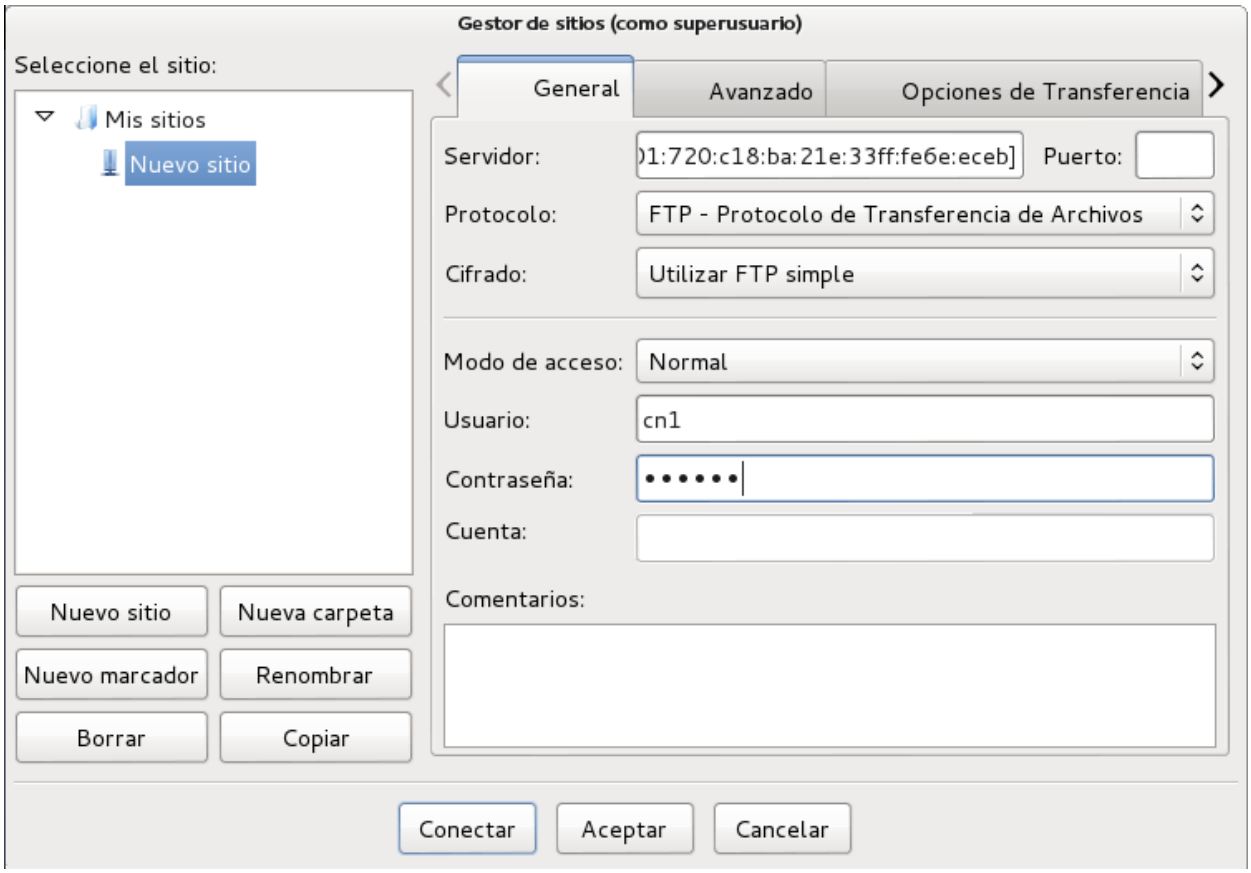


Figura C.0.2 Conectar a servidor

Ya solo falta pulsar en conectar y ya podremos subir ficheros al servidor FTP o descargárnoslo desde él.

ANEXO D. SCRIPT DE ARRANQUE UMIP

Para utilizar este script de arranque del software de movilidad de UMIP hay que ubicarlo en */etc/init.d*

```
#!/bin/sh

# Copyright © 2006,2007 USAGI/WIDE Proyect. All rights reserved.
# Adapted by Martin Andre andre@hongo.wide.ad.jp
# Furthere modified by Arnaud Ebalard arno@natisbad.org

### BEGIN INIT INFO
# Provides: mip6d
# Required-Start: $network $syslog
# Required-Stop: $network $syslog
# Should-Start: $local_fs
# Should-Stop: $local_fs
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Start/Stop MIPv6 Deamon (UMIP)
# Description: (empty)
### END INIT INFO
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
DESC="UMIP daemon"
NAME=mip6d
MIP6D=/usr/sbin/mip6d
MIP6D_CONF=/etc/mip6d.conf
MIP6D_DEBUG_LOG=/var/log/mip6d.log
PIDFILE=/var/run/mip6d.pid
FORCE_IPV6_FORWARDING="no"
RUN="no"
. /lib/lsb/init-functions
DIETIME=1
# Include defaults if available
if [ -f /etc/default/$NAME ] ; then
. /etc/default/$NAME
fi
if [ "x$RUN" != "xyes" ] ; then
```

```
log_failure_msg "$NAME disabled, please adjust the configuration to your needs"
log_failure_msg "and then set RUN to 'yes' in /etc/default/$NAME to enable it."
exit 1
fi
if [ ! -e $MIP6D_CONF ]; then
log_failure_msg "ERROR: $MIP6D_CONF does not exist."
log_failure_msg " See mip6d.conf(5) for configuration file syntax and"
log_failure_msg " sample configuration elements."
log_failure_msg " => $DESC will not be started"
exit 0
fi
if [ ! -x $MIP6D ]; then
log_failure_msg "ERROR: While trying to start $DESC, found its binary was"
log_failure_msg " missing ($MIP6D)."
log_failure_msg " => $DESC will not be started"
exit 0
fi
if [ ! -e /proc/sys/net/ipv6 ]; then
log_failure_msg "ERROR: In-kernel IPv6 is required for $DESC to work."
log_failure_msg " => $DESC will not be started."
exit 0
fi
set -e
MIP6D_OPTS="-c ${MIP6D_CONF}"
if [ x"$MIP6D_DEBUG_LOG" != x"" ]; then
MIP6D_OPTS="${MIP6D_OPTS} -l ${MIP6D_DEBUG_LOG}"
fi
post_war_cleaning()
{
# clean-up XFRM (BCE/BUL)
for t in sub main; do
ip xfrm policy flush ptype ${t} > /dev/null 2>&1 || true
done
for p in esp ah comp route2 hao; do
ip xfrm state flush proto ${p} > /dev/null 2>&1 || true
```

```
done
# clean-up tunnel device
tnls=`ifconfig -a | awk '/^ip6tnl/ { print $1 }'`
for tnl in $tnls; do
ip -6 tunnel del $tnl > /dev/null 2>&1 || true
done
# clean-up neighbor cache
devices=`ip link 2>&1 | grep '^[0-9]' | awk '{print $2}' | sed -e
's/:$//'`
for dev in $devices; do
ip neigh flush dev $dev > /dev/null 2>&1 || true
done
return 0
}
case "$1" in
start)
       echo -n "Starting MIP6D: "
       PID=x`pgrep -f $MIP6D || true`
       if [ "${PID}" != "x" ] ; then
             echo "failed (already started)."
             exit 0
        fi
if [ x"$FORCE_IPV6_FORWARDING" = x"yes" ]; then
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
fi
start-stop-daemon --start --quiet --exec ${MIP6D} -- ${MIP6D_OPTS} echo
"done."
;;
   stop)
         echo -n "Stopping MIP6D: "
         PID=x`pgrep -f $MIP6D || true`
         if [ "${PID}" = "x" ] ; then
             echo "done (none found)."
             exit 0
          fi
# Be nice ...
HOW=""
         pkill -f ${MIP6D}
         sleep $DIETIME
```

```
# Hum, you did not understand. Louder ...
         PID=x`pgrep -f $MIP6D || true`
              if [ "${PID}" != "x" ] ; then
                  pkill -TERM -f ${MIP6D} || true
                  sleep $DIETIME
                  PID=x`pgrep -f $MIP6D || true`
                  if [ "${PID}" != "x" ] ; then
post_war_cleaning
fi
HOW=" (TERMinated)"
fi
# Ok, go to hell ...
                 PID=x`pgrep -f $MIP6D || true`
                 if [ "${PID}" != "x" ] ; then
                     pkill -KILL -f ${MIP6D} || true
sleep $DIETIME
post_war_cleaning
HOW=" (KILLed)"
                  exit 0
                fi
                echo "done.${HOW}"
                ;;
reload)
                echo -n "Reloading MIP6D: "
pkill -HUP -f ${MIP6D} || true
echo "done."
;;
          restart|force-reload)
               $0 stop
               $0 start
                ;;
            status)
status=" NOT"
            PID=x`pgrep -f $MIP6D || true`
            if [ "${PID}" != "x" ] ; then
status=""
            fi
```

```
echo "${DESC} (${NAME}) is${status} running."
exit 0
;;
     *)
           N=/etc/init.d/$NAME
           echo "Usage: $N {start|stop|restart|force-reload|status}" >&2
           exit 1
           ;;
esac
exit 0
```

ANEXO E. CONFIGURACIÓN ROUTERS

En este anexo se recoge el conjunto de instrucciones necesarias para configurar los dos routers locales empleados en el escenario de pruebas. Empecemos por el Router 1:

```
Router>enable
Router#config terminal
%% Primero activamos el enrutamiento ipv6 y activamos el protocolo
%%ospf %%
Router(config)#ipv6 unicast-routing
Router(config)#ipv6 router ospf 1
Router(config-rtr)#router-id 12.12.12.12
Router(config-rtr)#exit
%% El puerto 8 es el único al que se le puede asignar directamente IP. Lo
%% usaremos para la conexión con el router CICA. Además usaremos el área
%% 0.0.2.98 ya que es la que recibimos en los mensajes OSPF del router
%% CICA %%
Router(config)#int fa8
Router(config-if)#ipv6 enable
Router(config-if)#ipv6 address autoconfig
Router(config-if)#ipv6 ospf 1 area 0.0.2.98
Router(config-if)#no shutdown
Router(config-if)#exit
%% Creamos las distintas Vlans. Vlan 20 para la red del nodo
%% corresponsal,la 30 para la red del home agent y del nodo móvil
%% inicialmente, y la 90 para la red donde migrará el nodo móvil.
Router(config)#vlan 20
Router(config-vlan)#name B1
Router(config-vlan)#exit
Router(config)#int vlan 20
Router(config-if)#no ip address
Router(config-if)#ipv6 enable
Router(config-if)#ipv6 address 2001:720:c18:b1::20/64
Router(config-if)#ipv6 address FE80::20 link-local
Router(config-if)#ipv6 ospf 1 area 0.0.2.98
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#vlan 30
```

```
Router(config-vlan)#name B2
Router(config-vlan)#exit
Router(config)#int vlan 30
Router(config-if)#no ip address
Router(config-if)#ipv6 enable
Router(config-if)#ipv6 address 2001:720:c18:b2::30/64
Router(config-if)#ipv6 address FE80::30 link-local
Router(config-if)#ipv6 ospf 1 area 0.0.2.98
Router(config-if)#ipv6 nd ra lifetime 5
Router(config-if)#ipv6 nd ra interval msec 500
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#vlan 90
Router(config-vlan)#name B9
Router(config-vlan)#exit
Router(config)#int vlan 90
Router(config-if)#no ip address
Router(config-if)#ipv6 enable
Router(config-if)#no shutdown
Router(config-if)#exit
%% El puerto 1 se usara como trunk para conectarlo al router 2. El resto
%% de puertos necesarios se asignarán a sus vlans correspondientes. El
%% spanning-tree portfast se activa en los puertos que se van a usar para
%% la movilidad del nodo móvil para agilizar el proceso de spanning-tree
%% saltándose los pasos de escucha y aprendizaje %%
Router(config)#int fa1
Router(config-if)#switchport mode trunk
Router(config-if)#no sh
Router(config-if)#exit
Router(config)#int fa3
Router(config-if)#switchport mode access
Router(config-if)#switchport access vlan 20
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#int fa6
Router(config-if)#switchport mode access
Router(config-if)#switchport access vlan 30
Router(config-if)#no shutdown
```

```
Router(config-if)#exit
Router(config)#int fa7
Router(config-if)#switchport mode access
Router(config-if)#switchport access vlan 30
Router(config-if)#spanning-tree portfast
```

A continuación, el Router 2:

```
Router>enable
Router#config terminal
Router(config)#ipv6 unicast-routing
Router(config)#ipv6 router ospf 1
Router(config-rtr)#router-id 2.2.2.2
Router(config-rtr)#exit
Router(config)#int fa8
Router(config-if)#ipv6 enable
Router(config-if)#ipv6 address autoconfig
Router(config-if)#ipv6 ospf 1 area 0.0.2.98
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#vlan 90
Router(config-vlan)#name B9
Router(config-vlan)#exit
Router(config)#int vlan 90
Router(config-if)#no ip address
Router(config-if)#ipv6 enable
Router(config-if)#ipv6 address 2001:720:c18:b9::90/64
Router(config-if)#ipv6 address FE80::90 link-local
Router(config-if)#ipv6 nd ra interval msec 1000
Router(config-if)#ipv6 ospf 1 area 0.0.2.98
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#int fa1
Router(config-if)#switchport mode trunk
Router(config-if)#no sh
Router(config-if)#exit
```

ANEXO F. SCRIPT DE TRASPASO

En este anexo recogemos el conjunto de instrucciones que se le pasan en bloque al Router 1 para provocar el cambio de VLAN del MN de forma que se simula el traspaso de una red a otra del dispositivo móvil.

Traspaso de la red local (VLAN 30) a la red foránea (VLAN 90):

```
Router>enable
Router#config terminal
Router(config)#int fa7
Router(config-if)# sw access vlan 90
Router(config-if)# exit
Router(config)# exit
```

Traspaso de la red foránea (VLAN 90) a la red local (VLAN 30):

```
Router>enable
Router#config terminal
Router(config)#int fa7
Router(config-if)# sw access vlan 30
Router(config-if)# exit
Router(config)# exit
```

Es posible que con estas instrucciones el MN no detecte bien el cambio de red. Para evitar esto, podemos optar por incluir instrucciones de deshabilitar la interfaz y volver a habilitarla para facilitar así la detección del cambio de red. Las instrucciones quedarían así:

```
Router>enable
Router#config terminal
Router(config)#int fa7
Router(config-if)# shutdown
Router(config-if)# exit
Router(config)# exit
Router(config)#int fa7
Router(config-if)# sw access vlan 90
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# exit
```

Y al contrario para el trapaso de vuelta a la red local:

```
Router>enable
Router#config terminal
Router(config)#int fa7
Router(config-if)# shutdown
Router(config-if)# exit
Router(config)# exit
Router(config)#int fa7
Router(config-if)# sw access vlan 30
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# exit
```

ANEXO G. INSTALACIÓN IPERF Y JPERF

Iperf es una herramienta que se utiliza para hacer pruebas en redes informáticas. El funcionamiento habitual es crear flujos de datos TCP y UDP y medir el rendimiento de la red.

Hay dos formas de instalar Iperf. La primera de ellas es la más sencilla. Consiste en abrir un terminal como root y ejecutar: apt-get install iperf

La segunda de ellas, radica en descargar el código fuente desde el repositorio oficial:

https://code.google.com/p/walami/downloads/detail?name=iperf-2.0.5.tar.gz&can=2&q=

Descomprimimos el fichero descargado con la orden tar –xvzf. Una vez descomprimido, entramos en la carpeta y ejecutamos las siguientes instrucciones:

```
./configure
```

```
make
```

```
make install
```

Una vez completada la instalación ya podremos usar Iperf. Pero debido a que su utilización es a través de línea de comandos, se recomienda emplear Jperf que es una interfaz gráfica desarrollada en Java que nos facilita el manejo de Iperf, puesto que solo tenemos que ir rellenando una serie de campos y él se encarga de traducirlos a un orden Iperf.

Para instalar Jperf, descargamos el código desde: *https://code.google.com/p/xjperf/*, y descomprimimos con unzip. Para empezar a utilizarlo, basta con entrar en la carpeta descomprimida y ejecutar jperf.sh. Aparecerá una ventana donde vamos seleccionando si queremos usar TCP o UDP, si usamos IPv6, etc. además de ver una gráfica de ancho de banda, si ejecutamos el programa como cliente, y de ancho de banda más jitter, si lo hacemos como servidor. Por último, abajo a la derecha aparece una ventana que es la salida por pantalla que mostraría Iperf por línea de comandos.

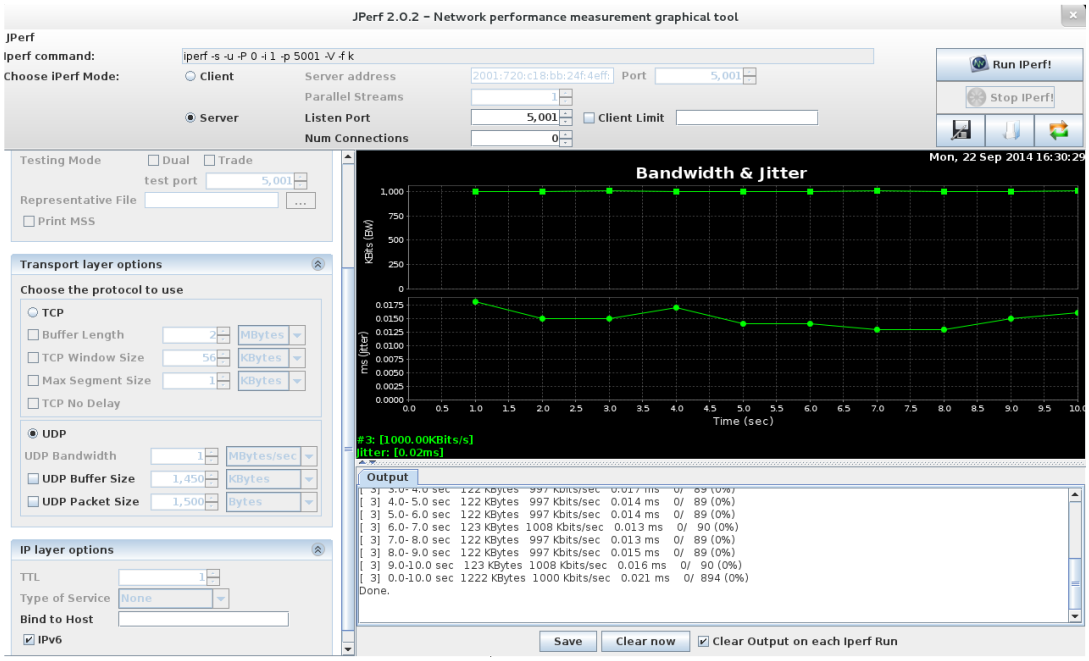


Figura F.0.1 Jperf como servidor.

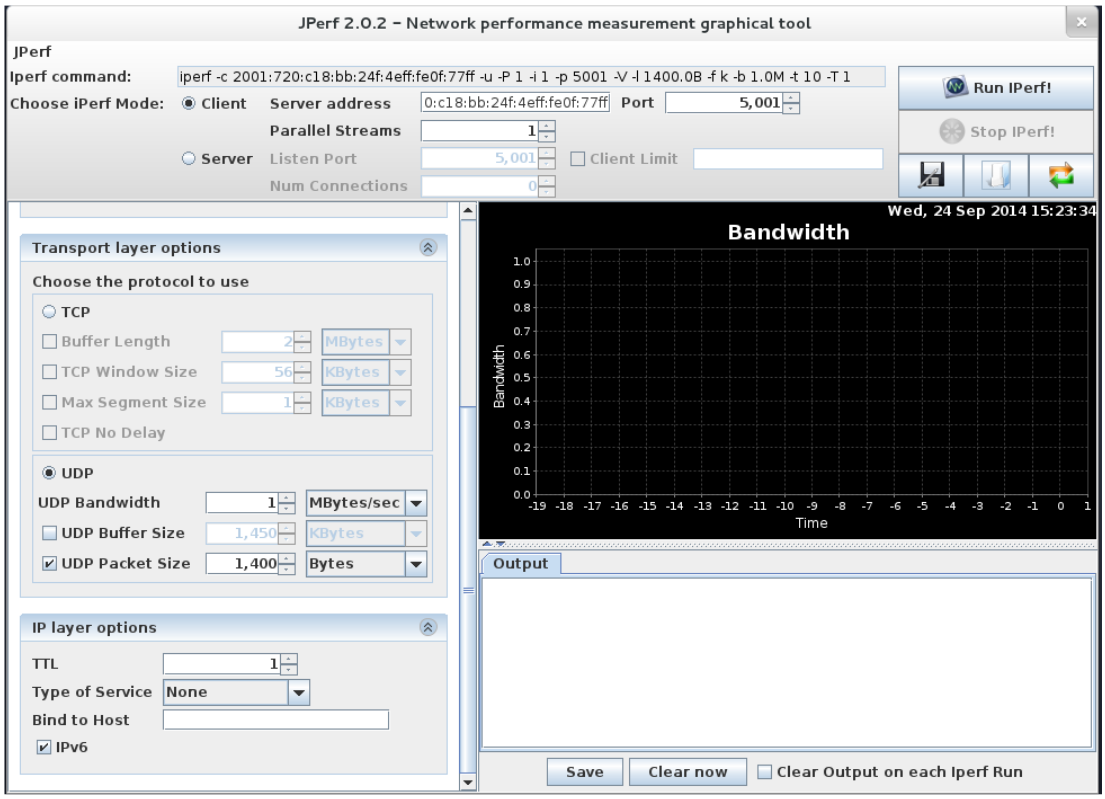


Figura F.0.2 Jperf como cliente

ANEXO H. GRÁFICAS CON WIRESHARK

En este anexo mostraremos los pasos para obtener las gráficas de Wireshark utilizadas en este proyecto.

Empezaremos por la gráfica Stevens. Para ello necesitamos una captura de tráfico TCP en la que, tras seleccionar un segmento en el sentido de la comunicación que se desea analizar, pincharemos en "Statistics – TCP Stream Graphs – Time Sequence (Stevens)".

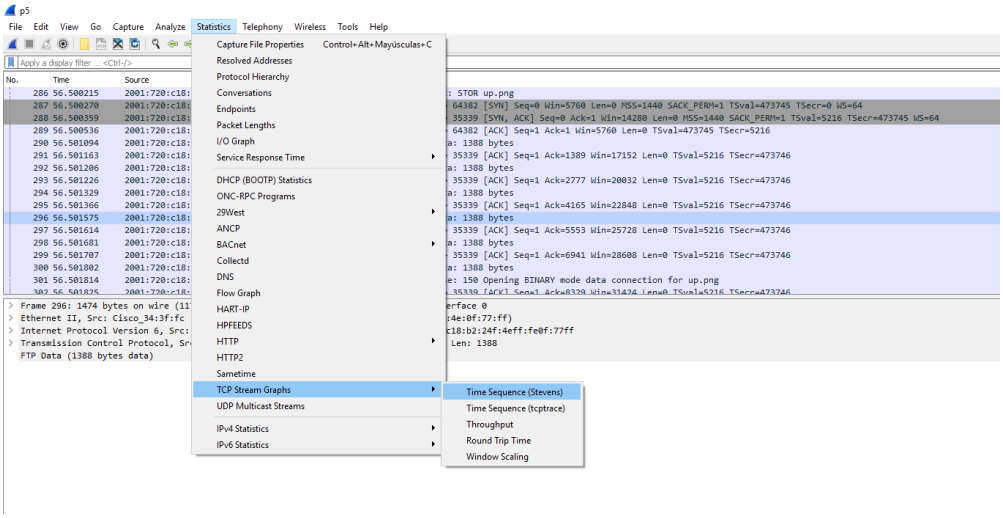


Figura H.0.1 Statistics – TCP Stream Graphs – Gráfica Stevens

Tras esto, se mostrará la gráfica por pantalla. Dicha gráfica tendrá un aspecto similar a este:

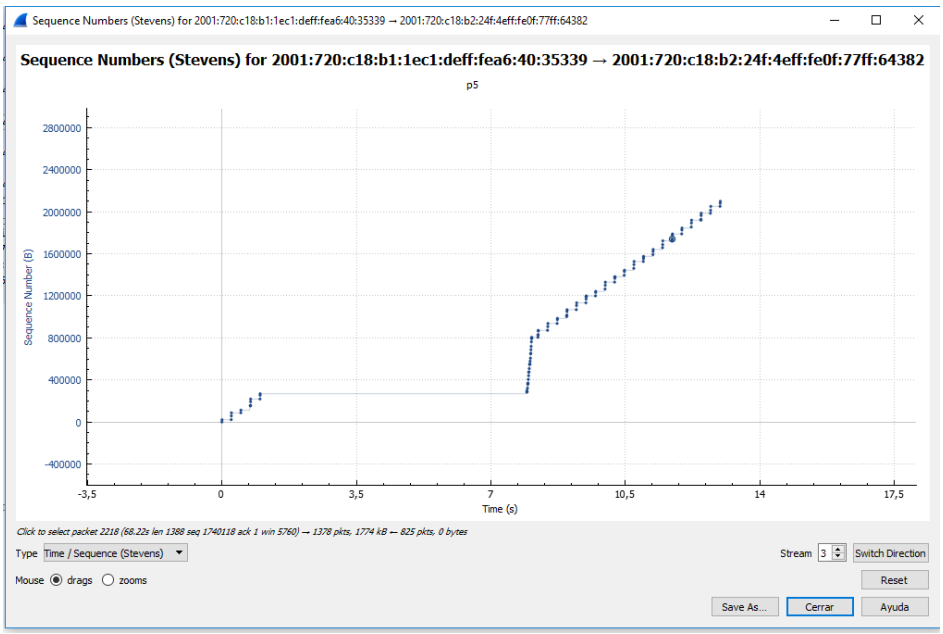


Figura H.0.2 Gráfica Stevens

Si hacemos click derecho sobre esta gráfica se desplegará un menú donde podremos configurar diferentes parámetros:

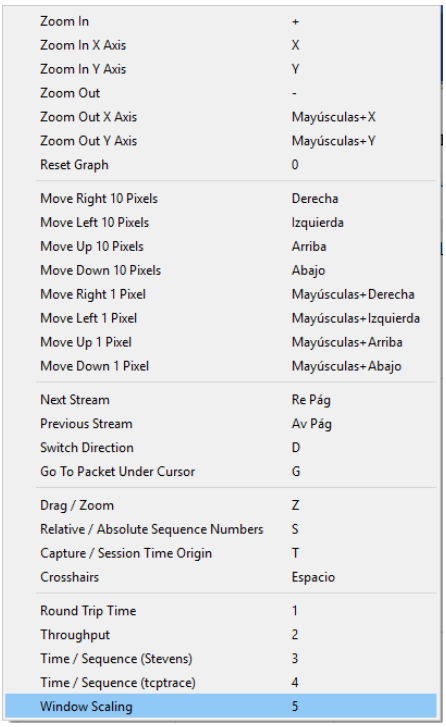


Figura H.0.3 Barra de opciones en la gráfica Stevens

Además, en la propia gráfica podremos ajustar ciertos parámetros. Por ejemplo moviendo la rueda del raton podremos hacer zoom, si hacemos click izquierdo podremos movernos por la gráfica, además de guardar la gráfica para poder consultarla en otro momento o resetear todos los parámetros.

Otra gráfica utilizada durante el proyecto es la RTT (Round Trip Time). El proceso para mostrarla es similar a la gráfica Stevens, sólo que seleccionando la opción “Round Trip Time” dentro de las gráficas TCP.

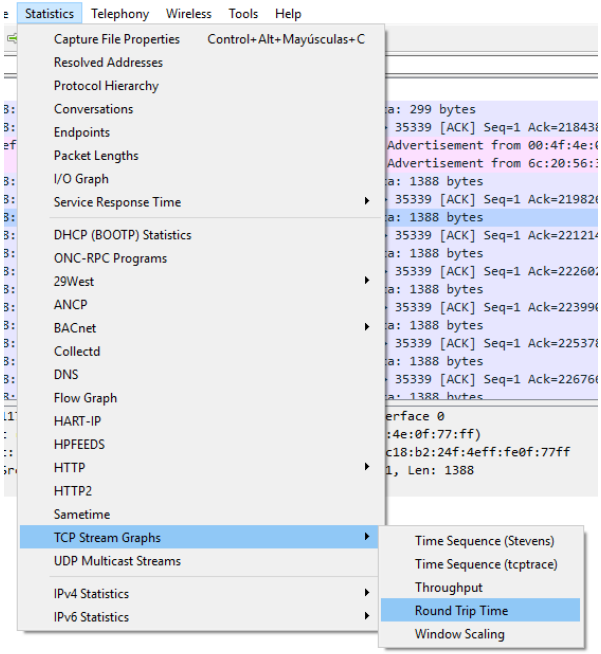


Figura H.0.4 Statistics – TCP Stream Graphs – Round Trip Time

De esta forma se mostrará una gráfica con este aspecto:

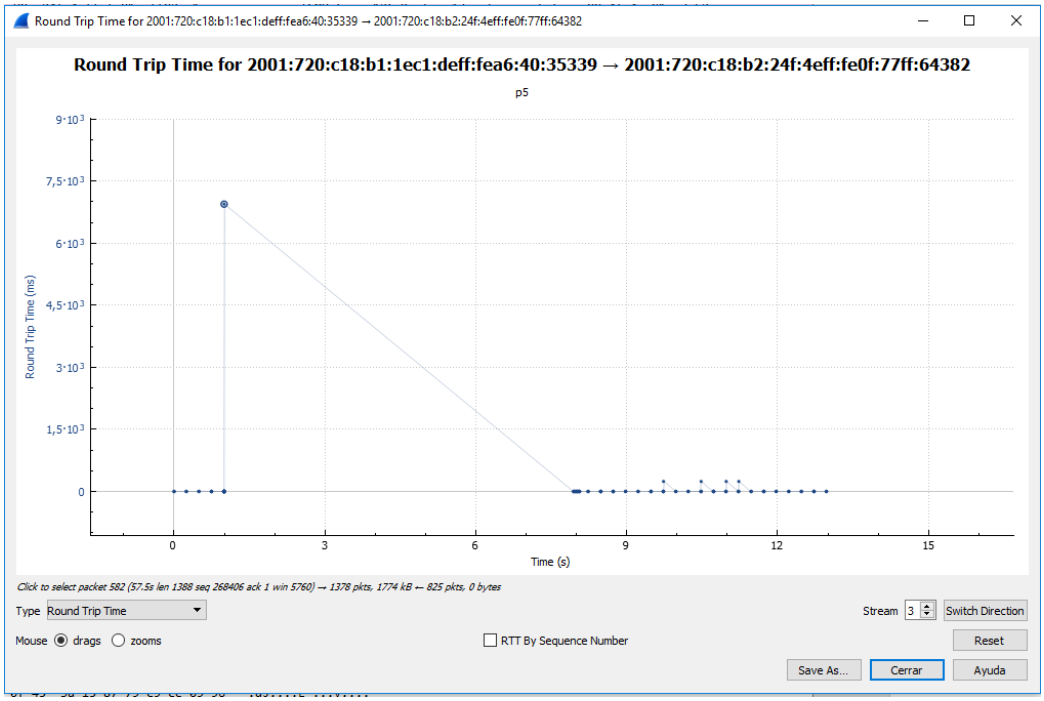


Figura H.0.5 Gráfica Round Trip Time (RTT)

Al igual que con la gráfica Stevens, tendremos parámetros configurables en la propia ventana de la gráfica y, haciendo click derecho, se mostrará un menú desplegable con más opciones.

Por último, otra gráfica utilizada ha sido la "Input/Output". Para mostrarla debemos seleccionar "Statistics – I/O Graph" en el menú de Wireshark.

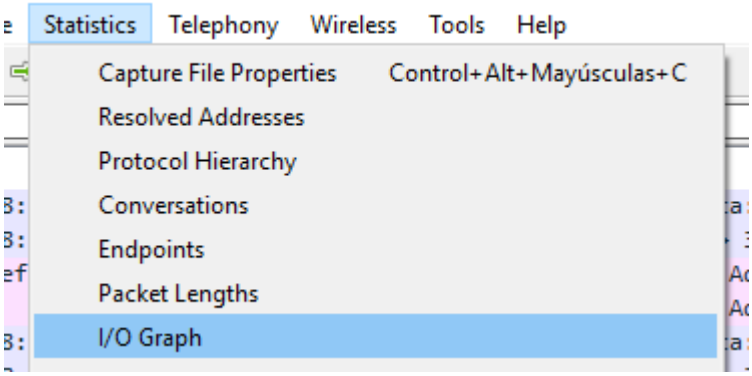


Figura H.0.6 Statistics – I/O Graph

Tras seleccionar esta opción se nos mostrará una nueva ventana con una gráfica con este aspecto:

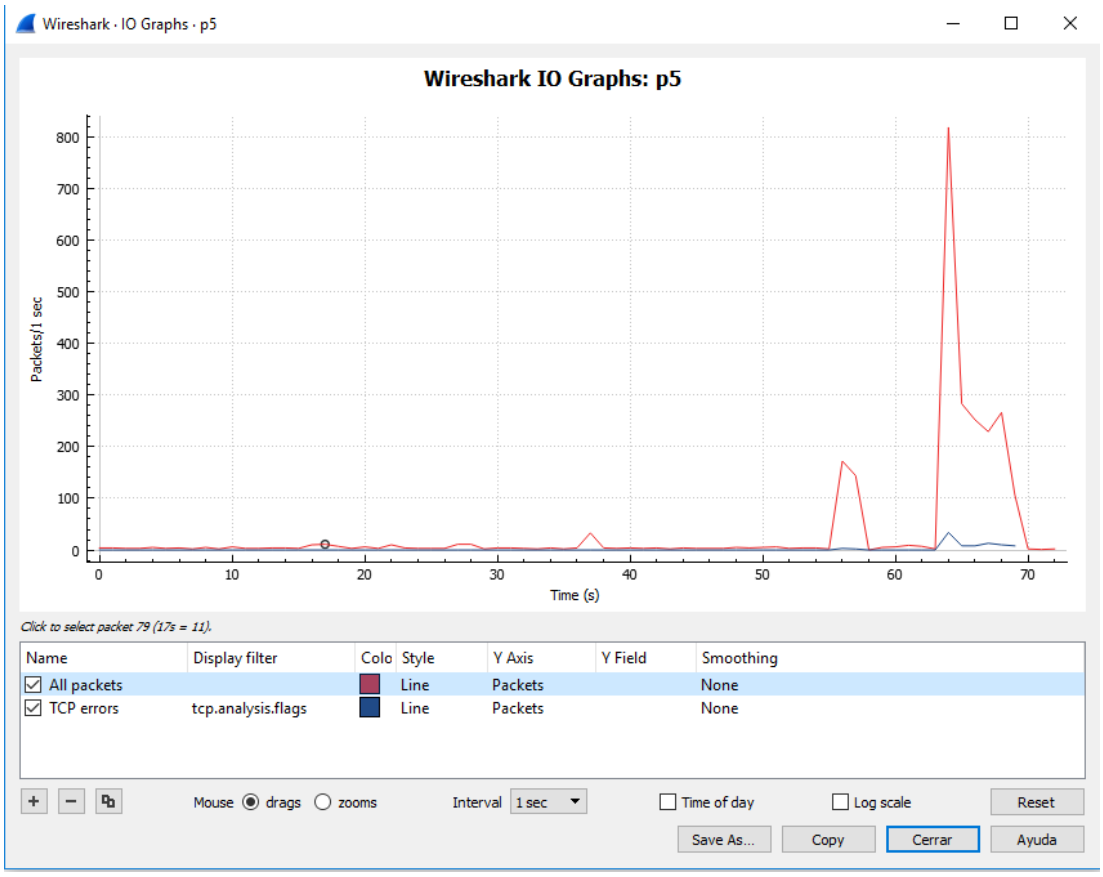


Figura H.7 Gráfica Input/Output

En la parte inferior de la ventana podremos configurar distintos filtros para que la gráfica nos muestre los paquetes deseamos. Por ejemplo, puede filtrarse para que solo muestre paquetes TCP, o con una IP de origen o destino determinada.

Además, puede cambiarse el tipo de gráfica que se muestra (Lineas, barras, etc.) o el color para poder comparar los distintos tipos de tráfico mostrados.

Por último, al igual que en las gráficas anteriores, haciendo click derecho se mostrará un menú desplegable con más opciones de configuración.

REFERENCIAS

I. Introducción y Estado del Arte

[1] M. Miralles, Desarrollo de Escenario para el Análisis de la Movilidad en IPv6, vol. 2, 2014.

[2] UMIP, «UMIP Project,» [En línea]. Available: http://www.umip.org/.

[3] J. d. Andalucía, «CICA,» [En línea]. Available: https://www.cica.es/.

[4] R. J. M. Tejedor, «IPv6,» [En línea]. Available: http://www.ramonmillan.com/tutoriales/ipv6_parte1.php.

[5] M. &. Dykes, «rfc 3069 - VLAN,» [En línea]. Available: https://tools.ietf.org/html/rfc3069.

[6] G. Combs, «Wireshark,» [En línea]. Available: https://www.wireshark.org/.

[7] P. S. a. K. Egevang, «rfc 3022 - NAT,» [En línea]. Available: https://tools.ietf.org/html/rfc3022.

[8] R. Droms, «RFC 2131 - DHCP,» [En línea]. Available: https://www.ietf.org/rfc/rfc2131.txt.

[9] IANA, «IANA,» [En línea]. Available: https://www.iana.org/.

[10] Wikipedia, «RIR,» [En línea]. Available: https://en.wikipedia.org/wiki/Regional_Internet_registry.

[11] Wikipedia, «ISP,» [En línea]. Available: https://en.wikipedia.org/wiki/Internet_service_provider.

[12] «MIPv6,» [En línea]. Available: https://www.ipv6.com/mobile/what-is-mobile-ipv6/.

[13] K. Shima, «KAME project,» [En línea]. Available: http://www.kame.net/newsletter/20031007/.

[14] J. Murai, «USAGI Project,» [En línea]. Available: http://www.linux-ipv6.org/.

[15] S. Gundavelli, «rfc 5213 - PMIPv6,» [En línea]. Available: https://tools.ietf.org/html/rfc5213.

[16] R. Wakikawa, «rfc 5648 - MCoA,» [En línea]. Available: https://tools.ietf.org/html/rfc5648.

[17] H. Soliman, «rfc 5555 - DSMIPv6,» [En línea]. Available: https://tools.ietf.org/html/rfc5555.

II. Internet Protocol Version 6 (IPv6)

[18] V. Fuller, «rfc 4632 - CIDR,» [En línea]. Available: https://tools.ietf.org/html/rfc4632.

[19] S. Deering, «rfc 2460 - IPv6,» [En línea]. Available: https://tools.ietf.org/html/rfc2460.

[20] 6SOS, «El protocolo IPv6 v4.0,» [En línea]. Available: http://www.6sos.org/documentos/6SOS_El_Protocolo_IPv6_v4_0.pdf.

[21] S. Frankel, «rfc 6071 - IPsec,» [En línea]. Available: https://tools.ietf.org/html/rfc6071.

[22] S. Thomson, «rfc 4862 - SLAAC,» [En línea]. Available: https://tools.ietf.org/html/rfc4862.

[23] -, «IPv6,» [En línea]. Available: https://www.ipv6.com/general/ipv6-header-deconstructed/.

[24] R. Hinden, «rfc 4291 - IPv6 Adressing Architecture,» [En línea]. Available: https://tools.ietf.org/html/rfc4291.

[25] A. Conta, «rfc 4443 - ICMPv6,» [En línea]. Available: https://tools.ietf.org/html/rfc4443.

[26] T. Narten, «rfc 4861 - Neighbor Discovery for IPv6,» [En línea]. Available: https://tools.ietf.org/html/rfc4861.

[27] R. Vida, «rfc 3810 - Multicast Listener Discovery v2 for IPv6,» [En línea]. Available: https://tools.ietf.org/html/rfc3810.

[28] D. C. Plummer, «rfc 826 - ARP,» [En línea]. Available: https://tools.ietf.org/html/rfc826.

[29] S. Kent, «rfc 4302 - Autentication Header,» [En línea]. Available: https://www.ietf.org/rfc/rfc4302.txt.

[30] S. Kent, «rfc 4303 - Encapsulating Security Payload,» [En línea]. Available: https://tools.ietf.org/html/rfc4303.

[31] D. Harkins, «rfc 2409 - Internet Key Exchange Protocol,» [En línea]. Available: https://tools.ietf.org/html/rfc2409.

[32] C. Kaufman, «rfc 7296 - IKE v2,» [En línea]. Available: https://tools.ietf.org/html/rfc7296.

[33] J. Moy, «rfc 2328 - OSPF v2,» [En línea]. Available: https://tools.ietf.org/html/rfc2328.

[34] R. Coltun, «rfc 5340 - OSPF for IPv6,» [En línea]. Available: https://tools.ietf.org/html/rfc5340.

III. Movilidad IPv6

[35] C. Perkins, «rfc 5944 - IP Mobility Support for IPv4,» [En línea]. Available: https://tools.ietf.org/html/rfc5944.

[36] C. Perkins, «rfc 6275 - Mobility in IPv6,» [En línea]. Available: https://tools.ietf.org/html/rfc6275.

[37] D. Johnson, «rfc 3775 - Mobility Support in IPv6,» [En línea]. Available: https://tools.ietf.org/html/rfc3775.

[38] T. E. Community, «Proyecto ENABLE,» [En línea]. Available: http://www.ist-enable.eu/.

IV. Escenario Práctico MIPv6

[39] Cisco, «Cisco 892 Router,» [En línea]. Available: https://www.cisco.com/c/en/us/support/routers/892-integrated-services-router-isr/model.html#~tab-documents.

[40] U. community, «Minicom, Ubuntu Installation,» [En línea]. Available: https://help.ubuntu.com/community/Minicom.

[41] C. S. Forums, «Cisco website,» [En línea]. Available: https://supportforums.cisco.com/t5/network-infrastructure-documents/resetting-router-to-factory-default-removing-the-startup/ta-p/3131114.

[42] Cisco, «Cisco Website - Basic Router Configuration,» [En línea]. Available: https://www.cisco.com/c/en/us/td/docs/routers/access/800M/software/800MSCG/routconf.html.

[43] I. Cisco Systems, «Cisco 800 Series Integrated Services Routers Software Configuration,» [En línea]. Available: https://cdn.cnetcontent.com/00/3b/003b3477-d8b9-4826-95b9-9b61d49aa562.pdf.

[44] C. S. Forums, «Cisco Website - Spanning tree,» [En línea]. Available: https://supportforums.cisco.com/t5/routing-y-switching-blogs/caracter%C3%ADsticas-avanzadas-de-spanningtree-portfast-bpdu-guard-y/ba-p/3104851.

[45] Desconocido, «Wikipedia - Spanning Tree Protocol,» [En línea]. Available: https://en.wikipedia.org/wiki/Spanning_Tree_Protocol.

[46] UMIP, «Instalación de UMIP,» [En línea]. Available: http://www.umip.org/docs/umip-install.html.

[47] UMIP, «UMIP configuration,» [En línea]. Available: http://www.umip.org/docs/umip-mip6.html.

[48] Ubuntu, «Ubuntu manpages - radvd,» [En línea]. Available: http://manpages.ubuntu.com/manpages/xenial/en/man5/radvd.conf.5.html.

V. Pruebas Realizadas

[49] Desconocido, «Seguridad y redes wordpress. Graficas Wireshark I,» [En línea]. Available: https://seguridadyredes.wordpress.com/2008/11/25/analisis-de-red-con-wireshark-interpretando-las-graficas-i-parte/.

[50] Desconocido, «Seguridad y Redes WordPress. Gráficas con Wireshark II,» [En línea]. Available: https://seguridadyredes.wordpress.com/2008/12/01/graficas-con-wireshark-ii-parte-tcptrace/.

ÍNDICE DE CONCEPTOS

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 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.
- CIDR: Notación mediante la cual se expresan los prefijos de red. Tiene la forma del prefijo (en hexadecimal) / longitud del prefijo (en decimal).
- Correspondent Node: Un nodo que se comunica con un nodo móvil que se encuentra fuera de su red habitual.
- 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.
- Enlace: Segmento o segmentos de una red de área local limitados por encaminadores.
- Home Agent: Nodo ubicado en la red local del nodo móvil y que ejerce de nexo entre este y su red local, permitiendo que dicho nodo móvil siga recibiendo los paquetes dirigidos a su HoA.
- Host: Véase nodo.
- 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: nodo que residiendo en la red de su "casa", tiene la capacidad de migrar a otras redes.
- 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.
- 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: 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.

- 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.
- Vecinos: Nodos conectados en el mismo enlace.