Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicaciones Móviles, Inalámbricas y Distribuidas (ICARO)

Desarrollo de una Solución para la Movilidad en IPv6 con Soporte para Dispositivos Móviles Android

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
por el (los) Bachiller(es):

Br. Christian Graffe
C.I.: 17.125.457
E-mail: cags12@gmail.com

Br. Rafael Emmi
C.I.: 18.941.450
E-mail: rafaemmi@hotmail.com

para optar al título de Licenciado en Computación

Tutora: Profa. María E. Villapol

Caracas, Mayo 2012

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Laboratorio de Comunicaciones Móviles, Inalámbricas y Distribuidas (ICARO)

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación para examinar el Trabajo Especial de Grado, presentado por los Bachilleres Christian Graffe C.I.: 17.125.457 y Rafael Emmi C.I.: 18.941.450, con el título **"Desarrollo de una solución para la movilidad en IPv6 con soporte para dispositivos móviles Android",** a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día 25 de mayo del 2012, a las 9:00 am, para que sus autores lo defendieran en forma pública, en Laboratorio de Internet 2 – Galpón 10, lo cual estos realizaron mediante una exposición oral de su contenido, y luego respondieron satisfactoriamente a las preguntas que les fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente acta, en Caracas el 25 de mayo del 2012, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Tutor María E. Villapol.

Prof. María E. Villapol
(Tutora)

Prof. Rafael Angulo
(Jurado Principal)

Prof. Ana Morales
(Jurado Principal)

RESUMEN

Título:
Desarrollo de una Solución para la Movilidad en IPv6 con Soporte para Dispositivos Móviles Android.

Autor(es):
Christian Graffe
Rafael Emmi

Tutora:
Prof. María E. Villapol

En la actualidad existe un auge en la utilización de dispositivos móviles, en específico el uso de teléfonos inteligentes y tabletas. Cada vez son más las personas que usan estos dispositivos y están en constante comunicación a través del Internet en todo lugar a donde vayan. Dada que una de sus características fundamentales es la movilidad, suena conveniente la posibilidad que ellos mantengan una dirección IP constante donde quiera que vayan y mantengan activas las conexiones en curso de manera transparente. Actualmente, los sistemas operativos que corren en teléfonos móviles inteligentes, desafortunadamente no cuentan con ninguna implementación de algún protocolo encargado de mantener la macro movilidad, es decir, el reenvío de paquetes entre subredes. En este trabajo se ha desarrollado una solución de macro movilidad para teléfonos inteligentes que usan el sistema operativo Android.

IPv6 móvil es la versión posterior a IPv4 móvil, que incluye mejoras en el protocolo inicial como lo son: menor número de agentes que intervienen en el proceso de comunicación, incorporación de la optimización de rutas, uso del encabezado del paquete IPv6 para mejorar los mecanismos de macro movilidad, desvinculación de mecanismos presentes en otras capas de la pila de protocolos, etc. Así, el objetivo de este trabajo es desarrollar una solución que proporcione soporte del protocolo IPv6 Móvil en dispositivos móviles, teléfonos inteligentes y tabletas, corriendo el sistema operativo móvil Google Android, a la vez de proporcionar la facilidad para que el usuario de estos dispositivos pueda realizar la configuración y gestión del protocolo. Para lograr esto, se realizó una modificación del *kernel* del Sistema Operativo Android para dar soporte a la pila de protocolos de red, las nuevas cabeceras y definición de mensajes que el protocolo IPv6 móvil introduce para su funcionamiento. Una vez realizado esto, se implementó la lógica del protocolo, la cual corre a nivel de usuario en forma de un demonio del sistema. Por último, se desarrolló una aplicación nativa de Android, que permite al usuario realizar la configuración y el manejo del demonio del protocolo, de tal forma que facilite y estimule su uso, a su vez de proveer funcionalidades propias del sistema operativo que permitan hacer uso del demonio del protocolo en un ambiente de producción y no solo de pruebas. La solución planteada fue probada en diversos escenarios de macro movilidad para validar su correcto funcionamiento. Adicionalmente, se realizaron pruebas de rendimiento para conocer como los mecanismos del protocolo IPv6 móvil afectan el desempeño de la red. Dichas

pruebas arrojaron resultados satisfactorios al demostrar que los mecanismos del protocolo IPv6 móvil no agregan una sobrecarga significativa a la comunicación lo cual se ve reflejado en el rendimiento de la red el cual resulto casi igual al rendimiento de la red sin el uso del protocolo IPv6 móvil.

Palabras Claves: IPv6, movilidad, dispositivos móviles, Android.

Tabla de contenido

Índice de figuras

Índice de tablas

1. Introducción

Las necesidades de los usuarios de Internet en sus inicios eran muy diferentes a las de los usuarios actuales, muchos protocolos se han quedado cortos en relación con las exigencias que se presentan como, por ejemplo, la cantidad de direcciones IP versión 4. Estas no han sido suficientes para todos los usuarios de Internet quienes han tenido que utilizar mecanismos alternos como la traducción de direcciones de red (NAT - *Network Address Translation*) la cual es solo un alivio y no una solución. Para solventar este y otros problemas surgió el Protocolo de Internet versión 6 (IPv6 - *Internet Protocol version 6*), que ofrece un mayor espacio de direcciones.

Por otro lado, en Internet, la ubicación de un nodo está fuertemente limitada por el esquema de enrutamiento. Una dirección IP que pertenece a un prefijo de red asignado a un proveedor de servicios de Internet venezolano no se puede utilizar para recibir paquetes en Japón. En consecuencia, la arquitectura actual de Internet hace que sea imposible mantener la misma dirección IP cuando se mueven los nodos. Este problema está relacionado con la doble función de las direcciones IP. En primer lugar, establece implícitamente la posición de un nodo en el mundo, este papel es llamado localizador. La segunda función se llama "identificador", que identifica de manera única el nodo en toda la topología de Internet.

Los protocolos de movilidad de IP, tales como MIPv4 y MIPv6 proporcionan una solución para separar estas dos funciones. MIPv6 (IPv6 Móvil) permite a los nodos móviles moverse dentro de Internet mientras que se mantienen las sesiones en curso y la accesibilidad usando una dirección IPv6 permanente. En estos esquemas, un nodo móvil mantiene una dirección local la cual es permanente y una dirección adquirida en la red visitada, la cual es llamada dirección de cuidado.

Otro aspecto importante a considerar es, el ritmo sin precedentes de crecimiento en el uso de teléfonos inteligentes los cuales se prevé que pasarán pronto a las laptops como plataforma móvil/portátil de elección. Esto hace pensar en la conveniencia de hacer uso de los protocolos de movilidad en dichos dispositivos los cuales explotaran al 100% su condición de dispositivos móviles. El desarrollo de una implementación de un protocolo de movilidad, traería grandes beneficios al usuario final al poder mantener las conexiones existentes activas mientras se mueven geográficamente y topológicamente.

1.1. Planteamiento del problema

La arquitectura actual de Internet hace que sea imposible mantener la misma dirección IP cuando se mueven los dispositivos dado que la función de ellas implica, en primer lugar, la posición de un nodo en el mundo y en segundo lugar, la identificación de manera única del nodo en toda la topología de Internet. En algunos casos puede ser extremadamente favorable mantener la misma dirección IP sin importar la localización geográfica donde se encuentre actualmente o incluso durante el constante movimiento por lo tanto, las conexiones previamente establecidas podrían seguir comunicándose sin problema logrando que el

movimiento libre por Internet sea transparente para el usuario. Esta problemática la resuelve IP Móvil e IPv6 Móvil. Sin embargo, encontramos la falta de implementaciones de estos protocolos que sean soportados en sistemas operativos para dispositivos móviles (Teléfonos inteligentes y tabletas) que son los que pueden realmente beneficiarse de las funcionalidades de movilidad IP.

1.2. Objetivo general

Desarrollar una solución para la movilidad en IPv6 basada en los estándares descritos en el RFC 3775 [1] Mobile IPv6 con soporte en dispositivos móviles basados en el sistema operativo Android.

1.3. Objetivos específicos

Los objetivos específicos de este trabajo son:

- Diseñar la solución de movilidad para dispositivos móviles corriendo el sistema operativo Android.
- Implementar la solución de movilidad.
- Probar la funcionalidad y rendimiento de la solución de movilidad.
- Analizar los resultados obtenidos en las pruebas.

1.4. Justificación

En la actualidad existe un auge en la utilización de dispositivos móviles, en específico el uso de teléfonos inteligentes y tabletas. Cada vez son más las personas que usan estos dispositivos y están en constante comunicación a través del Internet en todo lugar a donde vayan. La característica fundamental de dichos dispositivos es la movilidad, suena conveniente la posibilidad que ellos mantengan una dirección IP constante donde quiera que vayan y mantener activas las conexiones en curso de manera transparente. Ambiciosamente, seria aun más conveniente que cada dispositivo tenga una dirección IP pública fija, pero, como es ampliamente conocido, el problema de las escases de direcciones IPv4 públicas imposibilita lo antes mencionado; no así ocurre con las direcciones IPv6, en donde dicho escenario es perfectamente posible.

1.5. Estructura del documento

Este trabajo se organiza de la siguiente manera:

- **Capítulo 1 - Introducción:** Se describe el funcionamiento del protocolo IPv4 Móvil, el formato de las tramas enviadas y los procedimientos que apoyan su funcionamiento.

- **Capítulo 2 - IPv6 Móvil:** Se describe el funcionamiento del protocolo IPv6 Móvil, el formato de las tramas enviadas y los procedimientos que apoyan su funcionamiento.

- **Capítulo 3 – Metodología y herramientas:** Se describen la metodología a seguir para el desarrollo del T.E.G y las herramientas y software utilizado para el cumplimiento de los objetivos planteados.

- **Capítulo 4 – Implementación de la solución**: Se detalla el desarrollo de la solución de la problemática planteada.

- **Capitulo 5 – Desarrollo de la aplicación nativa:** Se detalla el desarrollo de la aplicación MIPv6Droid la cual forma parte de la solución.

- **Capítulo 7 – Pruebas y análisis de los resultados:** Se muestran las pruebas realizadas y un análisis de los resultados.

- **Capítulo 8 – Conclusiones:** Se plantean las conclusiones de los autores del documento.

2. IPv6 Móvil

El diseño de IPv6 Móvil (MIPv6) RFC 3775 [1], se beneficia de las experiencias adquiridas del desarrollo de IPv4 Móvil (MIP) RFC 5944 [2] y de las ventajas presentadas por el nuevo protocolo IPv6 [3] [4]. Aunque ambas versiones comparten muchas características, IPv6 Móvil se encuentra integrado en IPv6 y ofrece muchas otras mejoras, entre las cuales se pueden mencionar:

- Eliminación de la necesidad de los agentes foráneos. IPv6 Móvil opera en cualquier locación sin necesidad de un soporte especial del enrutador local.
- Soporte para optimización de rutas (*route optimization*) que es una parte fundamental del protocolo.
- La mayoría de los paquetes enviados a un nodo móvil mientras se encuentra fuera de su red local son enviados usando cabeceras de enrutamiento IPV6 en vez de encapsulación IP, reduciendo la cantidad de sobrecarga comparado a IPv4 Móvil.
- Se elimina el enrutamiento triangular.
- Se desvincula de cualquier capa de enlace en particular ya que usa *IPv6 Neighbor Discovery* en lugar de *Address Resolution Protocol* (ARP), esto aumenta la robustez del protocolo.
- Optimización de rutas puede operar de una forma segura aun sin asociaciones de seguridad pre-estipuladas.
- El uso de encapsulación IPv6 y de cabeceras de enrutamiento elimina la necesidad de administrar "*tunnel soft state*".

2.1. Terminología

En esta sección se presenta y define la terminología en relación a IPv6 Móvil:

- **Enrutador (*Router*):** Dispositivo encargado de enrutar paquetes entre redes informáticas.
- **Nodo móvil (*Mobile Node*):** Un host o enrutador que cambia su punto de conexión de una red o subred a otra. Un nodo móvil puede cambiar su ubicación sin modificar su dirección IP; y continuar comunicándose con otros nodos en Internet en cualquier ubicación usando una dirección IP constante.
- **Agente local (*Home Agent*):** Un enrutador en la red local del nodo móvil que envía los datagramas a través de un túnel para su entrega al nodo móvil cuando está fuera de casa, y mantiene información de la ubicación actual del nodo móvil.
- **Nodo Correspondiente (*Correspondent Node*):** Un host con el cual el nodo móvil se comunica. Puede ser móvil o fijo.
- **Agentes Móviles (*Mobility Agents*):** Cualquiera de los dos agentes tanto local como foráneo.
- **Red Local (*Home Network*):** Una red que posee el prefijo de red que coincide con la dirección local del nodo móvil.
- **Red Foránea (*Foreign Network*):** Cualquier otra red distinta a la red local del nodo móvil, a la que el nodo móvil está conectado (la red visitada).

- **Enlace Local (*Home Link*):** El enlace físico en donde el prefijo local se encuentra presente y es anunciado.
- **Enlace Foráneo (*Foreign Link*):** El enlace físico en donde el prefijo foráneo se encuentra presente y es anunciado.
- ***Cache de asociaciones (Binding Cache)*:** Cache de asociaciones mantenida por los agentes locales y nodos correspondientes, la cual mantiene las asociaciones entre las direcciones locales y direcciones de cuidado. Debe existir una cache de asociaciones por cada una de las direcciones *unicast* enrutables existentes en el nodo IPv6.
- **Lista de actualización de asociación (*Binding Update List)*:** Esta lista es mantenida por cada nodo móvil. La lista tiene una entrada por cada asociación que el nodo móvil tiene o está tratando de establecer con un nodo específico ya sea el agente local o el nodo correspondiente.
- **Lista de Agentes Locales *(Home Agents Lists)*:** Los agente locales necesitan saber cuáles otros agentes locales existen en el mismo enlace. Esta información es guardada en la *l*ista de agentes locales. Esta lista es usada por el mecanismo de *Dynamic Home Agent Address Discovery* (ver sección 2.3).
- **Dirección Local (*Home Address*):** Una dirección IP que se asigna a un nodo móvil por un período de tiempo extendido. Esta no cambia, sin importar el punto en donde el nodo se conecte a Internet.
- ***Dirección de Cuidado* (Care-of Address):** Es una dirección que el nodo móvil obtiene en la red foránea.
- **Prefijo Local (*Home prefix*)**: Prefijo de red que corresponde con la dirección local del nodo móvil.
- **Asociación (*Binding*):** La asociación entre la dirección local de un nodo móvil y su dirección de cuidado.
- **Actualización de la Asociación (*Binding Update*):** Es un mensaje que actualiza las asociaciones como por ejemplo, el tiempo de vida de la asociación.
- **Túnel (Tunnel):** El camino seguido por un datagrama mientras se encuentra encapsulado.

2.2. Funcionamiento/Operación de IPv6 Móvil

Un nodo móvil siempre espera a ser direccionado por medio de su dirección local sin importar si se encuentra conectado a su enlace local o si se encuentra lejos de él. Mientras que un nodo móvil se encuentra en su red local, los paquetes destinados a su dirección local son enrutados hacia su enlace local, usando los mecanismos convencionales de enrutamiento.

Mientras que el nodo móvil se encuentre conectado a algún enlace foráneo, es igualmente direccionado por medio de una o varias direcciones de cuidado e incluso su dirección local. El también puede aceptar paquetes dirigidos hacia varias direcciones de cuidado, como cuando este se encuentra moviéndose pero sigue siendo alcanzable en su enlace previo que también es foráneo.

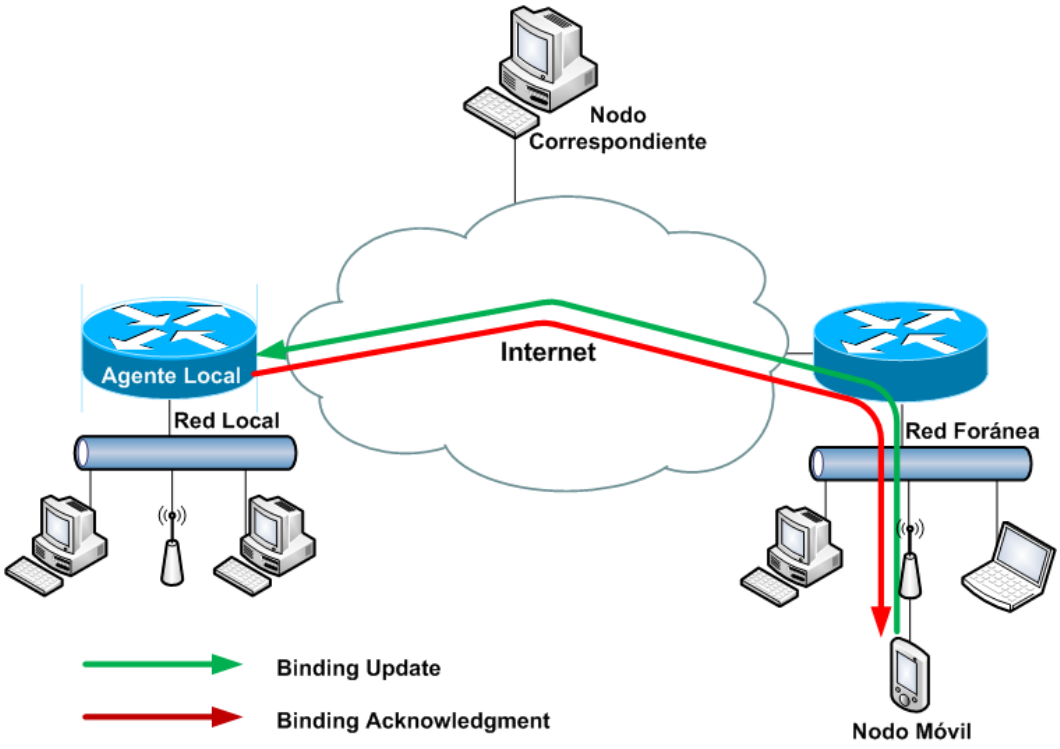


Figura 2.1 Registro Local

La Figura 2.1 representa el registro que el nodo móvil efectúa con su agente local luego que el nodo móvil ha detectado que se ha movido.

Detección de movimiento: Cuando el nodo móvil se mueve a una nueva red, en primer lugar, configura una nueva dirección de cuidado utilizando el prefijo anunciado por el enrutador de acceso (*Access Router*), y la dirección de su interfaz de red MAC. El nodo móvil puede obtener su dirección de cuidado a través de mecanismos IPv6 convencionales, tales como auto configuración sin estado (*stateless*) o con estado (*statefull*). La detección de movimiento se explica detalladamente en la sección 2.2.4.

Registro local: El nodo móvil realiza el registro local con su agente local enviándole un mensaje *Binding Update* conteniendo su dirección local y su nueva dirección de cuidado. Después de la recepción de este paquete, el agente local llena la *Binding Cache* asociando la dirección local con la dirección de cuidado. Por último, se envía un mensaje *Binding Acknowledgment* para notificar al nodo móvil de la terminación del registro.

A partir de este momento el agente local interceptará todos los paquetes dirigidos a la dirección local del nodo móvil y posteriormente se los redirigirá por medio de un túnel IPv6 en IPv6 en donde se encuentre.

En IPv6 Móvil existen dos posibles modos de comunicación entre un nodo móvil y un nodo correspondiente sin perder las conexiones ya establecidas entre ellos.

.

El primero se refiere a un túnel bidireccional (*bidirectional tunneling) (*ver sección 2.2.1), el cual es comúnmente usado cuando el nodo correspondiente no tiene ningún soporte de IPv6 Móvil y es necesaria la continua comunicación transparente entre ambos nodos.

La otra forma es optimización de rutas (*route optimization) (ver sección 2.2.2),* permite la comunicación entre ambos nodos directamente usando el camino más corto entre ellos. Este modo de operación requiere que ambas partes tengan soporte de IPv6 Móvil y que el nodo móvil realice el registro correspondiente enviándole al nodo correspondiente un mensaje *Binding Update* de manera análoga a como previamente lo hizo con el agente local en el registro local. Hasta que esto ocurra, la comunicación entre ambos nodos continuara realizándose por medio de túnel bidireccional.

Es posible que el nodo móvil vuelva a moverse de subred hacia su enlace local o incluso hacia alguna otra red foránea; en el primer caso se eliminaría la asociación con el nodo correspondiente y el agente local por medio de un mensaje *Binding Update*, y la transmisión de los paquetes se efectuará normalmente sin soporte de IPv6 Móvil. En la sección 2.2.4 se explica detalladamente este escenario. En el segundo caso, si se está usando optimización de rutas como modo de comunicación, se debe notificar tanto al agente local como al nodo correspondiente de la nueva localidad por medio de un mensaje *Binding Update* y volver hacer todos los pasos exigidos para continuar la comunicación transparentemente; en el caso de usar túnel bidireccional solo se le notificaría al agente local.

2.2.1. Túnel bidireccional

Cuando un nodo móvil y un agente local completan el registro local, estos nodos crean una conexión de túnel IPv6 en IPv6 entre ellos. Los puntos finales del túnel son las direcciones del agente local y la dirección de cuidado del nodo móvil. Este túnel es usado para ocultar la locación del nodo móvil al nodo correspondiente y mantener la transparencia de que el nodo móvil se ha movido de su red local como se puede apreciar en la Figura 2.2.

Un nodo móvil usualmente usa su dirección local como dirección lógica cuando envía paquetes. Esto asegura que la comunicación entre un nodo móvil y otros nodos sobreviva cuando el nodo móvil se mueve de una red a otra dado que la dirección local nunca cambia. Sin embargo, el nodo móvil no puede simplemente enviar un paquete con dirección fuente siendo la dirección local. Esos paquetes serian topológicamente incorrectos y el enrutador (AR) que sirve a dicha red foránea los descartaría. Para evitar este problema, es necesario el uso de la conexión de túnel creada entre el nodo móvil y el agente local o el uso de optimización de rutas.

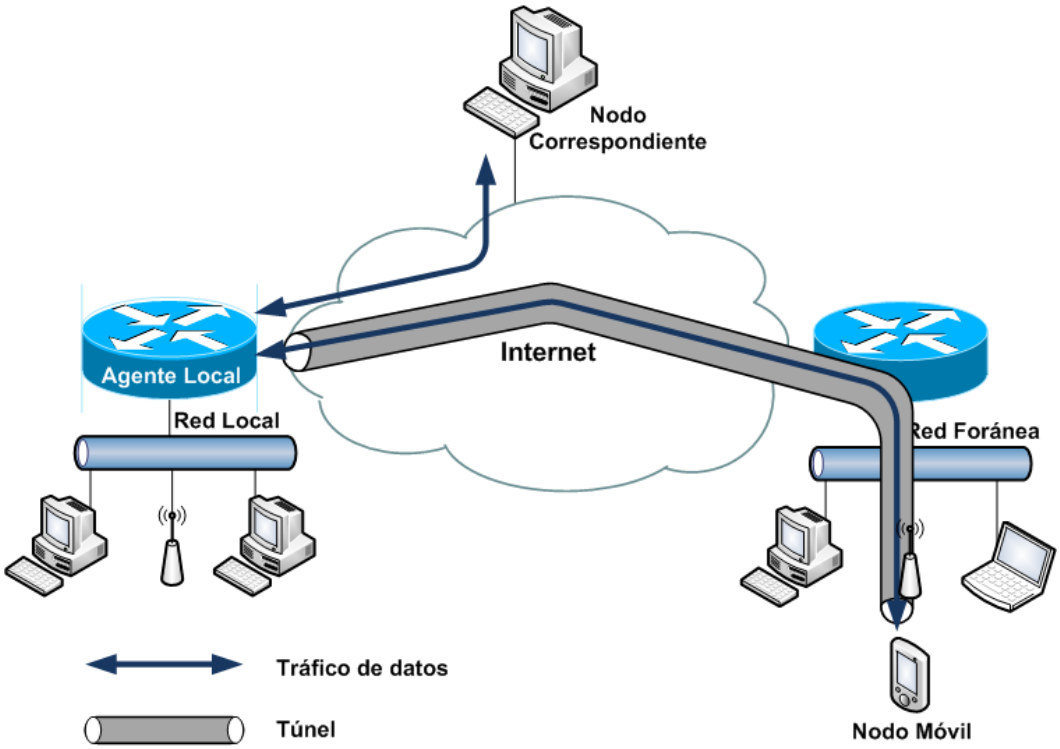


Figura 2.2 Esquema de Túnel Bidireccional

El paquete es encapsulado dentro de otra cabecera IPv6 en la cual la dirección origen y destino son la dirección de cuidado del nodo móvil y la dirección del agente local respectivamente. El paquete es des encapsulado en el agente local y este lo reenvía hacia su destino final. De esta forma el paquete se ve como si hubiese sido enviado por un nodo conectado a la red local.

Cuando el nodo correspondiente envía paquetes al nodo móvil, la conexión del túnel es también usada en dirección contraria. Todos los paquetes los cuales la dirección destino es la dirección local del nodo móvil son entregados a la red local del nodo móvil. Estos paquetes son interceptados por el agente local si este tiene una entrada válida en la *Binding Cache* para el nodo móvil, y luego son enviados usando encapsulación IPv6 sobre IPv6. La dirección origen y destino de la cabecera IPv6 externa son la dirección del agente local y la dirección de cuidado del nodo móvil respectivamente.

2.2.2. Optimización de rutas

Para poder utilizar este modo de operación el nodo móvil debe completar el registro correspondiente con el nodo correspondiente como se mencionó anteriormente. Para lograrlo se debe pasar por un procedimiento llamado *Return Routability Procedure* (*RRP*) (ver sección 2.2.3) como requisito para completar dicho registro correspondiente y empezar la comunicación utilizando optimización de rutas.

Los paquetes provenientes del nodo correspondiente podrán ser enrutados directamente por medio de la dirección de cuidado del nodo móvil. Al mandar algún paquete hacia cualquier

destino, el nodo correspondiente verifica su *Binding Update List* en busca de una entrada para la dirección destino del paquete, si es encontrada una entrada, el nodo usa un nuevo tipo de cabecera de enrutamiento IPV6 "*Type 2 Routing Header*" (ver sección 2.5.4) y la opción *Home Address Option* para asegurarse que la dirección IPv6 contenida en la cabecera IPv6 sea topológicamente correcta y dirigir el paquete hacia el nodo móvil por el camino directo hacia la dirección de cuidado indicada por la asociación.

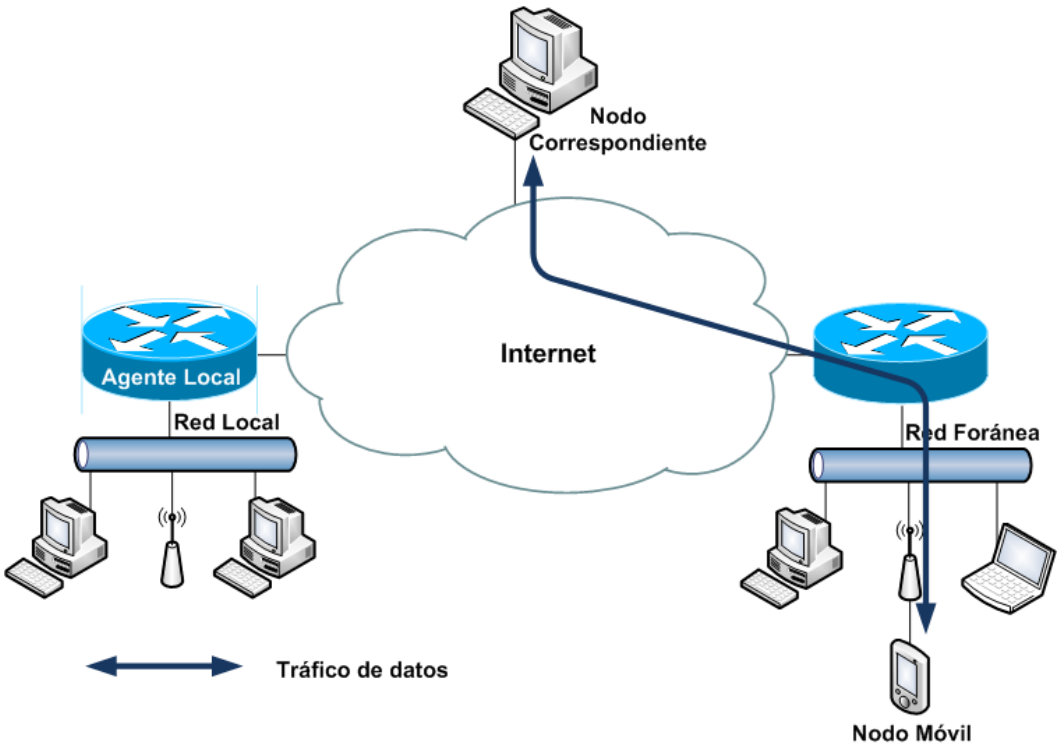


Figura 2.3 Esquema de optimización de rutas

Encaminar los paquetes directamente a la dirección de cuidado del nodo móvil permite la utilización del camino de comunicación más corto como se puede apreciar en la Figura 2.3. También elimina la congestión que se presenta en el enlace local y agente local del nodo móvil. Adicionalmente, el impacto de alguna posible falla del agente local.

2.2.3. *Return Routability Procedure* (*RRP*)

El *RRP* es un procedimiento en el cual se habilita al nodo correspondiente una forma de obtener una seguridad razonable de que el nodo móvil es verdaderamente quien dice ser y si es alcanzable por medio de la dirección de cuidado y la dirección local que este está afirmando que posee. Solamente con esta garantía, el nodo correspondiente podrá aceptar mensajes *Binding Update* provenientes del nodo móvil para completar el registro correspondiente y que le indicaran al nodo correspondiente hacia dónde dirigir el tráfico destinado al nodo móvil por medio de su dirección de cuidado.

La Figura 2.4 describe el intercambio de paquetes entre el nodo móvil y el nodo correspondiente durante el *RRP*.

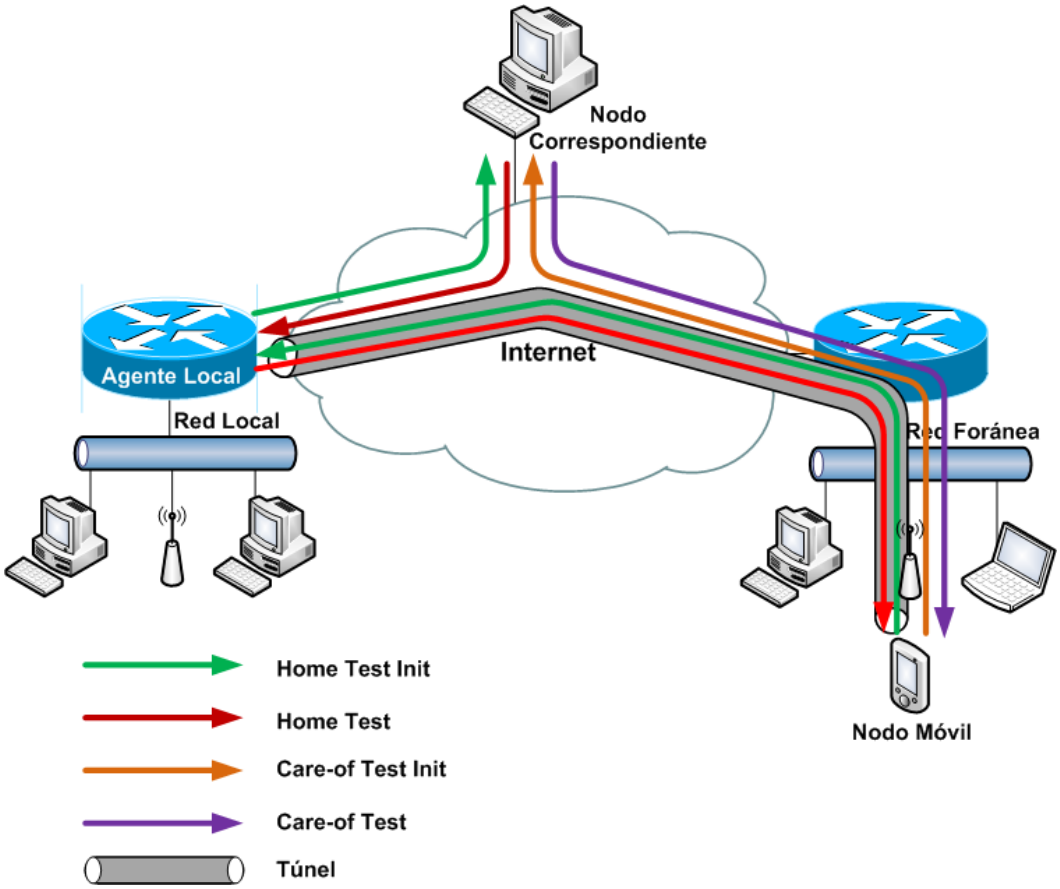


Figura 2.4 *Return Routability Procedure*

Solo el nodo móvil puede iniciar el *RRP*. Cuando un nodo móvil quiere iniciar una comunicación utilizando optimización de rutas, este envía dos mensajes iniciales llamados *Home Test Init* y *Care-of Test Init*, posteriormente el nodo correspondiente si es capaz de soportar optimización de rutas responderá al nodo móvil con otros dos mensajes llamados *Home Test y Care-of Test*, estos cuatro mensajes conforman el *RRP*.

Los mensajes *Home Test Init* y *Home Test* son usados para asegurarse de que el nodo móvil es capaz de enviar y recibir paquetes usando su dirección local. Estos mensajes son intercambiados a través del túnel establecido con el agente local.

Igualmente, el intercambio de los mensajes *Care-of Test Init* y *Care-of Test* verifica que el nodo móvil también puede enviar y recibir paquetes usando la dirección de cuidado. Estos mensajes a diferencia son enviados directamente entre el nodo móvil y el correspondiente.

Los cuatros mensajes intercambiados durante el *RRP* también son usados para generar una llave criptográfica (*Binding management key:* Kbm) que es usada por el nodo móvil para firmar el mensaje *Binding Update* que completará el registro correspondiente. Dicha llave criptográfica es generada creando un valor hash entre los dos *tokens* que son enviados desde el nodo correspondiente al nodo móvil en los mensajes *Home Test* y *Care-of Test*.

Para este momento se puede completar el registro correspondiente enviando dicho mensaje *Binding Update* hacia el nodo correspondiente como se puede ver en la Figura 2.5 y posteriormente la introducción de la asociación en la *Binding Cache* por parte del nodo correspondiente.

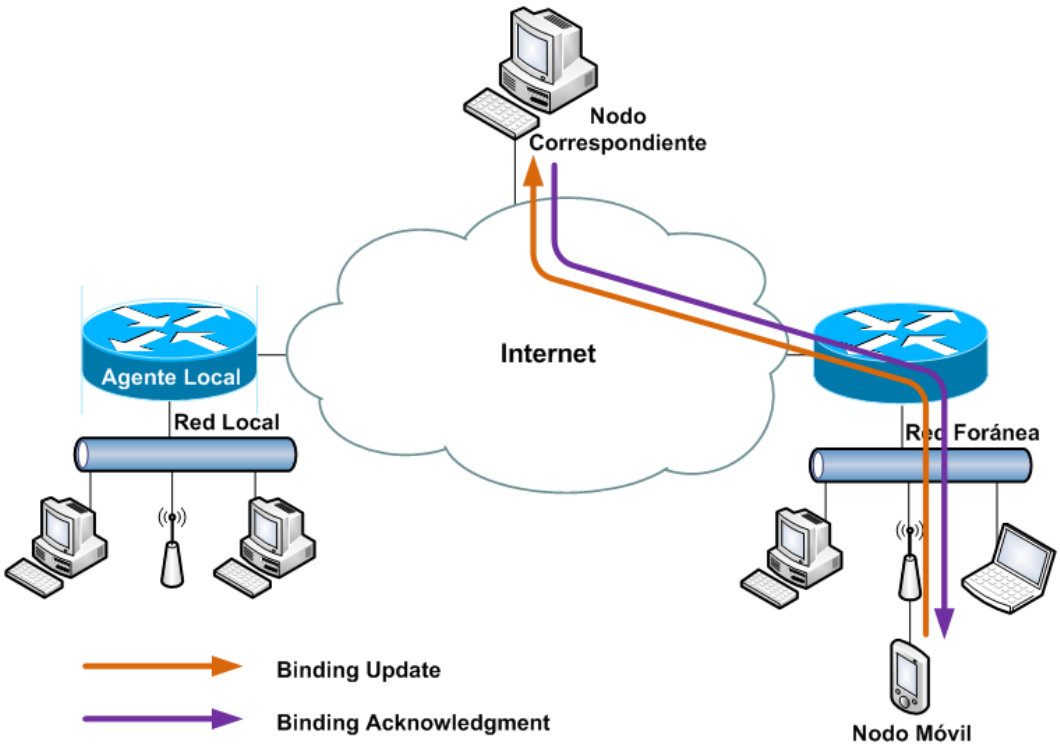


Figura 2.5 Registro correspondiente

2.2.4. Detección de movimiento

El principal objetivo de la detección de movimiento es detectar cuando se produce un movimiento en capa 3 (L3 *handover*). Para esto se utilizan mecanismos propios de IPv6 facilitando así la implementación y detección de los movimientos del nodo, entre estos mecanismos se encuentra *Neighbor Discovery,* incluyendo *Router Discovery* y *Neighbor Unreachability Detection.*

Se utiliza *Neighbor Unreachability Detection* para detectar cuando el enrutador por defecto ya no es bidireccionalmente alcanzable, en dado caso el nodo móvil debe descubrir un nuevo enrutador por defecto. Esta detección solo es posible cuando el nodo móvil aun tiene paquetes que enviar, y en la ausencia de frecuentes mensajes *Router Advertisement* provenientes de su actual enrutador por defecto o indicaciones de la capa de enlace, el nodo móvil podría llegar a

ser consciente de que un movimiento en capa 3 ha ocurrido. Por lo tanto, el nodo móvil debe complementar este método con otros indicios como pueden ser, el recibimiento de mensajes *Router Advertisement* provenientes de otro enrutador con anuncios de prefijo distintos, vencimiento de los intervalos de tiempo esperados en los que llegan los mensajes *Router Advertisement* provenientes del enrutador por defecto o cualquier otra información que esté disponible.

Cuando el nodo móvil detecta un movimiento en capa 3 lo cual lleva implícito el movimiento del nodo, este realiza el procedimiento *Duplicate Address Detection,* selecciona un nuevo enrutador como consecuencia del procedimiento *Router Discovery,* y luego ejecuta *Prefix Discovery* con ese nuevo enrutador para formar una o varias dirección de cuidado, ya sea por medio de autoconfiguración con estado o sin estado. Luego registra su nueva dirección de cuidado principal con su agente local y luego hace lo propio con el nodo correspondiente en el caso de que se puedan comunicar por medio de optimización de rutas.

2.2.5. Volviendo al enlace local

El nodo móvil detectará que ha vuelto a su enlace local a través del algoritmo de detección de movimiento explicado en el punto anterior, cuando el nodo móvil se da cuenta de la presencia de que su prefijo local de subred *se* encuentra en su enlace actual.

El nodo móvil puede ahora enviar un mensaje *Binding Update* a su agente local, para indicarle que no continúe sirviéndole como su agente local ni interceptando o haciendo túnel de los paquetes dirigidos a él. En este *mensaje Binding Update*, el nodo móvil debe activar el bit de *Acknowledge* (A) y el de *Home Registration* (H), establecer el campo de *Lifetime* en cero, y establecer el campo dirección de cuidado con la dirección local del nodo.

El agente local responderá a esto con un mensaje *Binding Acknowledgement*, el nodo móvil al recibir dicho mensaje, debe enviar un mensaje *Unsolicited Neighbor Advertisement* en *Multicast* hacia todos los nodos de su *e*nlace local, para indicarles su dirección de la capa de enlace de datos ligada a su dirección local y dirección *link-local,* de esta manera y a partir de ahora el nodo móvil empezará a funcionar como cualquier otro nodo de la red local.

La Figura 2.6 muestra el intercambio sucesivo de paquetes que se efectúa cuando el nodo móvil vuelve a su enlace local.

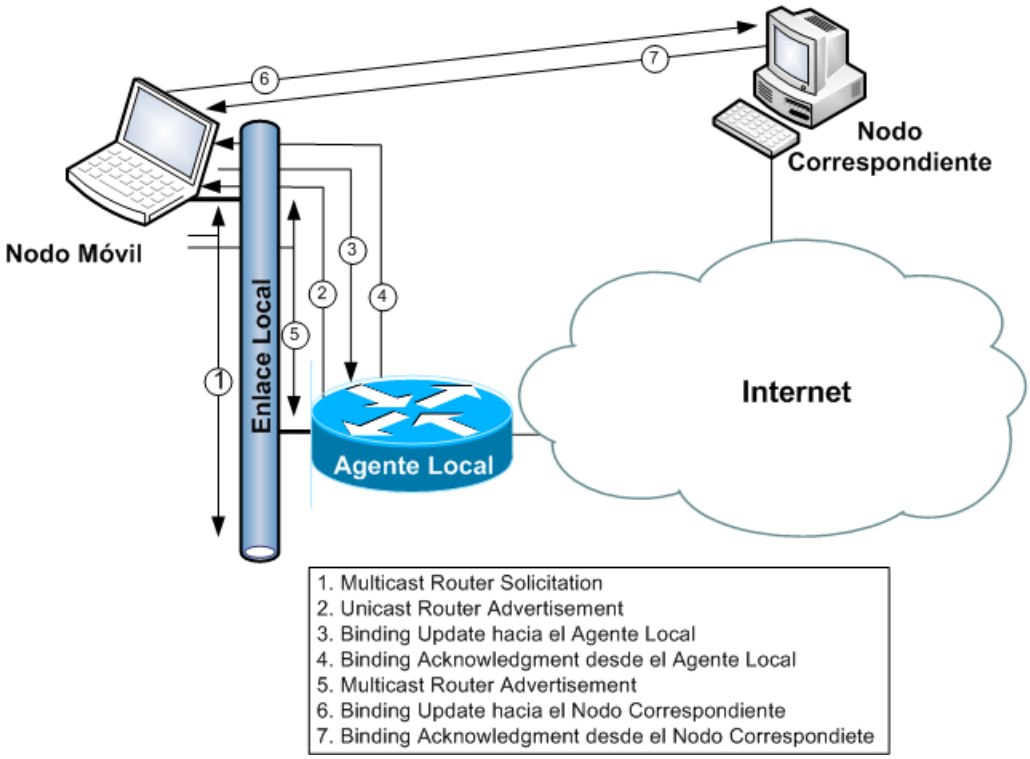


Figura 2.6 Volviendo al enlace local

2.3. Dynamic Home Agent Address Discovery

A veces cuando un nodo móvil necesita enviar un mensaje *Binding Update* a su agente local para registrar su nueva dirección de cuidado principal, el nodo móvil podría no saber la dirección de algún enrutador en su enlace local que pueda servirle como agente local. Por ejemplo, si un nodo móvil se reinicia en una red foránea, no hay información sobre el agente local a menos que dicha información este pre-configurada.

El mecanismo *Dynamic Home Agent Address Discovery* es usado para obtener las direcciones de agentes locales cuando el nodo móvil está en una red foránea.

El nodo móvil envía un mensaje *ICMP Home Agent Address Discovery Request* a una dirección *anycast* específica construida usando el prefijo local del nodo móvil. El agente local en su enlace local que reciba esta solicitud retornará un mensaje *ICMP Home Agent Address Discovery Reply* conteniendo todas las direcciones de los agentes locales que operan en el enlace local que se sepan en el momento.

Si el nodo móvil no recibe una respuesta al mensaje, se debe renviar el mensaje de solicitud. El nodo móvil luego de recibir el mensaje *Home Agent Address Discovery Reply* podrá enviar un mensaje *Binding Update* para realizar el registro local a cualquiera de las direcciones listadas en el campo *Home Agents Address* del mensaje antes mencionado. Si el nodo móvil tiene un registro actual con alguno de los agentes locales, entonces debe intentar cualquier nuevo registro primero con ese agente local.

2.4. Mobile Prefix Discovery

Las direcciones IPv6 tienen un tiempo de vida que es derivado del tiempo de vida del prefijo de red de ella. Cuando la dirección local del nodo móvil está por expirar, este debe enviar un mensaje *Mobile Prefix Solicitation* hacia el agente local para adquirir la información más reciente acerca de su prefijo local por medio de un mensaje *Mobile Prefix Advertisement* enviado de vuelta desde el agente local. El nodo móvil debe soportar y usar *IPsec* para proteger esta solicitud.

Mobile Prefix Discovery es similar a *Router Discovery* usado en *Neighbor Discovery*, excepto que es enrutado desde el nodo móvil en la red foránea hacia el agente local ubicado en la red local por métodos habituales de enrutamiento *unicast y* viceversa.

Al enviar el mensaje de solicitud *Mobile Prefix Solicitation,* la dirección fuente es establecida a la dirección de cuidado del nodo móvil, el destino del mensaje es la dirección del agente local con el cual el nodo móvil esta actualmente registrado. El mensaje debe incluir una opción *Home Address Option* la cual contendrá la dirección local del nodo móvil.

El agente local al recibir el mensaje antes descrito, debe responder al nodo móvil con un mensaje *Mobile Prefix Advertisement*. La dirección destino debe ser la dirección de cuidado del nodo móvil que realizó la solicitud. Debe existir una cabecera *Type 2 Routing Header* que contenga la dirección local del nodo móvil. En las opciones del mensaje debe incluirse la lista de prefijos de red que fueron solicitados.

Un agente local puede incluso enviar mensajes *Mobile Prefix Advertisement* aunque el nodo móvil no ha solicitado (*unsolicited*) ninguna información de prefijos en los siguientes casos:

- El estado de las banderas del prefijo local que un nodo móvil está usando cambia.
- El valor *valid* o *preferred lifetime* de un prefijo local es reconfigurado.
- Un nuevo prefijo local es agregado.
- El estado de las bandera o valores *lifetime* de un prefijo local que no es usado por ningún nodo móvil cambia.

Cuando cualquiera de las dos primeras condiciones ocurre, el agente local debe por obligación enviar un mensaje *unsolicited Mobile Prefix Advertisement*. Cuando la tercera condición ocurre se debería enviar el mensaje. Y cuando la última condición ocurre, el agente local podría enviar el mensaje.

Dicho procesamiento puede resultar en la configuración por parte del nodo móvil de una nueva dirección local, aunque debido a la separación entre el valor *preferred lifetime* y el valor *valid lifetime*, estos cambios no deben afectar a la mayoría de las comunicaciones del nodo móvil, de la misma manera como a otros nodos que se encuentran en el enlace local.

2.5. Tipos de Mensajes Definidos en MIPv6

IPv6 Móvil define una serie de tipos de mensajes y opciones necesarios para el correcto funcionamiento del protocolo, también aprovecha de las ventajas provistas por IPv6 en cuanto a las cabeceras de extensión.

2.5.1. Cabecera de movilidad

La cabecera de movilidad la cual es mostrada en la Figura 2.7, es una nueva cabecera de extensión introducida para llevar los mensajes de señalización de IPv6 Móvil.

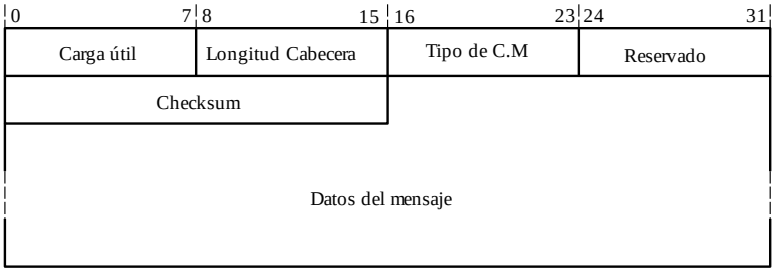


Figura 2.7 Cabecera de Movilidad

- Carga útil (*Payload Proto):* Indica el tipo de cabecera que le sigue, sin embargo, la especificación actual no permite que exista alguna cabecera que le siga a la cabecera de movilidad. Esta debe ser la última cabecera existente en la cadena de cabecera de un paquete IPv6. Este campo debe establecerse siempre a 58 (IPV6-NONXT), que indica que no hay una cabecera que le sigue.
- Longitud de cabecera (*Header Len):* Indica la longitud de una cabecera de movilidad en unidades de 8 bytes excluyendo los primeros 8 bytes.
- Tipo de Cabecera de Movilidad *(MH Type):* Indica el tipo de mensaje de la cabecera de movilidad. La Tabla 2.1 muestra todas las posibilidades.
- Reservado*:* Esta reservado para uso futuro. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.
- *Checksum:* Guarda el valor *checksum* de un mensaje cabecera de movilidad.
- El resto de la cabecera es definido según el tipo de cabecera de movilidad. Además puede contener algunas opciones de movilidad (ver sección 2.5.2).

La Tabla 2.1 muestra los tipos de cabecera de movilidad.

Tipo	Descripción
0	Binding Refresh Request
1	Home Test Init
2	Care-of Test Init
3	Home Test
4	Care-of Test
5	Binding Update
6	Binding Acknowledgment
7	Binding Error

Tabla 2.1 Tipos de cabecera de movilidad

Binding Refresh Request

El mensaje *Binding Refresh Request (BRR)* que se muestra en la Figura 2.8, es usado cuando un nodo correspondiente necesita extender el tiempo de vida de un *Binding* para un nodo móvil. Este mensaje usa el valor “0” para Tipo de Cabecera de Movilidad (*MH type*) y es enviado desde el nodo correspondiente a través del agente local hacia el nodo móvil mediante un túnel.

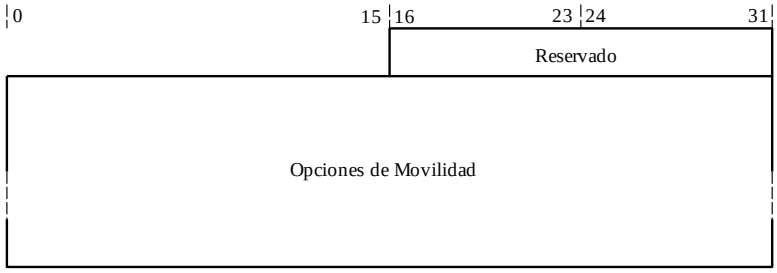


Figura 2.8 *Binding Refresh Request*

- Reservado: Esta reservado para uso futuro. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.
- Opciones de movilidad: RFC 3775 [1] no define alguna opción válida que se pueda utilizar con este mensaje.

Home Test Init

El mensaje *Home Test Init* (*HoTI*) que se muestra en la Figura 2.9, es usado para iniciar el *RRP* (ver sección 2.2.3). Este mensaje usa el valor “1” para Tipo de Cabecera de Movilidad (*MH type*).

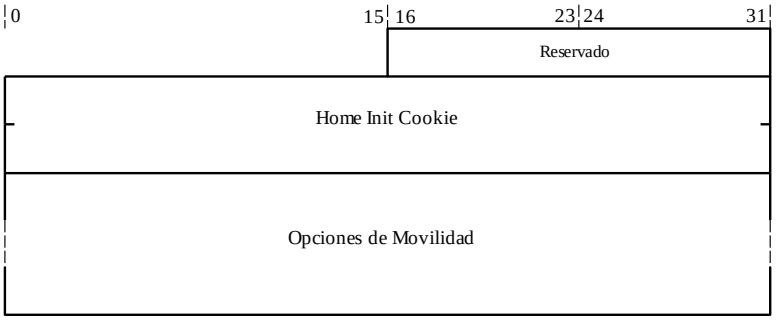


Figura 2.9 *Home Test Init*

- Reservado: Esta reservado para uso futuro. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.
- *Home Init Cookie:* Contiene un valor aleatorio, el *home init cookie*.
- Opciones de movilidad: RFC 3775 [1] no define alguna opción válida que se pueda utilizar con este mensaje.

Care-of Test Init

El mensaje *Care-of Test Init* (*CoTI*) que se muestra en la Figura 2.10, es usado para iniciar el *RRP* (ver sección 2.2.3). Este mensaje usa el valor “2” para Tipo de Cabecera de Movilidad (MH *type*).

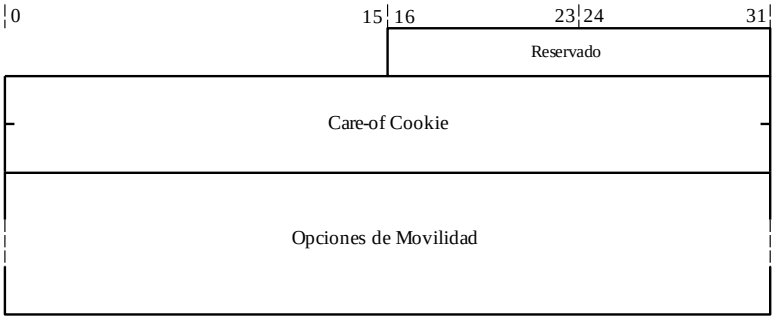


Figura 2.10 *Care-of Test Init*

- Reservado: Esta reservado para uso futuro. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.
- Care-of Init Cookie: Contiene un valor aleatorio, el care-of init cookie.
- Opciones de movilidad: RFC 3775 [1] no define alguna opción válida que se pueda utilizar con este mensaje.

Home Test

El mensaje *Home Test* (*HoT*) que se muestra en la Figura 2.11, es usado como respuesta al mensaje HoTI enviado desde el nodo móvil hacia el nodo correspondiente. Este mensaje usa el valor “3” para Tipo de Cabecera de Movilidad (*MH type*).

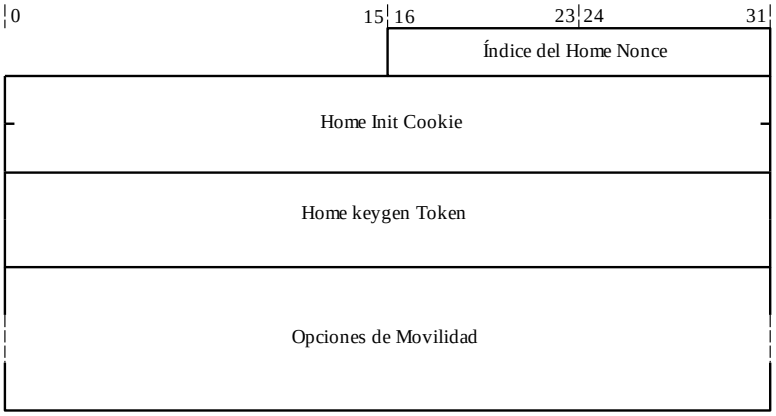


Figura 2.11 *Home Test*

- Índice del *Home Nonce* (*Home nonce Index):* Contiene el valor de un índice en el arreglo de *home nonce* que mantiene el nodo correspondiente.
- *Home Init Cookie*: Contiene el *Home init cookie*.
- *Home Keygen Token*: Contiene el valor *Home Keygen Token* usado en el *RRP*.
- Opciones de movilidad: RFC 3775 [1] no define alguna opción válida que se pueda utilizar con este mensaje.

Care-of Test

El mensaje *Care-of Test* (*CoT*) que se muestra en la Figura 2.12, es usado como respuesta al mensaje *CoTI* enviado desde el nodo móvil hacia el nodo correspondiente. Este mensaje usa el valor “4”para Tipo de Cabecera de Movilidad (*MH type*).

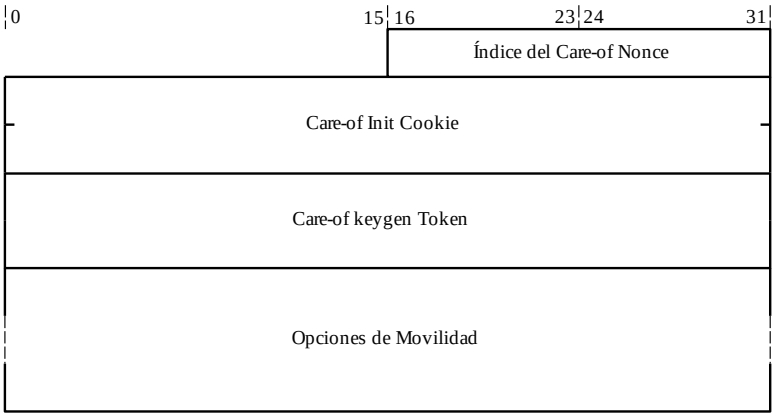


Figura 2.12 *Care-of Test*

- Índice del *Home Nonce* (*Home nonce Index):* Contiene el valor de un índice en el arreglo de *care-of nonce* que mantiene el nodo correspondiente.
- *Care-of Init Cookie*: Contiene el Care-of init cookie.
- *Care-of Keygen Token*: Contiene el valor *Care-of Keygen Token* usado en el *RRP*.
- Opciones de movilidad: RFC 3775 [1] no define alguna opción válida que se pueda utilizar con este mensaje.

Actualización de Asociación (*Binding Update*)

El mensaje *Binding Update* (*BU*) que se muestra en la Figura 2.13, es usado por el nodo móvil para notificar al nodo correspondiente o al agente local sobre información de su asociación de una dirección de cuidado y la dirección local del mismo. Este mensaje usa el valor “5” para Tipo de Cabecera de Movilidad (*MH type*).

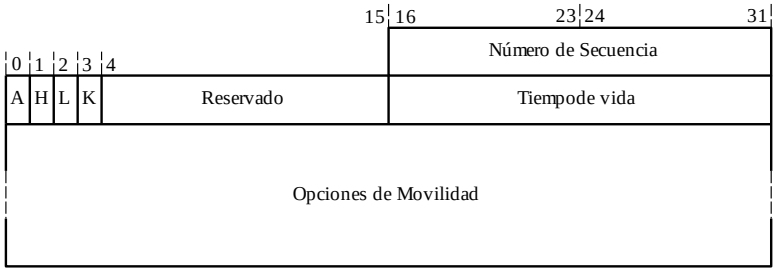


Figura 2.13 *Binding Update*

- Número de Secuencia: Número entero usado para secuenciar y verificar coincidencia de los mensajes *Binding Update* con sus respectivos mensajes *Binding Ackowledgment*.
- *Acknowledge (A):* Esta bandera es encendida cuando se solicita un mensaje *Binding Acknowledgment* como respuesta al presente mensaje *Binding Update*.
- *Home Registration (H):* Esta bandera significa que el presente mensaje *Binding Update* es un mensaje para realizar un *Home Registration*.
- *Link-Local Address Compatibility (L):* Esta bandera significa que la dirección *link-local* del nodo móvil tiene el mismo identificador de interfaz que su dirección local.
- *Key Management Mobility Capability (K):* Esta bandera significa que el protocolo usado para establecer la asociación de seguridad *IPSec* entre el nodo móvil y el agente local sobrevive a movimientos. De lo contrario deberá volver a ser ejecutado.
- Reservado: Este campo no es usado. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.
- Tiempo de vida (*Lifetime):* Contiene el número de unidades de tiempo restantes antes de que la asociación sea considerado como expirado. La unidad de tiempo es de 4 segundos.
- Opciones de movilidad: Este campo contendrá las distintos opciones de movilidad (ver sección 2.5.2) validos en un *Binding Update*, los cuales son:
    - *Binding Authorization Data*
    - *Nonce indices*
    - *Alternate Care-of Address*

Binding Acknowledgment

El mensaje *Binding Acknowledgment* (*BA*) que se muestra en la Figura 2.14, es usado como respuesta a un mensaje BU indicando que este último se ha recibido. Este mensaje usa el valor “6” para Tipo de Cabecera de Movilidad (*MH type*).

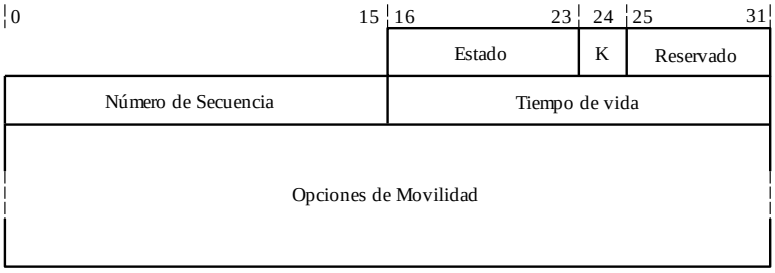


Figura 2.14 *Binding Acknowledgment*

- *Key Management Mobility Capability* (*K*): Esta bandera significa que el protocolo usado por el agente local para establecer la asociación de seguridad IPSec entre el nodo móvil y el agente local sobrevive a movimientos. De lo contrario deberá volver a ser ejecutado.
- Reservado: Este campo no es usado. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.
- Número de Secuencia (*Sequence #*): Copiado del mismo campo del BU usado para secuenciar y verificar coincidencia de los mensajes Binding Update con sus respectivos mensajes Binding Acknowledgment.
- Tiempo de vida (*Lifetime*): Contiene el tiempo de vida otorgado en unidades de tiempo de 4 segundos, para el cual este nodo debe conservar la entrada para este nodo móvil en su Binding Cache.
- Estado (*Status*): Este campo especifica el resultado del procesamiento del mensaje BU recibido. La Tabla 2.2 muestra la lista de los distintos códigos de estado (*status*).
- Opciones de movilidad: Este campo contendrá los distintas opciones de movilidad (ver sección 2.5.2) válidos en un mensaje Binding Update, los cuales son:
    - *Binding Authorization Data*
    - *Binding Refresh Advice*

Código	Descripción
0	Binding Update accepted
1	Accepted but prefix discovery necessary
128	Reason unspecified
129	Administratively prohibited
130	Insufficient resources
131	Home registration not supported
132	Not home subnet
133	Not home agent for this mobile node
134	Duplicate Address Detection failed
135	Sequence number out of window
136	Expired home nonce index
137	Expired care-of nonce index
138	Expired nonces
139	Registration type change disallowed

Tabla 2.2 Códigos de estado del mensaje *Binding Acknowledgment*

Binding Error

El mensaje *Binding Error* (*BE*) que se muestra en la Figura 2.15, es usado para indicar que ha ocurrido un error durante el procesamiento de señalizaciones de movilidad. Este mensaje usa el valor “7” para Tipo de Cabecera de Movilidad (*MH type*).

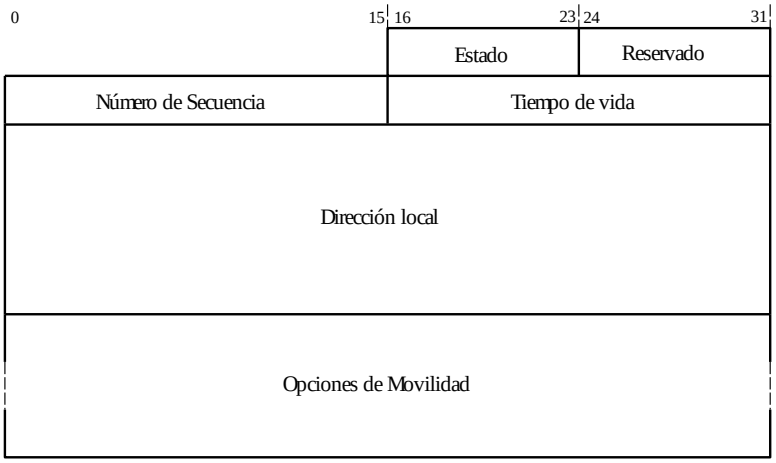


Figura 2.15 *Binding Error*

- Estado (*Status*)*:* Este campo especifica la razón para este mensaje.
- Reservado: Esta reservado para uso futuro. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.
- Dirección local: La dirección local que estaba contenida en la opción *Home Address Destination Option*.
- Opciones de movilidad: RFC 3775 [1] no define alguna opción válida que se pueda utilizar con este mensaje.

Código	Descripción
1	Unknown Binding for Home Address Destination Option
2	Unrecognized MH Type value

Tabla 2.3 Códigos de estado del mensaje *Binding Error*

2.5.2. Opciones de Movilidad

Los mensajes de movilidad pueden incluir o no ciertas opciones llamadas opciones de movilidad que se muestra en la Figura 2.16, las cuales proveen de información adicional. Existen seis opciones definidas en IPv6 Móvil.

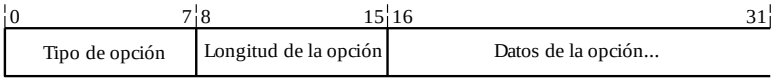


Figura 2.16 Opciones de Movilidad

- Tipo de opción: Identificador del tipo de opción. Los tipos de opción y su descripción se pueden apreciar en la Tabla 2.4.
- Longitud de la opción: Representa la longitud en octetos de los datos de opción sin incluir los campos Tipo de opción y Datos de la opción.

- Datos de la opción: Campo de longitud variable el cual contendrá los datos específicos a la opción.

Tipo	Descripción
0	Pad1
1	PadN
2	Binding Refresh Advice
3	Alternate Care-of Address
4	Nonce indices
5	Binding Authorization Data

Tabla 2.4 Tipos de Opciones de movilidad

Las opciones de movilidad pueden tener ciertos requerimientos de alineación siguiendo la convención en IPv6.

Pad1

La opción *Pad1* que se muestra en la Figura 2.17 es usada cuando 1 byte de relleno es necesario para cumplir los requisitos de alineación de otras opciones de movilidad y no tiene requisitos de alineación, esta opción debe ser ignorada por el receptor.

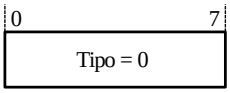


Figura 2.17 Pad1

Si es necesario más de 1 byte de relleno, se debe utilizar la opción *PadN*, en vez de utilizar múltiples *Pad1*.

PadN

La opción *PadN* que se muestra en la Figura 2.18 es usada cuando dos o más bytes de relleno son necesarios para cumplir los requisitos de alineación.

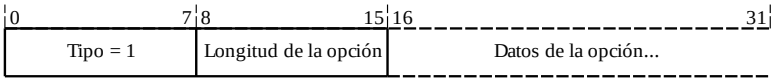


Figura 2.18 PadN

- Longitud de la opción (*Option Length*): Este campo está establecido por la cantidad de espacio de relleno necesario menos dos bytes.
- Datos de la opción (*Option Data*): Este campo deberá contener la cantidad de bytes indicado por *Option lenght* y deberán ser rellenados con ceros.

Binding Refresh Advice

La opción *Binding Refresh Advice* que se muestra en la Figura 2.19, es usada para especificar el intervalo recomendado entre los mensajes BU para actualizar la información de

un *Binding* con el agente local. Esta opción es usada con el mensaje BA que es enviado desde el agente local hacia el nodo móvil.

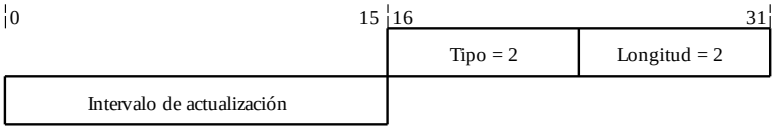


Figura 2.19 *Binding Refresh Advice*

- Intervalo de actualización (Refresh Interval): Este campo indica el valor del intervalo en unidades de 4 segundos.

Alternate Care-of Address

La opción *Alternate Care-of Address* que se muestra en la Figura 2.20, es usada en dos casos con el mensaje *BU*. El primer caso es cuando el nodo móvil desea asociar su dirección local a una dirección diferente a la que se encuentra en el campo de dirección origen del *BU*. El segundo caso es para proteger la información de la dirección de cuidado de un atacante presente en el camino.

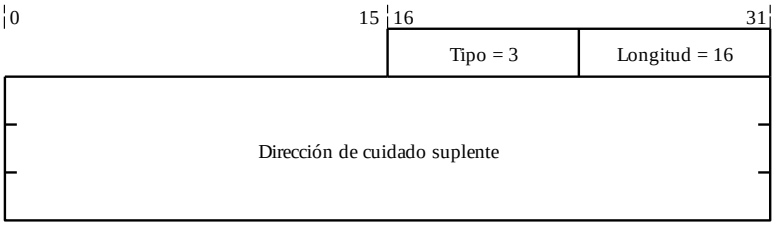


Figura 2.20 *Alternate Care-of Address*

Dirección de cuidado suplente *(Alternate Care-of Address):* Contiene la dirección que será utilizada como dirección de cuidado para la asociación.

Nonce Indices

La opción *Nonce Indices* que se muestra en la Figura 2.21, es válida solo en el mensaje BU enviado a un nodo correspondiente y solo si es presentado junto a la opción *Binding Authorization Data*. Esta opción es usada para indicar los valores que son usados para calcular el valor autenticador especificado en la opción *Binding Authorization Data*.

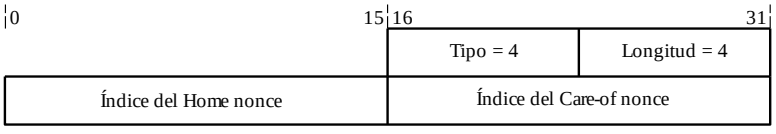


Figura 2.21 *Nonce Indices*

- Índice del Home nonce (*Home nonce Index*): Contiene el valor *Home nonce* copiado del mensaje *HoT*.
- Índice del Home nonce (*Care-of nonce Index*): Contiene el valor *Care-of nonce* copiado del mensaje *CoT*.

Binding Authorization Data

La opción *Binding Authorization Data* que se muestra en la Figura 2.22, es usada para enviar un valor *hash (*autenticador) necesario para completar el registro correspondiente. Esta opción solo es válida en el mensaje *BU* y el *BA*.

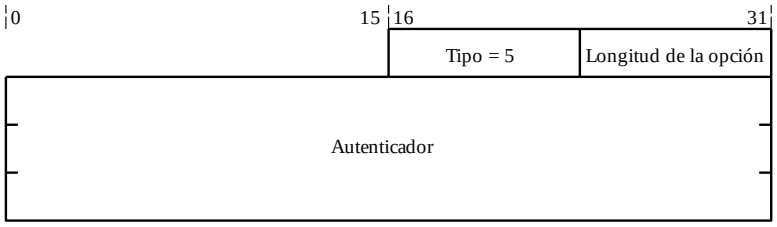


Figura 2.22 *Binding Authorization Data*

- Longitud de la opción (*Option Length*): Contiene la longitud del autenticador en bytes.
- Autenticador (*Authenticator*): Contiene un valor criptográfico.

2.5.3. Home Address Option

La *Home address Destination Option* que se muestra en la Figura 2.23, es llevada por la cabecera de extensión *Destination Option* (*Next Header* = 60). Es usada en un paquete enviado por el nodo móvil cuando está lejos de casa para informar al receptor del paquete la dirección local del nodo móvil.

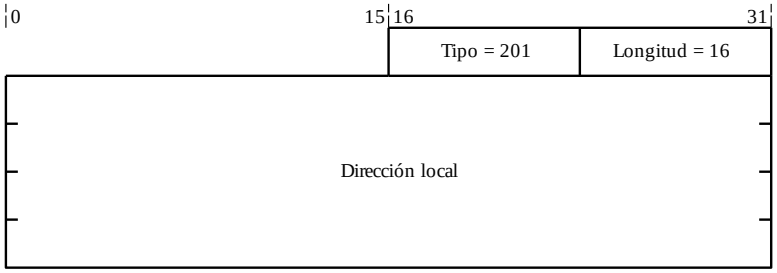


Figura 2.23 *Home Address Option*

- Dirección local (*Home Address*): Contiene la dirección local del nodo móvil que envía el paquete.

Los tres bits de orden más alto del campo Tipo de opción (*Option Type)* son codificados para determinar la acción a tomar en el nodo receptor del paquete cuando la opción *Home Address Option* no es soportada; estos tres bits son establecidos a 110. Esto indica las siguientes acciones:

- El paquete que incluye la opción debe ser descartado.
- Un mensaje *ICMPv6 Parameter Problem* debe ser enviado si la dirección de destino del paquete no es una dirección *multicast*.

Esto provee un mecanismo para detectar cuando un nodo par soporta la opción *Home Address Destination*, si no es soportada, el nodo móvil no puede utilizar el mecanismo de *Route Optimization* con ese nodo en particular.

La opción *Home Address Option* debe ser colocada de la siguiente manera:

- Después de la cabecera de enrutamiento, si esa cabecera está presente.
- Antes de la cabecera *Fragment Header*, si esa cabecera está presente.
- Antes de la cabecera AH o ESP, si alguna de estas cabeceras están presentes.

2.5.4. Type 2 Routing Header

Esta es una nueva cabecera de enrutamiento definida por IPv6 Móvil la cual se muestra en la Figura 2.24. Es usada por el agente local o el nodo correspondiente para llevar una dirección local del nodo móvil cuando los paquetes son enviados a él. La dirección de cuidado del nodo móvil es insertada en el campo dirección destino del paquete IPv6 y luego del que paquete arribe a su destino, el nodo móvil recupera su dirección local por medio de la cabecera de enrutamiento, y luego es usada como dirección de destino final para el paquete.

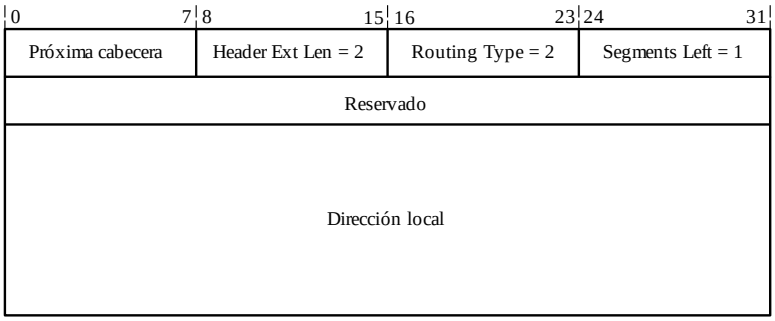


Figura 2.24 *Type 2 Routing Header*

- Próxima cabecera (*Next Header*): Identifica el tipo de cabecera que sigue inmediatamente al presente.
- Reservado: Campo reservado. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.
- Dirección local: Contiene la dirección local del nodo móvil a cual va destinado el paquete.

El *Type 2 Routing Header* es usado en los siguientes tres casos:

- Cuando el nodo móvil envía un mensaje Binding Acknowledgment
- Cuando el agente local o el nodo correspondiente hacen uso de *Route Optimization*
- Cuando el agente local envía un mensaje *Mobile Prefix Advertisement*

2.5.5. Mensajes ICMPv6

IPv6 Móvil define cuatro nuevos tipos de mensajes ICMPv6.

ICMP Home Agent Address Discovery Request

Este mensaje que se muestra en la Figura 2.25, es usado por un nodo móvil cuando desea iniciar el mecanismo *Dynamic Home Agent Address Discovery* (ver sección 2.3).

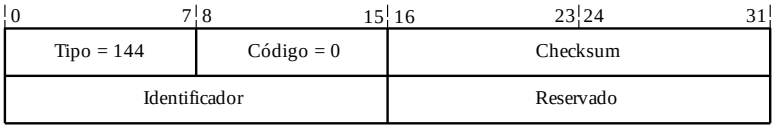


Figura 2.25 ICMP *Home Agent Address Discovery Request*

- *Checksum:* Suma de comprobación computada como es especificada por ICMPv6.
- Identificador*:* Identificador para verificar coincidencia con el respectivo mensaje de respuesta ICMP *Home Agent Address Discovery Reply*.
- Reservado: Este campo no es usado. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.

ICMP Home Agent Address Discovery Reply

Este mensaje, que se muestra en la Figura 2.26, es usado por el agente local en respuesta al nodo móvil cuando este último usa el mecanismo *Dynamic Home Agent Address Discovery* (ver sección 2.3).

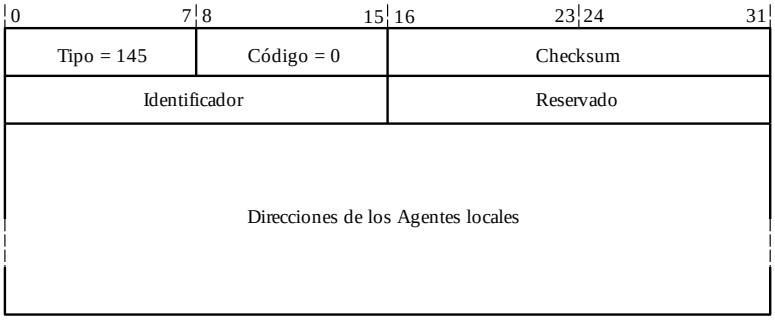


Figura 2.26 ICMP Home Agent Address Discovery Reply

- *Checksum:* Suma de comprobación computada como es especificada por ICMPv6.
- *Identifier:* Identificador copiado del respectivo mensaje de solicitud *ICMP Home Agent Address Discovery Request.*
- *Reserved:* Este campo no es usado. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.

ICMP Mobile Prefix Solicitation

Este mensaje, que se muestra en la Figura 2.28, es usado por el nodo móvil cuando desea saber la última información más actualizada sobre su red local. Este mensaje es típicamente usado para extender el tiempo de vida de su dirección local antes de que esta expire (ver sección 2.4).

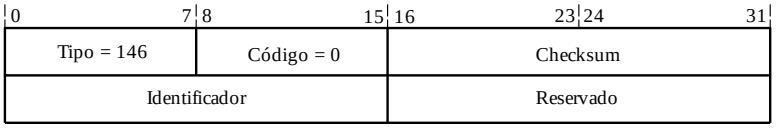


Figura 2.27 ***ICMP Mobile Prefix Solicitation***

- *Checksum:* Suma de comprobación computada como es especificada por ICMPv6.
- Identificador: Identificador para verificar coincidencia con el respectivo mensaje de respuesta *ICMP Mobile Prefix Advertisement.*
- Reservado: Este campo no es usado. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.

ICMP Mobile Prefix Advertisement

Este mensaje, que se muestra en la Figura 2.28, es usado para proporcionar al nodo móvil la información sobre su red local (ver sección 2.4).

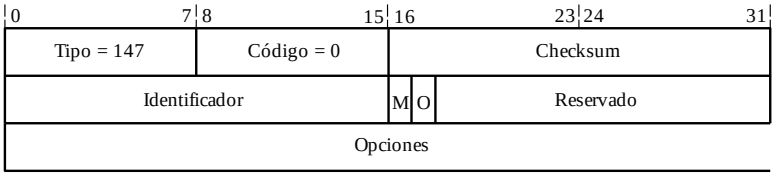


Figura 2.28 ***ICMP Mobile Prefix Advertisement***

- *Checksum*: Suma de comprobación computada como es especificada por ICMPv6.
- Identificador: Identificador copiado del respectivo mensaje de solicitud *ICMP Mobile Prefix Solicitation.*
- *Managed Address Configuration* (M): Esta bandera es encendida cuando en la red local se utiliza autoconfiguración con estado en adición con direcciones autoconfiguradas usando autoconfiguración sin estado.
- *Other Stateful Configuration* (O): Esta bandera es encendida cuando en la red local se utiliza *stateful autoconfiguration* para otra información que no son direcciones.
- Reservado: Este campo no es usado. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.
- Opciones: Las opciones de este mensaje deben seguir el formato definido en el RFC 2461. IPv6 Móvil define una opción que puede ser llevada en este mensaje (*Prefix Information Option*) que es la que proveerá de la información solicitada al nodo móvil.

2.5.6. Mensajes Neighbor Discovery

IPv6 Móvil modifica el formato de los mensajes *Router Advertisement* y *Prefix Information Option* de tal manera que se pueda distribuir la información acerca de un agente local. Además se introducen dos opciones para *Neighbor Discovery*.

Mensaje *Router Advertisement* modificado

El mensaje *Router Advertisement*, que se muestra en la Figura 2.29, es modificado de tal forma que se define una nueva bandera llamada *Home Agent* (H) que indica que el enrutador que envía el mensaje está sirviendo como agente local en el enlace. Esta información también es usada por cada agente local cuando se crea la lista de agente local del enlace en caso de que exista más de uno en el.

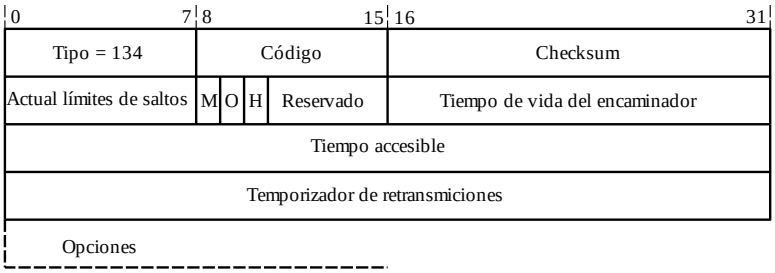


Figura 2.29 *Router Advertisement* modificado

- *Home Agent (H):* Bandera agregada para indicar que el enrutador que envía el mensaje está sirviendo como agente local en el enlace.
- Reservado*:* Campo reservado reducido de 6 bits a 5 bits para poder soportar la adición de la bandera H.

Opción *Prefix Information* modificada

Esta opción, que se muestra en la Figura 2.30, es usada en conjunto con el mensaje *Router Advertisement* para distribuir la información del prefijo a los nodos que están en la red en la que se está conectado. En IPv6 Móvil esta opción es modificada para incluir la dirección del agente local incluyendo la parte de identificador de interfaz. La bandera *Router Address* (R) es agregada para cumplir el propósito antes mencionado.

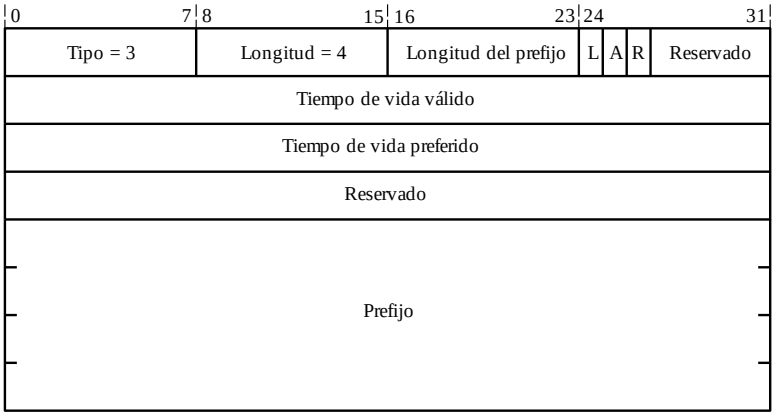


Figura 2.30 *Prefix Information Option*

- Dirección del enrutador *(R*): Si esta bandera está encendida significa que el campo Prefijo incluye una dirección completa IPV6 del agente local, no solo la parte del prefijo.

- Reservado: Campo reservado reducido de 6 bits a 5 bits para poder soportar la adición de la bandera R.

Advertisement Interval Option

Esta opción, que se muestra en la Figura 2.31, es definida y usada en conjunto con el mensaje *Router Advertisement* para anunciar el intervalo de tiempo en el cual el enrutador envía mensajes *Unsolicited Multicast Router Advertisement*.

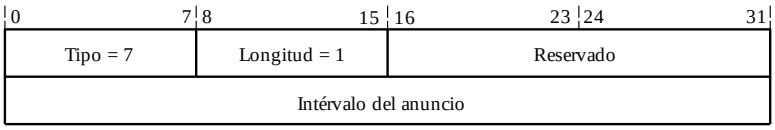


Figura 2.31 ***Advertisement Interval Option***

- Reservado: Este campo no es usado. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.
- Intervalo de anuncio *(Advertisement Interval):* Contiene el máximo tiempo entre envíos de mensaje *Unsolicited Router Advertisement* sucesivos por el presente nodo en esa interfaz de red, expresado en milisegundos.

Home Agent Information Option

Esta opción, que se muestra en la Figura 2.32, es definida para distribuir la información acerca de un agente local y es usada en conjunto con el mensaje *Router Advertisement.*

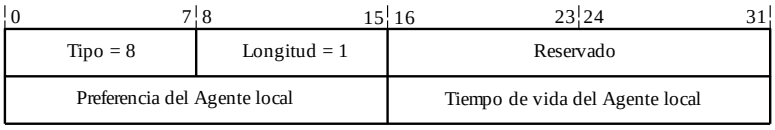


Figura 2.32 ***Home Agent Information Option***

- Reservado: Este campo no es usado. Debe ser inicializado en cero por el transmisor e ignorado por el receptor.
- Preferencia del Agente local *(Home Agent Preference):* Este campo especifica el valor de preferencia del agente local que envía la presente opción. Valores altos significan mayor preferencia.
- Tiempo de vida del Agente local *(Home Agent Lifetime):* Contiene el tiempo de vida restante hasta el cual el agente local prestara servicio.

2.6. Consideraciones de seguridad

IPv6 Móvil y sus extensiones no deberían traer nuevos problemas de seguridad dentro de la arquitectura del internet. Esta sección describe los posibles ataques que pueden realizarse contra IPv6 Móvil si las comunicaciones no son protegidas.

2.6.1. Posibles ataques contra IPv6 Móvil

A continuación se describen los posibles ataques en contra IPv6 Móvil.

a) Suplantación de identidad de los mensajes *Binding Update* (*Binding Update spoofing*)

Las comunicaciones entre un nodo móvil y su agente local son objetivos de interés para un atacante. Por defecto, los mensajes *Binding Update* no son autenticados. El atacante sólo necesita conocer la dirección local del nodo móvil con el fin de inyectar mensajes *Binding Update* falsos. Como resultado, el atacante puede hacer lo que quiera para perturbar al nodo móvil como la recuperación de su tráfico que prohíbe que se comunique, o re-direccionar el tráfico para realizar un ataque de denegación de servicio. Por esta razón, los despliegues en la vida real de IPv6 Móvil fuera de redes cerradas no se pueden hacer sin al menos la autenticación de los mensajes de señalización.

b) Inyección de tráfico

Si el túnel entre el nodo móvil y su agente local no está protegido, un atacante, que conozca la dirección local y de cuidado del nodo móvil, fácilmente podría inyectar paquetes de datos en el túnel y pretenden enviar tráfico desde la dirección local. Al igual que los mensajes de señalización, los paquetes enviados por el túnel deben ser autenticados y protegidos.

c) Return Routability Procedure

Desde su concepción, el *RRP* ha sido desarrollado para limitar los ataques de denegación de servicio contra la infraestructura de red y los nodos correspondientes. Si un atacante desea enviar un mensaje *Binding Update* falso a un nodo correspondiente, debe conocer las dos cookies de autenticación enviados en los mensajes *Home Test* y *Care-of Test*. En otras palabras, debe ser capaz de interceptar el tráfico en la red local y en la red foránea, lo cual no es una tarea fácil.

2.6.2. Protección *IPsec*

IPsec son una serie de protocolos (llamados AH, ESP e IKE) y algoritmos diseñados para asegurar comunicaciones basadas en IP. Como *IPsec* opera en capa 3, es una solución conveniente para asegurar transparentemente aplicaciones para usuarios finales y proveer servicios de seguridad para protocolos de capas superiores. *IPsec* tiene dos modos de operación: modo túnel (túneles IP-en-IP son protegidos) y modo transporte (comunicaciones extremo-a-extremo son protegidas).

Con el fin de tener protección sobre inyección de paquetes y escuchas, los mensajes de señalización, así como el túnel entre el nodo móvil y el agente local deben estar protegidos con AH y ESP como se define en el RFC 3776 [6].

Autenticación del mensaje Binding Update

La protección de *IPsec* es a menudo considerada por las operadoras de telecomunicaciones demasiado complicada para ser integrado en dispositivos móviles. Por otra parte, consideran que *IPsec* no es obligatorio en sus redes básicas si los paquetes son filtrados

eficientemente de manera interna. Como consecuencia, un mecanismo alternativo para proteger los mensajes de señalización se definió con base en IPv4 Móvil.

Se basa en una autenticación y opciones de protección de la respuesta que son llevados por el mensaje *Binding Update*. Usando estas dos opciones, el agente local puede autenticar a un nodo móvil, y evitar la inyección de falsos mensajes de señalización. Si la dirección de cuidado no se revela, esta protección es, en consecuencia, suficiente para garantizar las comunicaciones de IPv6 Móvil, si se revela, un atacante todavía puede inyectar paquetes en el túnel.

3. Metodología y Herramientas

En este capítulo se describen las fases en que se distribuyó el desarrollo de este trabajo con la finalidad de lograr los objetivos propuestos. Adicionalmente, se describen las herramientas que soportan la realización del mismo.

3.1. Fases para el desarrollo del presente trabajo

Para el desarrollo de este trabajo, se plantea la realización de las siguientes fases:

- Análisis de la solución.
- Configuración del entorno en el dispositivo.
- Desarrollo de la aplicación nativa.
- Diseño y definición de los escenarios de prueba.
- Implementación de escenarios de prueba.
- Pruebas.
- Análisis de los resultados.

3.1.1. Análisis de la solución

En esta primera fase se realiza un análisis de cómo estará compuesta la solución. Todo ello con la finalidad de determinar los requerimientos mínimos de hardware y software necesarios para poder obtener una compatibilidad entre el sistema operativo y la implementación para poder así desplegar la solución de movilidad que se desarrolla en este trabajo.

3.1.2. Configuración del entorno en el dispositivo

Para esta fase de la realización del trabajo, se dedicará el tiempo a la puesta a punto del entorno en el cual correrá la solución de movilidad desarrollada y de esta forma cumplir con los requerimientos mínimos de hardware y software necesarios para este fin. Esto es principalmente, la modificación, configuración e instalación del sistema operativo en el dispositivo así como otras configuraciones necesarias.

3.1.3. Desarrollo de la aplicación nativa

En esta tercera fase se realiza el desarrollo de una aplicación nativa de Android, la cual será una parte fundamental para la solución de movilidad presentada en este trabajo.

3.1.4. Diseño del escenario de prueba

El objetivo principal de esta fase consiste en el diseño del escenario de prueba. Se denomina escenario a la topología de una red en la cual se realizarán las pruebas de rendimiento y funcionalidad de la solución de movilidad desarrollada en el presente trabajo.

3.1.5. Implementación de escenario de prueba

Esta fase engloba todas las configuraciones de los dispositivos que forman parte de la topología diseñada en la fase previa de forma tal que se puedan interconectar acordemente a como está diseñado el escenario. Básicamente consiste en la configuración de los enrutadores, puntos de acceso, agente local, nodo correspondiente y nodo móvil.

3.1.6. Pruebas

La finalidad de esta fase es la escogencia y realización de un conjunto de pruebas soportadas sobre el escenario implementado en la fase anterior, las cuales serán necesarias para la evaluación de la solución desarrollada en este trabajo.

3.1.7. Análisis de los resultados

Una vez recopilados los resultados consecuentes de las pruebas realizadas en la fase anterior, estos serán interpretados y examinados de tal forma para poder llegar a unas conclusiones que demuestren si se cumplieron o no los objetivos planteados en el presente trabajo. Este análisis será crucial a la hora de determinar si fue un existo o no la solución de movilidad desarrollada.

3.2. Herramientas utilizadas para el desarrollo del trabajo

Para la elaboración del presente trabajo hizo uso de distintas herramientas de software y hardware que se especifican a continuación.

3.2.1. Software

A continuación se describen las herramientas usadas para el desarrollo de este trabajo:

- **Android:** es un sistema operativo para dispositivos móviles (teléfonos inteligentes y tabletas) basado en el *kernel* de Linux que está siendo desarrollado por Google con la colaboración de la *Open Handset Alliance,* el cual está conformado por un conjunto de fabricantes y desarrolladores de hardware y software los cuales son los que hacen uso de este sistema operativo.

  Android se desarrolla en forma abierta y se puede acceder al código fuente, no solo del *kernel* por ser Linux sino de todo el sistema operativo y aplicaciones por defecto. Está disponible bajo licencia Apache [9].

- **Android SDK / NDK / ADT:** El SDK (*Software Development Kit*) de Android incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. La plataforma integral de desarrollo soportada oficialmente es Eclipse junto con el complemento ADT (*Android Development Tools plugin*).

El NDK (*Native Development Kit*) permite instalar bibliotecas escritas en C y otros lenguajes de programación, una vez compiladas para ARM. Los programas Java corriendo en la máquina virtual Dalvik (Dalvik VM) pueden llamar a clases nativas por medio de la función *System.loadLibrary*, que forma parte de las clases estándar Java en Android [10].

- **Eclipse:** es un entorno de desarrollo integrado multilenguaje de código abierto multiplataforma con soporte para un sistema extensible de plug-in [11].

- **Android Linux *kernel*:** es un núcleo libre de sistema operativo basado en Unix que es utilizado por Android en el cual específicamente incluye soporte para dispositivos móviles que corren Android [12].

- **UMIP:** (*USAGI-patched Mobile IPv6 for Linux*) es una implementación del protocolo de movilidad IPv6 Móvil para los sistemas operativos GNU/Linux desarrollada y mantenida por distintos grupos de trabajo a lo largo de su existencia, entre esos por el proyecto USAGI/WIDE y actualmente por una comunidad de desarrolladores voluntarios UMIP.org el cual integró el soporte para NEMO (*Network Mobility*), nuevas características, corrección de errores y simplificación del código [13].

- **ARM *Toolchain*:** es un conjunto de herramientas de software que se usan para crear un programa o sistema informático. Específicamente una serie de herramientas que permiten la compilación y construcción de software para la arquitectura ARM [13].

- **Su** (***s****ubstitute* ***u****ser*)**:** es una utilidad de los sistemas operativos del tipo Unix que permite usar el intérprete de comandos de otro usuario sin necesidad de cerrar la sesión. Comúnmente se usa para obtener permisos de *root* para operaciones administrativas, sin tener que salir y re ingresar al sistema [15].

- **BusyBox:** es un programa que combina muchas utilidades estándares de Unix en un solo ejecutable pequeño. Es capaz de proveer la mayoría de las utilidades que están especificadas para los sistemas Unix además de muchas de las utilidades que esperas ver en los sistemas GNU/Linux. Busybox es utilizada normalmente en sistemas con Linux embebido [16].

- **Odin:** es un software utilizado para realizar un reprogramado o reescritura (*Flash*) de un firmware en la memoria ROM de los dispositivos Samsung Galaxy S, interactuando con ellos mediante el modo de recuperación "*downloading mode*" vía USB. Una alternativa a este software es Heimdall [17].

- **VMware VSphere:** es un sistema operativo basado en computación en la nube que contiene una serie de herramientas para realizar virtualización de otros sistemas operativos, desarrollado por VMware Inc [17].

- **Debian GNU/Linux:** es un sistema operativo basado en Linux. Debian GNU/Linux incluye las herramientas GNU y el *kernel* de Linux [19].

- **Iperf:** es una herramienta que se utiliza para realizar un análisis del desempeño de las redes, la cual puede generar flujos de datos tanto en TCP y UDP[1].

- **Wireshark:** es un analizador de paquetes y protocolos. Utilizado para realizar análisis, solución de problemas en redes de comunicación y desarrollo de software y protocolos de comunicación [20].

3.2.2. Hardware

A continuación se describen los distintos dispositivos de hardware utilizados para la elaboración de este trabajo:

- **Teléfono inteligente Samsung Galaxy S GT-i9000:** es un teléfono móvil inteligente de gama alta producido por Samsung Electronics. Es un dispositivo el cual usa el sistema operativo Android con pantalla táctil, soporte de WiFi y 3G [21].

- **AP CISCO AIRONET 1200:** es un punto de acceso inalámbrico producido por CISCO [21].

[1] http://sourceforge.net/projects/iperf/

4. Implementación de la Solución

En este capítulo se describirá el desarrollo del trabajo siguiendo un orden coherente de tareas realizadas para lograr los objetivos planteados.

4.1. Análisis de la solución

La idea principal de la solución para la movilidad planteada en este trabajo es la de proveer soporte del protocolo IPv6 Móvil a los dispositivos móviles, teléfonos inteligentes y tabletas, corriendo el sistema operativo móvil Google Android, a la vez de proveer facilidad para un usuario de estos dispositivos de realizar la configuración del protocolo y el manejo de él. En la Figura 4.1 se muestra el esquema de la arquitectura de la solución, donde se puede apreciar en líneas punteadas, los elementos claves del sistema Android que se debieron desarrollar, modificar, adaptar y configurar de tal forma que permitió poner en marcha dicha solución.

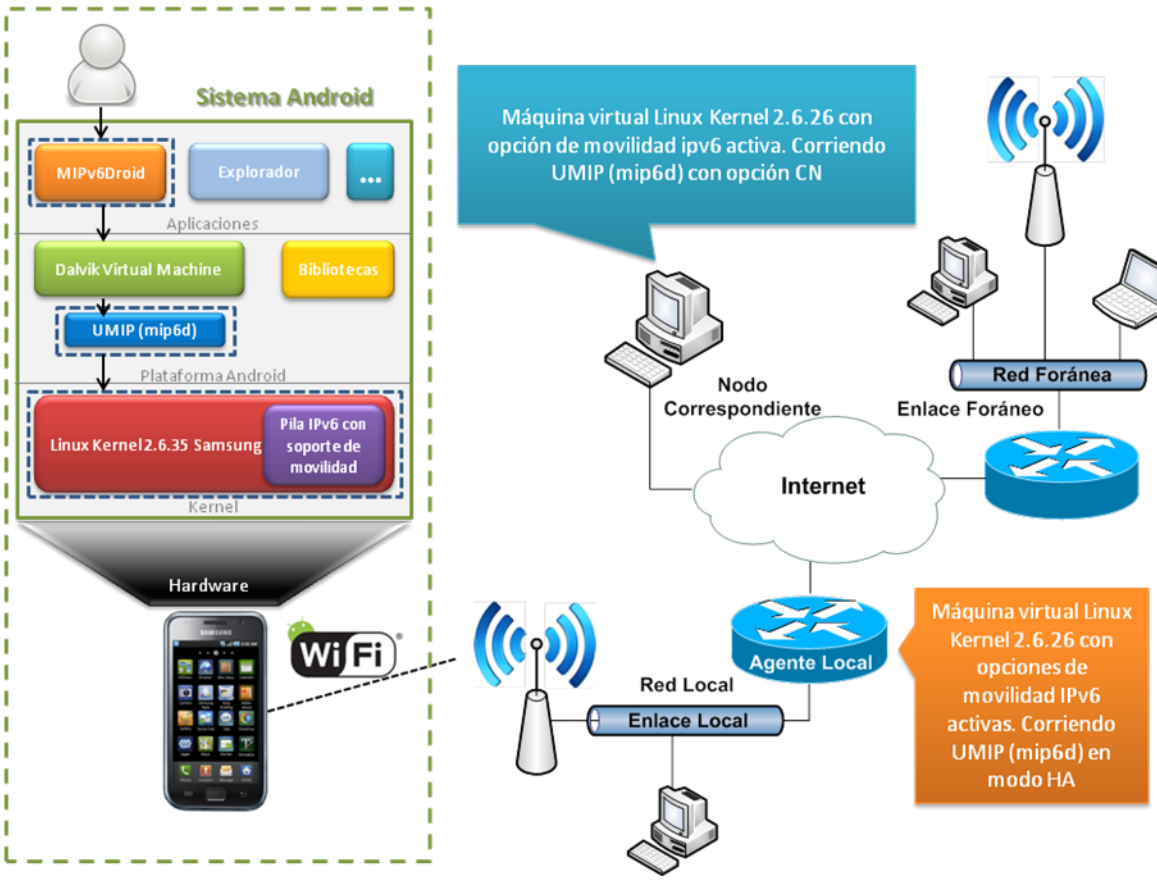


Figura 4.1 Arquitectura de la solución

En las secciones siguientes se explicará más detalladamente los pasos que se debieron seguir para lograr la solución planteada en el esquema antes mencionado.

La razón por la que se decidió realizar la solución de movilidad bajo el sistema Android, viene de una investigación previa [23] profunda que se realizó sobre las distintas plataformas

disponibles para los dispositivos móviles de la actualidad, donde se encontró sumamente conveniente hacer uso del sistema operativo Android. La razón de esta decisión viene de la mano de sus características, la naturaleza de su arquitectura y su posicionamiento en el mercado actual. Adicionalmente un factor influyente fue el resultado de la investigación sobre las implementaciones del protocolo MIPv6, dado que la única implementación libre llamada UMIP que aun tiene soporte y sigue en constante desarrollo, puede correr sobre Android.

4.2. Configuración del entorno en el dispositivo

El dispositivo que se uso en el desarrollo del presente trabajo es un Samsung Galaxy S GT-I9000 corriendo el sistema operativo Google Android 2.3.3 Gingerbread (*build* XWJVB) con versión del *kernel* 2.6.35. Para configurar el entorno del sistema del dispositivo para que pueda correr la solución de movilidad, fue necesario realizar una serie de pasos los cuales son descritos a continuación:

- Compilación del *kernel* de Android.
- Instalación del *kernel* compilado en el dispositivo.
- Compilación de la implementación del protocolo IPv6 Móvil “UMIP”.

4.2.1. Compilación del *kernel* de Android

Para poder implementar la solución en el dispositivo antes mencionado fue necesario realizar una modificación al *kernel* de Android. Esto es debido a que solo de esta forma se pudo dar soporte a la pila de protocolos de red, las nuevas cabeceras y definición de mensajes que el protocolo IPv6 móvil introduce para su funcionamiento. Afortunadamente, Android hace uso del *kernel* de Linux el cual ya tiene implementado el soporte del protocolo IPv6 Móvil a partir de su versión 2.6.x, por consiguiente solo hizo falta realizar una re compilación del mismo, habilitando el soporte del protocolo ya que por defecto no lo trae. Adicionalmente en este paso se procedió a habilitar el acceso a ejecución de comandos en modo súper usuario para los usuarios regulares al sistema, el cual viene deshabilitado por defecto en Android

Es necesario destacar que los dispositivos móviles basados en Android, por lo general y es el caso específico del GT-I9000, cuentan con procesadores con arquitectura ARM (*Advanced RISC Machine*), por lo que cualquier código que se desea que corra en dicho dispositivo debe ser compilado específicamente para la antes mencionada arquitectura. Como no es posible realizar la compilación de código fuente nativamente en el dispositivo por falta de herramientas y por incomodidad de trabajar en dicho dispositivo fue necesario hacer una compilación cruzada (*Cross Compile*) desde un sistema huésped con arquitectura x86 (Estación de trabajo corriendo Linux). La compilación cruzada se puede definir como la compilación de código fuente, realizada bajo una determinada arquitectura que genera código ejecutable para una arquitectura diferente a la que se está usando.

Para poder contar con un *kernel* compilado con el soporte de IPv6 Móvil se debieron seguir los siguientes pasos:

1. Configuración del sistema huésped para poder realizar compilación cruzada a arquitectura ARM: fue necesario realizar una configuración del sistema huésped, esto es, descarga e instalación de las herramientas de compilación cruzada para arquitectura ARM y configuración de variables de entorno del sistema necesarias para la posterior compilación.

2. Descarga del código fuente del *kernel* específico para el dispositivo que se utilizó: aunque el *kernel* de Android se encuentra basado en Linux, fue necesaria la ubicación y descarga del código fuente del *kernel* específico para el dispositivo que se utilizó, esto es debido a que el *kernel* debe tener soporte de los controladores específicos para los elementos de hardware con el que el dispositivo cuenta.

3. Habilitación de las opciones del *kernel* necesarias para dar soporte del protocolo IPv6 Móvil: las opciones específicas del *kernel* para habilitar el soporte del protocolo IPv6 Móvil, se encuentran deshabilitadas por defecto. Por esta razón se procedió a su habilitación para que sean incluidas posteriormente en la imagen final del *kernel* compilado.

4. Configuración de un disco RAM inicial personalizado el cual sirvió para dar permiso de ejecución de comandos en modo súper usuario: fue necesario crear una imagen personalizada del disco RAM inicial que es usada al momento de arranque del sistema, incluyéndole ciertos binarios y scripts dentro. Esto es debido a que de esta forma se pudo dar permiso de ejecución de comandos en modo súper usuario. Lo cual es vital para el correcto funcionamiento de la implementación del protocolo IPv6 Móvil

5. Compilación del *kernel*: en este paso se realizó la compilación efectiva del *kernel*, generando a su finalización la imagen final la cual fue posteriormente instalada en el dispositivo.

6. Empaquetado del *kernel:* este último paso se refiere al empaquetado de la imagen del *kernel* de la forma en el que el dispositivo admite al momento de su instalación.

El Anexo N° 1 muestra de forma detallada la ejecución de cada uno de los pasos anteriores.

4.2.2. Instalación del *kernel* compilado en el dispositivo.

La instalación del *kernel* en el dispositivo fue una tarea sumamente delicada ya que si no se realizaba correctamente podía ocasionar un daño irreparable en dicho dispositivo dejándolo inoperable. Afortunadamente Samsung ha provisto de un modo de recuperación al dispositivo llamado *Downloading Mode,* que sirve para solventar dichas circunstancias.

Para la instalación del *kernel* compilado en el dispositivo, se uso de un software llamado Odin v1.30[2] [3]. Este software que corre sobre la plataforma Windows, contiene utilidades de recuperación y de reescritura de las imágenes del sistema en el firmware de los dispositivos Samsung. Una vez descargado y ejecutado Odin, se puso al dispositivo en el modo *Downloading Mode* y se procedió a la rescritura del *kernel* con el nuevo *kernel* compilado.

El Anexo N° 2 describe de forma detallada la ejecución de cada uno de los pasos anteriores.

4.2.3. Compilación cruzada de UMIP (demonio mip6d)

La solución de movilidad del presente trabajo, hace uso de la implementación UMIP. Al igual de cómo se hace con el *kernel* del dispositivo, el código fuente de UMIP debe ser compilado para la arquitectura ARM.

Hay que destacar que *UMIP* hace uso de llamadas a bibliotecas del sistema en tiempo de ejecución. Android no cuenta con las bibliotecas estándar GNU Glibc que cuentan los sistemas GNU/Linux en general, por su parte Android dispone de una biblioteca reducida para sistemas embebidos que suplanta a Glibc llamada Bionic. Desafortunadamente UMIP hace uso de funciones que no se encuentran en Bionic, por lo cual es necesario realizar la compilación del software de modo estático incluyendo las bibliotecas estándar Glibc dentro del binario resultante para evitar problemas en tiempo de ejecución al no poderse realizar el enlace dinámico con las bibliotecas que no encontraría presentes en el sistema.

Para realizar la compilación de *UMIP* se debieron seguir los siguientes pasos:

1. Configuración del sistema huésped para poder realizar compilación cruzada a arquitectura ARM. AL igual que cuando se realizó la configuración cuando se realizó la compilación del *kernel,* se debió hacer lo propio al momento de realizar la compilación cruzada de la implementación UMIP.

2. Descarga del código fuente: este paso fue tan sencillo como realizar un clonado de los repositorios GIT donde se encuentra hospedado el código fuente hacia nuestro sistema de trabajo.

3. Configuración de la compilación: como se explicó antes en esta sección, Android hace uso de bibliotecas distintas a las que utilizan los sistemas Linux estándar. Por lo que se debió configurar la compilación de tal forma de que se incluyeran las bibliotecas Glibc estándares que utiliza UMIP dentro del binario resultante de la compilación (enlace estático). Adicionalmente otras opciones necesarias para su correcto funcionamiento en el dispositivo.

[2] http://forum.xda-developers.com/wiki/Samsung_Galaxy_S_Series
[3] http://forum.xda-developers.com/showthread.php?t=1467128

4. Compilación del código fuente: este último paso se refiere al proceso de compilación como tal, generando como resultado un binario compatible con la arquitectura ARM.

La instalación del UMIP en el dispositivo, se realizó incluyéndolo automáticamente en el paquete de la aplicación nativa como se describirá en la siguiente sección.

El Anexo N° 3 describe de forma detallada la ejecución de cada uno de los pasos anteriores.

5. Desarrollo de la aplicación nativa

En este capítulo se describirá el desarrollo de la aplicación MIPv6Droid siguiendo un orden lógico de pasos: análisis, levantamiento de requerimientos, modelado e implementación.

5.1. Análisis de los requerimientos del sistema

El desarrollo de un sistema comienza por analizar sus requerimientos. Para establecer los requerimientos del sistema, tanto funcionales como no funcionales, analizamos los posibles problemas que podrían enfrentar el usuario final al interactuar con la aplicación. Además se identificaron, clasificaron y priorizaron los requerimientos que reflejan los objetivos del sistema. A continuación se listan los requerimientos establecidos con la finalidad de cumplir con las características deseadas:

- Desarrollar una aplicación con el API de la plataforma Android para manejar el demonio de una manera más sencilla y usable. La aplicación debe ser lo más estándar posible para correr en la mayoría de dispositivos Android sin importar la versión del sistema operativo, independientemente del fabricante o capacidad del dispositivo. Esto con la finalidad de no obligar a utilizar un dispositivo específico.

- El sistema debe tener la capacidad de iniciar y detener el demonio mip6d sin tener que utilizar comandos de consola. El demonio mip6d estará siendo manejado por la aplicación sin la necesidad de utilizar una consola como medio para la inicialización del demonio.

- El sistema debe tener la capacidad de manejar direcciones IPv6. El *kernel* del sistema operativo Android soporta direcciones IPv6 sin embargo el API no soporta el manejo de direcciones IPv6.

- El sistema debe tener la capacidad de verificar opciones del *kernel*. Como se menciona en el capítulo anterior para poder correr mip6d en espacio de usuario es necesario un *kernel* que tengan las opciones de movilidad activas por lo tanto el sistema debe tener la capacidad de verificar si las opciones del *kernel* están activas.

- El sistema debe tener la capacidad de verificar si el sistema Android tiene permisos de correr comandos en modo privilegiado.

- El sistema debe tener la capacidad de configurar todas las opciones del demonio tanto para el NM como el CN. El sistema debe proveer una interfaz que pueda configurar todas las opciones del demonio sin tener que modificar directamente un archivo de configuración. La idea es que la configuración del demonio sea sencilla y que sea asistida por el sistema.

- El sistema debe permitir utilizar un archivo de configuración distinto al archivo que genera el sistema. El sistema debe proveer un módulo para explorar en el almacenamiento externo (*sdcard*) y poder elegir un archivo de configuración distinto al que genera el sistema en el caso que el usuario lo desea.

- El sistema debe permitir mostrar el *Debug* del demonio de una forma agradable. Debe proveer la capacidad de retener por un tiempo la salida del *Debug* del sistema para que el usuario sepa lo que está pasando. También realizar una exportación del mismo para permitir ser analizado posteriormente por el usuario si así lo requiere.

- Se debe crear una interfaz usable que permita al usuario trabajar fácilmente, y tratando que cometa la menor cantidad de errores, además de proporcionarle una fácil navegación por la aplicación. Simultáneamente, la aplicación debe tener la característica de fiabilidad para proporcionar la capacidad del buen funcionamiento de sus funcionalidades.

5.2. Modelado de la aplicación MIPv6Droid

El modelado de una aplicación se desarrolla para comprender el sistema que se va a desarrollar. Para modelar, analizar y diseñar sistemas se utiliza un conjunto de herramientas, llamado UML (*Unified Modeling Language*, Lenguaje Unificado de Modelado). Entre las herramientas que proporciona UML se encuentran: el diagrama de casos de usos, diagrama de clases, diagrama de estados, diagrama de secuencias, entre otros. Para el modelo de esta la aplicación se van a utilizar los diagramas de casos de usos y el diagrama de secuencia.

5.2.1. Diagramas de casos de uso

Los casos de usos forman parte del análisis del sistema, modelando sus funcionalidades, ayudando a describir las tareas del sistema y define cómo el usuario interactúa con el sistema.

5.2.2. Casa de uso - Nivel 0

En este nivel se modela de forma generalizada la iteración del usuario con el sistema como se muestra en la Figura 5.1.

Usuario: El usuario se comunica directamente con el sistema para administrar a MIPv6Droid.

Figura 5.1 Diagrama de caso de uso - nivel 0

5.2.3. Caso de uso - Nivel 1

Este caso refleja en términos general la interacción de los actores con el sistema como se muestra en la Figura 5.2. Se puede notar que el usuario interactúa con todas las opciones del sistema. La descripción del funcionamiento del caso de uso del nivel 1 se expone en las Tablas Tabla 5.1 a la Tabla 5.7, indicando todos los elementos que interactúan con el usuario en cada caso de uso.

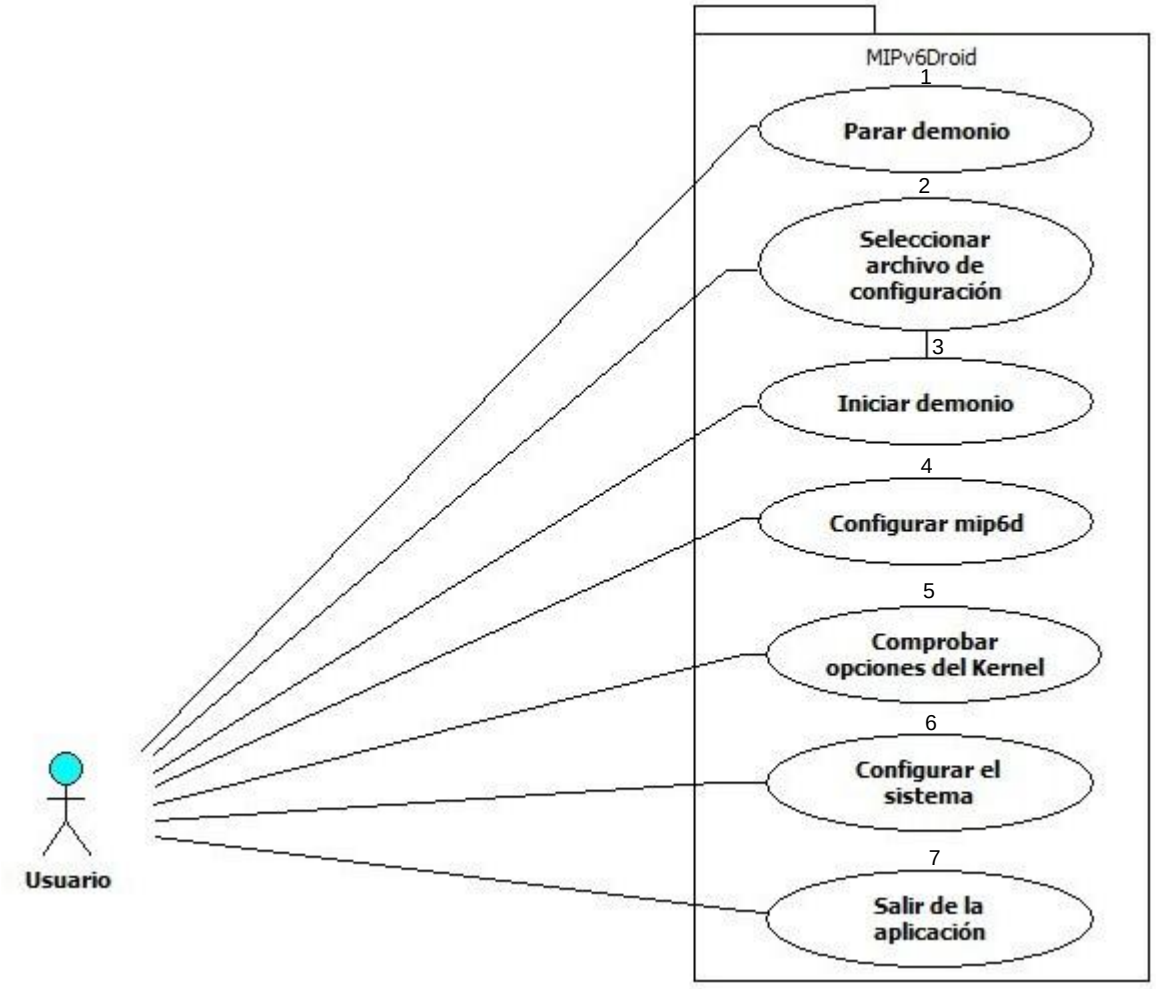


Figura 5.2 Diagrama de caso de uso - nivel 1

Caso de uso 1 (Parar demonio)	
Nombre	Parar demonio
Actor	Usuario Móvil
Descripción	Permite finalizar la ejecución del demonio mip6d.
Pre-condición	El demonio debe estar en ejecución.
Flujo básico	El usuario finaliza la ejecución del demonio. Esta función se realiza enviado el comando para la finalizar la ejecución del demonio y se muestra por

	pantalla el *debug* de la finalización del demonio.
Post-condición	La opción iniciar demonio es activada.

Tabla 5.1 Caso de uso 1 (Parar demonio)

Caso de uso 2 (Seleccionar archivo de configuración)	
Nombre	Seleccionar archivo de configuración
Actor	Usuario Móvil
Descripción	Permite seleccionar un archivo de configuración diferente al creado por el sistema MIPv6Droid.
Pre-condición	La *sdcard* debe estar desmontada.
Flujo básico	El usuario selecciona un archivo de configuración que se encuentra en la memoria interna del dispositivo desde un explorador.
Post-condición	Se configura la ruta donde se encuentra el archivo de configuración.

Tabla 5.2 caso de uso 2 (Seleccionar archivo de configuración)

Caso de uso 3 (Iniciar demonio)	
Nombre	Iniciar demonio
Actor	Usuario Móvil y Demonio mip6d
Descripción	Permite iniciar la ejecución demonio mip6d.
Pre-condición	Debe haber un archivo de configuración definido, el sistema debe tener la capacidad de iniciar aplicaciones con privilegios de administrador y las opciones de movilidad del *kernel* deben estar activas.
Flujo básico	El usuario ejecuta el demonio. Es enviado el comando para la ejecución y se muestra por pantalla el *debug* del demonio.
Post-condición	El sistema puede utilizar todas las ventajas que le ofrece el demonio mip6d. La opción iniciar demonio es desactivada.

Tabla 5.3 caso de uso 3 (Iniciar demonio)

Caso de uso 4 (Configurar mip6d)	
Nombre	Configurar mip6d
Actor	Usuario Móvil
Descripción	Permite configurar todas las opciones del mip6d en una interfaz más usable.
Pre-condición	Ninguna.
Flujo básico	Al activar esta opción el sistema despliega un menú con las siguientes todas las opciones para configurar mip6d
Post-condición	

Tabla 5.4 Caso de uso 4 (Configurar mip6d)

Caso de uso 5 (Comprobar opciones del *kernel*)	
Nombre	Comprobar opciones del *kernel*
Actor	Usuario Móvil
Descripción	Permite comprobar si las opciones de movilidad del *kernel* están activas.

Pre-condición	Debe existir el archivo *config.gz* en el sistema.
Flujo básico	Al activar esta opción el sistema chequea que dentro del archivo config.gz se encuentran las opciones de movilidad del *kernel* activas.
Post-condición	Si las opciones de movilidad del *kernel* están activadas, el demonio puede ejecutarse correctamente en el caso contrario el demonio no se ejecutará.

Tabla 5.5 Caso de uso 5 (Comprobar opciones del *kernel*)

Caso de uso 6 (Configurar el Sistema)	
Nombre	Configurar el Sistema
Actor	Usuario Móvil
Descripción	Permite configurar el sistema MIPv6Droid.
Pre-condición	Ninguna.
Flujo básico	Al activar esta opción el sistema despliega un menú con una serie de opciones.
Post-condición	Una vez desplegadas las opciones del menú el usuario tiene la capacidad de elegir cualquiera de ellas.

Tabla 5.6 Caso de uso 6 (Configurar el sistema)

Caso de uso 7 (Salir de la aplicación)	
Nombre	Salir de la aplicación
Actor	Usuario Móvil
Descripción	Permite terminar la aplicación MIPv6Droid
Pre-condición	Ninguna.
Flujo básico	Al activar esta opción el sistema se cierra.
Post-condición	Puede que el sistema se cierre en modo *Background*.

Tabla 5.7 Caso de uso 7 (Salir de la aplicación)

5.2.4. Caso de uso - Nivel 2

En este nivel se muestran las funcionalidades específicas del sistema como se muestra en la Figura 5.3. Las funcionalidades de este nivel se describen en las Tabla 5.8 a la Tabla 5.11.

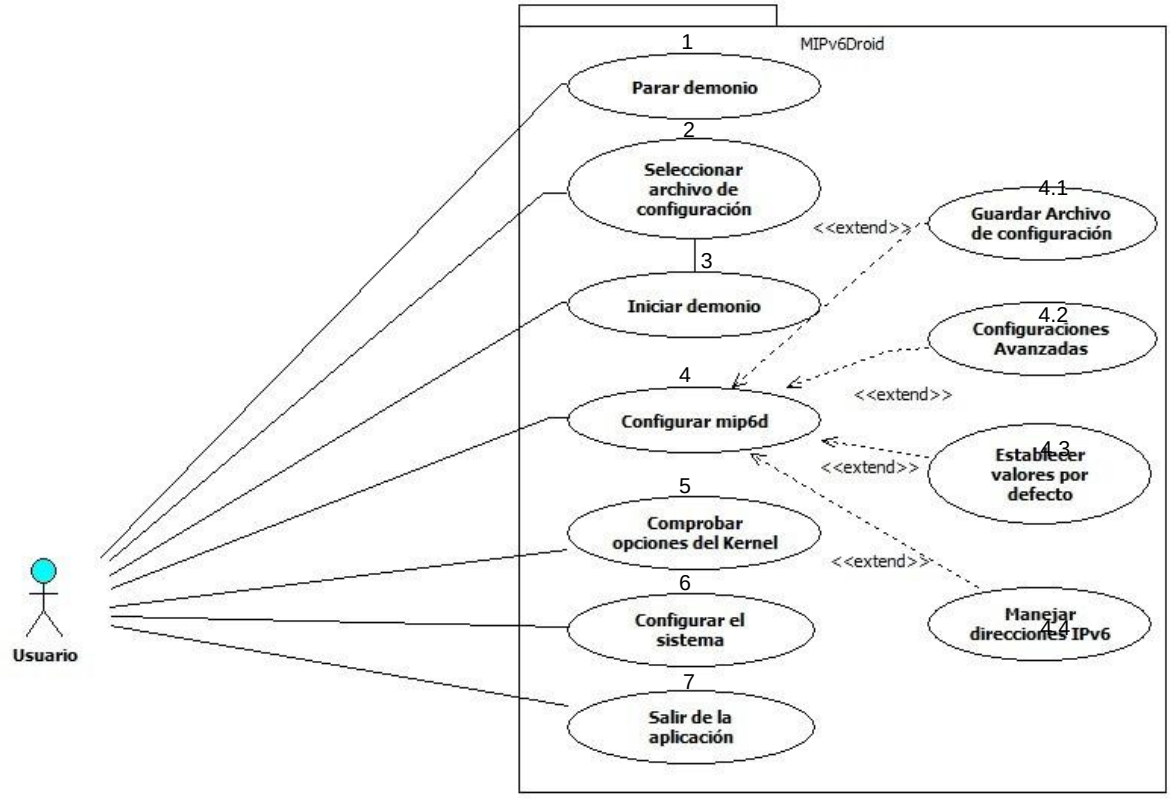


Figura 5.3 Diagrama de caso de uso – nivel 2

Caso de uso 4.1 (Guardar archivo de configuración)	
Nombre	Guardar archivo de configuración
Actor	Usuario Móvil
Descripción	Permite guardar el archivo de configuración manejado por la aplicación MIPv6Droid.
Pre-condición	Debe existir alguna configuración.
Flujo básico	Una vez configurada las opciones del sistema se debe guardar el archivo de configuración para que los cambios tengan efecto.
Post-condición	Este archivo de configuración es utilizado para configurar del demonio mip6d, si el usuario lo desea.

Tabla 5.8 Caso de uso 4.1 (Guardar archivo de configuración)

Caso de uso 4.2 (Configuraciones Avanzadas)	
Nombre	Configuraciones Avanzadas
Actor	Usuario Móvil
Descripción	Permite configurar las configuraciones avanzadas del demonio mip6d.
Pre-condición	El usuario debe estar dentro de la configuración de mip6d y la opción del modo debe ser MN.
Flujo básico	El usuario dentro de la configuración del mip6d puede acceder dentro de las

	configuraciones avanzadas del demonio en el caso que lo desee.
Post-condición	Ninguna

Tabla 5.9 Caso de uso 4.2 (Configuraciones Avanzadas)

Caso de uso 4.3 (Establecer valores por defecto)	
Nombre	Establecer valores por defecto
Actor	Usuario Móvil
Descripción	Permite establecer los valores por defecto de la configuración del demonio.
Pre-condición	La configuración actual debe ser distinta a la configuración por defecto.
Flujo básico	El usuario al seleccionar esta opción se establece los valores por defecto de la configuración del demonio.
Post-condición	Ninguna

Tabla 5.10 Caso de uso 4.3 (Establecer valores por defecto)

Caso de uso 4.4 (Manejar direcciones IPv6)	
Nombre	Manejar direcciones IPv6
Actor	Usuario Móvil
Descripción	Permite la administración de las direcciones IPv6 del sistema.
Pre-condición	El sistema debe de poseer la pila IPv6 activo.
Flujo básico	El usuario pulsa el botón manejar direcciones IPv6 y se desplegará una ventana con las opciones para administrar las direcciones.
Post-condición	Las direcciones agregadas pueden ser utilizadas para configurar el demonio mip6d.

Tabla 5.11 Caso de uso 4.4 (Manejar direcciones IPv6

5.2.5. Caso de uso - Nivel 3

En este nivel se muestra las funcionalidades más específicas del sistema como se muestra en la Figura 5.4. Los casos de uso de este nivel, son descritos en las tablas 6.10, 6.11 y 6.12.

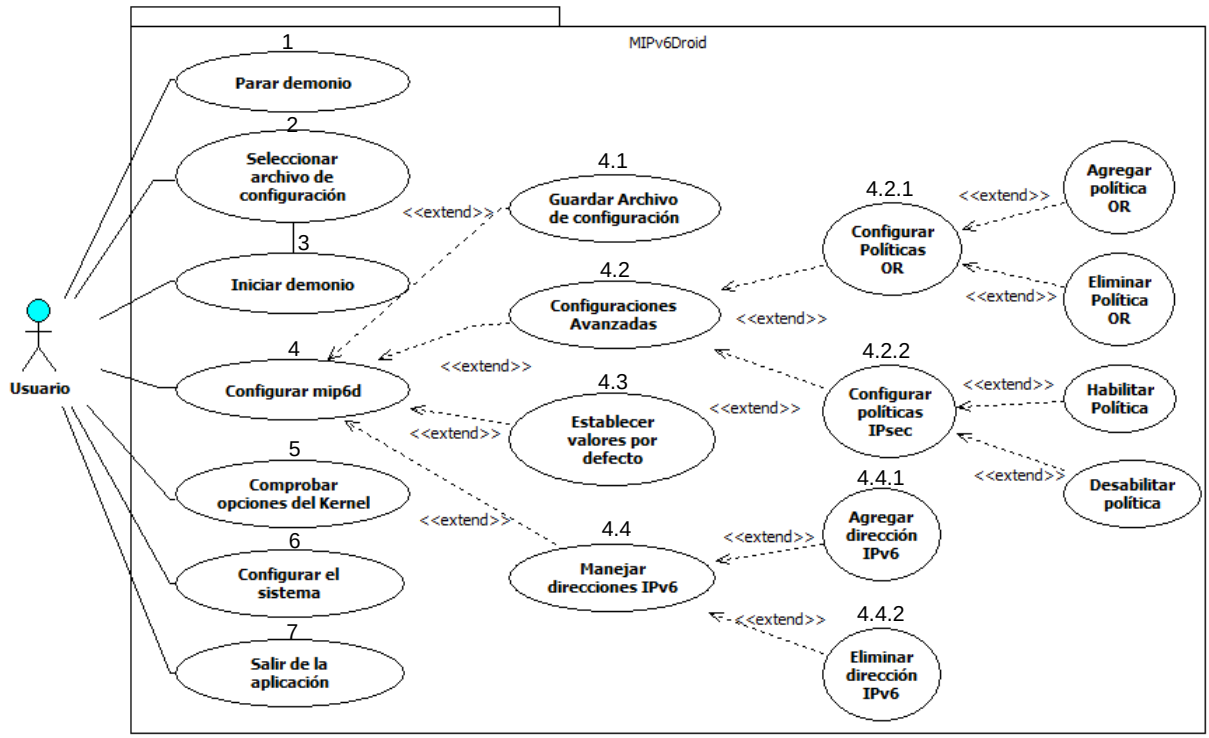


Figura 5.4 Diagrama de caso de uso – nivel 3

Caso de uso 4.2.1 (Configurar políticas OR)	
Nombre	Configurar políticas OR
Actor	Usuario Móvil
Descripción	Permite configurar las opciones de optimización de rutas para nodos correspondientes.
Pre-condición	Ninguna
Flujo básico	El usuario pulsa el botón configurar políticas OR y se desplegará una ventana con las siguientes opciones: agregar política y eliminar política.
Post-condición	Para que la política tenga efecto debe de estar activa.

Tabla 5.12 Caso de uso 4.2.1 (Configurar políticas OR)

Caso de uso 4.2.2 (Configurar políticas *IPsec*)	
Nombre	Configurar políticas *IPsec*
Actor	Usuario Móvil
Descripción	Permite configurar las opciones de *IPsec*.
Pre-condición	Deben de estar configuradas la dirección local y la dirección del agente local.
Flujo básico	El usuario pulsa el botón configurar políticas *IPsec* y se desplegará una ventana con todas las políticas permitidas. Al seleccionar alguna se podrá activar o desactivar y configurar.
Post-condición	Ninguna

Tabla 5.13 Caso de uso 4.2.2 (Configurar políticas IPsec)

Caso de uso 4.4.1 (Agregar dirección IPv6)	
Nombre	Agregar dirección IPv6
Actor	Usuario Móvil
Descripción	Agrega una dirección IPv6 al sistema.
Pre-condición	El sistema debe tener activo la pila IPv6.
Flujo básico	El usuario pulsa el botón agregar dirección y le muestra un dialogo donde se coloca la dirección IPv6 con su respetiva mascara.
Post-condición	Las direcciones agregadas pueden ser utilizadas para configurar el demonio mip6d.

Tabla 5.14 Caso de uso 4.4.1 (Agregar dirección IPv6)

Caso de uso 4.4.2 (Eliminar dirección IPv6)	
Nombre	Eliminar dirección IPv6
Actor	Usuario Móvil
Descripción	Permite eliminar una dirección IPv6 del sistema operativo.
Pre-condición	Debe existir una dirección IPv6 configurada.
Flujo básico	El usuario selecciona la opción eliminar dirección y se listan todas las direcciones configuradas al seleccionar la deseada la elimina.
Post-condición	Ninguna

Tabla 5.15 Caso de uso 4.4.2 (Eliminar dirección IPv6)

5.2.6. Diagrama de secuencia

El diagrama de secuencia, es un diagrama UML que muestra la iteración de los objetos para visualizar, especificar, construir y documentar los aspectos dinámicos de un sistema. Los diagramas de secuencia nos ayudan a modelar la comunicación entre los objetos y los mensajes que inicializan esas comunicaciones durante el tiempo.

En la Figura 5.5 se muestra cómo el usuario interactúa con el sistema al momento de inicializar el demonio mip6d.

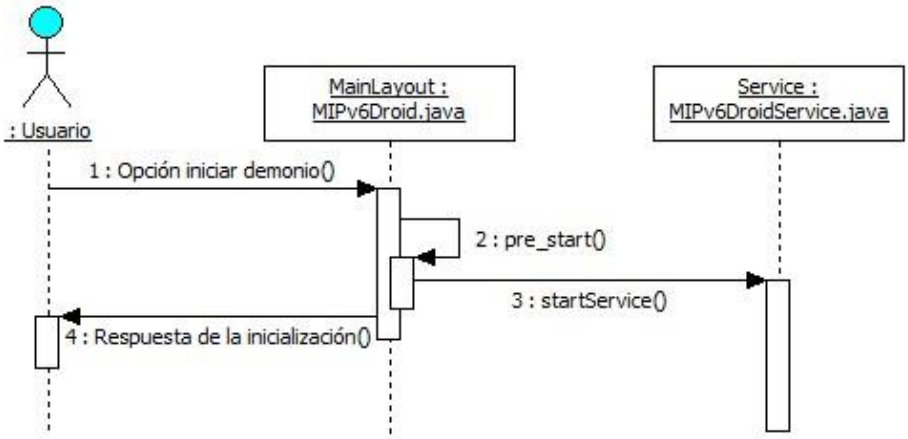


Figura 5.5 Diagrama de secuencia del caso de uso iniciar demonio

En la Figura 5.6 se muestra cómo el usuario interactúa con el sistema al momento de para el demonio mip6d.

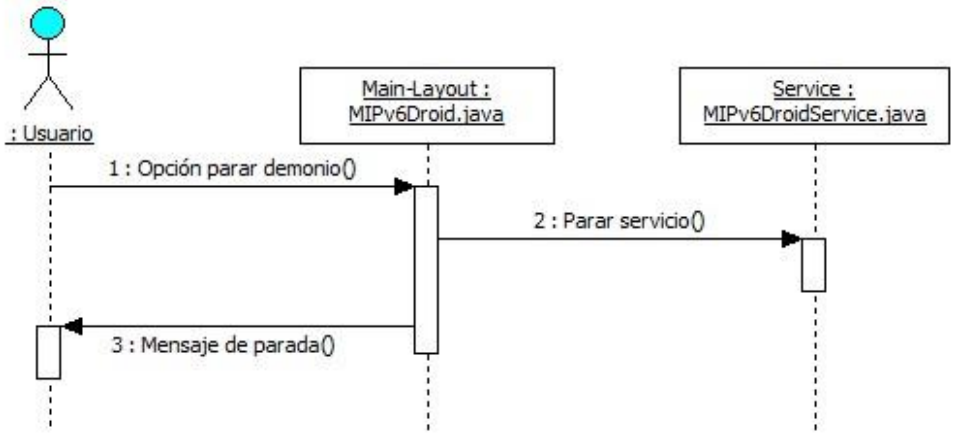


Figura 5.6 Diagrama de secuencia del caso de uso parar demonio

Para el caso de uso de configuración de mip6d, se modeló el diagrama de secuencia que se muestra en la Figura 5.7, mostrando los diversos objetos o instancias de clases involucradas en el proceso de configuración de mip6d.

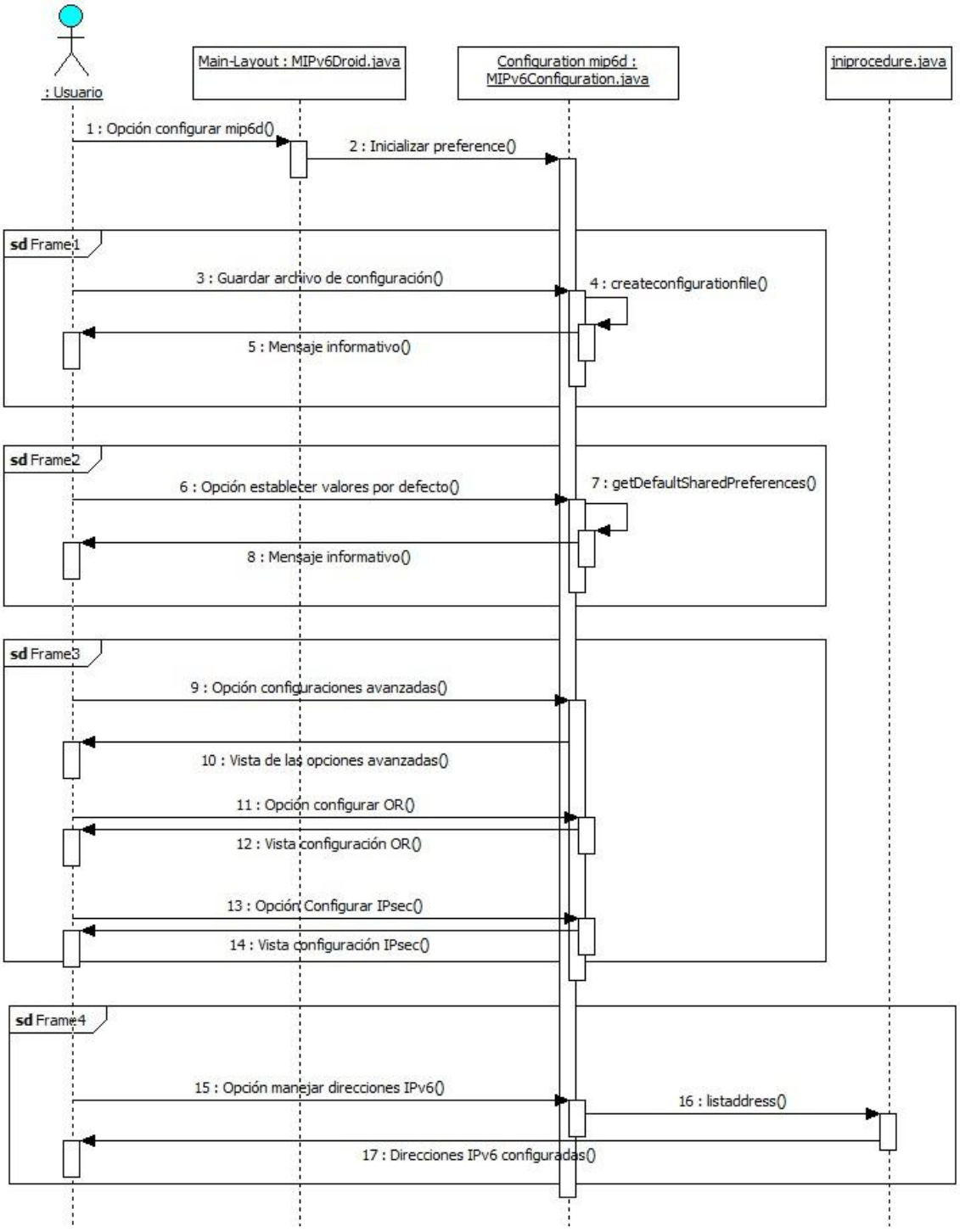


Figura 5.7 Diagrama de secuencia del caso de uso configuración de mip6d

5.3. Implementación de la aplicación

En este capítulo se describe detalladamente cómo se compone la arquitectura de la aplicación diseñada, y el modelo de comunicación entre aplicación MIPv6Droid y el demonio

mip6d. Se realizó un estudio previo sobre las características del SDK de Android para poder determinar cuál es la mejor opción de implementación y a su vez, cubrir los requerimientos de la aplicación. Como se mencionan en los requerimientos de la aplicación, el manejo del demonio mip6d debe realizarse mediante la aplicación MIPv6Droid sin embargo, mip6d es un binario ejecutable por lo tanto se buscó una manera de poder manejar el demonio mip6d desde una aplicación nativa de Android.

5.3.1. Implementación

La implementación del sistema está basada en una estructura que se asemeja al modelo Cliente/Servidor. El servidor está representado por un componente *Service* que forma parte del API de Android el cuál administra el demonio mip6d. El cliente está representado por la interfaz de la aplicación cuya funcionalidad es ofrecer al usuario la interfaz para que pueda dar la orden de inicialización, finalización, configuración del demonio mip6d y además, ofrecer la interfaz para mostrar la información recibida de la salida estándar del demonio. En la Figura 5.8 se muestra el modelo de comunicación que existe entre los diferentes componentes de la aplicación MIPv6Droid y el demonio mip6d.

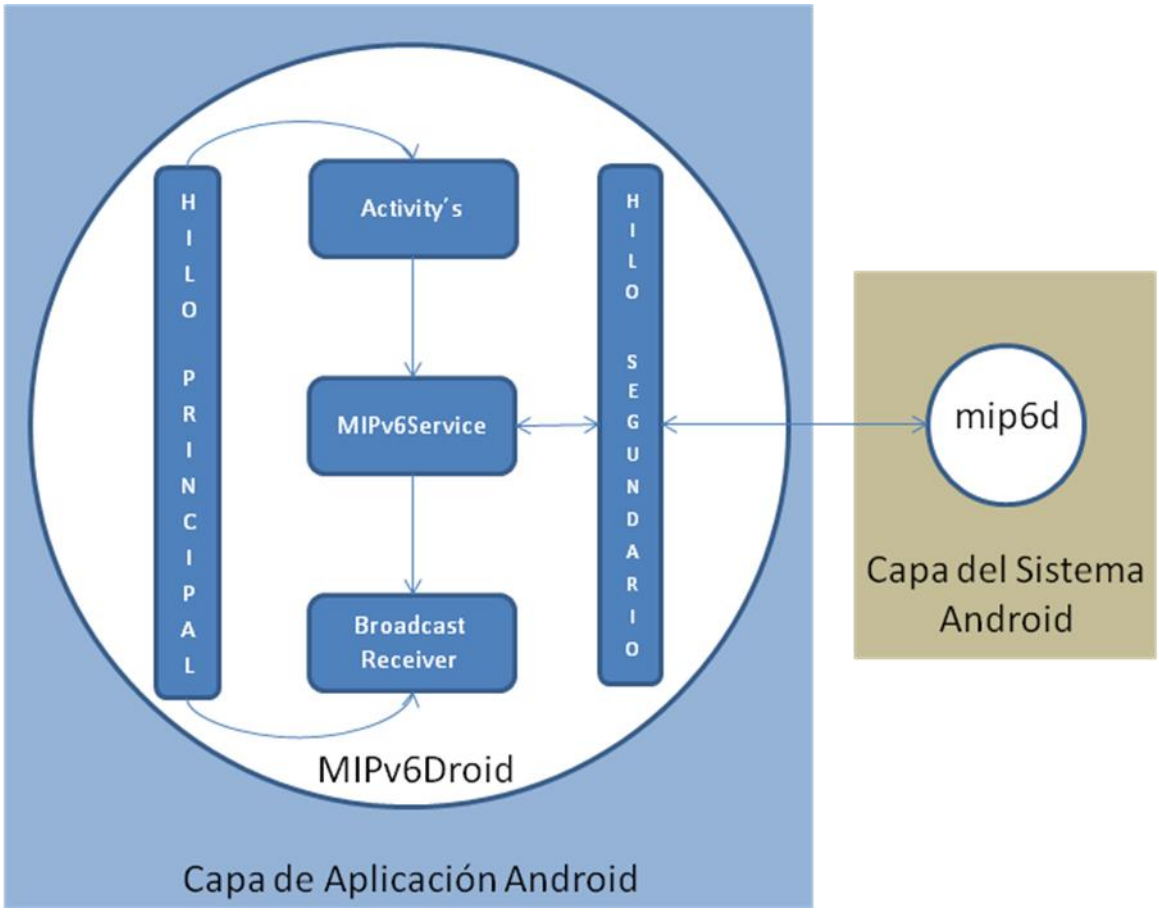


Figura 5.8 Modelado de la comunicación interna - MIPv6Droid

El objeto de tipo *Service* es un componente del API de Android que puede realizar operaciones en segundo plano de larga duración sin proveer una interfaz de usuario. El servicio

puede ser inicializado por otro componente de la aplicación y una vez inicializado correrá en segundo plano inclusive si el usuario navega hacia otra aplicación. Adicionalmente, un componente de la aplicación puede asociarse al servicio después de haberse cerrado. Estas características son bastante atractivas para cumplir con el diseño de la aplicación, es decir, poder ejecutar el demonio mip6d y mantenerlo ejecutándose mientras el usuario realiza otras tareas en el teléfono y después poder asociarse para comunicarse nuevamente con el demonio mip6d.

Unos de los componentes que se utilizó para dibujar la interfaz gráfica fue una actividad (*Activity*). Una actividad es un componente de la aplicación que ofrece una pantalla en donde los usuarios pueden interactuar con el fin de hacer algo, como por ejemplo marcar el teléfono, tomar una foto, enviar un correo electrónico, o ver un mapa. En la aplicación MIP6dDroid fue utilizado para ofrecer una interfaz usable para el usuario. En la Figura 5.9 se puede apreciar la actividad que se presenta al usuario al iniciar la aplicación, específicamente en la pestaña "INFO" donde se muestra información de interés para el usuario y algunas opciones de configuración.

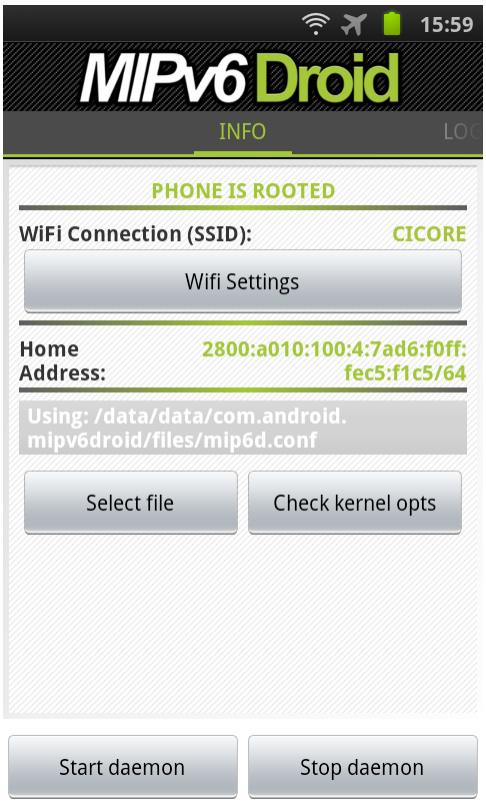


Figura 5.9 Actividad inicial de la aplicación

La aplicación se conforma de varias actividades. Cada una posee una ventana en donde se dibuja la interfaz de usuario la cual se define en un archivo con extensión .XML. Para dibujar la interfaz gráfica se utilizó el Android UI *toolkit* y opciones del menú para la comunicación efectiva con el usuario y así cumplir con los requerimientos antes descritos. Otras de las funcionalidades que provee este componerte es que permite la navegación en la aplicación.

Cada vez que se inicia una nueva actividad, la actividad anterior se detiene, pero el sistema preserva la actividad en una pila. Cuando la actividad se reanuda, se puede volver a adquirir los recursos necesarios y reanudar las acciones que fueron interrumpidas. Estas transiciones de estado son parte del ciclo de vida de la actividad y ayudan a mantener la consistencia de la navegación.

El binario resultado de la compilación cruzada de la implementación UMIP (ver sección 4.2.3), es incluido dentro del paquete que se genera al compilar la aplicación. Específicamente en la carpeta *assets* del proyecto, de donde posteriormente es extraído y ejecutado en tiempo de ejecución por parte de la aplicación.

5.3.2. Implementación de las funcionalidades

A continuación, se describe de forma detallada la implementación de cada una de las funcionalidades del sistema.

5.3.3. Configuración del demonio y del sistema

Dentro de la plataforma Android, existe varios métodos para guardar datos del sistema pero no todos ofrecen un modelo de jerarquías y objetos de interfaz ya definidos. La clase *PreferenceActivity* de Android, especifica la implementación de las configuraciones de manera estándar y con el mismo formato de las configuraciones de la plataforma. La ventaja de utilizar esta clase es que poseen objetos predefinidos de tipo *Preference* los cuales se les guarda sus valores automáticamente en un tipo de dato llamado *SharedPreference* a medida que el usuario interactúa con ellos. Específicamente un *SharedPreference* es una dupla que posee una llave que identifica unívocamente a cada objeto *Preference* y su valor correspondiente.

Para la configuración del demonio y del sistema MIPv6Droid se utilizó la clase *PreferenceActivity* porque otorga la facilidad de crear una jerarquía de preferencias (que puede ser mostrado en varias pantallas) definidas a través de un archivo XML sin la necesidad de desarrollar una forma de guardar las configuraciones del sistema. El objeto *PreferenceScreen* debe estar en la parte superior de la jerarquía de preferencia y es la raíz de todos los objetos tipo *Preference*, sin embargo, cada nueva pantalla de preferencia es un nuevo *PreferenceScreen* hijo.

Las configuraciones del sistema son accedidas desde el menú de la actividad principal de la aplicación (Figura 5.10). La ventana de configuración del sistema como se muestra en la Figura 5.11, se presenta al usuario algunas opciones. Los campos que conforman la ventana de configuración del sistema son: ejecutar el demonio en segundo plano, ejecutar el demonio al iniciar el sistema operativo, activar automáticamente el WIFI al iniciar la aplicación, mantener la interfaz WIFI activa mientras el demonio está activo y cambiar el idioma de la aplicación.

Figura 5.10 Menú de opciones de la actividad principal

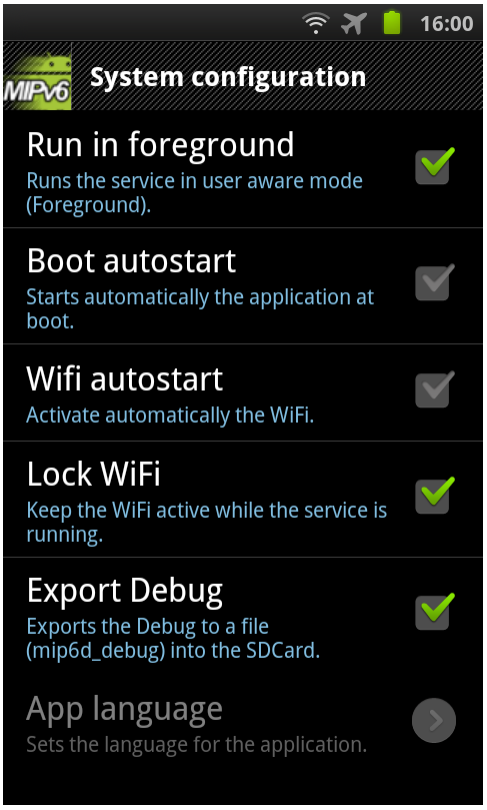


Figura 5.11 Configuraciones del sistema

La opción para configurar el demonio mip6d es accedida desde el menú de la actividad principal de la aplicación. La ventana de configuración del demonio como se muestra en la Figura 5.12 se presenta al usuario las opciones principales.

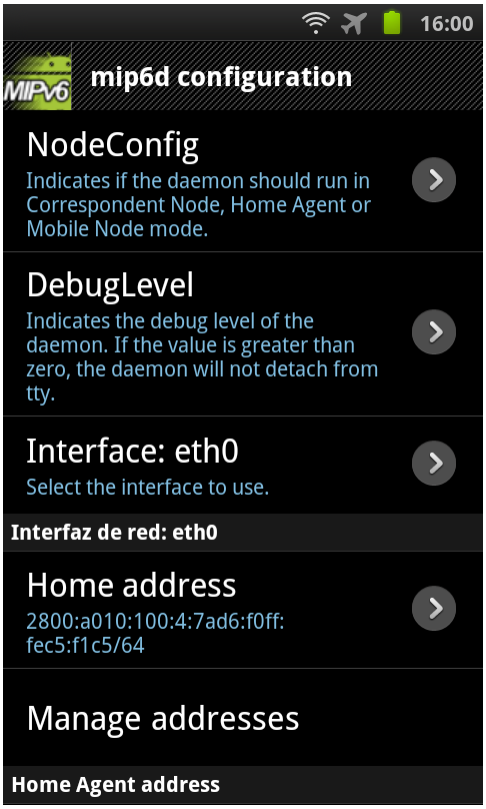


Figura 5.12 Configuración mip6d

En el Anexo N° 4 se encuentra la descripción detallada de cada una de las configuraciones del demonio mip6d.

A pesar de que Android Gingerbread trae soporte de IPv6, no hay una forma nativa que permita realizar la configuración de las interfaces de red, solo es posible la autoconfiguración propia de IPv6 mediante DHCPv6 o *Router Advertisement*. Mediante la utilización del Android NDK que permite implementar partes de las aplicaciones utilizando código nativo en lenguajes como C y C++. Se utilizó un procedimiento programado en lenguaje C que permite la consulta de las interfaces de red y de direcciones IPv6. Para la configuración de las direcciones se utilizó comandos de consola (*ipaddr*) a través de la aplicación para poder hacer la gestión de dichas direcciones.

Como se muestra en la descripción de la configuración de mip6d en el Anexo 1 cada campo posee valores por defecto. La aplicación MIPv6Droid puede restablecer los valores por defecto sin importar el estado de la configuración del demonio. Adicionalmente para la configuración del demonio se desarrolló un algoritmo con la capacidad de recorrer los valores en el *SharedPreferences* y construir un archivo de texto plano con el formato del archivo de configuración del demonio mip6d. El archivo de configuración es guardado dentro de la carpeta files donde está instalada la aplicación MIPv6Droid, comúnmente ubicada en /data/data/com.android.mipv6droid/files/mip6d.conf y si así se desea, pude ser utilizado para la ejecución del demonio.

5.3.4. Inicio del demonio

Para la inicialización del demonio mip6d dentro del componente servicio, se utiliza un objeto Java tipo *Process,* el cual representa un proceso externo a la aplicación y nos permite manejar la ejecución del demonio mip6d desde la aplicación nativa de Android. El botón para la iniciación del demonio se encuentra en la ventana principal de la aplicación y es el que llama a la ejecución del proceso dentro del servicio de la aplicación MIPv6Droid.

5.3.5. Detención del demonio

De igual manera para la finalización de la ejecución del demonio mip6d se utilizó la instancia del objeto *Process* que se encuentra en ejecución y se destruye con la función *destroy*().

5.3.6. Presentación del *debug* del demonio

Para obtener el flujo estándar de salida del demonio, es decir el *debug* o log, se utilizó el método *getInputStream*. El flujo estándar de entrada a la aplicación, realmente es el flujo de salida estándar (stout) del proceso nativo en este caso, representado por mip6d. En la Figura 5.13 se muestra la pestaña “LOG” de la actividad principal donde se presenta el *debug* del demonio al usuario.

Figura 5.13 Pestaña "LOG" donde se presenta el *debug* del demonio

En el Anexo N° 5 se muestra una serie de capturas de pantalla de la aplicación.

6. Pruebas y Análisis de los Resultados

Para la realización de las pruebas posteriores al desarrollo de la solución de movilidad en IPv6, inicialmente se realizó un diseño de unos escenarios en los cuales se puedan poner a prueba de forma efectiva el comportamiento y funcionamiento del protocolo MIPv6 corriendo en el dispositivo. Luego se procedió a realizar la implementación de dichos escenarios para posteriormente poner en marcha las pruebas y finalmente su análisis.

6.1. Diseño y definición de los escenarios de prueba

Para la definición de los escenarios de prueba, se tomaron en cuenta todos los posibles escenarios en los cuales el nodo móvil podría presentarse, esto es, la comunicación con nodos correspondientes presentes tanto en la red local, red foránea o red remota; movimiento del nodo móvil a más de una red foránea; descubrimiento dinámico de prefijos de red y agentes locales, uso de optimización de rutas o túnel bidireccional.

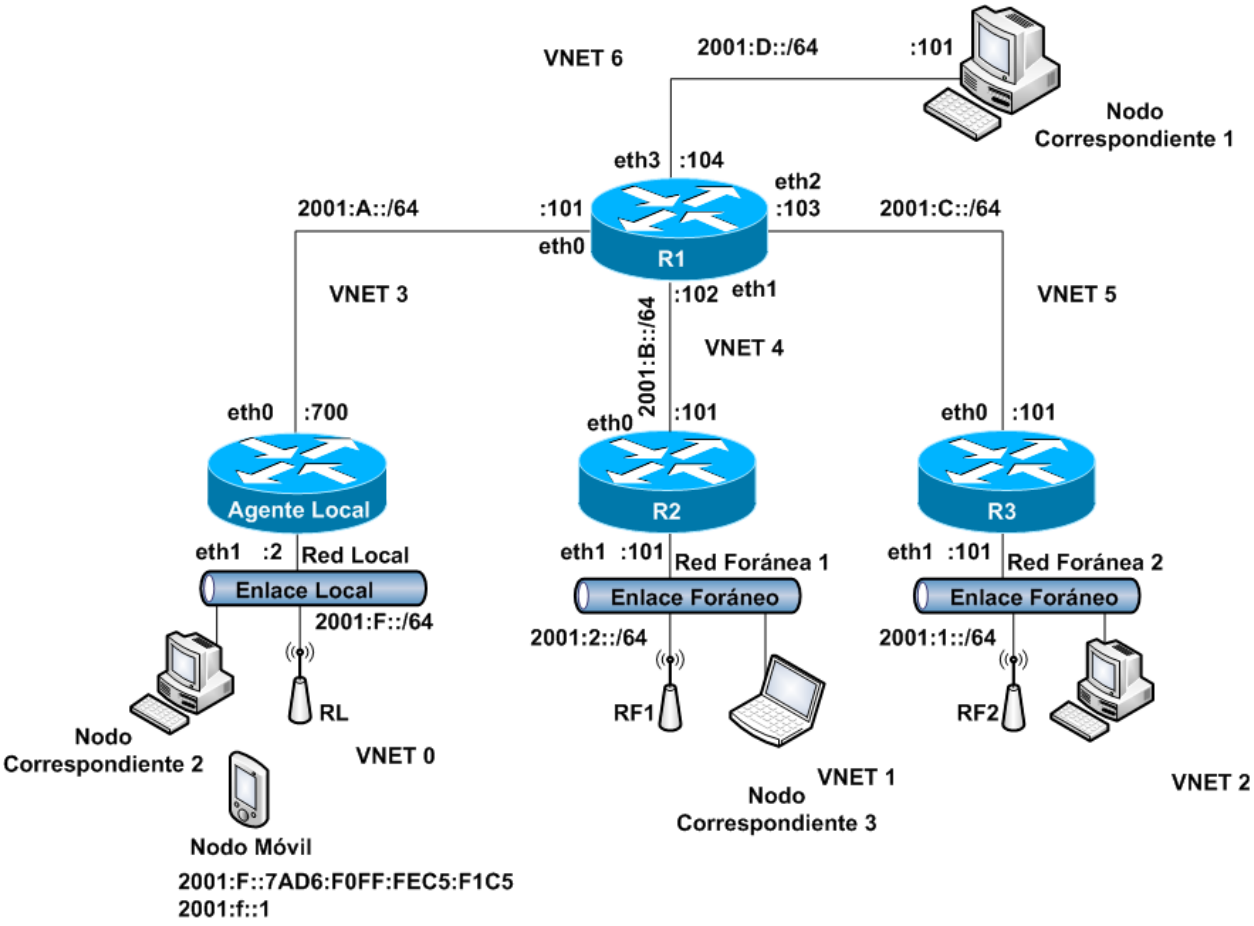


Figura 6.1 Topología de la red IPv6

Como se puede apreciar en la Figura 6.1, la topología de red IPv6 diseñada para los escenarios de prueba cuenta con una red local que sirve como casa del nodo móvil en donde reside el agente local como enrutador de salida con servicios de autoconfiguración de direcciones IPv6. Adicionalmente la topología cuenta con dos redes foráneas las cuales disponen de sus respectivos enrutadores de salida que proveen sus respectivos servicios de autoconfiguración de direcciones IPv6.

Los nodos correspondientes se encuentran en todas las redes en las cuales el nodo móvil puede moverse (local y foráneas) y también en una red atrás del enrutador R1 en donde el nodo móvil no tiene acceso físico.

Tanto la red local como las dos redes foráneas cuentan con sus respectivos puntos de acceso inalámbricos mediante los cuales el dispositivo nodo móvil puede conectarse a dichas redes.

Cada uno de los escenarios de prueba consta de tres sub escenarios, estos son: comunicación sin soporte del protocolo MIPv6, comunicación vía túnel bidireccional y comunicación vía optimización de rutas como está definido en el protocolo MIPv6.

A continuación se procederá a explicar en qué consisten cada uno de los escenarios seleccionados.

6.1.1. Escenario 1: Comunicación entre el Nodo Móvil y el Nodo Correspondiente 1

Este es el escenario que se puede presentar más comúnmente, consiste en la comunicación de un nodo móvil que se encuentra en una red foránea (Red Foránea 1) y quiere iniciar una comunicación con un nodo correspondiente (Nodo Correspondiente 1) que se encuentra en una red remota.

6.1.2. Escenario 2: Comunicación entre el Nodo Móvil y el Nodo Correspondiente 2

En este escenario se introduce la variable de la comunicación con un nodo correspondiente (2) que pertenece a la red local del nodo móvil.

6.1.3. Escenario 3: Comunicación entre el Nodo Móvil y el Nodo Correspondiente 3

La variable en este escenario consiste en la comunicación con un nodo correspondiente (Nodo Correspondiente 3) que se encuentra en la red foránea donde en el nodo móvil se encuentra en ese momento.

6.1.4. Escenario 4 (funcionalidad): Comunicación entre el Nodo Móvil y el Nodo Correspondiente 1 y 3 incluyendo más de un cambio de red y vuelta a casa, utilizando DHAAD

Este último escenario contempla múltiples movimientos de red incluyendo la vuelta a casa mientras que se mantienen activas las conexiones con los distintos nodos correspondientes existentes en la topología diseñada y se hace uso del mecanismo DHAAD.

6.2. Implementación de los escenarios de prueba

Todos los escenarios de prueba antes descritos cuentan con una misma topología de red como se muestra en la Figura 6.1, por lo que para la implementación de dichos escenarios simplemente se procedió a realizar el montaje y configuración de la antes mencionada topología.

Para el montaje de la topología de red, se utilizó un ambiente de máquinas virtuales con VMware VSphere con las cuales se puede crear un esquema de red, por lo que no hace falta el uso de piezas de hardware especializados (enrutadores, conmutadores, cables de red UTP, etc.). Las únicas piezas de hardware de las que no se puede prescindir son los puntos de acceso (AP) inalámbricos mediante los cuales el dispositivo nodo móvil que corre la solución de movilidad puede conectarse e interactuar con la red diseñada.

Para implementar los escenarios de pruebas se debieron realizar las siguientes tareas que se describen a continuación:

- Configuración de las máquinas virtuales.
- Configuración de los nodos.
- Instalación y configuración de UMIP.

6.2.1. Configuración de las máquinas virtuales

El agente local, el nodo correspondiente 1 junto con los enrutadores R1, R2 y R3 son máquinas virtuales corriendo el sistema operativo Debian GNU/Linux Lenny. Luego de crear dichas máquinas virtuales se deben configurar en VMware VSphere para agregarles la cantidad de interfaces de red necesarias a cada máquina y asignarlas a la red virtual (vnetX) específica para cumplir el esquema de la topología. En la Figura 6.2 se puede observar la configuración VMware VSphere del agente local y en la Figura 6.3 se observa la configuración de todas las redes virtuales de la topología.

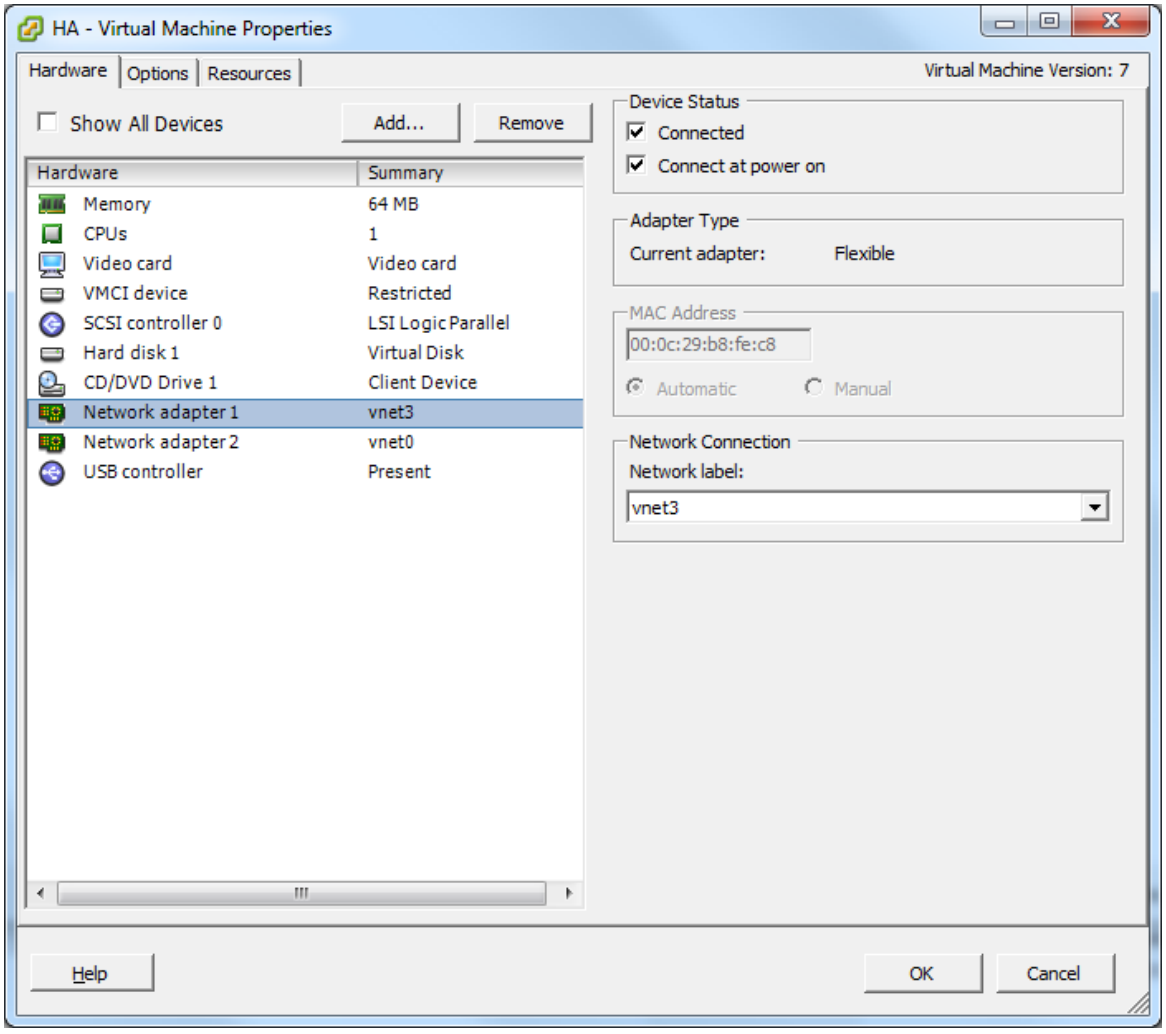


Figura 6.2 Configuración VMware de interfaces de red del agente local

6.2.2. Configuración de los nodos

Para la configuración del agente local y el nodo correspondiente1, la principal configuración que se debe hacer es en el *kernel*, este último se debe recompilar habilitando las opciones de movilidad de forma análoga a como se hace con el nodo móvil.

Recompilación del *kernel*

Lo primero que es necesario hacerse, es obtener el código fuente del *kernel* de Linux, ya sea por medio del gestor de paquetes “Aptitude” o ingresando directamente a la página de archivos de *kernel* de Linux [http://www.*kernel*.org/] donde se encuentra el código fuente de los *kernel* de Linux. La versión del *kernel* que se usa es la 2.6.26 [http://www.*kernel*.org/pub/linux/*kernel*/v2.6/linux-2.6.26.tar.gz] y luego de su descarga se procede a su descompresión utilizando el comando como se muestra en la Figura 6.4.

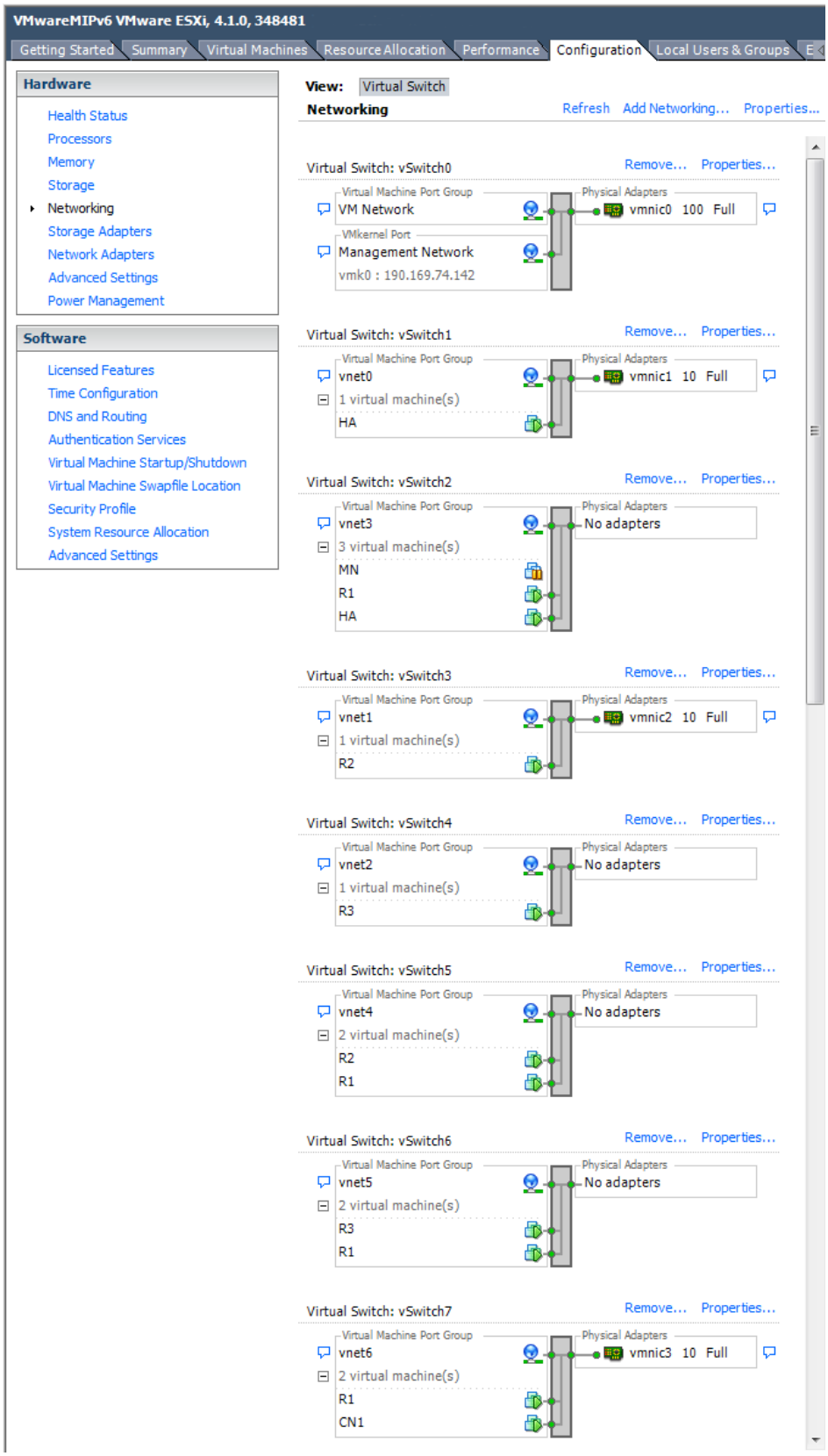


Figura 6.3 Configuración VMware de redes virtuales

```
# wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.26.tar.gz
# tar -zxvf linux-2.6.26.tar.gz
# cd linux-2.6.26
```

Figura 6.4 Comandos para descargar y descomprimir el *kernel*

Para realizar la configuración del *kernel* y habilitar las opciones de movilidad que se necesitan, se ejecutan los siguientes comandos Figura 6.5, y se activan las opciones que se muestran en la Figura 6.6.

```
# make oldconfig
# make menuconfig
```

Figura 6.5 Comando para realizar la configuración del *kernel*

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

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

Figura 6.6 Opciones de movilidad del *kernel*

Después de haber hecho la configuración del *kernel* se procede a realizar la compilación e instalación como se muestra en la Figura 6.7.

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

Figura 6.7 Comandos para compilar el *kernel*

Ahora se actualiza el *bootloader* del sistema, en este caso Grub para que se pueda iniciar el nuevo *kernel*.

Configuración del agente local, enrutadores R2 y R3

Para habilitar al agente local que funcione como el enrutador de la red local, se necesitan realizar ciertas configuraciones que también son necesarias para los enrutadores R2 y R3 para las redes Foráneas 1 y 2 respectivamente. Lo primero que se debe realizar es la configuración de unas banderas de configuración de red del sistema que son leídas en el arranque de este. Se edita el archivo /etc/sysctl.conf y se configuran las siguientes opciones.

```
# nano /etc/sysctl.conf
net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.accept_ra=0
net.ipv6.conf.eth1.proxy_ndp=1        ##→          (solo el Agente local)
```

Figura 6.8 Opciones del sistema

Para que el agente local, los enrutadores R2 y R3 tengan servicios de autoconfiguración de direcciones IPv6, se debe instalar el demonio RADVD (*Router ADVertisement Daemon*, Demonio de Anuncios de Enrutador) mediante el siguiente comando.

```
# aptitude install radvd
```

Figura 6.9 Comando para instalación de RADVD

Para la configuración del demonio radvd primero se deben configurar las interfaces de red del sistema de tal formar que corresponda con la topología diseñada. Un ejemplo de configuración de las interfaces de red del agente local se puede observar en la Figura 6.10. Luego de haber completado el paso anterior, se edita el archivo de configuración de radvd como se muestra en la Figura 6.11.

```
# nano /etc/network/interfaces
auto lo
iface lo inet6 loopback

auto eth0
iface eth0 inet6 static
        address 2001:a::700
        netmask 64
        gateway 2001:a::101

auto eth1
iface eth1 inet6 static
        address 2001:f::2
        netmask 64
```

Figura 6.10 Configuración interfaces de red del Agente Local

```
# nano /etc/radvd.conf
interface eth1
{
   AdvSendAdvert on;
   MaxRtrAdvInterval 10;
   MinRtrAdvInterval 3;

##Sección específica para el Agente local
   AdvHomeAgentFlag on;
   AdvHomeAgentInfo on;
   HomeAgentLifetime 10000;
   HomeAgentPreference 20
##---------------------------------

   prefix 2001:f::2/64
   {
      AdvOnLink on;
      AdvAutonomous on;
      AdvRouterAddr on;
   };
```

Figura 6.11 Configuración RADVD

Por último se reinicia el demonio RADVD con el siguiente comando para que tome las configuraciones realizadas.

```
# /etc/init.d/radvd restart
```

Figura 6.12 Comando para reiniciar el demonio RADVD

Configuración de R1

Aparte de las configuraciones antes descritas para el R2 y R3 que también se deben hacer para R1 exceptuando la instalación del demonio RADVD, es necesario configurar la tabla de rutas de este enrutador que se encargará de interconectar todas las redes. Para hacer la configuración de dicha de tabla de rutas persistente, se edita el archivo interfaces de R1 como se muestra en la Figura 6.13.

```
# nano /etc/network/interfaces
auto lo eth0 eth1 eth2 eth3
iface lo inet6 loopback

iface eth0 inet6 static
        address 2001:a::101
        netmask 64
up ip -6 route add 2001:f::/64 via 2001:a::700

iface eth1 inet6 static
        address 2001:b::102
        netmask 64
up ip -6 route add 2001:2::/64 via 2001:b::101

iface eth2 inet6 static
        address 2001:c::103
        netmask 64
up ip -6 route add 2001:1::/64 via 2001:c::101

iface eth3 inet6 static
        address 2001:d::104
        netmask 64
```

Figura 6.13 Configuración de interfaces y rutas de R1

6.2.3. Instalación y configuración de UMIP

El agente local junto con todos los nodos correspondientes, aparte de tener un *kernel* compilado con las opciones de movilidad activas como se describió en la sección 6.2.2; también deben tener la implementación del demonio IPv6 Móvil (UMIP), para esto, lo primero que se debe realizar es la descarga de su código fuente para la posterior compilación e instalación en dichos nodos. En la Figura 6.14 se muestran los comandos requeridos para obtener el código fuente de UMIP.

```
# git clone git://git.umip.org/umip.git
# cd umip/
```

Figura 6.14 Comandos para la descarga del código fuente de UMIP

Asumiendo que se realizaron los pasos descritos en la sección 6.2.2, se procede a la compilación e instalación de UMIP usando los comandos que se muestran en la Figura 6.15.

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

Figura 6.15 Comandos para la compilación e instalación de UMIP

La opción **--enable-vt** habilita una terminal virtual, la cual puede ser útil para obtener la información de la cache de asociaciones y la lista de actualización de asociaciones del agente local o el nodo móvil.

La configuración de UMIP para todos los tipos de nodo de MIPv6 se realiza sobre el mismo archivo de configuración llamado mip6d.conf. En este archivo se especifica el rol que el nodo cumplirá (HA - agente local, CN – nodo correspondiente, MN – nodo móvil), y también se pueden realizar distintas configuraciones más específicas al demonio.

Configuración del agente local

La configuración del agente local se puede observar en la Figura 6.16. Aquí se especifica que el agente local prestará sus servicios por la interfaz eth1.

```
# nano /etc/mip6d.conf

# This is an example of mip6d Home Agent configuration file
NodeConfig HA;
## If set to > 0, will not detach from tty
DebugLevel 10;
## List of interfaces where we serve as Home Agent
Interface "eth1";

##
## IPsec configuration
##

UseMnHaIPsec disabled;
```

Figura 6.16 UMIP: Configuración del Agente Local

Configuración de los nodos correspondientes

La configuración del agente local se puede observar en la Figura 6.17, donde se especifica si acepta optimización de rutas o no con los nodos móviles.

```
# nano /etc/mip6d.conf

# This is an example of mip6d Correspondent Node configuration file

NodeConfig CN;

## If set to > 0, will not detach from tty
DebugLevel 10;

## Support route optimization with MNs
DoRouteOptimizationCN enabled;
```

Figura 6.17 UMIP: Configuración del Nodo Correspondiente

Para iniciar el demonio, se digita el comando que se muestra en la Figura 6.18; esto también aplica para el agente local.

```
# mip6d –c /etc/mip6d.conf
```

Figura 6.18 Comando para iniciar el demonio mip6d

Configuración del nodo móvil

Como se mencionó en la sección 5.3.2, la configuración del nodo móvil se realiza mediante la aplicación MIPv6Droid. A pesar de que Android Gingerbread trae soporte de IPv6, no hay una forma nativa que permita realizar la configuración de las interfaces de red, solo es posible la autoconfiguración propia de IPv6 mediante *DHCPv6* o *Router Advertisement.* La aplicación MIPv6Droid no solo permite la configuración del demonio mip6d (UMIP), sino que también incluye una opción para la configuración de las interfaces de red y de direcciones IPv6 estáticas.

En la topología de red diseñada, se utiliza como dirección local del nodo móvil, la que se auto configura por el mecanismo *Router Advertisement,* así que simplemente es necesario encender la interfaz inalámbrica y conectarse al punto de acceso “RL” (SSID Red Local). Automáticamente se agregará la dirección 2001:F::7AD6:F0FF:FEC5:F1C5 a la interfaz eth0 (interfaz inalámbrica) del dispositivo. Se puede verificar que se agregó dicha dirección satisfactoriamente por medio de la aplicación MIPv6Droid.

Las opciones del demonio mip6d en particular que se deben configurar en el nodo móvil para los escenarios de prueba son:

- *NodeConfig*: MN
- *Interface*: eth0
- *Home address*: 2001:F::7AD6:F0FF:FEC5:F1C5
- *Home agent address*: DHAAD
- *Advanced – DoRouteOptimizationCN: Check*
- *Advanced – DoRouteOptimizationMN: Check*

6.3. Pruebas

Luego de la implementación de los escenarios de prueba y de sus debidas verificaciones de correcto funcionamiento, se procedió a la ejecución de una serie de pruebas para la evaluación en cada escenario en particular de la solución de movilidad. Las pruebas seleccionadas para ser realizadas en dichos escenarios fueron de rendimiento y funcionalidad, las cuales serán explicadas a continuación.

6.3.1. Pruebas de rendimiento

El factor tomado en cuenta para realizar las pruebas de rendimiento fue la tasa promedio de éxito de entrega de mensajes (*throughput*). Es un promedio del número de bits que se transfieren entre dos dispositivos en una unidad de tiempo. En este caso esta

medida esta expresada en Megabits por segundo (Mb/s). Como se menciona anteriormente (ver la sección 6.1), cada escenario, específicamente el 1, 2 y 3, cuentan con tres sub escenarios que sirven de base de comparación para medir el rendimiento del protocolo IPv6 Móvil. Estos son: transmisión sin soporte del protocolo IPv6 móvil, transmisión con soporte de IPv6 móvil utilizando optimización de rutas y transmisión utilizando túnel bidireccional.

Para llevar a cabo las pruebas en todos los escenarios y sub escenarios, se utilizó un software llamado Iperf (ver la sección 3.2.1). Las pruebas consisten en 10 corridas de Iperf configurado para generar tráfico TCP durante 5 min, de los resultados obtenidos se ordenaron y se calculó la mediana (es el valor medio de un conjunto de datos ordenados) para reflejar un resultado del rendimiento consistente en el sub escenario en el cual se está realizando la prueba.

6.3.2. Pruebas de funcionalidad

Se realizaron pruebas de funcionalidad con la finalidad de validar el funcionamiento correcto de la solución en términos de eventos claves de la misma como son: cambios de red durante comunicaciones activas con varios nodos correspondientes, configuración para el uso de optimización de rutas y túnel bidireccional, configuración para el uso del procedimiento *Dynamic Home Agent Address Discovery*. De tal forma, que se pudiera verificar si dichas funcionalidades cumplen con lo estipulado en el RFC 3775 [1].

Para la realización de estas pruebas se transfirieron archivos de video de gran tamaño mientras aparecían los eventos clave antes mencionados y se hizo uso del software Wireshark (ver sección 3.2.1), para hacer la captura de los paquetes de control propios de IPv6 Móvil. Mientras que se usaron otra serie de comandos básicos de administración de red como ifconfig, ipaddr y traceroute para la verificación de los estados de conexión y conectividad del nodo móvil en un momento dado. Finalmente se hizo uso de herramientas de comprobación de integridad de datos como la suma de comprobación MD5 para verificar que los datos transferidos están correctos. Estos comandos antes mencionados se pueden encontrar dentro de Busybox (ver sección 3.2.1).

Específicamente el escenario utilizado para realizar las pruebas de funcionalidad consiste en:

- El nodo móvil comunicándose con el nodo correspondiente 1 utilizando optimización de rutas.
- El nodo móvil comunicándose con el nodo correspondiente 3 utilizando túnel bidireccional.
- El nodo móvil realiza un cambio a la red foránea 1 luego a la red foránea 2 y por último vuelta a la red local.

El estado inicial del nodo móvil es conectado a la red local por medio del AP con SSID “Red Local” adquiriendo la dirección local mediante el mecanismo “*Router Discovery*” de parte del agente local. En la Figura 6.19 se puede apreciar la configuración inicial del nodo móvil.

```
# ipaddr show eth0
8: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
    link/ether 78:d6:f0:c5:f1:c5 brd ff:ff:ff:ff:ff:ff
    inet6 2001:f::7ad6:f0ff:fec5:f1c5/64 scope global dynamic
       valid_lft 2591992sec preferred_lft 604792sec
    inet6 fe80::7ad6:f0ff:fec5:f1c5/64 scope link
       valid_lft forever preferred_lft forever

# iwconfig eth0
eth0      IEEE 802.11-DS  ESSID:"RedLocal"  Nickname:""
          Mode:Managed  Frequency:2.452 GHz  Access Point:
00:23:04:2D:7A:30
          Bit Rate=54 Mb/s   Tx-Power:32 dBm
          Retry min limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Managementmode:All packets received
```

Figura 6.19 Configuración inicial del NM

El objetivo principal es lograr que el MN se mantenga al alcance tanto en su red local, como en redes foráneas. Cuando el MN está en su red local, los paquetes son direccionados, usando mecanismos convencionales de enrutamiento. Cuando el MN se conecta a otra red, debe poder ser direccionado a través de una o más direcciones de cuidado.

Con la configuración del demonio mip6d descrita en la sección 6.2.3, se procede a ejecutar el demonio mip6d haciendo uso de la aplicación MIPv6Droid como se muestra en la Figura 6.20.

Figura 6.20 Ejecución del demonio mip6d usando la aplicación MIPv6Droid

Una vez iniciado el demonio, se procedió a iniciar la comunicación entre el nodo móvil y los nodos correspondientes 1 y 3. Se realizó la transferencia de dos archivos de video por medio del protocolo SSH haciendo uso de una aplicación llamada SSHDroid.

Durante la transmisión de los archivos de video, cuando estos llevaban 25% de transmisión completada, se procedió a realizar un cambio de red hacia la red foránea 1. Esto se hizo mediante el manejador de conexiones inalámbricas de Android que se muestra en la Figura 6.21.

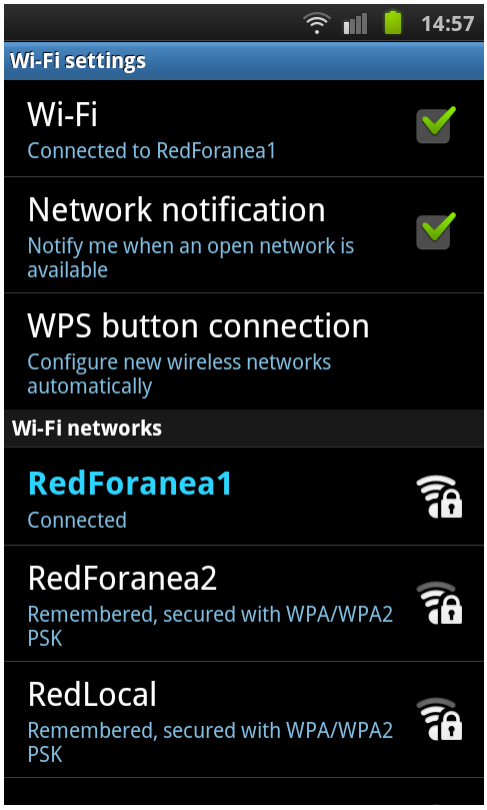


Figura 6.21 Cambio de red inalámbrica

Una vez que el nodo móvil se ha conectado a la red foránea 1, este obtiene una dirección de cuidado mediante el mecanismo *Router Advertisement* de parte del enrutador R2. En la Figura 6.22 se aprecia el estado y configuración del nodo móvil una vez se ha cambiado de red y ha adquirido la dirección de cuidado.

```
# ipaddr show eth0
8: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
    link/ether 78:d6:f0:c5:f1:c5 brd ff:ff:ff:ff:ff:ff
    inet6 2001:2::7ad6:f0ff:fec5:f1c5/64 scope global dynamic
       valid_lft 2591997sec preferred_lft 604797sec
    inet6 fe80::7ad6:f0ff:fec5:f1c5/64 scope link
       valid_lft forever preferred_lft forever

# iwconfig eth0
eth0      IEEE 802.11-DS  ESSID:"RedForanea1"  Nickname:""
          Mode:Managed  Frequency:2.452 GHz  Access Point:
00:23:04:2D:7F:40
          Bit Rate=54 Mb/s   Tx-Power:32 dBm
          Retry min limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Managementmode:All packets received
```

Figura 6.22 Configuración del NM en la red foránea 1

Luego de haberse configurado la dirección de cuidado, el demonio mip6d ya ha detectado que ha realizado un movimiento de red e inicia los procedimientos descritos en el RFC 3775 para continuar y preservar el alcance de la dirección local desde la red foránea 1. Lo primero que hace el demonio mip6d es el mecanismo *Dinamyc Home Agent Address Discovery* para descubrir la dirección del agente local. En la Figura 6.23 se aprecia una captura de Wireshark desde la interfaz eth0 del enrutador R1 donde se capturan los mensajes *Request* (paquete capturado No. 27) y *Reply* (paquete capturado No. 28) de dicho mecanismo en la cual se puede evidenciar que se le informa al nodo móvil que la dirección del agente local es 2001:f::2. Dicha dirección se utiliza para enviar el mensaje *Binding Update* inicial (paquete capturado No. 29) para completar el registro local.

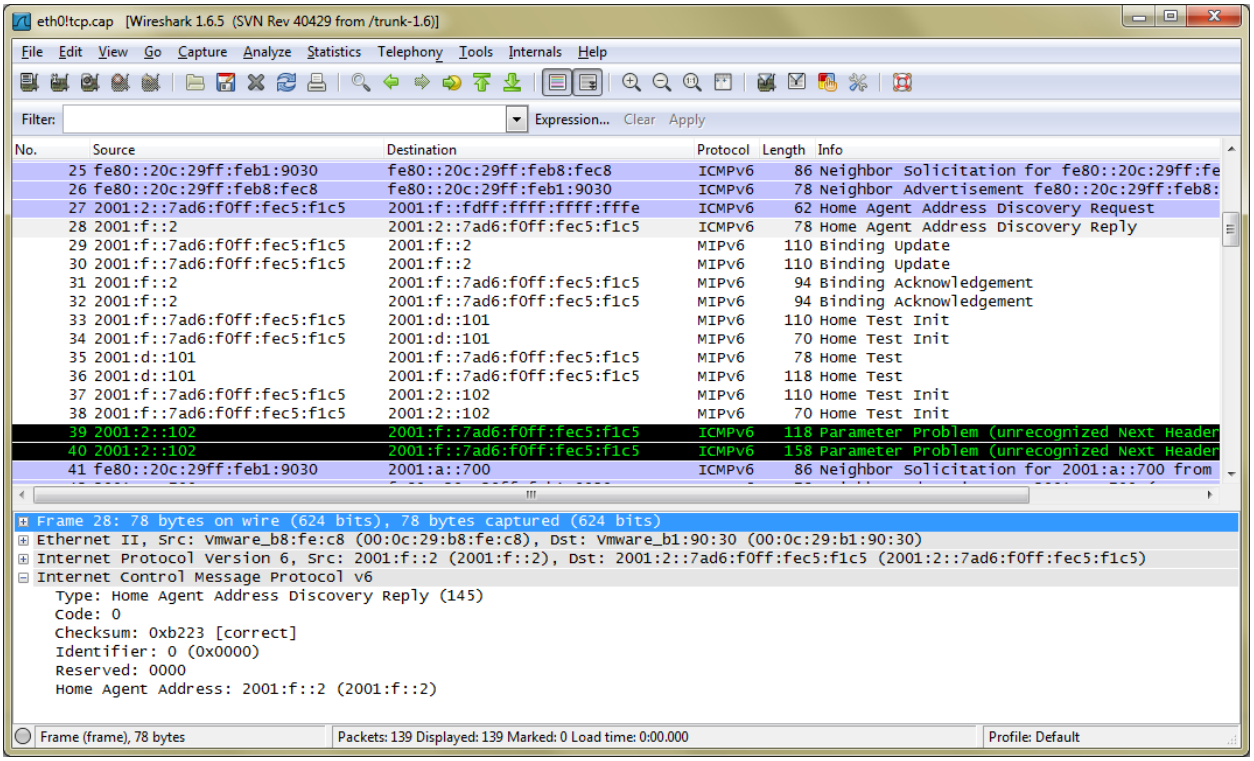


Figura 6.23 Mensajes Home Agent Address Discovery

Una vez que se ha completado el registro local con el agente local, el demonio mip6d en el nodo móvil se da cuenta que tiene conexiones en curso, específicamente las que tiene con los nodos correspondiente 1 y 3. El nodo móvil entonces inicia el procedimiento *Return Routability Procedure.* En la figura Figura 6.24 se puede apreciar una captura de Wireshark desde la interfaz eth3 de enrutador R1 donde se aprecian los mensajes involucrados en el *RRP* y el posterior *BU*.

El nodo móvil inicia el *RRP* contra ambos nodos correspondientes enviando los mensajes *Home Test Init* (paquete capturado No. 70902) y *Care-of Test Init* (paquete capturado No. 70904) respectivos. El nodo correspondiente 1 que está configurado para soportar optimización de rutas, responde a dichos mensajes con los mensajes *Home Test* (paquete capturado No. 70903) y *Care-of Test* (paquete capturado No. 70905) pertinentes para que posteriormente el

nodo móvil complete el registro correspondiente enviando un mensaje *Binding Update* (paquete capturado No. 70906) a dicho nodo correspondiente. En la figura Figura 6.24 se puede apreciar una captura de Wireshark desde la interfaz eth3 de enrutador R1 donde se aprecian todos los mensajes involucrados en el *RRP* y el posterior *BU*.

Por otro lado, cuando el nodo correspondiente 3 recibe los mensajes *Home Test Init* y *Care-of Test Init*, como no tiene soporte del protocolo IPv6 Móvil, le informa al nodo correspondiente por medio de un mensaje ICMPv6 – *Parameter Problem (unrecognized next header)* (paquete capturado No. 29) (Figura 6.25) que no tiene soporte de dicho protocolo. De esta forma no se puede completar el *RRP* contra este nodo correspondiente y entonces se procede a continuar la transferencia haciendo uso de túnel bidireccional.

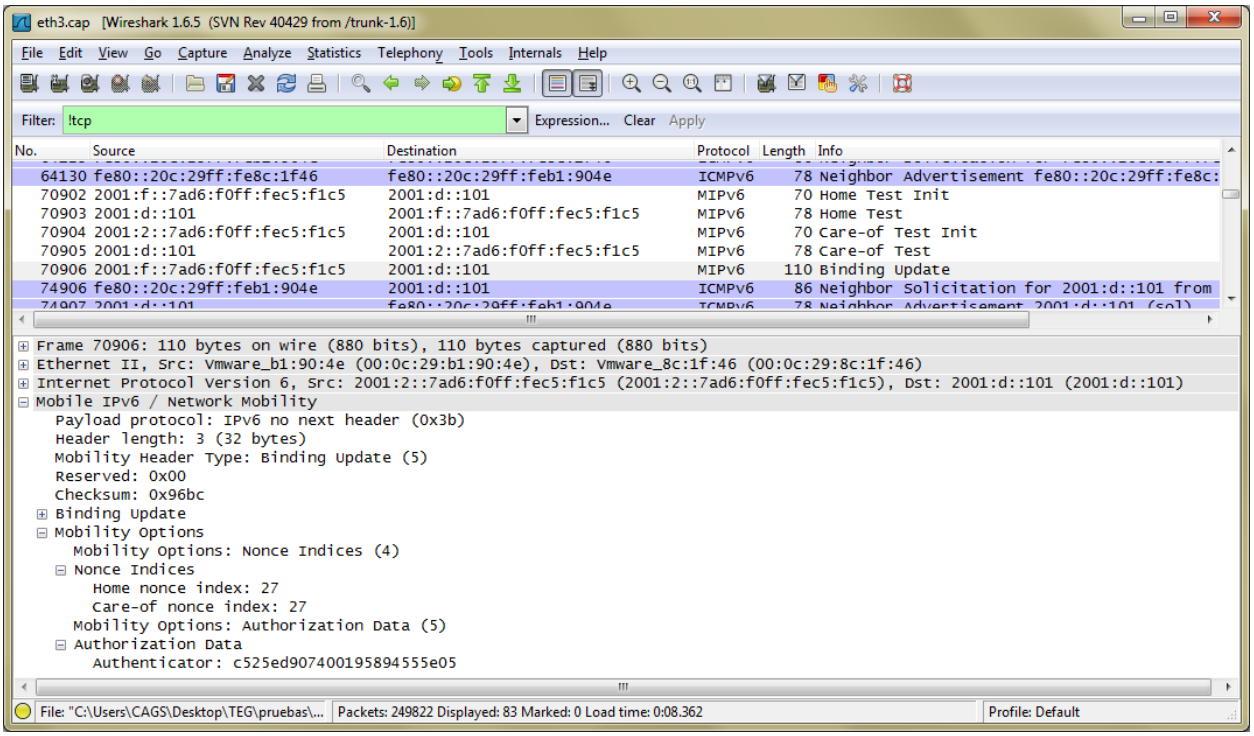


Figura 6.24 *RRP* con el nodo correspondiente 1

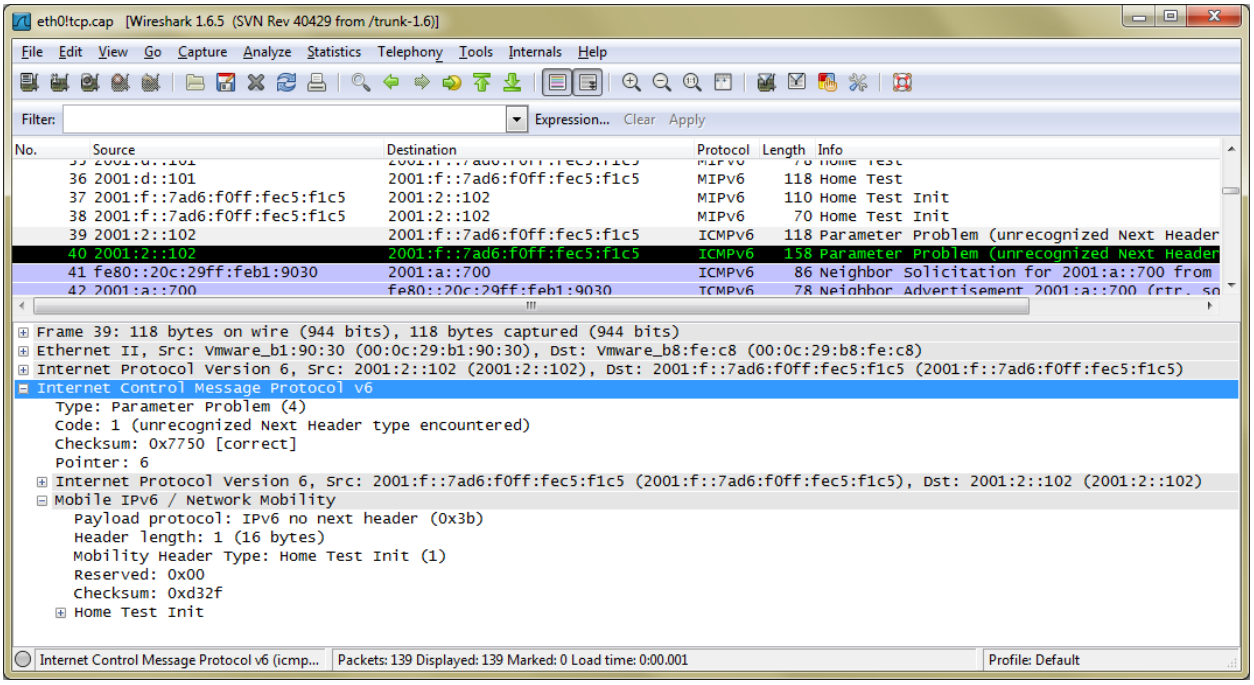


Figura 6.25 ICMPv6 – *Parameter Problem* (unrecognized next header)

Cuando la transmisión de los videos tenían 50% completado, se procedió a realizar otro cambio de red hacia la red foránea 2 de forma análoga a como se realizó anteriormente (Figura 6.21. El procedimiento para la configuración de la nueva dirección de cuidado y la detección de movimiento se realiza de la misma manera de cómo se hizo antes. En la Figura 6.26 se muestra el estado del nodo móvil luego del movimiento.

```
# ipaddr show eth0
8: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
    link/ether 78:d6:f0:c5:f1:c5 brd ff:ff:ff:ff:ff:ff
    inet6 2001:1::7ad6:f0ff:fec5:f1c5/64 scope global dynamic
       valid_lft 2591998sec preferred_lft 604798sec
    inet6 fe80::7ad6:f0ff:fec5:f1c5/64 scope link
       valid_lft forever preferred_lft forever

# iwconfig eth0
eth0      IEEE 802.11-DS  ESSID:"RedForanea2"  Nickname:""
          Mode:Managed  Frequency:2.452 GHz  Access Point:
00:23:04:2D:7F:A0
          Bit Rate=54 Mb/s   Tx-Power:32 dBm
          Retry min limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Managementmode:All packets received
```

Figura 6.26 Configuración del NM en la RedForanea2

El nodo móvil debe ahora actualizar las asociaciones antes realizadas con el agente local y el nodo correspondiente 1 para informar sobres su nueva ubicación, para esto simplemente

envía un mensaje *Binding Update* al agente local como también envía un *Care-of Test Init* (paquete capturado No. 133537) al nodo correspondiente para comprobar la nueva dirección de cuidado y posteriormente el respectivo Binding Update (paquete capturado No. 133539) como se muestra en la Figura 6.27. Luego de esto, la transferencia continuó su curso.

Finalmente cuando la transmisión de los videos tenían 75% completado, se procedió a realizarse otro cambio de red, esta vez hacia la red local. El nodo móvil se configura de tal forma que concuerda con la configuración inicial antes mostrada en la Figura 6.19. Esta vez el nodo móvil vuelve a mandar dos mensajes *Binding Update* (Figura 6.28), uno para el agente local y otro para el nodo correspondiente 1 (paquete capturado No. 199314), pero con la particularidad que el tiempo de vida (*lifetime*) indica 0 segundos, lo que quiere decir que se debe eliminar la asociación. Para ese momento el agente local deja de interceptar los paquetes dirigidos al nodo móvil para que este los reciba directamente.

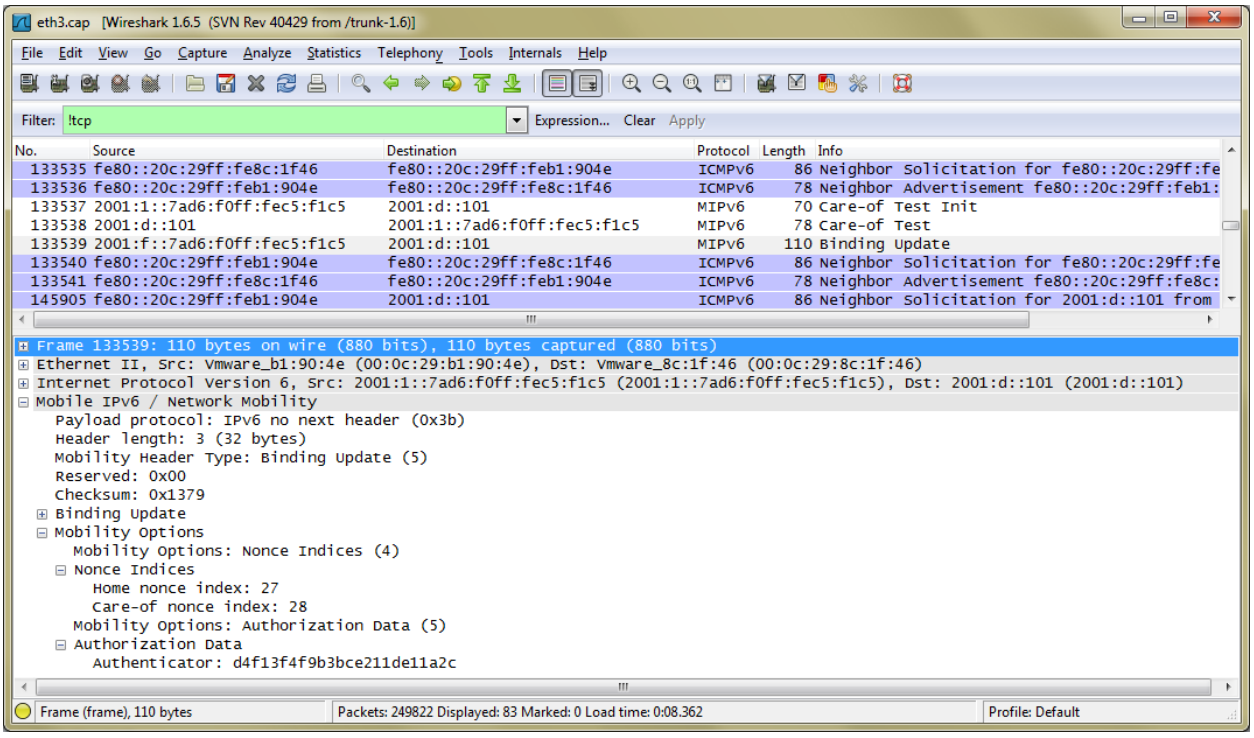


Figura 6.27 *Binding Update* después de cambio a la red foránea 2

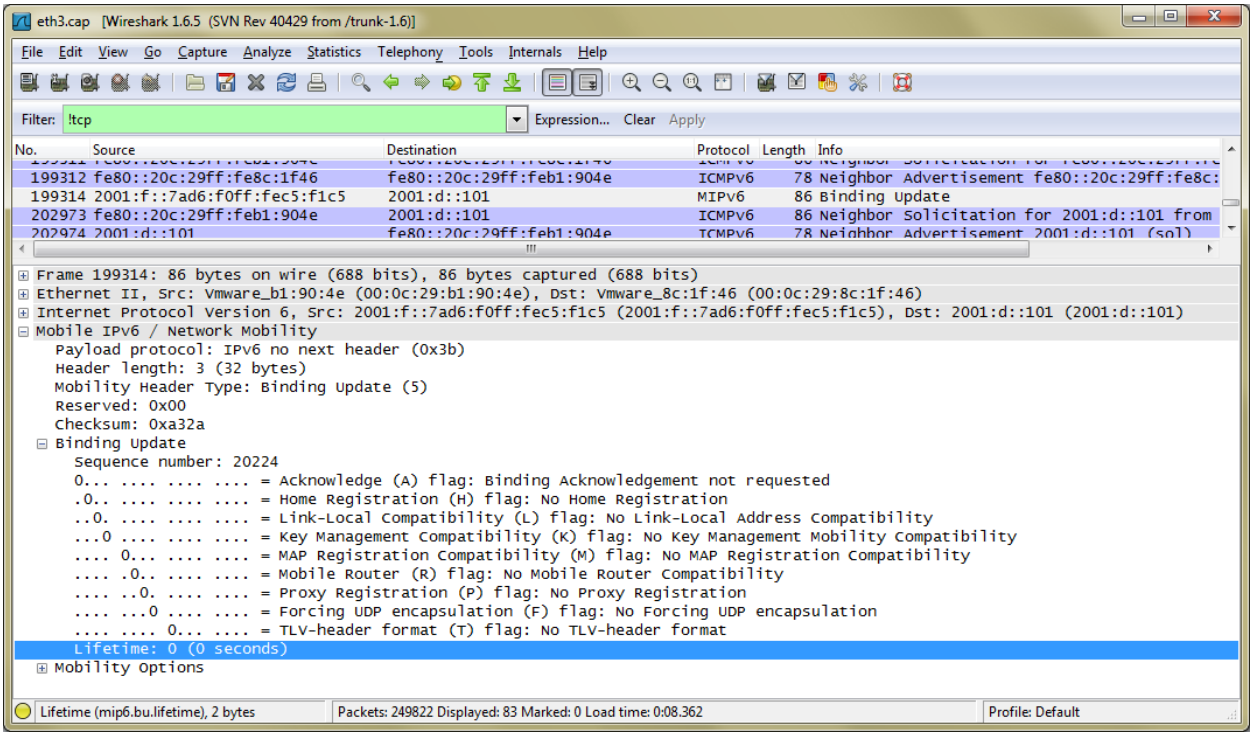


Figura 6.28 Eliminación de la asociación con el nodo correspondiente 1

Al culminar la transferencia de los archivos de video se comprobó su integridad por medio de una suma de comprobación MD5, donde arrojó un resultado indicando que la transferencia fue satisfactoria y que dichos archivos se transmitieron sin problema. Finalmente se detuvo el demonio mip6d.

6.4. Análisis de los resultados de rendimiento

Los resultados obtenidos ejecutando las pruebas antes mencionadas permitieron recopilar toda la información necesaria para realizar un análisis de rendimiento del protocolo IPv6 Móvil con sus variantes corriendo en el dispositivo móvil ejecutando la solución de movilidad. A continuación se analizarán por separado los resultados obtenidos en cada uno de los escenarios planteados anteriormente:

6.4.1. Escenario 1: Comunicación entre el Nodo Móvil y el Nodo Correspondiente 1.

La Figura 6.29 muestra los valores obtenidos durante las pruebas efectuadas en el escenario 1. Allí se muestra el *throughput* para los casos donde no se considera la movilidad del nodo (No MIPv6), cuando se considera movilidad con optimización de rutas (MIPv6 con OR) y cuando se considera movilidad usando túnel bidireccional (MIPv6 con TB).

Como se puede observar, el rendimiento del protocolo IPv6 móvil con optimización de rutas es muy similar al rendimiento de la transferencia sin el soporte de dicho protocolo. Por su parte

el rendimiento del protocolo haciendo uso de túnel bidireccional es considerablemente más bajo. Estos resultados eran los esperados desde el punto de vista teórico donde la mejora en la cantidad de saltos de optimización de rutas sobre túnel bidireccional se hace notar.

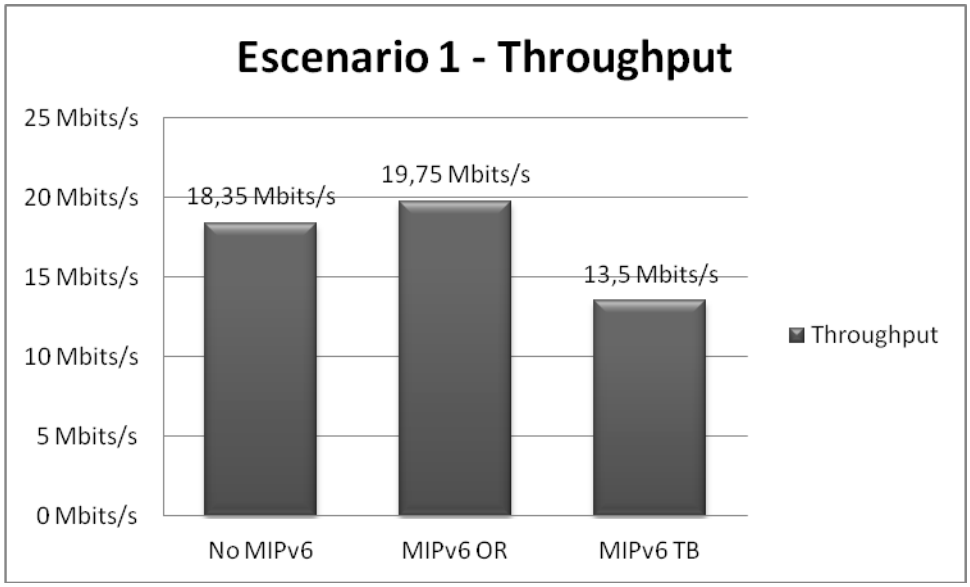


Figura 6.29 Valores de tasa de transferencia de escenario 1

6.4.2. Escenario 2: Comunicación entre el Nodo Móvil y el Nodo Correspondiente 2.

La Figura 6.30 muestra los valores obtenidos durante las pruebas en el escenario 2. Como se puede observar, en este caso el rendimiento de los tres sub-escenarios es similar, esto es porque los paquetes, sin importar el modo de comunicación, siguen la misma ruta y la diferencia que se nota con túnel bidireccional, se le puede atribuir al *overhead* que se presenta en encapsulamiento de los paquetes al realizarse el túnel.

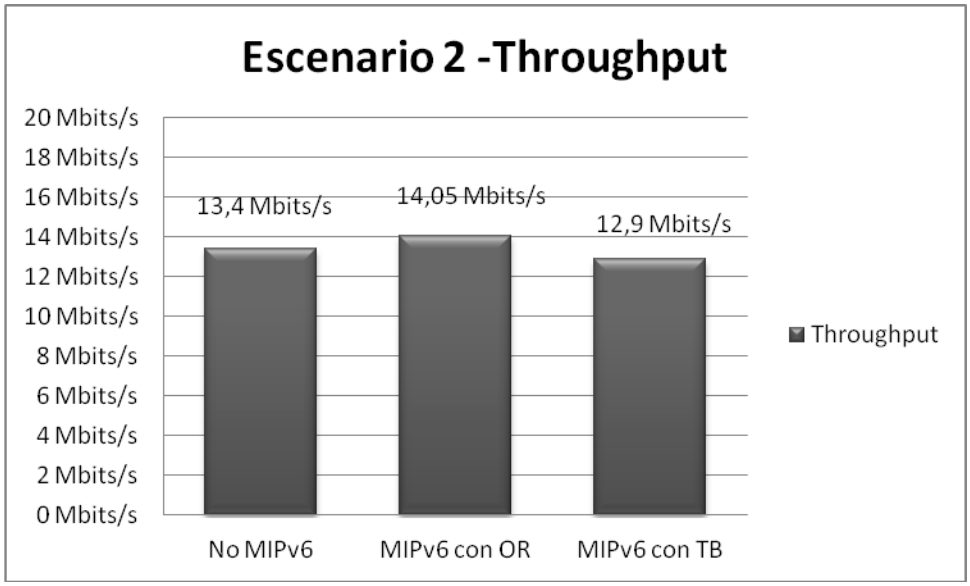


Figura 6.30 Valores de tasa de transferencia de escenario 2

6.4.3. Escenario 3: Comunicación entre el Nodo Móvil y el Nodo Correspondiente 3.

La Figura 6.31 muestra los valores obtenidos durante las pruebas en el escenario 3. El comportamiento que se puede observar en este escenario es muy similar al que se observa en el escenario 1. Mientras que el rendimiento de la comunicación utilizando optimización de rutas y sin soporte de IPv6 móvil son similares. El sub-escenario con uso de túnel bidireccional muestra una considerable baja en el rendimiento, esto es por la diferencia en la cantidad de saltos en comparación con los otros dos modos de comunicación que en este escenario es mucho mayor.

Mientras que al hacer uso de túnel bidireccional, los paquetes deben llegar hasta la red local para posteriormente volver a la red foránea y ser entregados al nodo correspondiente, con el uso de optimización de rutas se realiza una entrega directa al nodo correspondiente lo que implica una gran mejora en la comunicación y se ve reflejado en la tasa de transferencia, lo que es igual para la comunicación sin soporte de movilidad.

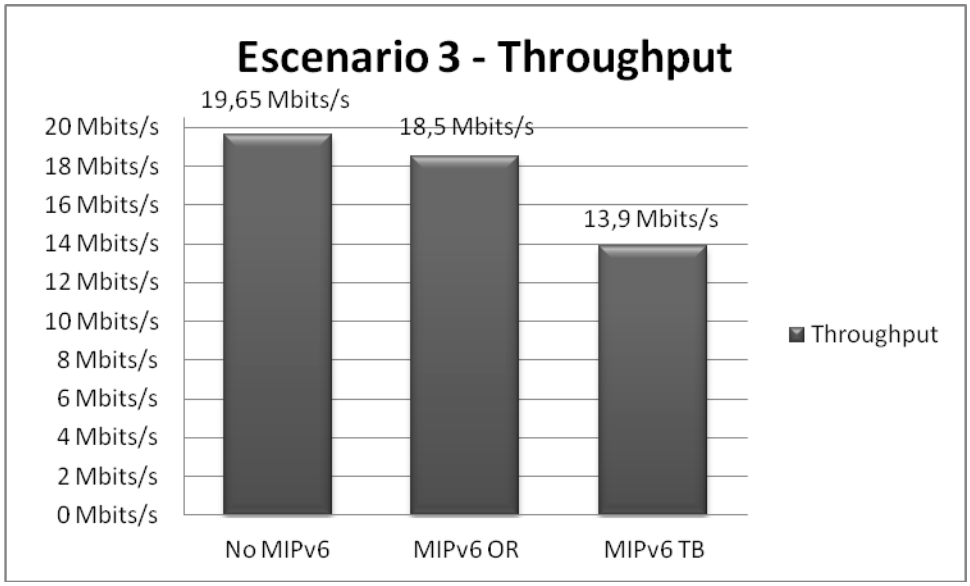


Figura 6.31 Valores de tasa de transferencia de escenario 3

6.5. Análisis de los resultados de funcionalidad

Los resultados obtenidos ejecutando las pruebas de funcionalidad antes mencionadas permitieron recopilar información necesaria para analizar las capacidades funcionales que la solución de movilidad posee y sus aplicaciones en el ambiente de los dispositivos móviles con Android.

Después de haber culminado las pruebas, se evidenció que la solución de movilidad implementada en el dispositivo móvil cumple con las funcionalidades descritas en la especificación del protocolo IPv6 móvil además de las planteadas en el presente trabajo. Los videos transmitidos durante las pruebas desde los distintos nodos hacia el nodo móvil, luego de haberse realizado varios cambios de red, se recibieron de forma satisfactoria y con integridad.

Se comprobó que la solución de movilidad puede operar en el dispositivo móvil de forma transparente y sin intervención explicita del usuario final de forma satisfactoria.

7. Conclusiones

IPv6 Móvil proporciona soporte a la movilidad en una red IPv6, pudiendo mantener las conexiones con la misma dirección IP de un nodo mientras este se encuentra en movimiento. Adicionalmente, las nuevas tendencias como por ejemplo el auge en el uso de dispositivos móviles (teléfonos inteligentes y tabletas), los servicios de computación en la nube, las redes sociales, etc. Crean la necesidad de estar siempre conectados. Por lo tanto, el soporte de los protocolos de movilidad dentro del ámbito de los dispositivos móviles hace sinergia y satisfacen las necesidades de los usuarios finales aunque estos no estén al tanto de lo que está sucediendo a bajo nivel.

Este trabajo se enfocó en el diseño, implementación y realización de pruebas de una solución de movilidad en IPv6 para dispositivos móviles, específicamente a los que corren el sistema operativo Google Android. Luego de haber concluido todas las fases de desarrollo del presente trabajo, y haber podido tener un producto final tangible y probado, se puede decir que el uso de este protocolo en un ambiente real de producción o de uso común para usuarios finales, es factible, útil y conveniente. Se pudo demostrar que se logran los objetivos que incentivaron al desarrollo de dicho protocolo mientras se evita un exceso de complejidad al hacer uso de él en dispositivos de esta índole. Manteniendo la transparencia para los protocolos de capas superiores y más importante aún, para el usuario final.

El resultado de las pruebas de rendimiento sobre el protocolo IPv6 Móvil también revela que la tasa de transferencia de datos no es afectada de forma significativa cuando se está usando optimización de rutas, no es así cuando se usa túnel bidireccional pero se tiene que considerar que aún mantiene las funcionalidades básicas del protocolo y que lo más conveniente y esperado es el uso de optimización de rutas. Esto es otro punto a favor en la decisión de hacer uso de dicho protocolo. Todo esto sin tocar el punto tan discutido a nivel mundial sobre la migración de IPv4 a IPv6.

Se puede concluir que los dispositivos móviles corriendo Android, están listos para hacer uso de los protocolos de vanguardia como IPv6 y de movilidad como los es IPv6 Móvil e interactuar con otros sistemas operativos. Sería ideal que el soporte a dichos protocolos venga incluido nativamente por defecto en los lanzamientos (*realeases*) futuros oficiales de este sistema operativo y así impulsar el uso de estas tecnologías.

7.1. Contribuciones

Este trabajo tiene como principal aporte el de proveer de una solución real y factible para la conectividad IPv6 con soporte de movilidad a los usuarios de dispositivos móviles corriendo Android. El disponer de una aplicación nativa que haga el manejo del demonio IPv6 móvil representa un gran beneficio para dichos usuarios, de tal forma que provee un nivel de transparencia que permite hacer uso de las ventajas que el protocolo provee sin agregar mucha complejidad. Para los usuarios más expertos (investigadores, desarrolladores del protocolo, etc), también sirve como herramienta de apoyo para la realización de futuras investigaciones y

pruebas que involucren ambientes donde es necesario preservar la movilidad IPv6 en dispositivos móviles. Todo esto también haciendo énfasis de que luego de una investigación previa [23] no se encontraron soluciones similares que pudieran proveer las mismas funcionalidades que las que el presente trabajo aporta.

Otro aporte del presente trabajo es el incentivo que la presente solución puede hacer para promover el uso de los protocolos de vanguardia como los son IPv6 y por supuesto IPv6 Móvil. Es ampliamente conocido que aun el protocolo IPv6 no tiene una amplia presencia como la tiene IPv4 y por consiguiente tampoco IPv6 Móvil al depender de la versión del protocolo de internet que se usa. Los usuarios y desarrolladores de aplicaciones tienen la oportunidad de darse cuenta de los beneficios de las ventajas que el protocolo IPv6 móvil ofrece y crear una tendencia o demandar su uso de tal forma que se estimule a los responsables de las plataformas existentes de integrar su soporte por defecto y estimular su uso.

Por último, hay que resaltar que el uso de los protocolos de movilidad, están más que justificados en sistemas los cuales su principal característica es la movilidad. El presente trabajo se enfocó precisamente en adaptar una implementación existente, a un sistema en el cual se puede explotar de una mejor manera las ventajas que el protocolo IPv6 provee. Lo cual resultó de una forma satisfactoria al poder cumplir los objetivos planteados inicialmente.

7.2. Limitaciones

Durante el desarrollo de la solución de movilidad fueron encontradas las siguientes limitaciones:

- Como se mencionó anteriormente, Android trabaja con la biblioteca *Bionic*. El intento de asegurar la compatibilidad de UMIP con dicha biblioteca conllevó mucho tiempo invertido y dificultad. Al final no se logró de esa forma y se optó por la solución que se realizó en el presente trabajo.
- El ineficiente sistema de distribución de Android, específicamente sobre las versiones del sistema que lanzan, trajeron limitaciones a la hora de instalar la versión especifica del sistema requerido. La disponibilidad de dicha versión del sistema depende del fabricante del dispositivo y la región para la cual se distribuirá, lo cual dificulta a los usuarios de los dispositivos a aplicar las actualizaciones deseadas.
- Información muy variada y no oficial sobre cómo realizar la instalación del *kernel* en el dispositivo, ya que el procedimiento varía mucho según la marca y modelo. Se realizaron muchos ensayos y error hasta lograr hacerlo efectivamente. Incluso se dejó inoperativo un teléfono HTC Magic que originalmente iba a ser el que se usaría para el desarrollo del presente trabajo.
- Una limitación propia de la solución es la de tener que realizar una modificación al *kernel* de los dispositivos en donde se usará. Desafortunadamente los *kernel* que vienen por defecto en la gran variedad de dispositivos que usan Android, difieren en el soporte del hardware específico de cada dispositivo, por lo que sería

estrictamente necesario realizar la compilación para cada uno. Tarea que no es sencilla para cualquier usuario de estos dispositivos.

7.3. Trabajos futuros

Se recomienda realizar una extensión al presente trabajo, proveer el soporte para el protocolo DSMIPv6 (*Dual Stack Mobile IPv6*) descrito en el RFC5555 [8] al demonio mip6d, para de esta forma dar soporte al protocolo IPv4 mientras dure la transición a IPv6. Dicha labor ya se encuentra adelantada a un nivel muy básico, el soporte completo del estándar no existe aún, para lograrlo haría falta la modificación del código fuente del *kernel* de Linux, del código fuente del demonio y agregar el soporte de las variables de configuración específicas a la aplicación MIPv6Droid.

Por otro lado, y muy interesante, se propone y recomienda proveer soporte del estándar 802.21[4], el cual describe como habilitar el *handover* e interoperabilidad entre redes heterogéneas, es decir, entre redes de distinta tecnología lo cual se denomina como "*handover* verticales". Dicho estándar provee información para realizar *handover* entre redes 802.3, 802.11, 802.15, 802.16, 3GPP y 3GPP2 por medio de diferentes mecanismos. Llevado a dispositivos móviles, significaría poder realizar *handover* entre la interfaz Wifi y la interfaz radio celular del dispositivo. Una implementación libre de este estándar se denomina ODTONE (*Open Dot Twenty One*)[5].

[4] http://www.ieee802.org/21/

[5] http://hng.av.it.pt/projects/odtone

8. Referencias

[1] **D. Johnson, C. Perkins y J. Arkko**. “Mobility Support in IPv6”. RFC 3775. Junio, 2004.

[2] **C. Perkins.** “IP Mobility Support for IPv4, Revised”. RFC 5944. Noviembre, 2010.

[3] **J. Davies**. “Understanding IPv6”. Segunda Edición. Microsoft Press. 2008.

[4] **Q.Li, T. Jinmei y K. Shima**. “Mobile IPv6 - Protocols and Implementations”. Morgan Kaufman. 2009.

[5] **T. Narten, E. Nordmark, W. Simpson.** “Neighbor Discovery for IP Version 6 (IPv6)”. RFC 2461. Diciembre, 1998.

[6] **J. Arkko, V. Devarapalli, F. Dupont**. “Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents”. RFC 3776. Junio, 2004.

[7] **B. Aboba, J. Carlson, S. Cheshire**. “Detecting Network Attachment in IPv4 (DNAv4)” RFC 4436. Marzo, 2006.

[8] **H. Soliman.** “Mobile IPv6 Support for Dual Stack Hosts and Routers”. RFC 5555. Junio, 2009.

[9] **Google. “**Android**”.** http://www.android.com/. Mayo 2012.

[10] **Google.** “Android Developers”. http://developer.android.com/index.html. Mayo 2012.

[11] **The Eclipse Foundation.** “Eclipse”. http://www.eclipse.org/. Mayo 2012.

[12] **Wikimedia Foundation, Inc**. “Android (operating system)”. http://en.wikipedia.org/wiki/Android_%28operating_system%29#Linux. Mayo 2012.

[13] **UMIP.org.** “UMIP”. http://www.umip.org/. Mayo 2012.

[14] **Mentor Graphics.** “CodeSourcery is now Mentor Graphics Sourcery Tools”. http://www.mentor.com/embedded-software/codesourcery. Mayo 2012.

[15] **The Linux Information Project.** “The su Command”. http://www.linfo.org/su.html. Mayo 2012.

[16] **E. Andersen.** “Busybox”. http://www.busybox.net/about.html. Mayo 2012.

[17] **Y. Gaire** "Smart Android Solutions". http://smartandroidsolutions.blogspot.com/2012/04/what-is-odin-how-do-i-upgrade-samsung.html. Abril, 2012.

[18] **VMware, Inc.** "VMware vSphere: Private Cloud Computing for Mid-Size & Enterprise Businesses". http://www.vmware.com/products/vsphere/mid-size-and-enterprise-business/overview.html

[19] **Software in the Public Interest, Inc.** "Debian -- The Universal Operating System". http://www.debian.org/. Mayo 2012.

[20] **Wireshark Foundation.** "Wireshark – Go deep". http://www.wireshark.org/. Mayo 2012.

[21] **SAMSUNG.** "Galaxy S – INFORMACIÓN GENERAL - SAMSUNG". http://www.samsung.com/ve/consumer/mobile-phones/mobile-phones/smartphone/GT-I9000HKTTTT. Abril, 2012.

[22] **Cisco Systems.** "Cisco Aironet 1200 Series". http://www.cisco.com/en/US/products/hw/wireless/ps430/index.html. Abril, 2012.

[23] **C. Graffe, R. Emmi.** "Propuesta de desarrollo de una solución para la movilidad IP con soporte para dispositivos móviles". Universidad Central de Venezuela. Junio, 2011.

Anexos

Anexo N° 1 Compilación cruzada del *kernel* de Android

Para la compilación cruzada del *kernel* se usó una estación de trabajo corriendo el sistema operativo GNU/Linux Ubuntu x86, en donde se instalaron todas las herramientas necesarias para poder realizar una compilación cruzada para la arquitectura ARM. En la Figura 9.1 Se muestra la instalación por medio del gestor de paquetes aptitude de una serie de herramientas básicas necesarias para poder realizar compilaciones complejas.

```
# aptitude install gnupg flex bison gperf build-essential zip curl \
zlib1g-dev  libc6-dev libncurses5-dev x11proto-core-dev libx11-dev \
libreadline6-dev libgl1-mesa-dev tofrodos python-markdown libxml2-utils \
xsltproc
```

Figura 0.1 Comando para instalación de herramientas básicas de compilación

Ahora es necesario realizar la descarga del compilador cruzado para ARM EABI el cual es una cadena de herramientas (*toolchain*) necesarias para la compilación para la arquitectura ARM. Para esto se accede al sitio web (http://www.codesourcery.com/) y se descarga la cadena de herramientas Sourcery G++ Lite 2009q3-68. En la Figura 9.2 se observan los comandos para la descarga e instalación.

```
# wget https://sourcery.mentor.com/sgpp/portal/package5355/public/arm-none-
eabi/arm-2009q3-68-arm-none-eabi.bin
# chmod +x arm-2009q3-68-arm-none-eabi.bin
# ./arm-2009q3-68-arm-none-eabi.bin
```

Figura 0.2 Comandos para la descarga e instalación de la cadena de herramientas ARM

Los comandos que se muestran en la Figura 9.3 sirven para configurar el entorno del sistema para realizar la compilación cruzada y para emparejar la versión local del sistema (LOCALVERSION) que en este caso es XWJVB con la del *kernel* en orden para que los módulos del *kernel* puedan cargarse de forma correcta.

```
# export ARCH=arm
# export CROSS_COMPILE="directorio de instalación"/CodeSourcery-2009q3-
68/Sourcery_G++_Lite/bin/arm-none-eabi-
# export LOCALVERSION=-I9000XWJVB-CL118186
# export KBUILD_BUILD_VERSION=MIPV6
```

Figura 0.3 Comandos para configurar el entorno de compilación

Completando el paso anterior ya se tiene el entorno del sistema configurado para realizar la compilación cruzada. Ahora se debe hacer la descarga del código fuente del *kernel* específico para el teléfono que se usó. Samsung dispone de una página web para la descarga de todo el contenido fuente abierta relacionado con sus productos (https://opensource.samsung.com) donde se puede conseguir exactamente lo que se necesita.

El archivo específico que se debe descargar y descomprimir es el GT-I9000_OpenSource_GB.zip que contiene los archivos del *kernel* y los del sistema Android.

Ahora que se ha descargado el código fuente del *kernel*, se debe proceder a realizar la configuración del mismo para habilitar la ejecución de comandos en modo súper usuario y habilitar las opciones de movilidad que son necesarias para la solución del presente T.E.G.

Para preparar el *kernel* para que tenga acceso a ejecución de comandos en modo súper usuario, se debe hacer una manipulación el disco RAM inicial (initramfs) en el cual se va a agregar el binario del programa su (*super user*) que permitirá al usuario ejecutar comandos en el dispositivo en modo privilegiado. Para esto se usa un script de instalación (install_su.sh) que es invocado por el script init.rc en el momento de arranque del sistema y también se usa el binario del programa Busybox (comando que combina muchas utilidades estándares de Unix en un solo ejecutable pequeño) para poder abrir una consola Shell y hacer uso de otros comandos necesarios para la ejecución satisfactoria del script de instalación de *su* antes mencionado. Es necesario destacar que el binario del comando Su y Busybox deben estar compilados específicamente para la arquitectura ARM, también se puede hacer una búsqueda en la web para descargar dichos binarios ya recompilados para la arquitectura ARM.

a) Para obtener el disco RAM inicial, se puede recurrir a distintos métodos, el primero es extrayéndolo del *kernel* que viene por defecto con un script o el segundo método, descargarlo de la web desde el sitio Git de un grupo de trabajo de terceros (Project-vodoo) que se dedica al desarrollo y mejoras del software para los dispositivos Samsung Galaxy. El siguiente enlace sirve para realizar dicha descarga.

   https://github.com/project-voodoo/samsung_initramfs/zipball/master

   Una vez descargado y descomprimido, se encontrará una serie de directorios con un esquema de nombres que viene precedido por la versión Android seguido de un código de distribución local, se debe seleccionar como disco RAM inicial al directorio que concuerde con la versión del sistema que se tiene instalado en el dispositivo, en este caso seria “gingerbread-i9000xwjvb”. Si se falla en realizar la selección correctamente, se podría tener problemas de arranque del *kernel* ya que los módulos (firmware) necesarios para el arranque localizados en el directorio lib/modules/ del disco RAM inicial, no concordaran con los que el *kernel* intentará cargar al momento del arranque del sistema.

b) Modificar el script init.rc que se encuentra en el directorio raíz del disco RAM inicial , agregando en la última línea los siguiente:

```
service install_su /sbin/install_su.sh
user root
oneshot
```

c) Agregar en el directorio tmp del disco RAM inicial los binarios de Su y Busybox.
d) Agregar en el directorio sbin del disco RAM inicial el script install_su.sh.

Una vez completados los pasos anteriores, se debe indicar a la configuración del *kernel* el nuevo disco RAM de inicio que se usara al momento de compilación, además se procede a activar las opciones de movilidad antes mencionadas, para esto se ejecuta el comando que se muestra en la Figura 9.4.

```
# cd "directorio raíz del Kernel"/
# make aries_eur_defconfig
```

Figura 0.4 Comando para configurar opciones del *Kernel*

En la opción CONFIG_INITRAMFS_SOURCE es donde se debe especificar el directorio del disco RAM de inicio que se ha preparado, en la Figura 9.5 se muestra donde se encuentra la opción junto con las opciones de movilidad que se deben activar en el *kernel*.

```
General setup
--> Prompt for development and/or incomplete code/drivers [CONFIG_EXPERIMENTAL]
--> System V IPC [CONFIG_SYSVIPC]
--> CONFIG_INITRAMFS_SOURCE="'directorio donde se encuentra el initramfs'"


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: IPv6-in-IPv6 tunnel [CONFIG_IPV6_TUNNEL]
        --> IPv6: Multiple Routing Tables [CONFIG_IPV6_MULTIPLE_TABLES]
        --> IPv6: source address based routing [CONFIG_IPV6_SUBTREES]

File systems
--> Pseudo filesystems
```

Figura 0.5 Configuración del *Kernel*

Ahora se procede a compilar el *kernel*. Si todas las configuraciones del entorno que se hicieron previamente están correctas, se hará la compilación de forma cruzada, para esto se usa el comando que se muestra en la Figura 9.6.

```
# make
```

Figura 0.6 Compilación *Kernel* de Android

Si todo resulta bien, la imagen empaquetada del *kernel* se debe encontrar en el directorio interno arch/arm/boot/zImage. Ahora para realizar la instalación primero se comprime como se muestra en la Figura 9.7.

```
# tar cf zImage.tar zImage
```

Figura 0.7 Comando para comprimir la imagen del *Kernel*

Anexo N° 2 Instalación del *kernel* compilado en el dispositivo

Para la instalación del *kernel* compilado en el dispositivo, se hace uso de un software llamado Odin v1.30 (se puede conseguir en la web haciendo una búsqueda en Google), este software contiene utilidades de recuperación y reescritura del sistema embebido para los dispositivos Samsung que corre sobre la plataforma Windows. Una vez descargado y ejecutado Odin, se debe poner al dispositivo en un modo especial de recuperación de Samsung que se llama *Downloading Mode,* para entrar a este modo se debe apagar el dispositivo y luego presionar el botón *Home*, *Vol Down* y *Power* por unos segundos para llegar a una pantalla como la que se muestra en la Figura 9.8.

Hay que destacar que los procedimientos que se deben realizar en este momento si no se hacen correctamente, pueden dejar el dispositivo inoperativo, aunque Samsung ha provisto del modo *Downloading Mode* para solventar estas circunstancias, no hay garantía de que pueda ocasionarse un *FULL BRICK* en donde el dispositivo no se puede recuperar.

Una vez que el dispositivo se encuentra en *Downloading Mode*, el mismo se debe conectar a la PC para que Odin lo reconozca y presionar botón PDA para seleccionar el archivo del *kernel* (zImage.tar) que se ha preparado con anterioridad como se muestra en la .

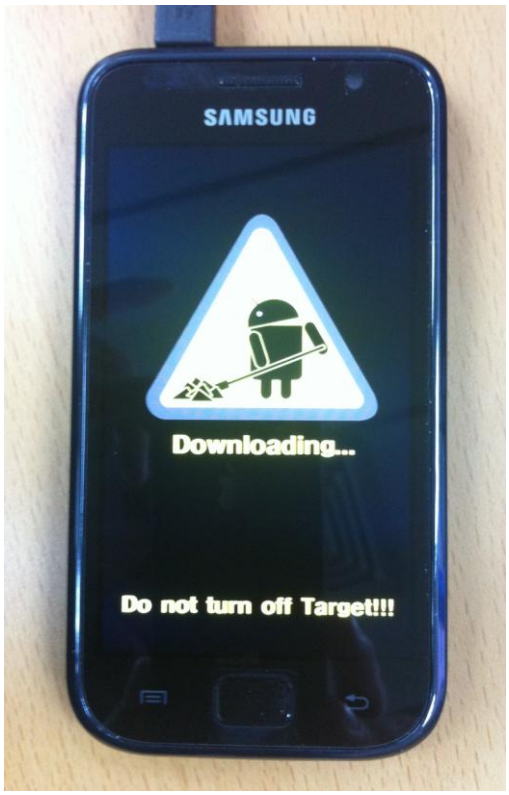


Figura 0.8 *Downloading Mode*

Ahora se procede a realizarse la instalación presionando el botón *Start*, y luego de una serie de pasos que el programa ejecuta (Figura 9.10), este retornará un mensaje de que se completo de forma exitosa *PASS*, en este momento el dispositivo se reiniciará iniciando el *kernel* que se ha compilado.

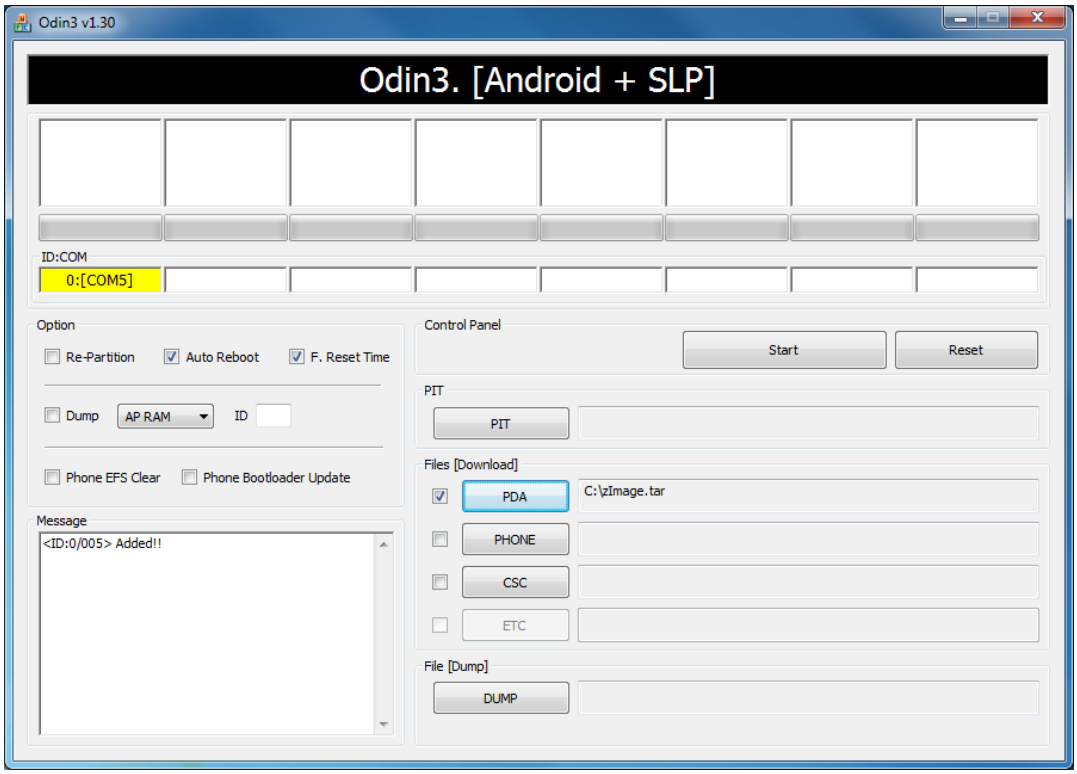


Figura 0.9 Software Odin

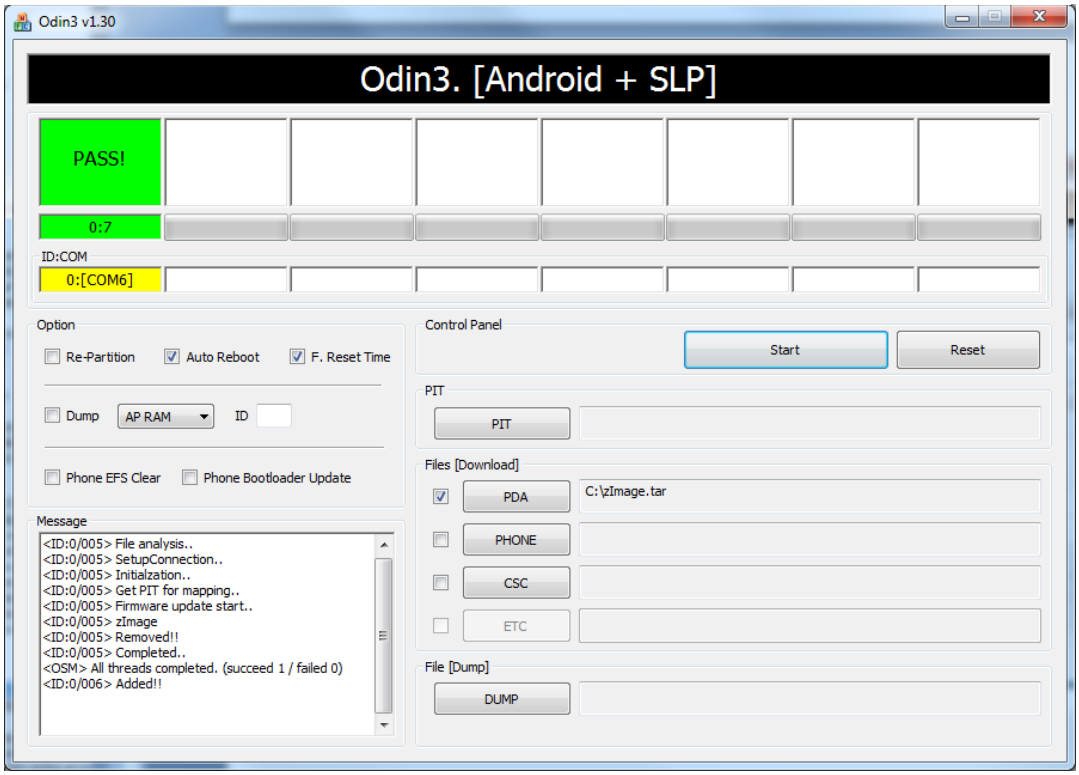


Figura 0.10 Software Odin - PASS

Una vez el dispositivo ha arrancado, se puede verificar la versión del *kernel* que se está ejecutando si se accede a *Settings* > *About Phone* como se muestra en la Figura 9.11.

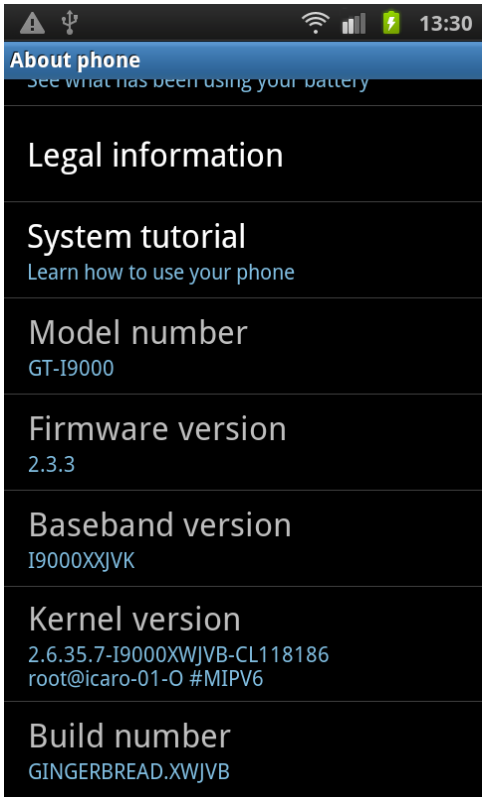


Figura 0.11 *About Phone*

Se puede observar que en el campo *kernel version* aparece el nombre de *kernel* que se le ha puesto MIPV6 y el código de *Local Version* que se debió especificar anteriormente. Además de mostrar el nombre de *Hostname* de la PC que realizo la compilación que en este caso es *icaro-01-O* perteneciente al laboratorio ICARO de la escuela de computación de la Universidad Central de Venezuela.

Por último, para dejar el dispositivo completamente listo, se deben descargar del *Android Market* (Tienda de aplicaciones de Android), dos aplicaciones gratuitas llamadas Superuser y Busybox; necesarias para el correcto funcionamiento de la solución de movilidad del presente trabajo.

Anexo N° 3 Compilación cruzada UMIP (demonio mip6d)

Lo primero que se debe realizar es la descarga de su código fuente para la posterior compilación e instalación en dichos nodos. En la Figura 9.12 se muestran los comandos requeridos para obtener el código fuente de UMIP.

```
# git clone git://git.umip.org/umip.git
# cd umip/
```

Figura 0.12 Comandos para la descarga del código fuente de UMIP

A diferencia del proceso de compilación del *kernel* de Android en la sección anterior, se debe usar una versión de la cadena de herramientas distinta. Para esto se accede al sitio web (http://www.codesourcery.com/) y se descarga la cadena de herramientas Sourcery G++ Lite 2009.03-41. En la Figura 9.13 se observan los comandos para la descarga e instalación.

```
# wget http://www.codesourcery.com/sgpp/lite/arm/portal/package8741/public
/arm-none-linux-gnueabi/arm-2011.03-41-arm-none-linux-gnueabi.bin
# chmod +x arm-2009.03-41-arm-none-eabi.bin
# ./arm-2009q3-68-arm-none-eabi.bin
```

Figura 0.13 Comandos para la descarga e instalación de la cadena de herramientas ARM

Los comandos que se muestran en la Figura 9.3 sirven para configurar el entorno del sistema para realizar la compilación cruzada.

```
# export PATH=$PATH:"directorio inst"/CodeSourcery/Sourcery_G++_Lite/bin/
```

Figura 0.14 Comando configurar PATH del sistema

Se procede a la compilación e instalación de UMIP usando los comandos que se muestran en la Figura 9.15.

```
# autoreconf -i
# ./configure --prefix=/system/local --disable-curses --disable-termcap --
disable-termidx --host=arm-none-linux-gnueabi ARCH=arm CC="arm-none-linux-
gnueabi-gcc" CROSS_COMPILE="arm-none-linux-gnueabi-"
sysconfdir=/system/usr/local/etc
# make CFLAGS="-static"
```

Figura 0.15 Comandos para la compilación e instalación ARM de UMIP

Si todo resulta bien, para este momento se debe disponer del archivo binario de la aplicación en el directorio src/ de umip/. Para verificar si efectivamente se ha compilado para la arquitectura ARM de forma estática, se debe ejecutar el comando que se muestra en la Figura 9.16 y analizar su salida.

```
# file src/mip6d
mip6d: ELF 32-bit LSB executable, ARM, version 1 (SYSV), statically linked,
for GNU/Linux 2.6.16, not stripped
```

Figura 0.16 Comando y salida del comando file sobre el binario de mip6d

Anexo N° 4 Descripción de las configuraciones del demonio mip6d

Abajo se encuentra la lista de opciones de configuración soportadas actualmente por el demonio mip6d (UMIP). Las opciones que están entre paréntesis y doble comillas son las que tienen sintaxis correctas para configurar el demonio mip6d.

- **Nivel de debug** ("DebugLevel " number ";")

Variable que indica el nivel de depuración del programa. La salida de la depuración se realiza en la consola en donde es llamado. El nivel de *debug* se contrala con una variable entero del 0 al 10. 0= es el nivel mínimo de *debug*. Mientras que el 10 es el nivel máximo.

- **Optimización de rutas con otros nodos** ("DoRouteOptimizationMN " "boolean")

Indica si un nodo móvil debe inicializar optimización de rutas con el nodo correspondiente.

- **Optimización de rutas con nodos correspondientes** ("DoRouteOptimizationCN " "boolean").

Indica si un nodo debe participar en la optimización de rutas con el nodo móvil.

- **UseCnBuAck** ("UseCnBuAck " "boolean" ";")

Indica si el bit de *Acknowledge* del mensaje Binding Update hacia el nodo correspondiente debe establecerse para que le respondan con un mensaje Ack.

- **MnDiscardHaParamProb** ("MnDiscardHaParamProb" "boolean";)

Alterna si el nodo móvil debe descartar paquetes *ICMPv6 Paramater Problems* (mensajes ICMPv6 con problemas en los parámetros) que son enviados por el nodo móvil. Como los mensajes de error de ICMPv6 normalmente no están protegidos con IPsec, una tercera persona maliciosa puede hacerse pasar fácilmente por el HA hacia el NM. Tener habilitada esta opción (es decir dejar que acepte este tipo de mensajes) abre una brecha de seguridad para ataques de negación de servicios, incluso aunque el registro hacia el HA estuvo protegido con IPsec. Por defecto: *disabled*.

- **Interfaz** ("Interface" "name";)

Configura la interfaz que se va a utilizar.

- **MnRouterProbes** ("MnRouterProbes " "Number";)

Indica cuantas veces el nodo móvil debe enviar mensajes *Neighbor Unreachability Detection* (NUD) probando a su antiguo Router después de haber recibido un mensaje *Router Advertisement* de un nuevo Router. Si esta opción está fijada a cero o el nuevo mensaje *Router Advertisement* posee un valor estrictamente mayor al valor por defecto del *Router Advertisement* anterior (como se define en el RFC 4191), el nodo móvil se moverá hacia el nuevo Router inmediatamente. El valor por defecto es "0".

- **Configuración de la red local del nodo móvil.**

Esta definición, encerrada entre llaves ({}), definen la configuración de la interfaz del nodo móvil.

```
"MnHomeLink " "name " "{"
"HomeAddress " "address/length MNP list" ";"
"HomeAgentAddress " "address" ";"
"MnRoPolicy ..."
"          ..."
"}"
```

Cada definición de configuración de la red local del nodo móvil (MnHomeLink) está definido por un nombre. Ese nombre corresponde (Encerrado entre comillas) a la interfaz utilizada para la conexión física al enlace local. Para poder configurar múltiples direcciones locales del nodo móvil, se debe definir múltiples estructuras MnHomeLink. Los nombres de las interfaces no tienen que ser única en estas definiciones. Todas las definiciones del enlace local se especifican detalladamente a continuación:

Configuración de la dirección del agente local. ("HomeAddress " "address/length ";")

Es una dirección IPv6, y el prefijo de mascara, usualmente 64. Esta opción debe ser incluida en la definición del enlace local (MnHomeLink).

Configuración de la dirección local del nodo móvil. ("HomeAgentAddress " "address" ";")

Especifica la dirección del agente local del nodo móvil. DHAAD es usado si este campo no se especifica. Por defecto: “::”

Mobile Node Route Optimization Policy ("MnRoPolicy " "address" "boolean" ";")

Activa o desactiva la optimización de rutas hacia un nodo correspondiente específico, identificado por su dirección IPv6. Los campos para la política de optimización de rutas son las siguientes:

Dirección ("address")

Posee la dirección del nodo correspondiente al que se le va a aplicar la política, (if left undefined the uspecified address is used as a wildcard)

Valor buleano ("enabled | disabled"):

Establece la optimización de ruta activada o desactivada para paquetes que coincidan con esta entrada. Cualquier número de estas políticas se pueden definir. Si no se definen las políticas de comportamiento por defecto depende de la opción DoRouteOptimizationMN.

- **Configuraciones de IPsec**

En esta parte se describen las configuraciones de IPsec para el nodo móvil:

“UseMnHaIPsec ” “boolean” “;”

Indica si los mensajes de control entre el nodo móvil y el agente local deben estar protegidos con IPSec.

Key Management Mobility Capability **(“KeyMngMobCapability ” “boolean” “;”)**

Si se utiliza llaves dinámicas con MIPv6-aware IKE si las llaves dinámicas de MIPv6-aware IKE son utilizadas, esta opción debe estar habilitada. Esta opción enciende el K-bit para los mensajes de *Binding Update* y *Binding Acknowledgements*. Por defecto está deshabilitado.

IPsecPolicySet

Es un conjunto de políticas aplicadas a los paquetes. Las configuraciones de las políticas pueden contener múltiples opciones asociadas a la dirección local del nodo móvil (*HomeAddress*) pero, solo una opción asociada a la dirección del agente local *(HomeAgentAddress*). Por ejemplo, para los agentes locales, la dirección local del agente local contiene su propia dirección, y en el campo dirección local del nodo móvil puede contener cualquier cantidad de nodos móviles para la cual aplica esa política.

IPsecPolicy ("IPsecPolicy " "type " "UseESP" "number number" ";")

El campo tipo (*type*) puede ser una de HomeRegBinding, Mh, MobPfxDisc, ICMP, any, TunnelMh, TunnelHomeTesting, or TunnelPayload. Por ahora solo ESP de IPsec es soportado entre el nodo móvil y el agente local. Los dos campos numéricos faltantes especifican los valores requeridos por IPsec. El primero es usado entre el MN-HA, el segundo entre HA-MN.

"MnMaxHaBindingLife " "number" ";"

Limita el máximo tiempo de vida (en segundos) para los registros locales del nodo móvil. Por defecto: 262140

"MnMaxCnBindingLife " "number" ";"

Limita el máximo tiempo de vida (en segundos) para los registros del nodo móvil con los nodos correspondientes. Por defecto: 420

"**SendMobPfxSols " "boolean" ";"**

Controla si el nodo móvil envía mensajes *Mobile Prefix Solicitations* para la red local.

"MobRtrUseExplicitMode " "enabled | disabled" ";"

Alterna entre los modo explícito o implícito de registro en la *Mobile Router*. Por defecto: activado.

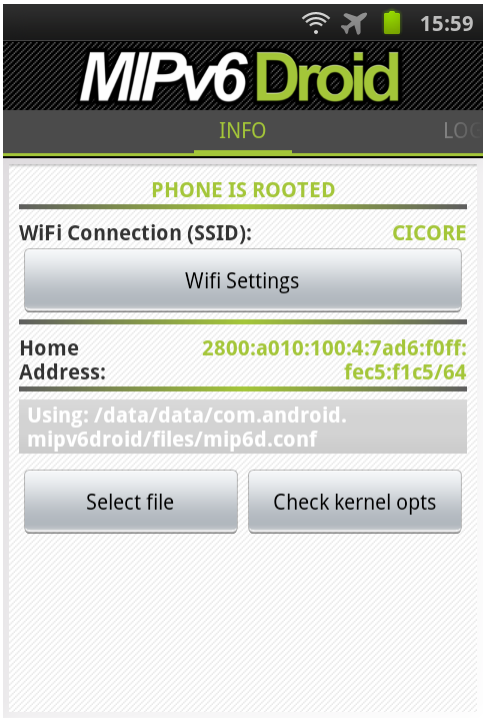
"MnRouterProbeTimeout " "decimal" ";"

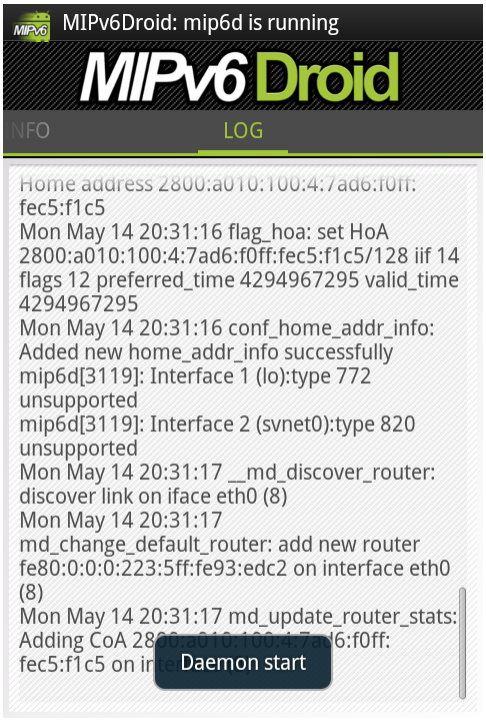
Indica cuanto tiempo (en segundos) el nodo móvil debe esperar por una respuesta durante el acceso a un enrutador al probar mediante el *Neighbor Unreachability Detection*. Si está habilitado, se sobrescribe cualquier valor por defecto del *Neighbor Solicitation Retransmit Timer* que sea mayor que el valor de MnRouterProbeTimeout. Por ejemplo, si el valor para retransmitir es de 1 segundo pero, MnRouteProbeTimeout es de 0,2 segundos, el nodo móvil solo va a esperar 0,2 segundos por un mensaje, antes de proceder con el *handoff*. Por defecto: 0.

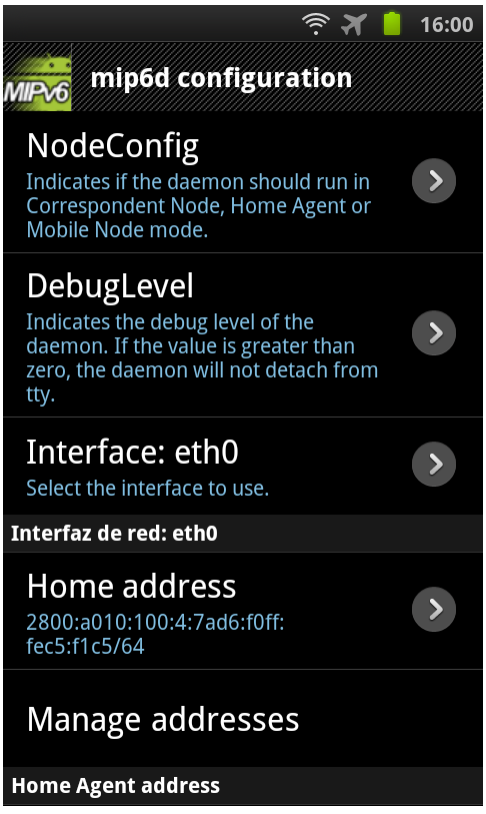
"OptimisticHandoff " "enabled | disabled" ";"

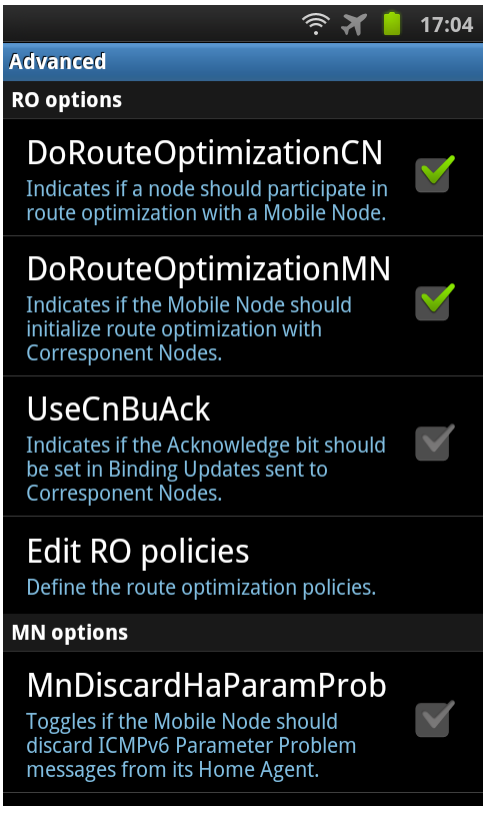
Cuando el nodo móvil envía un mensaje *Binding Update* hacia su agente local por defecto, no se envía ningún tráfico ya sea por optimización de rutas o túnel bidireccional hasta que un mensaje de *Binding Acknowlegment* es recibido. Cuando esta opción está habilitada, permite que el nodo móvil asuma que la asociación ha sido exitosa. Por lo tanto, una vez enviado el mensaje *Binding Update*, no espera por el mensaje positivo de respuesta y sin importar si está utilizando optimización de rutas o túnel bidireccional. Por defecto: disabled.

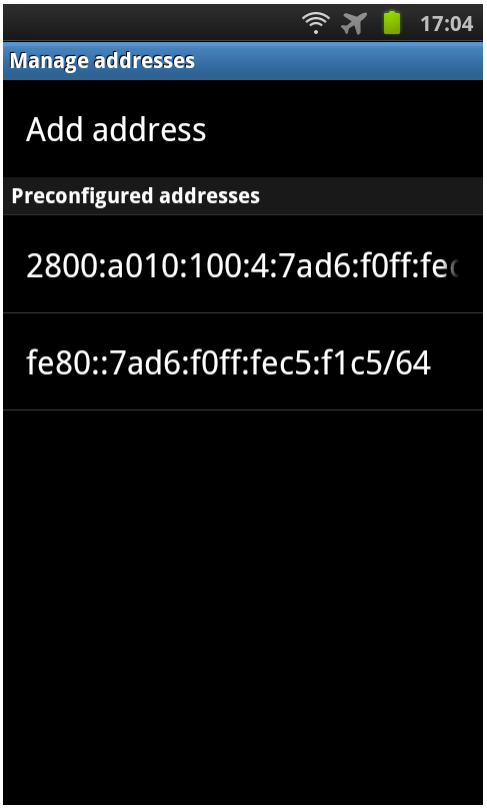
Anexo N° 5 Capturas de pantalla MIPv6Droid

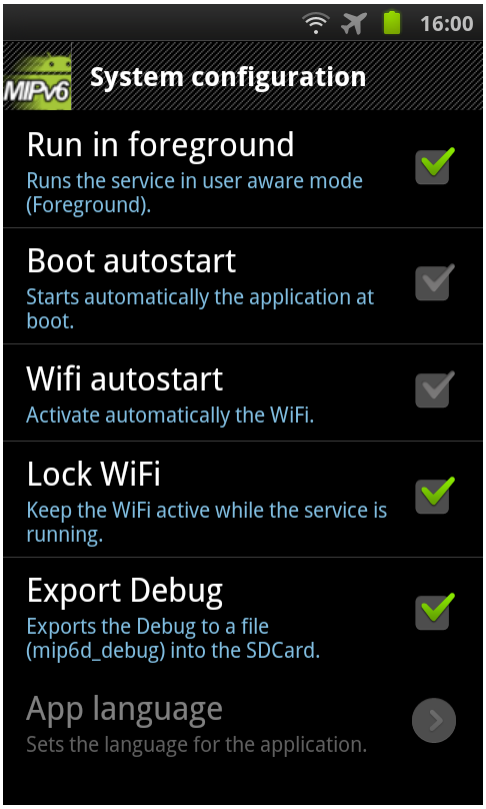




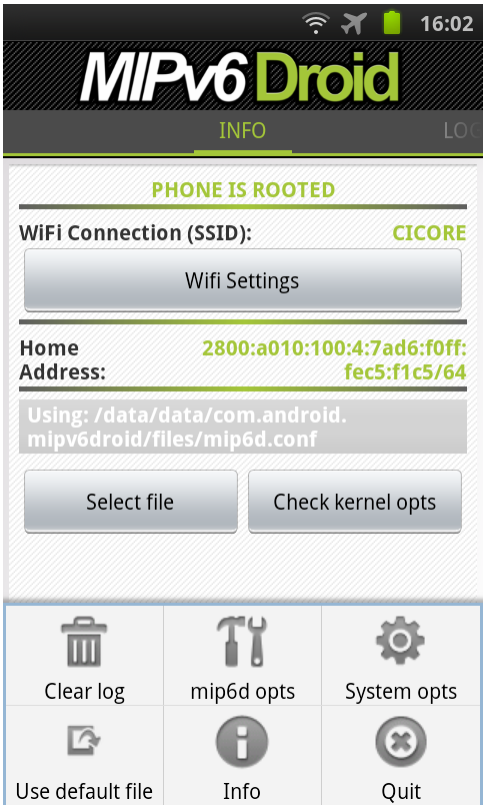





17:04
Manage addresses
Add address
Preconfigured addresses
2800:a010:100:4:7ad6:f0ff:fe
fe80::7ad6:f0ff:fec5:f1c5/64


16:00
System configuration
Run in foreground
Runs the service in user aware mode (Foreground).
Boot autostart
Starts automatically the application at boot.
Wifi autostart
Activate automatically the WiFi.
Lock WiFi
Keep the WiFi active while the service is running.
Export Debug
Exports the Debug to a file (mip6d_debug) into the SDCard.
App language
Sets the language for the application.


15:59
List files
Location: /sdcard
TEGF.doc
Resumen-Christian-Rafael.docx
BCV Doc/
BCV.rar
BBCV_A1_1941_N2_alta.pdf
calibre-0.8.47.msi
metadata.calibre
driveinfo.calibre
mipv6droid.rar
eclipse.rar
Christian
TEG/
script.sh


16:02
MIPv6 Droid
INFO
PHONE IS ROOTED
WiFi Connection (SSID):
CICORE
Wifi Settings
Home Address:
2800:a010:100:4:7ad6:f0ff:fec5:f1c5/64
Using: /data/data/com.android.mipv6droid/files/mip6d.conf
Select file
Check kernel opts
Clear log
mip6d opts
System opts
Use default file
Info
Quit