

**MODELO DE SIMULACIÓN PARA EL PROTOCOLO HIERARCHICAL MOBILE  
IPv6 MEDIANTE EL NETWORK SIMULATOR**



**DARWIN ARLEY COLMENARES DELGADO  
HEILER JOSÉ RODRIGUEZ RODRIGUEZ**

**UNIVERSIDAD PONTIFICIA BOLIVARIANA  
ESCUELA DE INGENIERÍA Y ADMINISTRACIÓN  
FACULTAD DE INGENIERÍA ELECTRÓNICA  
BUCARAMANGA  
2008**

**MODELO DE SIMULACIÓN PARA EL PROTOCOLO HIERARCHICAL MOBILE  
IPv6 MEDIANTE EL NETWORK SIMULATOR**



**DARWIN ARLEY COLMENARES DELGADO  
HEILER JOSÉ RODRIGUEZ RODRIGUEZ**

**TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARA OPTAR AL  
TÍTULO DE INGENIERO ELECTRÓNICO**

Directora:  
**ING. LINE YASMÍN BECERRA SÁNCHEZ**

**UNIVERSIDAD PONTIFICIA BOLIVARIANA  
ESCUELA DE INGENIERÍA Y ADMINISTRACIÓN  
FACULTAD DE INGENIERÍA ELECTRÓNICA  
BUCARAMANGA  
2008**

**Nota de aceptación:**

---

---

---

---

---

---

**Presidente del jurado**

---

**Firma del jurado**

---

**Firma del jurado**

A Dios, a nuestras familias,  
amigos y todas las  
personas que de algún  
modo han ayudado ha que  
esta meta sea alcanzada.

## **AGRADECIMIENTOS**

A la **Ing. Line Yasmín Becerra Sánchez** por su incondicional apoyo en el desarrollo del presente trabajo, por su tiempo, disposición y sobretodo por creer en nosotros.

Agradecemos a los docentes de la Universidad Pontificia Bolivariana y personal en general por su amabilidad y colaboración.

## OBJETIVOS

### Objetivo General

- ✓ Crear un modelo de simulación para el protocolo *Hierarchical Mobile IPv6* [1] que permita evaluar el comportamiento del protocolo para diferentes topologías de red, diferentes fuentes de tráfico y que sea versátil y de fácil acceso.

### Objetivos Específicos

- ✓ Estudiar los protocolos *Hierarchical Mobile IPv6* [1], *Mobile IPv6* [2] y la herramienta de simulación *Network Simulator* [3].
- ✓ Diseñar el modelo de simulación para el protocolo *Hierarchical Mobile IPv6* [1].
- ✓ Realizar pruebas con diferentes topologías y diferentes fuentes de tráfico.

## TABLA DE CONTENIDO

	pág.
INTRODUCCIÓN .....	23
1. MOBILE IPv6 .....	24
1.1 TERMINOLOGÍA DE MOBILE IPv6.....	24
1.2 GENERALIDADES .....	25
1.3 MENSAJES .....	28
1.4 ESTRUCTURA DE DATOS DE MOBILE IPv6 .....	28
1.5 OPERACIÓN DE MOBILE IPv6 .....	29
1.5.1 Registro con el Agente Local.....	30
1.5.2 Enrutamiento Triángulo .....	31
1.5.3 Optimización de la Ruta .....	33
2. HIERARCHICAL MOBILE IPv6 .....	36
2.1 INTRODUCCIÓN.....	36
2.2 OPERACIÓN DE HMIPv6 .....	37
2.3 OPERACIÓN DE LOS ELEMENTOS DE RED .....	40
2.3.1 Operación del Nodo Móvil .....	40
2.3.1.1 Local Binding Update .....	42
2.3.1.2 Actualización de MAPs.....	42
2.3.1.3 Envío de paquetes al Nodo Correspondiente.....	43
2.3.2 Operación del MAP .....	43

2.4	DESCUBRIMIENTO DE MAP .....	44
2.4.1	Descubrimiento dinámico. ....	45
2.4.1.1	Operación del Router .....	46
2.4.1.2	Operación del MAP .....	46
2.4.2	Descubrimiento manual.....	47
2.4.2.1	Operación del Nodo Móvil .....	47
2.5	SELECCIÓN DEL MAP POR EL NODO MÓVIL .....	47
2.5.1	Selección del MAP en un ambiente de MAP distribuido.....	48
2.5.2	Selección del MAP en una arquitectura de gestión de movilidad plana.....	49
2.6	FAST HANDOVER PARA HMIPv6.....	49
2.6.1	Mensajes .....	50
2.6.2	Operación de F-HMIPv6.....	51
3.	CONCEPTOS BÁSICOS DEL LENGUAJE DE PROGRAMACIÓN C++ .....	53
3.1	MATRICES .....	53
3.1.1	Matrices Multidimensionales y de referencias. ....	54
3.2	FUNCIONES AMIGAS DE UNA CLASE .....	57
4.	METODOLOGÍA .....	62
5.	DESARROLLO DEL PROYECTO .....	64
5.1	HERRAMIENTA DE SIMULACIÓN HMIPv6_SM .....	64
5.1.1	Estructura de la Herramienta FHMIPv6-ext de Robert Hsieh .....	65
5.1.2	Inconvenientes y Soluciones de la Herramienta F-HMIPv6-ext.....	68



5.1.3	Estructura de la Herramienta HMIPv6_SM.....	69
5.2	MODIFICACIONES REALIZADAS PARA LA CREACIÓN DE HMIPv6_SM .....	71
5.2.1	Creación de diferentes topologías .....	72
5.2.1.1	Clase “MAPA”.....	72
5.2.1.2	Matriz “map” .....	73
5.2.1.3	Procesos del Método “registrar” .....	75
5.2.2	Optimización de Ruta .....	81
5.2.3	Modificación del desencapsulamiento en los ARs.....	83
5.2.3.1	Valores de “optimy_” según el tipo de Enrutamiento.....	84
5.2.3.1.1	Valores de “optimy_” en el Enrutamiento Triangulo .....	84
5.2.3.1.2	Valores de “optimy_” en Optimización de Ruta.....	86
6.	SIMULACIONES CON HMIPv6_SM.....	87
6.1	SIMULACIÓN 1 .....	87
6.1.1	Configuración de la Simulación .....	87
6.1.1.1	Configuración de la Red Fija .....	88
6.1.1.2	Configuración de la Red Inalámbrica .....	89
6.1.2	Configuración de Movimiento.....	90
6.1.2.1	Comportamiento de la Simulación.....	90
6.1.2.2	Resultados Obtenidos .....	95
6.1.3	Configuración de Movimiento con Optimización de Ruta .....	97
6.1.3.1	Comportamiento de la Simulación.....	97

6.1.3.2	Resultados Obtenidos .....	101
6.2	SIMULACIÓN 2 .....	101
6.2.1	Configuración de Red.....	102
6.2.1.1	Configuración de Red Fija.....	103
6.2.1.2	Configuración de Red Inalámbrica .....	104
6.2.2	Configuración de Movimiento .....	104
6.2.2.1	Comportamiento de la Simulación.....	106
6.2.2.2	Resultados Obtenidos .....	107
6.2.3	Configuración de Movimiento con Optimización de Ruta .....	109
6.2.3.1	Comportamiento de la Simulación.....	111
6.2.3.2	Resultados Obtenidos .....	111
6.3	SIMULACIÓN 3 .....	111
6.3.1	Configuración de Movimiento .....	113
6.3.1.1	Resultados Obtenidos .....	113
6.3.2	Configuración de Movimiento Optimización de Ruta .....	115
6.3.2.1	Resultados Obtenidos .....	116
6.4	Simulación 4 .....	116
6.4.1	Configuración de Red.....	116
6.4.1.1	Configuración de la Red fija .....	121
6.4.1.2	Configuración de la Red Inalámbrica. ....	125
6.4.2	Configuración de Movimiento .....	125
6.4.3	Resultados Obtenidos .....	130

7. CONCLUSIONES .....	134
8. RECOMENDACIONES.....	135
GLOSARIO .....	136
BIBLIOGRAFÍA.....	140
ANEXOS.....	142

## ÍNDICE DE FIGURAS

	pág.
Figura 1: Escenario para Mobile IPv6.....	26
Figura 2: Operación de Mobile IPv6.....	30
Figura 3: Registro del Nodo Móvil con su agente local.....	31
Figura 4: Enrutamiento Triángulo.....	32
Figura 5: Optimización de la Ruta.....	33
Figura 6: Cabecera de enrutamiento de un paquete enviado directamente al MN. .	34
Figura 7: Dominio de HMIPv6.....	36
Figura 8: Desplazamiento del MN.....	38
Figura 9: Mensajes de actualización y registro del MN.....	39
Figura 10: Operación del Nodo Móvil.....	41
Figura 11: Local Binding Update.....	42
Figura 12: Operación del MAP.....	43
Figura 13: Descubrimiento del MAP. ....	45
Figura 14: Diálogo de traspaso con Fast Handover.....	51
Figura 15: Matriz unidimensional de 10 elementos.....	54
Figura 16: Matriz bidimensional vista como dos matrices unidimensionales. ....	55
Figura 17: Matriz bidimensional vista como una tabla. ....	55
Figura 18: Matriz unidimensional con 2 elementos.....	57

Figura 19: Estructura de Archivos Modificados por Robert Hsieh.....	65
Figura 20: Fast Handover Predictivo.....	66
Figura 21: Estructura de archivos modificados para la herramienta HMIPv6_SM y por Robert Hsieh.....	70
Figura 22: Detección del HA a partir del Nodo Móvil .....	78
Figura 23: Registro y asignación del encapsulado en el Home Agent.....	82
Figura 24: Registro y asignación de proceso al Correspondent Node.....	83
Figura 25: Valores de la Variable “optimy_” en el Enrutamiento Triangulo .....	85
Figura 26: Valor de la Variable “optimy_” en la Optimización de Ruta.....	86
Figura 27: Topología de Simulación 1 .....	88
Figura 28: Envío de paquete del CN al MAP en Enrutamiento Triangulo .....	90
Figura 29: Envío de paquete del MAP al MN en Enrutamiento Triangulo.....	93
Figura 30: Movimiento del Nodo Móvil de Simulación1 .....	94
Figura 31: Pérdida de paquetes y paquetes recibidos de Simulación 1.....	96
Figura 32: Paquetes recibidos de Simulación 1.....	96
Figura 33: Pérdida de paquetes Simulación 1. ....	97
Figura 34: Enrutamiento Triangulo en Simulación1 .....	99
Figura 35: Optimización de Ruta en Simulación1. ....	100
Figura 36: Topología de Simulación 2 .....	102
Figura 37: Movimiento del Nodo Móvil en Simulacion2 de AR4 a AR3.....	104
Figura 38: Movimiento del Nodo Móvil en Simulacion2 de AR3 a AR2.....	104
Figura 39: Movimiento del Nodo Móvil en Simulacion2 de AR3 a AR2.....	105

Figura 40: Pérdida de paquetes y paquetes recibidos de Simulación 2.....	108
Figura 41: Pérdida de paquetes de Simulación 2 .....	108
Figura 42: Paquetes recibidos de Simulación 2.....	109
Figura 43: Movimiento del Nodo Móvil en Simulación2 de AR1 a AR2.....	110
Figura 44: Movimiento del Nodo Móvil en Simulacion2 de AR2 a AR3.....	110
Figura 45: Movimiento del Nodo Móvil en Simulación2 de AR3 a AR4.....	111
Figura 46: Topología de Simulación 3 .....	112
Figura 47: Movimiento del nodo móvil en Simulación3 de AR2 a AR1 .....	113
Figura 48: Pérdida de paquetes y paquetes recibidos de Simulación 3.....	114
Figura 49: Paquetes recibidos de Simulación 3.....	114
Figura 50: Paquetes Perdidos de Simulación 3 .....	115
Figura 51: Movimiento del nodo móvil en Simulacion3 de AR1 a AR2 .....	116
Figura 52: Topología Simulación 4. ....	117
Figura 53: Topología Simulación 4 (MAP1). ....	118
Figura 54: Topología Simulación 4 (MAP2). ....	118
Figura 55: Topología Simulación 4 (MAP3). ....	119
Figura 56: Topología Simulación 4 (MAP4). ....	119
Figura 57: Topología Simulación 4 (MAP5) .....	120
Figura 58: Movimiento de MN1 en Simulación 4.....	126
Figura 59: Movimiento de MN2 en Simulación 4.....	126
Figura 60: Movimiento de MN3 en Simulación 4.....	127
Figura 61: Movimiento de MN4 en Simulación 4.....	128

Figura 62: Movimiento de MN5 en Simulación 4.....	128
Figura 63: Movimiento de MN6 en Simulación 4.....	129
Figura 64: Movimiento de MN7 en Simulación 4.....	130
Figura 65: Visualización de Simulación 4 en el NAM.....	131
Figura 66: Pérdida de paquetes y paquetes recibidos de MN7 en Simulación 4 ...	132
Figura 67: Paquetes perdidos de MN7 en Simulación 4 .....	132
Figura 68: Paquetes recibidos por MN7 en Simulación 4 .....	133

## ÍNDICE DE TABLAS

	pág.
Tabla 1: Distribución de las columnas de la matriz “map” .....	74
Tabla 2: Jerarquía de direcciones del MN y HA.....	76
Tabla 3: Direcciones y número de nodo de los componentes de Simulación1 .....	87
Tabla 4: Direcciones y número de nodo de los componentes de Simulación2 .....	102
Tabla 5: Direcciones y número de nodo de los componentes de Simulación2 .....	120



## ÍNDICE DE ANEXOS

### ANEXO A

#### A.1 Instalación del Programa NS y la Herramienta HMIPv6\_SM

##### A.1.1 Instalación de ns-2.30

##### A.1.2 Instalación de la Herramienta HMIPv6\_SM

### ANEXO B

#### B.1 Tutorial HMIPv6\_SM

##### B.1.1 Creación del Objeto Simulador

##### B.1.2 Estructura Jerárquica

##### B.1.3 Creación e Inicialización de los Archivos de Salida

##### B.1.4 Asignación de Grilla

##### B.1.5 Red Fija

##### B.1.6 Configuración de la Red Inalámbrica

##### B.1.7 Ubicación de los Nodos en la Grilla

##### B.1.8 Enlaces de la Red Fija

##### B.1.9 Configuración del Nodo MAP

##### B.1.10 Creación de los Agentes Transporte y las Fuentes de Tráfico

##### B.1.11 Movimiento del Nodo Móvil

##### B.1.12 Creación de los Procesos de Grabación y Finalización

#### B.2 Ejecución del Script

## ANEXO C

### C.1 Formato para Trazas de una Red Fija

#### C.1.1 Tipo de Evento

#### C.1.2 Tiempo de Simulación

#### C.1.3 Nodo Fuente y Nodo Destino

#### C.1.5 Nombre del Paquete

#### C.1.6 Tamaño del Paquete

#### C.1.7 Banderas

#### C.1.8 Flow ID

#### C.1.9 Dirección Fuente y Dirección Destino

#### C.1.10 Número de Secuencia

#### C.1.11 Unique Packet ID

### C.2 Formato para Trazas de una Red Inalámbrica

#### C.2.1 Tipo de Evento

#### C.2.2 Tag Generales

#### C.2.3 Tags de Propiedades del Nodo

#### C.2.4 Información del Paquete de nivel IP

#### C.2.5 Información Siguiendo Destino

#### C.2.6 Información del paquete en el nivel MAC

#### C.2.7 Información del paquete del nivel de Aplicación.

## LISTA DE ACRONIMOS

<b>AR</b>	Access Router
<b>BACK</b>	Binding AcKnowledgement
<b>BR</b>	Binding Request
<b>BU</b>	Binding Update
<b>CoA</b>	Care-of Address
<b>CN</b>	Correspondent Node
<b>FBACK</b>	Fast Binding Acknowledgment
<b>FBU</b>	Fast Binding Update
<b>FNA</b>	Fast Neighbor Advertisement
<b>HACK</b>	Handover Acknowledge
<b>HI</b>	Handover Initiate
<b>HA</b>	Home Agent
<b>LBU</b>	Local Binding Update
<b>LCoA</b>	On-link Care-of Address
<b>MN</b>	Mobile Node
<b>MAP</b>	Mobility Anchor Point
<b>PrRtAdv</b>	Proxy Router Advertisement
<b>RCoA</b>	Regional Care-of Address
<b>RA</b>	Router Advertisement
<b>RtSolPr</b>	Router Solicitation for Proxy Advertisement

## RESUMEN

Este trabajo contiene una descripción detallada del funcionamiento del protocolo *HMIPv6* [1] y el desarrollo del modelo de simulación *HMIPv6\_SM*. Este modelo se creó basado en modificaciones realizadas al *Network Simulator (NS)* [3], el cual, es un software que es utilizado para realizar simulaciones de redes fijas e inalámbricas, y de protocolos de red.

Inicialmente se llevo a cabo un estudio acerca de los protocolos *MIPv6* [2] y *HMIPv6* [1] con el fin de conocer el comportamiento y funcionamiento de los protocolos. A continuación se hicieron simulaciones y algunas pruebas en el *Network Simulator (NS)* [3] con el objetivo de aprender todo lo necesario acerca de *NS* [3] y de igual manera se estudiaron los lenguajes de programación en los cuales esta diseñado el *Network Simulator (NS)* [3].

Posteriormente se consultó información acerca de herramientas de simulación para *HMIPv6* [1] existentes, se hallaron algunos trabajos realizados los cuales no cumplían con las especificaciones básicas del protocolo *HMIPv6* [1] o presentaban limitaciones y además estaban desactualizadas. Uno de los trabajos más próximos al comportamiento de *HMIPv6* y que además estaba disponible para los usuarios fue el realizado por *Robert Hsieh* [4]. La herramienta de *Robert* estaba diseñada para *NS 2.1b7* por tanto, el primer paso a seguir fue implementarla en la versión de *NS 2.30* [3]. Luego se fueron realizando modificaciones al código con el propósito de obtener un modelo de simulación que permitiera efectuar simulaciones para el protocolo *HMIPv6* [1], teniendo en cuenta las características básicas del protocolo especificadas en la *RFC 4140*.

Por ultimo se realizo el diseño y programación del modelo de simulación para el protocolo *HMIPv6* [1] el cual permite la simulación de una topología de hasta 6 *MAPs*, cada *MAP* con la capacidad de manejar 5 *routers* de acceso (*Access Routers, ARs*) en los cuales se pueden conectar 5 nodos móviles (*Mobile Nodes, MNs*) por cada *AR*. De igual manera, realizar diferentes configuraciones de topologías para el estudio de dicho protocolo.

Este documento esta organizado de la siguiente manera: En el capítulo 1 se describe el funcionamiento y operación del protocolo *MIPv6* [2], el capítulo 2 muestra una explicación detallada del protocolo *HMIPv6* [1], en la cual se hace una descripción de la operación del protocolo y el funcionamiento de los elementos de red, entre estos la operación del *MAP*. En el capítulo 3 se describen algunos conceptos básicos acerca del lenguaje de programación *C++* [5], esenciales para el desarrollo del proyecto. El capítulo 4 menciona la metodología que se siguió en la realización del modelo de simulación denominado

*HMIPv6\_SM*. El capítulo 5 contiene la explicación detallada de la creación del modelo de simulación *HMIPv6\_SM*, en la cual se exponen las modificaciones hechas al *Network Simulator* [3], los archivos y carpetas modificados y todo lo referente a la programación. El capítulo 6 muestra el modelo de simulación *HMIPv6\_SM* en funcionamiento donde se realizan diferentes simulaciones con el fin de comprobar la operación del protocolo con las respectivas modificaciones hechas.

Por último se dan las conclusiones, recomendaciones y referencias bibliográficas.

## **ABSTRACT**

Considering the benefits that the protocol Hierarchical Mobile IPv6 with respect to Mobile IPv6 offers, and which the simulators which they are within reach of the users do not have designed a modulate of simulation for Hierarchical Mobile IPv6, it is necessary to design new tools that allow to thus simulate the protocols which they are rising and to facilitate the evaluation of their operation and basic characteristics. This work contains a detailed description of the development of the model of simulation for the protocol Hierarchical Mobile IPv6, denominated HMIPv6\_SM; which allows the simulation of designed topologies of network for this protocol, considering specified the basic characteristics of the protocol in RFC 4140. The HMIPv6\_SM model admits the simulation of a topology of up to 6 MAPs, each MAP with the capacity to handle 5 Access Routers in as 5 Mobile Nodes by each can be connected to access routers. In the same way, to make different configurations from topologies for the study of this protocol. This model was created based on modifications made to the Simulator Network, which, is a software that is used to make simulations of fixed and wireless networks, and of network protocols.

## INTRODUCCIÓN

El uso de dispositivos móviles es cada vez más común, estos nos permiten estar en contacto con otras personas y tener acceso a la red de Internet ya sea enviando o recibiendo información a través de esta.

*Mobile IPv6 (MIPv6)* [2] es un protocolo que permite a los usuarios de dispositivos móviles mantener una conexión permanente a Internet, aun cuando este abandona su red local. Este protocolo ofrece una solución a los problemas de movilidad cuando se trata de macromovilidad, pero presenta ciertas limitaciones en lo que se refiere a la micromovilidad.

El protocolo *Hierarchical Mobile IPv6 (HMIPv6)* [1] es una extensión al protocolo *MIPv6* [2] y se creó para solucionar los problemas de micromovilidad, adicionando un nuevo nodo llamado *Mobility Anchor Point (MAP)*. El *MAP* es el encargado de la movilidad local cuyas funciones son similares al las de un agente local, cada vez que el nodo correspondiente (*Correspondent Node, CN*) envía un paquete hacia el nodo móvil (*Mobile Node, MN*), el *MAP* lo intercepta y lo tuneliza hacia el nodo móvil (*MN*), el nodo móvil quita la cabecera del túnel y procesa el paquete. Siempre que el nodo móvil se mueve dentro de un dominio *MAP* debe actualizarse a su respectivo *MAP*, cuando cambie de dominio *MAP* entonces debe actualizarse con su agente local (*Home Agent, HA*) y nodo correspondiente (*CN*). Este proceso permite una reducción de la señalización entre el *MN*, *HA* y el *CN*, y así mismo disminuye la pérdida de paquetes y las posibles interrupciones en la comunicación, garantizando una mayor eficiencia del protocolo.

Considerando los beneficios que ofrece el protocolo *HMIPv6* [1] con respecto a *MIPv6* [2], y que los simuladores que están al alcance de los usuarios no tienen diseñado un modulo de simulación para *HMIPv6* [1], es necesario diseñar nuevas herramientas que permitan simular los protocolos que van surgiendo y así facilitar la evaluación de su funcionamiento y características básicas.

Teniendo en cuenta que los trabajos realizados por otros autores que implementan algunas de las características básicas del protocolo *HMIP*, están desactualizados y considerando que estos estudios presentaban muchas limitaciones y deficiencias con respecto a las especificaciones descritas en la *RFC 4140 (HMIPv6)* [1]; se planteó un modelo de simulación denominado *HMIPv6\_SM*, el cual permite desarrollar simulaciones en diferentes topologías de red, fuentes de tráfico y que cumple con las características básicas del protocolo *HMIPv6* [1].

## 1. MOBILE IPv6

### 1.1 TERMINOLOGÍA DE MOBILE IPv6

- ✓ **Router:** Es un nodo que envía paquetes IP, los paquetes no necesariamente deben estar direccionados a él.
- ✓ **Mobile Node:** Es un nodo que puede cambiar su punto de conexión de un enlace a otro, manteniendo la comunicación por medio de su dirección local (*Home Address*).
- ✓ **Interface:** Es una conexión de un nodo móvil (*Mobile Node, MN*) a un enlace.
- ✓ **Subnet prefix:** El prefijo de subred es una cadena de bits que consiste en los bits iniciales de la dirección IP.
- ✓ **Interface identifier:** Es un número usado para identificar la interfase de un nodo en un enlace. El identificador de interfase es el resto de bits de bajo orden de la dirección IP del nodo, que están después del prefijo de subred.
- ✓ **Home address:** Es la dirección asignada a un nodo móvil (*Mobile Node, MN*), usada como la dirección permanente del nodo móvil (*Mobile Node, MN*). Esta dirección se encuentra dentro del enlace local del *MN*. Los mecanismos de enrutamiento IP Standard entregarán los paquetes destinados para la dirección local del *MN* a su enlace local. Los *MNs* pueden tener múltiples direcciones locales cuando existen múltiples prefijos locales en un enlace local.
- ✓ **Home subnet prefix:** Es el prefijo de subred IP correspondiente a la dirección local (*Home Address*) de un nodo móvil (*Mobile Node, MN*).
- ✓ **Home link:** Es el enlace en el cual se define el prefijo de subred local del nodo móvil (*Mobile Node, MN*).
- ✓ **Correspondente node:** Es un nodo con el cual el nodo móvil (*Mobile Node, MN*) se está comunicando. El nodo correspondiente (*Correspondent Node, CN*) puede ser cualquier nodo estacionario o móvil.
- ✓ **Foreign subnet prefix:** Es cualquier prefijo de subred IP diferente al prefijo de subred local del nodo móvil (*Mobile Node, MN*).



- ✓ **Foreign link:** El enlace foráneo es cualquier enlace diferente al enlace local del nodo móvil (*Mobile Node, MN*).
- ✓ **Care-of address:** Es la dirección asociada con un nodo móvil (*Mobile Node, MN*) mientras visita un enlace foráneo; el prefijo de subred de esta dirección IP es el prefijo de subred foráneo. Entre las múltiples direcciones *Care-of address (CoA)* que un nodo móvil (*Mobile Node, MN*) puede tener en algún momento, la primera dirección registrada con el agente local (*Home Agent, HA*) del nodo móvil (*Mobile Node, MN*) para una dirección local (*Home Address*) dada es llamada su *Care-of address* primaria.
- ✓ **Home agent:** Es un *router* en el enlace local del nodo móvil (*Mobile Node, MN*) con el cual el *MN* tiene registrada su actual *Care-of address (CoA)*. Mientras el *MN* esta lejos de su enlace local, el agente local (*Home Agent, HA*) intercepta los paquetes en el enlace local destinados a la dirección local (*Home Address*) del *MN*, los encapsula y los tuneliza a la *CoA* registrada del *MN*.
- ✓ **Binding:** Un *binding* equivale a la asociación de la dirección local (*Home Address*) de un nodo móvil (*Mobile Node, MN*) con una *Care-of address (CoA)* para ese *MN*. [2]

## 1.2 GENERALIDADES

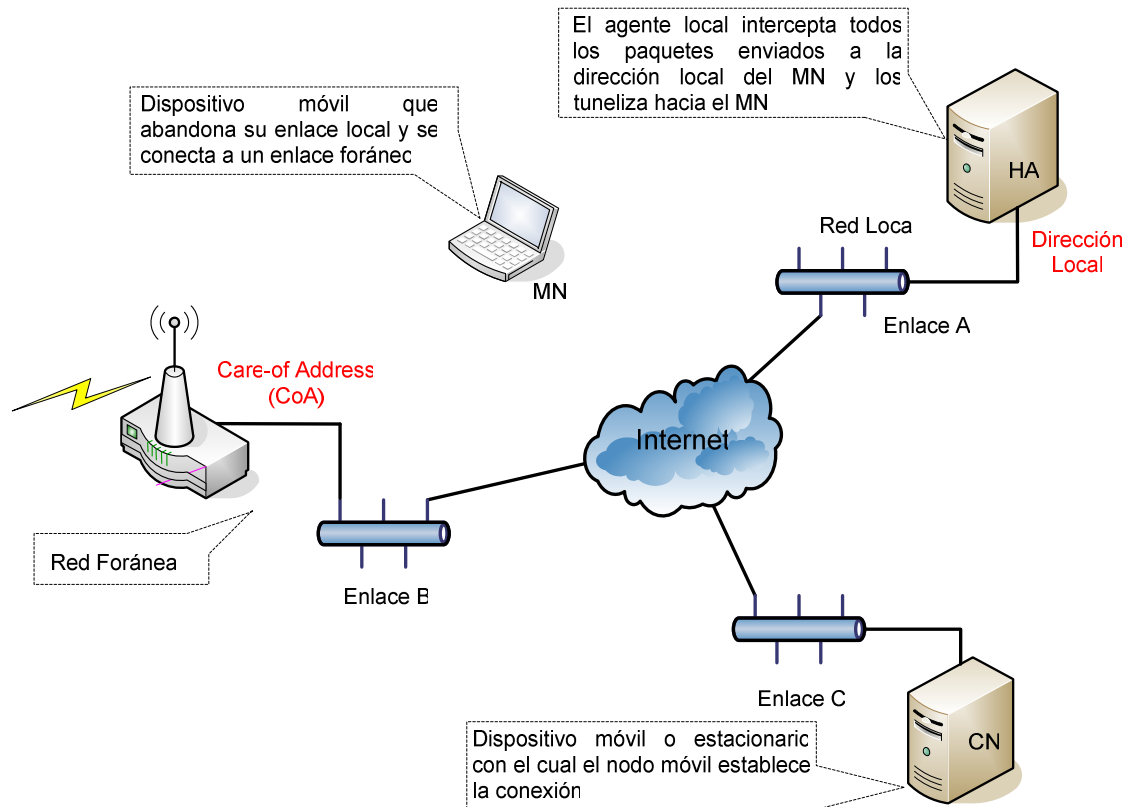
El soporte de movilidad para redes *MIPv6* [2] se describe en la *RFC 3775* y se conoce como *Mobile IPv6* [2], este protocolo tiene como característica fundamental permitir a los nodos móviles mantener la comunicación con otros nodos aun cuando los nodos móviles se desplacen de un enlace a otro, haciendo transparente el movimiento realizado.

Se espera que un *MN* siempre sea direccionable en su dirección local (*Home Address*), si está actualmente vinculado a su enlace local o fuera de él. La dirección local es una dirección *IP* asignada al *MN* dentro de su prefijo de subred local en su enlace local. Mientras un *MN* esta dentro de su área local, los paquetes direccionados a su dirección local son dirigidos al enlace local del *MN*.

Mientras un *MN* esta conectado a algún enlace foráneo lejos de su área local, este también es direccionable a una o mas *Care-of Addresses (CoAs)*. Una *CoA* es una dirección *IP* asociada con un *MN* que tiene el prefijo de subred de un enlace foráneo particular. El *MN* puede adquirir su *CoA* a través de mecanismos habituales de *IPv6* [6]. Mientras el *MN* permanece en esta localización, los paquetes direccionados a esta *CoA* son dirigidos al *MN*. El *MN* puede además

aceptar paquetes de varias CoAs, cuando se esta moviendo pero todavía es accesible al enlace previo. En la figura 1 se muestra el escenario para *Mobile IPv6* [2].

Figura 1: Escenario para Mobile IPv6.



Fuente: Autores.

La asociación entre la dirección local (*Home Address*) de un *MN* y la *CoA* es conocida como un *binding* por el *MN*. Mientras esta lejos de su área local, un *MN* registra su *CoA* primaria con un *router* el cual cumple las funciones de agente local (*Home Agent, HA*) y se encuentra en su enlace local. El *MN* realiza este registro enviando un mensaje *Binding Update (BU)* al *HA*. El *HA* responde al *MN* retornando un mensaje *Binding Acknowledgement*.

Cualquier nodo que establezca comunicación con el *MN* es conocido como nodo correspondiente (*Correspondent Node, CN*) y puede ser cualquier nodo fijo o móvil. Los *MNs* pueden proporcionar información acerca de sus actuales localizaciones a los *CNs*. Esto sucede con el registro correspondiente.

Existen dos posibles maneras para la comunicación entre el *MN* y un *CN*. El primer modelo, consiste en establecer un túnel bidireccional, en el *CN* no requiere soporte de *Mobile IPv6* [2] y esta disponible incluso si el *MN* no tiene registrado su actual conexión con el *CN*. Los paquetes del *CN* son enrutados al *HA* y tunelizados al *MN*. Los paquetes para el *CN* son tunelizados de el *MN* al *HA* y enrutados normalmente de la red local al *CN*. En este modelo, el *HA* usa descubrimiento de *proxy* vecino para interceptar algunos paquetes direccionados a la dirección local (*Home Address*) del *MN* en el enlace local. Cada paquete interceptado es tunelizado hacia la *CoA* primaria del *MN*. Esta tunelización es realizada utilizando encapsulación para *IPv6* [6].

El segundo modelo, se refiere a la optimización de la ruta, requiere que el *MN* registre su actual conexión en el *CN*. Los paquetes del *CN* pueden ser enrutados directamente a la *CoA* del *MN*. Cuando es enviado un paquete, el *CN* verifica si el *MN* esta registrado, si es así los paquetes son enrutados a la *CoA* con destino del *MN*.

Enrutar los paquetes directamente a la *CoA* del *MN* permite una ruta de comunicación mas corta. De igual manera se elimina congestión en el *HA* del *MN* y el enlace local. Asimismo, se reduce el impacto de cualquier posible falla de el *HA* o las redes que se comunican con el.

Cuando son enrutados los paquetes directamente al *MN*, el *CN* establece como dirección destino en la cabecera, la *CoA* del *MN*. Un nuevo tipo de cabecera de enrutamiento es adicionada al paquete para llevar la dirección local (*Home Address*) del *MN*. El *MN* establece como dirección fuente en la cabecera del paquete a su actual *CoA* y adiciona una nueva cabecera que contendrá su dirección local. La inclusión de la dirección local en estos paquetes hace que el uso de la *CoA* se transparente sobre la capa de red.

*MIPv6* [2] provee soporte para múltiples *HAs* y limita el soporte para la reconfiguración de la red local. En estos casos, el *MN* puede no saber la dirección *IP* de su propio *HA* e incluso el prefijo de subred local puede cambiar con el tiempo. Un mecanismo conoce cuando permitir que un *MN* descubra dinámicamente la dirección *IP* de un *HA* en su enlace local, incluso cuando el *MN* es de su enlace local. Los *MNs* pueden también adquirir nueva información acerca de los prefijos de subred local a través del mecanismo. [2]

### 1.3 MENSAJES

*Mobile IPv6* [2] requiere el intercambio de información adicional. Todos los nuevos mensajes usados en *Mobile IPv6* [2] son definidos como opciones de destino de *IPv6* [6]. Estas opciones son usadas en *IPv6* [6] para llevar información adicional que necesita ser examinada solamente por el nodo de destino del paquete.

Las siguientes cuatro nuevas opciones de destino son definidas en *Mobile IPv6* [2]:

- ✓ **Binding Update (BU):** Esta opción es usada por un nodo móvil (*Mobile Node, MN*) para informar a su agente local (*Home Agent, HA*) o a cualquier otro nodo correspondiente (*Correspondent Node, CN*) a cerca de su actual *Care-of Address (CoA)*.
- ✓ **Binding Acknowledgement (BAck):** Esta opción es usada para confirmar la recepción de un *Binding Update (BU)*, si el reconocimiento fuese pedido.
- ✓ **Binding Request (BR):** Esta opción es usada por cualquier nodo para solicitar a un nodo móvil (*Mobile Node, MN*) el envío de un *BU* con la actual *CoA*.
- ✓ **Home Address (HA):** Es usada en un paquete enviado por un *MN* para informar al receptor de este paquete, sobre de la dirección local del *MN*. [7]

### 1.4 ESTRUCTURA DE DATOS DE MOBILE IPv6

La especificación de *Mobile IPv6* [2] describe el protocolo en términos a las siguientes estructuras de datos conceptuales:

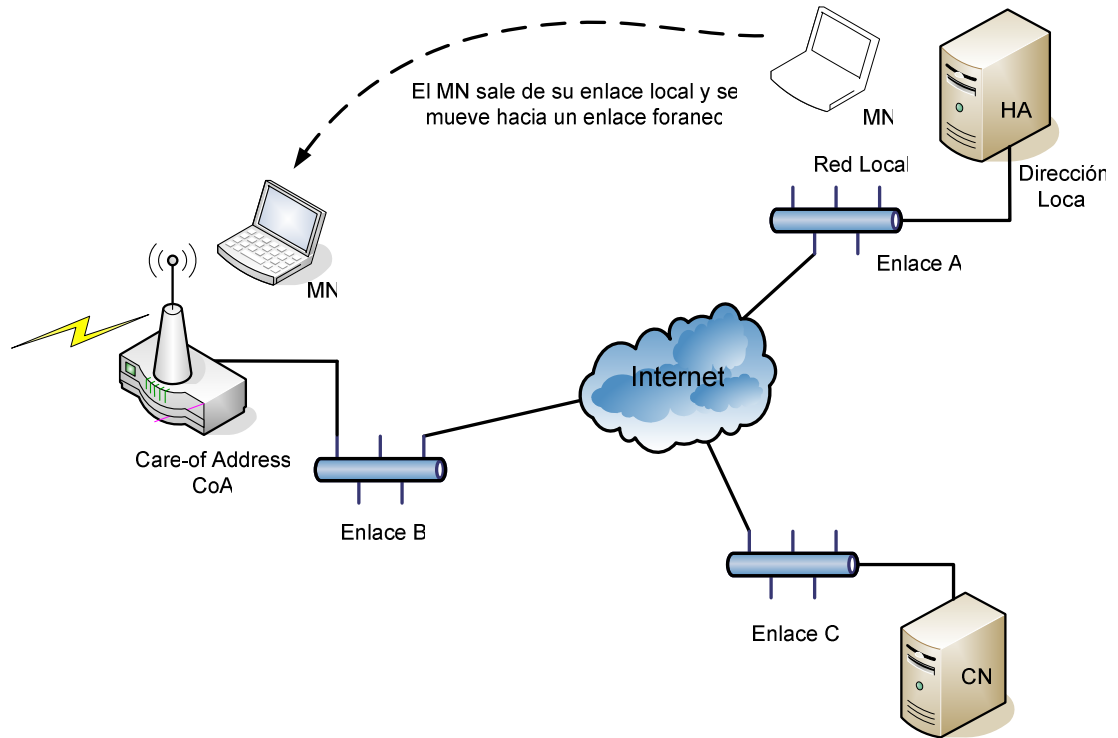
- ✓ **Binding Cache:** Cada nodo *IPv6* [6] tiene un *Binding Caché* o listado de ubicaciones de las *CoAs* que se usa para almacenar las ubicaciones de otros nodos. Si un nodo recibe un mensaje *BU*, agregará la nueva ubicación *CoA* a su *Binding Caché*. Cuando quiere enviar un paquete, buscará la *CoA* en el *Binding Caché* y envía el paquete a la *CoA* del nodo, usando una cabecera de enrutamiento.
- ✓ **Binding Update List:** Cada *MN* tiene una tabla llamada *Binding Update List* que se usa para guardar la información sobre cada actualización enviada por este *MN* cuando el tiempo de vida (*lifetime*) no ha expirado aún. Contiene todas los *BU* enviados a su *HA* y a cualquier *CN* (móvil o estacionario), es decir nodos que se están comunicando o se han comunicado en un tiempo corto con el *MN*.

- ✓ **Home Agents List:** Cada *HA* genera una lista, la cual contiene información sobre otros *HAs* en su subred. La información en esta lista es tomada de los anuncios de router multicast no solicitados (*Unsolicited Multicast Router Advertisements*), los cuales son enviados por todos los *HAs*. La información acerca de todos los *HAs* es usada por el mecanismo de descubrimiento de agente local dinámico (*Dynamic Home Agent Discovery Mechanism*). [7]

## 1.5 OPERACIÓN DE MOBILE IPv6

Los mecanismos de *Mobile IPv6* [2] descritos en la *RFC 3775*, se explicarán usando el escenario mostrado en la figura 2. El escenario muestra tres subredes o enlaces y tres nodos. Sobre el enlace A reside un *router* el cual ofrece servicios de agente local (*Home Agent, HA*). Este enlace es también el enlace local del nodo móvil (*Mobile Node, MN*). El *MN* se ha movido desde el enlace A hacia el enlace B. Adicionalmente hay un nodo correspondiente (*Correspondent Node, CN*) sobre el enlace C. Este nodo puede ser móvil o estacionario. La operación del protocolo consiste básicamente en que primero el *MN* obtiene su *Care-of Address (CoA)* en la red visitada en este caso el enlace B, a continuación el *MN* registra su *CoA* en su *HA*. El *CN* envía los paquetes al *HA* y el *HA* los tuneliza hacia el *MN*, a su vez el *MN* envía *Binding Updates (BUs)* al *HA* y *CN* donde se registra con este ultimo con el fin de optimizar la ruta, por consiguiente el *CN* envía los paquetes al *MN* directamente.

Figura 2: Operación de Mobile IPv6.



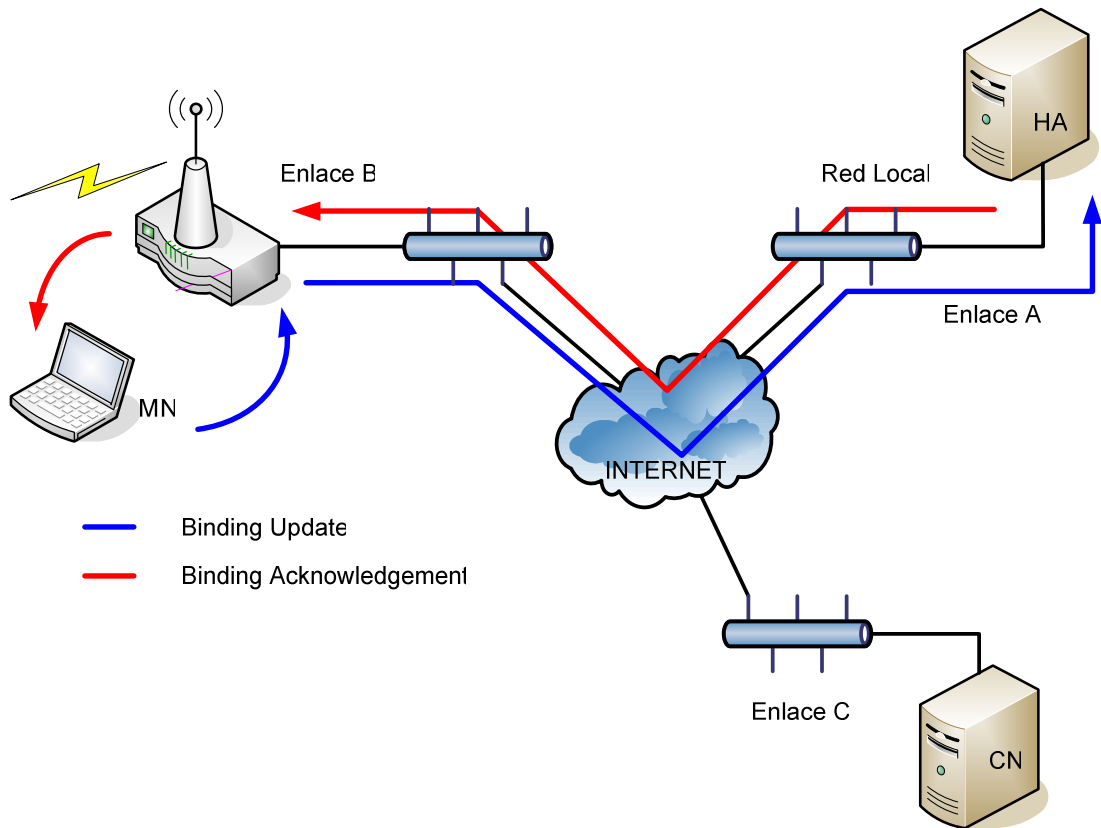
Fuente: Autores.

### 1.5.1 Registro con el Agente Local

Una vez un nodo móvil (*Mobile Node, MN*) detecta que se ha movido de un enlace a otro y ha descubierto un nuevo *router*, el *MN* realiza una autoconfiguración *stateful* o *stateless*. El *MN* usa la nueva dirección conformada como su *Care-of Address (CoA)*. El prefijo de su *CoA* es el prefijo del enlace visitado por el *MN*. Todos los paquetes direccionados a su *CoA* alcanzarán el *MN* sobre su actual enlace.

El *MN* registra su *CoA* con su agente local (*Home Agent, HA*) ubicado el enlace local. Es así que el *MN* envía un paquete a su *HA* el cual contiene una opción de destinación *Binding Update (BU)* tal como se puede observar en la figura 3. El *HA* registra su *binding* y retorna un paquete con una opción de destino *Binding Acknowledgement* hacia el *MN*. [2]

Figura 3: Registro del Nodo Móvil con su agente local.



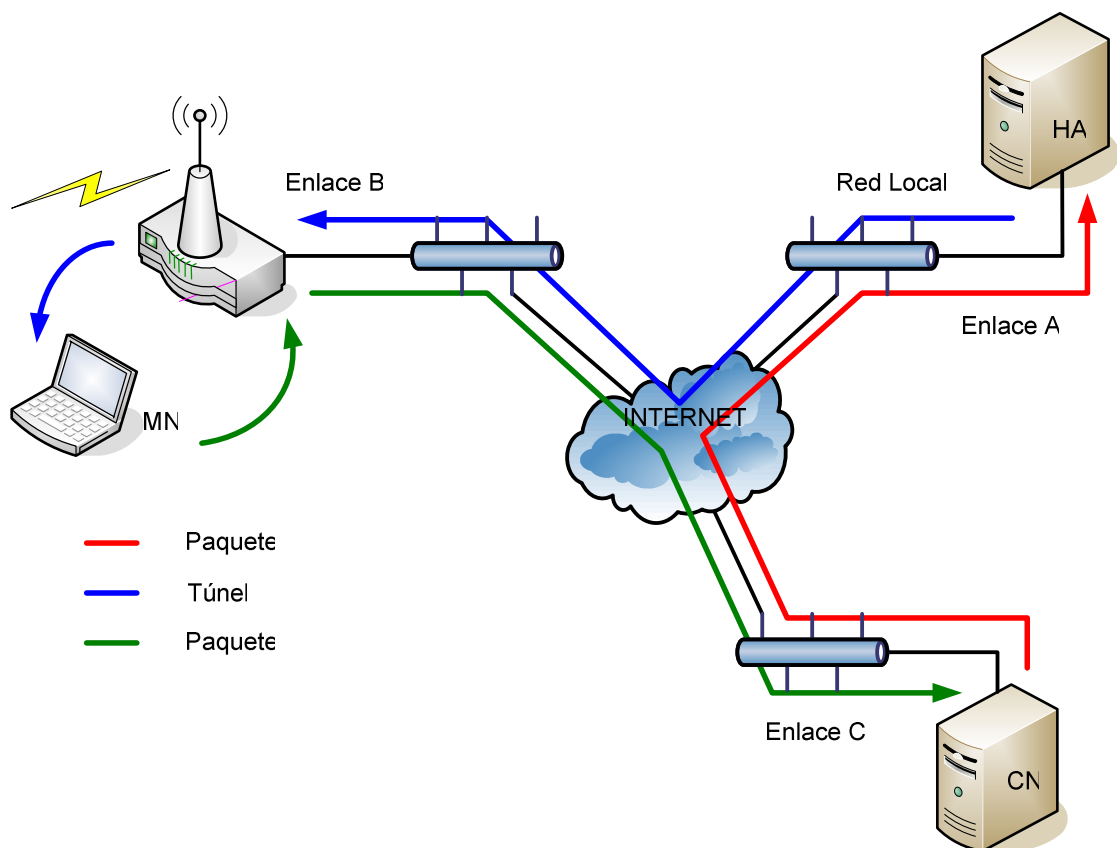
Fuente: Becerra S, Line Y, Amaya S, Cecilia. Simulación del protocolo Mobile IPv6 mediante la herramienta MOBIWAN bajo el simulador Network Simulator. Universidad Pontificia Bolivariana – Seccional Bucaramanga. 2005.

### 1.5.2 Enrutamiento Triángulo

El HA intercepta cualquier paquete direccionado hacia la dirección local (Home Address) del MN. Es así que este usa el protocolo *Proxy Neighbour Discovery* por lo que el HA envía varios anuncios de vecino sobre el enlace local en nombre del MN. Estos anuncian el cambio de la dirección de capa de enlace propia del HA por la dirección local del MN. El HA responde también a solicitudes de vecino en nombre del MN. Cada paquete interceptado es tunelizado hacia la CoA registrada del MN usando encapsulación IPv6 [6].

Si el *MN* envía paquetes a algún otro nodo, estos son enviados directamente al destino. El *MN* escribe en la dirección fuente de su paquete la *CoA* e incluye una opción de destino "*home address*", ya que la dirección local es estática (en contraste a la *CoA*), esto permite a cada *CN* el uso transparente de la *CoA*. [7]

Figura 4: Enrutamiento Triángulo.



Fuente: Becerra S, Line Y, Amaya S, Cecilia. Simulación del protocolo Mobile IPv6 mediante la herramienta MOBIWAN bajo el simulador Network Simulator. Universidad Pontificia Bolivariana – Seccional Bucaramanga. 2005.

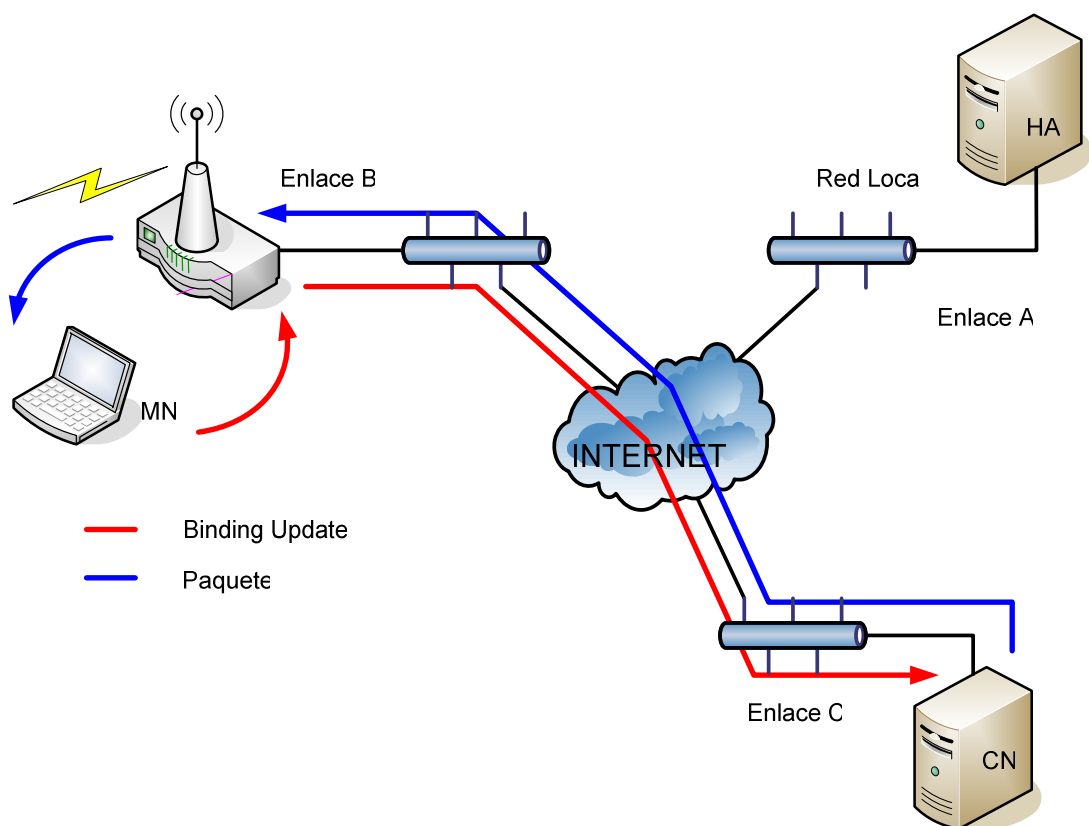
Si un *MN* se comunica con un *CN* cuando esta fuera de su red local, los paquetes son enrutados desde el *CN* hacia el *HA*, del *HA* al *MN* y desde el *MN* hacia el *CN*. Este enrutamiento es llamado "enrutamiento triángulo" y se puede observar en la figura 4. [2]



### 1.5.3 Optimización de la Ruta

Para evitar el enrutamiento triángulo un nodo móvil (*Mobile Node, MN*) puede enviar *Binding Updates (BUs)* hacia cualquier nodo correspondiente (*Correspondent Node, CN*) ya sea móvil o estacionario. Esto permite a los nodos *IPv6* [6] correspondientes almacenar la actual *Care-of Address (CoA)* y enviar paquetes directamente al *MN*. Ver figura 5.

Figura 5: Optimización de la Ruta



Fuente: Becerra S, Line Y, Amaya S, Cecilia. Simulación del protocolo Mobile IPv6 mediante la herramienta MOBIWAN bajo el simulador Network Simulator. Universidad Pontificia Bolivariana – Seccional Bucaramanga. 2005.

Cuando un nodo *IPv6* [6] va a enviar un paquete, primero chequea su *Binding Cache* para ésta dirección de destino. Si hay una entrada, éste enviará el paquete al *MN* usando una cabecera de enrutamiento (encapsulación *IPv6* [6]). La ruta especificada por esta cabecera de enrutamiento tiene dos saltos (Ver figura 6). El

primer salto es la CoA y el segundo salto es la dirección local (*Home Address*) del MN. Como resultado envía directamente el paquete a la CoA del MN. El MN recibe este paquete y lo envía al próximo salto especificado en la cabecera de enrutamiento. El próximo y final salto es la dirección local del MN, es así que el paquete quedará dentro del MN. Luego el paquete será procesado en la misma forma como si el MN estuviera en su red local. [7]

Figura 6: Cabecera de enrutamiento de un paquete enviado directamente al MN.



Fuente: Becerra S, Line Y, Amaya S, Cecilia. Simulación del protocolo Mobile IPv6 mediante la herramienta MOBIWAN bajo el simulador Network Simulator. Universidad Pontificia Bolivariana – Seccional Bucaramanga. 2005.

Salto 1= Care-of address del Nodo Móvil; Salto 2 = Dirección local del Nodo Móvil.

Si la tabla *Binding Cache* no tiene una entrada, este paquete será enviado normalmente. Luego este paquete es enrutado hacia la red especificada y recibido por el nodo destino. En caso de que el destino sea un MN, el cual esta fuera de su red local, este paquete será interceptado por el HA sobre el enlace local y entunelado hacia el MN.

Un MN, que configuró una nueva CoA como CoA primaria, tiene que registrar su nueva dirección a su agente local (*Home Agent, HA*) y al CN. Para este propósito el MN envía un BU informando su nueva CoA. Para asegurar, que el receptor destinado recibe este BU, el MN puede forzar al receptor a reconocer la recepción del BU respondiendo con un *Binding Acknowledgement*. Para provocar este reconocimiento, el MN fija el bit de *acknowledge* en el BU. Mientras espera la recepción del *Binding Acknowledgement* el MN continúa retransmitiendo los BU periódicamente. [7]

En caso de que un MN reciba un paquete desde un CN este es capaz de detectar, si el CN tiene ya una entrada en la tabla *Binding Cache*. Si el CN tiene una entrada del MN en la tabla *Binding Cache* direccionará el paquete directamente a la CoA del MN. De otra forma el CN envía este paquete a la dirección local (*Home*

*Address*) del *MN*. Allí el paquete es tunelizado por el *HA* hacia la *CoA* del *MN*. En este caso el *MN* recibe un paquete tunelizado. El *MN* puede enviar un *BU* al *CN* para habilitarlo, para enviar futuros paquetes directamente al *MN* sin ser tunelizados por el *HA*.

Al igual que el *MN* debe fijar el bit *acknowledgement* en los *BU* direccionados al *HA*, el *MN* puede también fijar el bit *acknowledgement* en los *BUs* enviados al *CN*. Si el *BU* no fue recibido por el *CN*, el *MN* reconocería esto al recibir paquetes tunelizados desde el *HA*.

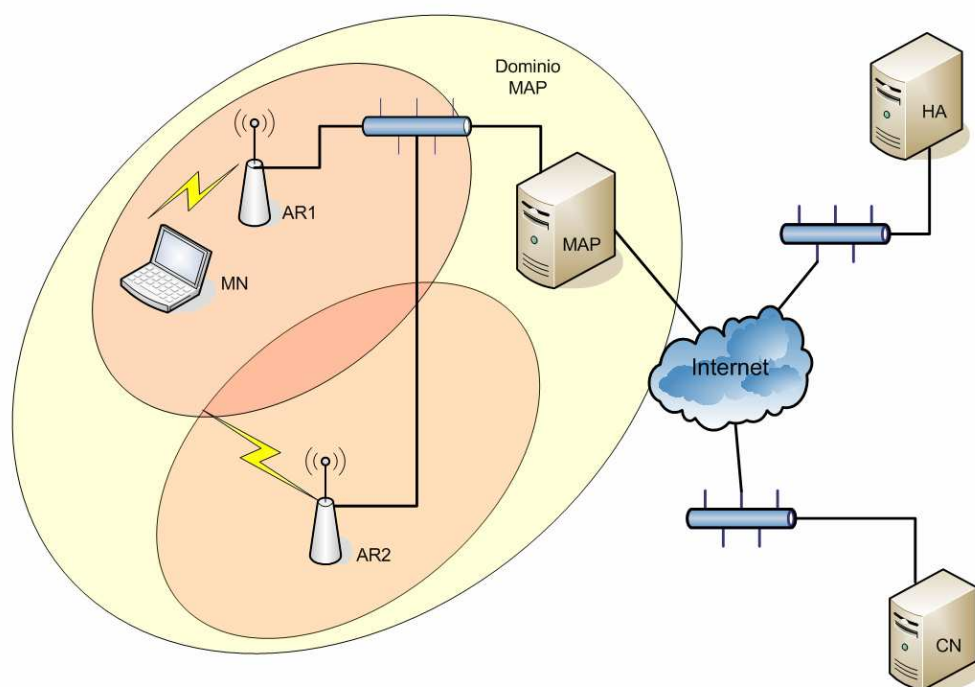
Antes de la expiración de una entrada de un *MN* en la *Binding Cache* el *CN* puede iniciar un refresco del *binding* enviando un *Binding Request* (solicitud de enlace) hacia el *MN*. Al recibir *Binding Request* el *MN* puede responder con un *BU*. [7]

## 2. HIERARCHICAL MOBILE IPv6

### 2.1 INTRODUCCIÓN

*Hierarchical Mobile IPv6 (HMIPv6)* [1] se creó como una extensión al protocolo *Mobile IPv6 (MIPv6)* [2], con el fin de mejorar la gestión de movilidad local principalmente mediante la reducción de la cantidad de señalización entre el nodo móvil (*Mobile Node, MN*), nodo correspondiente (*Correspondent Node, CN*) y su agente local (*Home Agent, HA*), esto se logra adicionando un nuevo nodo llamado *Mobility Anchor Point (MAP)*, el cual es el encargado de gestionar la movilidad del *MN* dentro de su dominio permitiendo un mejor desempeño en términos de velocidad de *handover*. *HMIPv6* [1] permite al igual que *MIPv6* [2] movilidad dentro o entre diferentes tipos de redes de acceso y reduce significativamente la latencia del *handover* en los casos de micromovilidad en comparación con *MIPv6* [2].

Figura 7: Dominio de HMIPv6.



Fuente: H. Soliman, C. Castelluccia, K. El Malki, L. Bellver. *Hierarchical Mobile IPv6 Mobility Management (HMIPv6)*. RFC 4140. Agosto 2005.

Básicamente el *MAP* actúa como un agente local, donde todo *MN* que entra al dominio del *MAP* recibe anuncios de *router* (*Router Advertisements, RAs*), los cuales contienen información sobre uno o más *MAPs* locales. El *MN* puede enlazar su ubicación actual *On-link Care-of Address (LCoA)* con una dirección *Regional Care-of Address (RCoA)* de subred de los *MAPs*. Por consiguiente el *MAP* recibirá todos los paquetes dirigidos al *MN*, los encapsula y los envía directamente a la dirección actual del *MN*. Si el *MN* cambia su dirección actual dentro del dominio del *MAP*, este solamente necesita registrar la nueva dirección *LCoA* con el *MAP* por lo tanto la *RCoA* no cambiara haciendo transparente la movilidad del *MN* ante el *CN*. Por otra parte la *RCoA* solamente necesita ser registrada con el *CN* y el *HA*. La figura 7 muestra el dominio de *HMIPv6* [1].

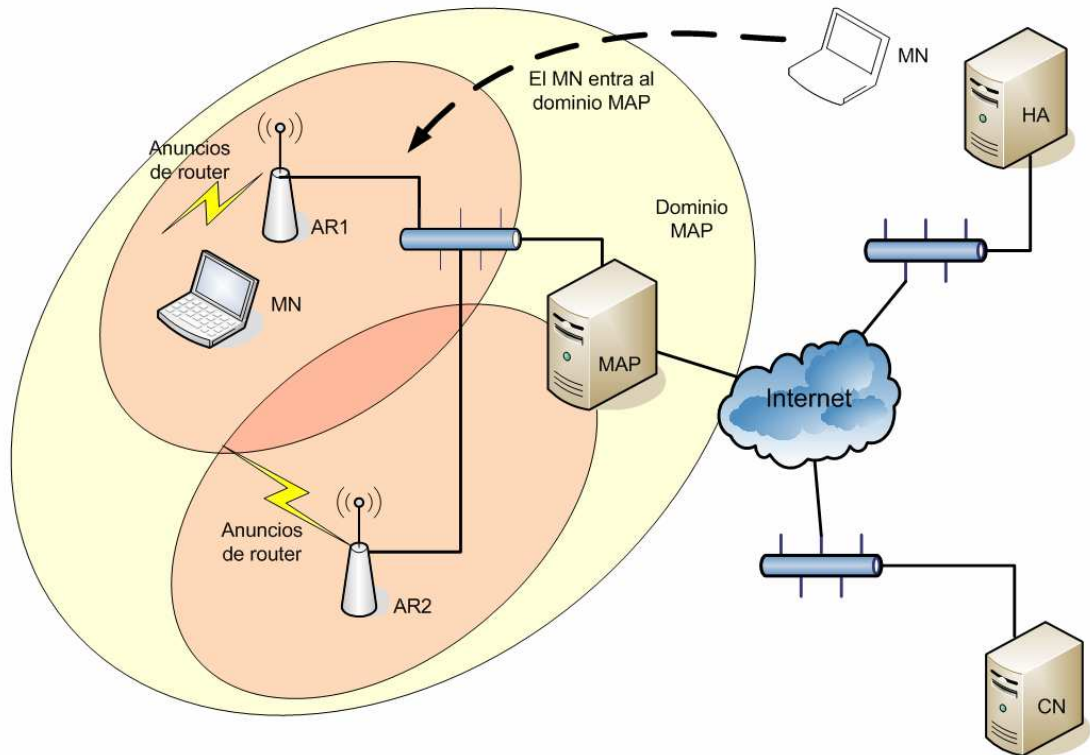
El dominio del *MAP* esta definido por la cobertura de los *routers* de acceso (*Access Routers, ARs*), lo cuales están encargados de anunciar información del *MAP* al *MN* por medio de los *RAs*. [1]

## 2.2 OPERACIÓN DE HMIPv6

A continuación se describe en términos generales la operación del protocolo *HMIPv6* [1]. La operación de los elementos de red se muestra en las secciones siguientes.

El nodo móvil (*Mobile Node, MN*) descubre la dirección global del *MAP* cada vez que llega al dominio del *MAP* de una red visitada. Con el fin de informar al *MN* acerca de la presencia del *MAP* se implemento una nueva opción para los anuncios de *router* (*Router Advertisements, RAs*), esto hace parte del procedimiento de descubrimiento del *MAP* (Ver figura 8). La dirección del *MAP* es almacenada en el *router* de acceso (*Access Router, AR*) y comunicada al *MN* mediante los *RAs*, por consiguiente todos los *router* de acceso (*Access Routers, ARs*) pertenecientes al dominio del *MAP* deben advertir la dirección *IP* del *MAP*. La fase de descubrimiento del *MAP* también informa al *MN* acerca de la distancia desde el *MN* al *MAP*.

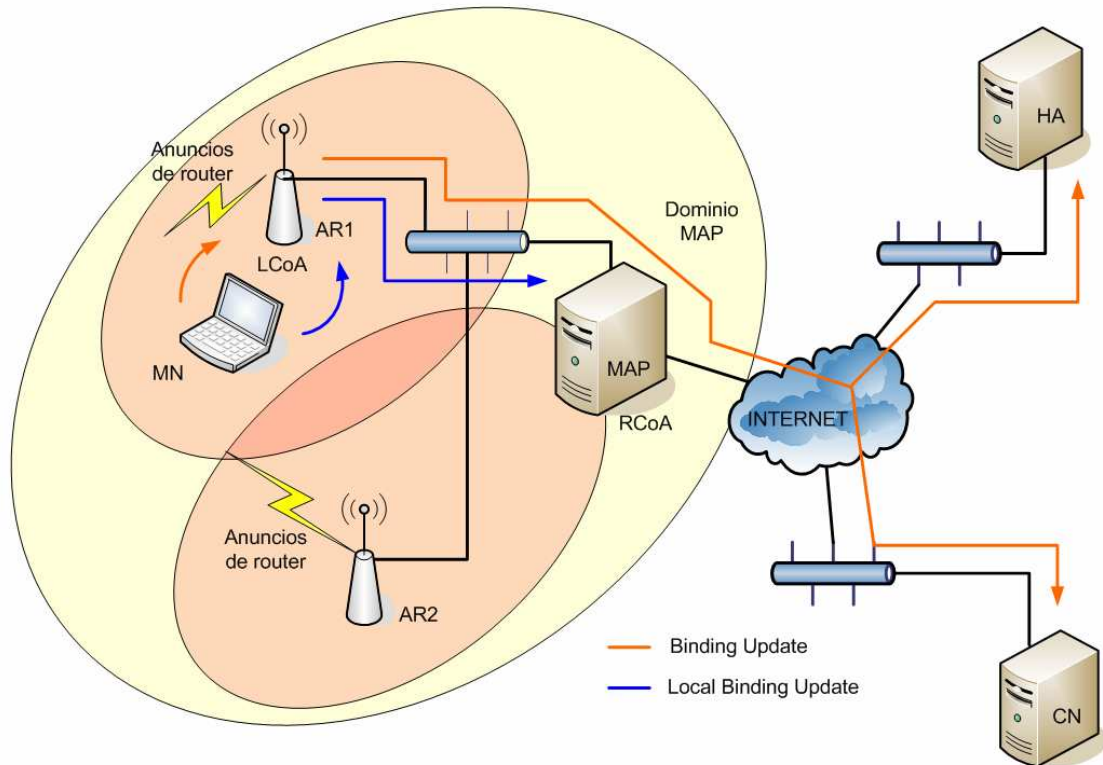
Figura 8: Desplazamiento del MN.



Fuente: Autores.

Cada vez que el *MN* se mueve desde una subred a la próxima se realiza el proceso de descubrimiento del *MAP*. A si mismo el *MN* detectará si esta en el mismo dominio *MAP*, mediante la opción *MAP* contenida en el anuncio de *router* (*Router Advertisement, RA*) enviado desde el *AR*. Si el *MN* continua recibiendo la misma opción *MAP* indica que éste se mueve dentro del mismo dominio *MAP*. De lo contrario el *MN* debe actuar sobre el cambio enviando *Bindind Updates (BUs)* a su agente local (*Home Agent, HA*) y nodo correspondiente (*Node Correspondent, NC*) tal como se observa en la figura 9. Si el *MN* no tiene la opción de *MAP* entonces el descubrimiento del *MAP* no será desarrollado, en este caso el *MN* usará el protocolo *MIPv6* [2] para su gestión de movilidad. [1]

Figura 9: Mensajes de actualización y registro del MN.



Fuente: Autores.

Por otra parte si el MN tiene la opción de MAP este debe escoger usar dicha implementación. Si es así, el MN se registra con el MAP enviando un *Local Binding Update (LBU)* (Ver figura 9). El MAP almacena esta información en su *Binding Cache* para poder enviar paquetes a su destino final cuando los recibe desde el CN o HA.

El MN necesita conocer el emisor original de cualquier paquete recibido para determinar si la optimización de la ruta es requerida. Debido a que el MAP no modifica el contenido del paquete original esta información esta disponible al MN. El procesamiento original de los paquetes recibidos se realiza de igual manera que en *MIPv6* [2] permitiendo al MN obtener la información necesaria.

Un MN puede registrarse con más de un MAP simultáneamente y usar cada dirección de MAP para un grupo específico de CNs. Esto evita enviar todos los paquetes vía al más alto MAP en la jerarquía, teniendo como resultado un uso

mas eficiente del ancho de banda. El *MN* también puede usar su actual *LCoA* como una *Care-of Address (CoA)* debido a que este no debe presentar una *RCoA* desde una subred del *MAP* como una *LCoA* en un *BU* enviado a otro *MAP* por que la *LCoA* incluida en el *BU* debe ser la dirección del *MN* derivada desde el prefijo anunciado sobre su enlace. [1]

## 2.3 OPERACIÓN DE LOS ELEMENTOS DE RED

### 2.3.1 Operación del Nodo Móvil

La movilidad del nodo móvil (*Mobile Node, MN*) dentro del dominio de un *MAP* requiere que este configure dos *Care-of Address (CoA)*, una *Regional Care-of Address (RCoA)* sobre el enlace del *MAP* y una *On-link Care-of Address (LCoA)*. La *RCoA* es formada combinando el identificador de interfase del *MN* y el prefijo de subred en la opción del *MAP*, el *MN* envía un *Local Binding Update (LBU)* al *MAP* con las banderas A y M fijadas. El *LBU* incluye la *RCoA* del *MN* en la opción de "home address". La *LCoA* es usada como la dirección fuente del *LBU* y representa la dirección en la que se encuentra el *MN*. Este *LBU* enlazara la *RCoA* del *MN* a su *LCoA*. El *MAP* el cual actúa como un agente local, envía un *Binding Acknowledgement* al *MN*, con el fin de determinar si el *binding* fue exitoso (Ver figura 10). De lo contrario el *Binding Acknowledgement* informara al *MN* con el código de falla apropiado si se presento un error. El *MN* debe silenciosamente ignorar un *Binding Acknowledgement* que no contenga una cabecera de enrutamiento la cual incluye la *RCoA* del *MN*.

Luego de registrarse con el *MAP*, el *MN* debe registrar su nueva *RCoA* con su agente local (*Home Agent, HA*) enviando un *Binding Update (BU)* de igual manera como se realiza en *MIPv6* [2]. La dirección local (*Home Address*) del *MN* es usada en la opción de "home address" y la *RCoA* es usada como la *CoA* en el campo de dirección fuente. El *MN* puede también enviar un *BU* similar que especifique el enlace entre la dirección local (*Home Address*) y la *RCoA* a su actual nodo correspondiente (*Correspondent Node, CN*).

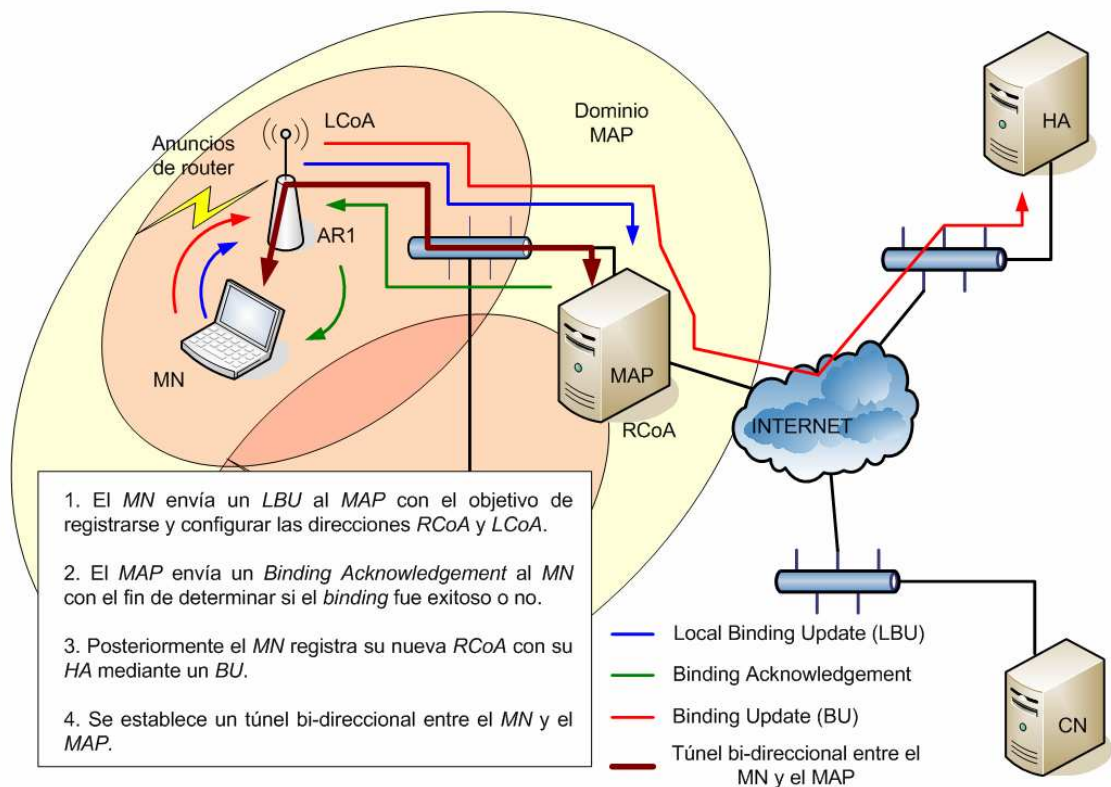
Antes de registrarse con su *HA*, el *MN* debe esperar por el *Binding Acknowledgement* enviado desde el *MAP*. Debido a que cuando enlaza la *RCoA* con el *HA* y el *CN*, el tiempo de vida del enlace no debe ser más largo que el tiempo de vida del enlace del *MN* con el *MAP*, el cual se recibe en el *Binding Acknowledgement*.

Siguiendo un registro exitoso con el *MAP*, se establece un túnel bi-direccional entre el *MN* y el *MAP* tal como se puede observar en la figura 10. Todos los paquetes enviados por el *MN* son tunelizados hacia el *MAP*. La cabecera más



externa contiene la LCoA del MN en el campo de dirección fuente y la dirección del MAP en el campo de dirección destino. La cabecera más interna contiene la RCoA del MN en el campo de dirección fuente y la dirección del CN en el campo de dirección destino. Similarmente todos los paquetes direccionados a la RCoA del MN son interceptados por el MAP y tunelizados hacia la LCoA del MN. Esta especificación permite al MN usar más de una RCoA si este recibe más de una opción MAP. En este caso, el MN debe desarrollar el procedimiento de LBU para cada RCoA por separado. Además, el MN no debe usar una RCoA derivada desde el prefijo de un MAP como una CoA en su LBU para ser enviado a otro MAP diferente debido a que se forzaría a los paquetes a ser encapsulados varias veces cuando estos son enviados al MN. Esta forma de jerarquía multinivel reducirá el desempeño y la eficiencia del protocolo. [1]

Figura 10: Operación del Nodo Móvil.



Fuente: Autores.

El *MAP* recibirá paquetes direccionados hacia la *RCoA* del *MN* desde el *HA* o *CNs*. Los paquetes serán tunelizados desde el *MAP* hacia la *LCoA* del *MN*. El *MN* desencapsulará los paquetes y los procesara en la manera normal.

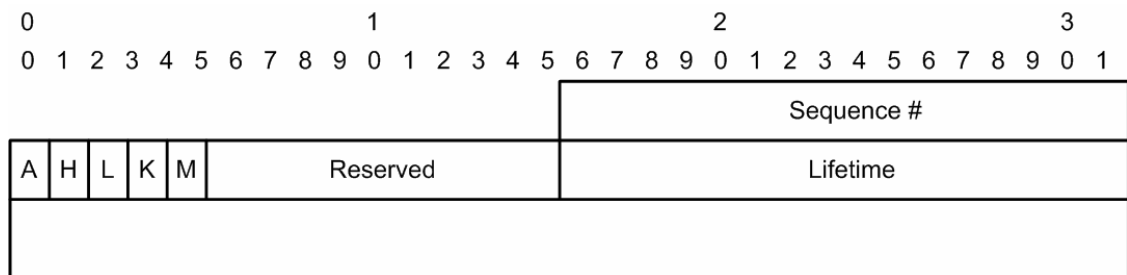
Cuando el *MN* se mueve dentro del mismo dominio del *MAP*, este debe solamente registrar su nueva *LCoA* con su *MAP*. En este caso, la *RCoA* permanece igual.

Por otra parte el *MN* puede enviar un *BU* conteniendo su *LCoA* en vez de su *RCoA* a los *CNs*, si los *CNs* están conectados a su mismo enlace. Los paquetes serán luego enrutados directamente sin ir a través del *MAP*.

### 2.3.1.1 Local Binding Update

Con el propósito de mantener la comunicación el *MN* necesita enviar mensajes de actualización su *HA* y *CN*, estos mensajes son conocidos con el nombre de *Binding Update (BU)*. En *HMIPv6* [1] el *MN* cuando entra a un nuevo dominio *MAP* debe registrarse con este, por lo tanto también requiere enviar *BUs* al *MAP*, este *binding* recibe el nombre de *Local Binding Update (LBU)* y se diferencia del *BU* en que al *LBU* se le adiciona una bandera *M* como se puede observar en la figura 9. Si *M* esta fijo en 1 esto indica que se ha efectuado un registro de *MAP*. [1]

Figura 11: Local Binding Update.



Fuente: H. Soliman, C. Castelluccia, K. El Malki, L. Bellver. Hierarchical Mobile IPv6 Mobility Management (HMIPv6). RFC 4140. Agosto 2005.

### 2.3.1.2 Actualización de MAPs

Cuando un *MN* se mueve dentro de un nuevo dominio *MAP* y con el propósito de acelerar el *handover* entre *MAPs* y reducir la pérdida de paquetes, el *MN* debe enviar un *LBU* a su *MAP* previo o anterior, especificando su nueva *LCoA*. Los paquetes en transito que alcanzan el *MAP* previo son luego enviados a la nueva *LCoA*.

El administrador puede limitar al *MAP* de enviar paquetes a las *LCoAs* afuera del dominio *MAP*. Sin embargo, se recomienda que se les permita a los *MAPs* enviar paquetes a *LCoAs* asociadas con algunos de los *router* de acceso (*Access Routers, ARs*) de dominios del *MAP* cercanos, siempre y cuando estén localizados dentro del mismo dominio administrativo.

### 2.3.1.3 Envío de paquetes al Nodo Correspondiente

El *MN* puede comunicarse con un *CN* a través de el *HA* o mediante una ruta optimizada. Si el *MN* se comunica directamente con el *CN*, el *CN* tiene una entrada de *Binding Cache* para el *MN*, el *MN* usa la *RCoA* como la *CoA* y crea una entrada en la *Binding Cache* del *CN*, esta dirección es usada como dirección fuente. Tal como se describe en *MIPv6* [2], el *MN* debe también incluir una opción de “*home address*” en los paquetes salientes. La opción de “*home address*” debe contener la dirección del *CN*.

### 2.3.2 Operación del MAP

El *MAP* actúa como un *HA*, este intercepta todos los paquetes direccionados hacia los *MNs* registrados y los entunela hacia la correspondiente *LCoA*, la cual es almacenada en su *Binding Cache*. Ver figura 12

Figura 12: Operación del MAP.



Fuente: Autores.

Las direcciones locales (*Home Address*) de los *MNs* son desconocidas para el *MAP*. El *MN* envía un *LBU* hacia el *MAP* con las banderas *M* y *A* establecidas. El propósito del *BU* es informar al *MAP* que el *MN* tiene formada una *RCoA* (contenida en el *BU* como una *home address*). Si el procedimiento del *LBU* se lleva a cabo con éxito, el *MAP* debe retornar un *Binding Acknowledgement* hacia el *MN*, indicando que el registro fue exitoso. El *Binding Acknowledgement* debe incluir una cabecera de enrutamiento que contiene la *RCoA* del *MN*.

El *MAP* permite aceptar paquetes tunelizados desde el *MN*, siendo el *MN* el punto de entrada al túnel y el *MAP* el punto de salida del túnel.

El *MAP* actúa como un *HA*. Los paquetes direccionados a la *RCoA* son interceptados por el *MAP*, y luego los encapsula y enruta hacia la *LCoA* del *MN*.

Un *MAP* puede ser configurado con la lista de prefijos válidos que los *MNs* pueden usar para derivar las *LCoAs*. Permitiendo a los operadores de red reconocer cuando un *MN* continúa usando un *MAP* después de moverse a otro dominio diferente. Si un *MN* envía un *LBU* conteniendo una *LCoA* que no está en la lista de prefijos válidos del *MAP*, este podrá rechazar el *LBU*. [1]

## 2.4 DESCUBRIMIENTO DE MAP

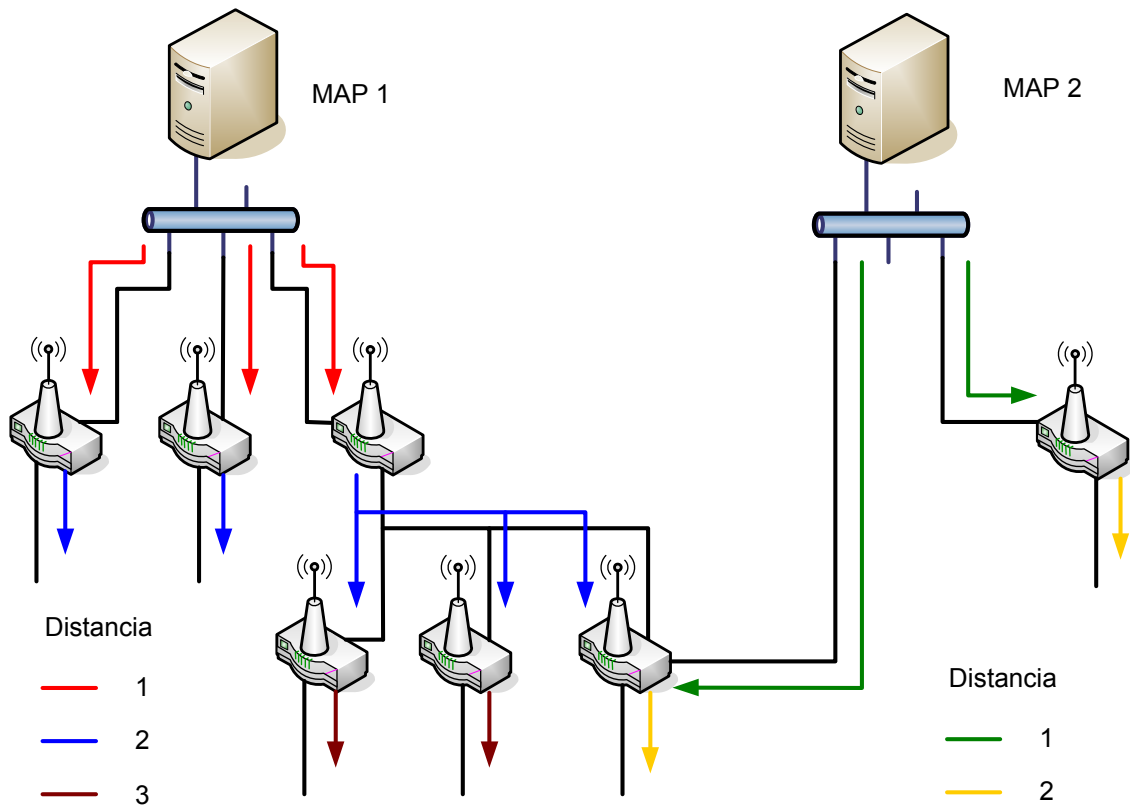
El descubrimiento del *MAP* conlleva a conocer como el nodo móvil (*Mobile Node, MN*) obtiene la dirección del *MAP* y el prefijo de red y como los *routers* de acceso (*Access Routers, ARs*) en un dominio descubren los *MAPs*. Para llevar a cabo este proceso se definen dos métodos diferentes.

El descubrimiento dinámico del *MAP* se basa en la propagación de la opción *MAP* contenida en los anuncios de *router* (*Router Advertisements, RAs*) enviados desde el *MAP* hacia el *MN* a través de los *routers* en una red de operadores. Esto requiere configuración manual del *MAP* y también que los *routers* reciban y propaguen la opción *MAP*.

Para garantizar una comunicación segura entre *routers*, los *RAs* que son enviados entre *routers* para el descubrimiento dinámico del *MAP* deben ser autenticados. En tal caso que esta autenticación no sea posible, un operador de red puede preferir configurar manualmente todos los *ARs* para enviar la opción de *MAP*.

La configuración manual de la información de la opción del *MAP* en *ARs* y otros *MAPs* en el mismo dominio es el mecanismo por defecto. Es posible configurar *ARs* y *MAPs* para habilitar mecanismos dinámicos para el descubrimiento del *MAP*. La figura 13 muestra en forma general el proceso de descubrimiento del *MAP*. [1]

Figura 13: Descubrimiento del MAP.



Fuente: Autores.

#### 2.4.1 Descubrimiento dinámico.

El proceso de descubrimiento del *MAP* puede ser desarrollado en diferentes formas. Los *RAs* son usados para el descubrimiento dinámico del *MAP* introduciendo una nueva opción. El *router* de acceso (*Access Router, AR*) se requiere para enviar la opción *MAP* en sus *RAs*. Esta opción incluye el vector de distancia del *MN*, la preferencia por un *MAP* en particular, el prefijo de red y la dirección *IP* global del *MAP*.

### 2.4.1.1 Operación del Router

Los *ARs* dentro de un dominio de *MAP* pueden ser configurados dinámicamente con la información relacionada respecto a las opciones *MAP*. Estos pueden obtener esta información mediante los *RAs* con opciones de *MAP*. Como se mencionó anteriormente cada *MAP* en la red necesita ser configurado con una preferencia por defecto, con la interfase correcta para enviar esta opción activa y la dirección *IP* a ser enviada. El valor inicial del campo (distancia) puede ser fijo a un valor defecto de 1 y no debe ser fijo a cero. Los *routers* en el dominio *MAP* deben ser configurados para reenviar la opción *MAP* sobre ciertas interfaces.

Una vez se recibe un *RA* con la opción *MAP*, el *router* receptor debe copiar la opción y reenviarla después incrementando el campo de la distancia por uno. Si el *router* receptor es un *MAP*, este debe enviar su propia opción, junto con la opción recibida, en el mismo anuncio. Si un *router* recibe más de una opción *MAP* enviada por el mismo *MAP* desde dos diferentes interfaces, este debe escoger la opción con el campo de distancia más pequeño.

De esta manera, la información acerca de uno o más *MAPs* puede ser transmitida a un *MN*. Además, desarrollando la fase de descubrimiento en esta forma, diferentes *MAPs* son capaces de cambiar sus preferencias dinámicamente basados en las políticas locales, en la sobrecarga del nodo o usando protocolos para compartir carga. [1]

### 2.4.1.2 Operación del MAP

Un *MAP* se configura para enviar su opción o transmitir opciones de *MAP* pertenecientes a otros *MAPs* sobre ciertas interfaces. La elección de interfaces es hecha por el administrador de red y depende de la topología de la red. Un valor de preferencia por defecto de 10 puede ser asignado a cada *MAP*. El *MAP* puede cambiar su valor de preferencia en cualquier momento debido a varias razones las cuales pueden ser sobrecarga del nodo o compartimiento de carga. Un valor de preferencia de cero significa que el *MAP* no debe ser escogido por nuevos *MNs*. Este valor podría ser alcanzado en casos de sobrecarga de nodo o fallas parciales de nodo.

La opción *MAP* es propagada hacia los *ARs* pertenecientes a este dominio. Cada *router* a lo largo del camino hacia un *AR* incrementara el campo de distancia por uno.

## 2.4.2 Descubrimiento manual

El descubrimiento manual depende directamente de la gestión que realiza el *MN* el cual se encarga de buscar la opción *MAP* cuando recibe un *RA* y de registrarse con el *MAP* de más alta preferencia, esto debido a que una o más opciones de *MAP* con direcciones *IP* diferentes pueden ser encontradas. Teniendo en cuenta que si no se encuentra por lo menos una opción *MAP*, el *MN* debe usar el protocolo *MIPv6* [2] para la comunicación con el nodo correspondiente (*Correspondent Node, CN*). [1]

### 2.4.2.1 Operación del Nodo Móvil

Un *MN* puede escoger registrarse con un *MAP* por encima de otro, dependiendo del valor recibido en el campo distancia, previniendo que el valor de preferencia debe ser superior a cero.

Una opción *MAP* que contiene un valor de tiempo de vida valido de cero significa que este *MAP* no puede ser seleccionado por el *MN*. Un tiempo de vida valido de cero indica que el *MAP* falla. Si esta opción es recibida el *MN* debe escoger otro *MAP* y crear nuevos *bindings*.

Los *MNs* que tienen acceso a varios *ARs* simultáneamente deben usar una *LCoA* sobre el enlace definido por el *AR* que advierte su *MAP* actual. Un *MN* debe almacenar las opciones recibidas para escoger al menos un *MAP* para registrarse con el. Almacenar las opciones es esencial, por que estas serán comparadas con otras opciones recibidas cada vez que se presenta un movimiento.

UN *MN* puede escoger registrarse con más de un *MAP* simultáneamente o usar ambas direcciones, la *Regional Care-of Address (RCoA)* y su *On-link Care-of Address (LCoA)* como *Care-of Address (CoA)* simultáneamente con diferentes *CNs*.

## 2.5 SELECCIÓN DEL MAP POR EL NODO MÓVIL

*HMIPv6* [1] provee un mecanismo flexible para la gestión de movilidad global dentro de una red visitada. Como se explicó anteriormente, un *MAP* puede existir en cualquier lugar en la red del operador. Varios *MAPs* pueden ser localizados dentro del mismo dominio independientemente uno del otro. Además, la superposición de dominios de *MAP* es permitida y recomendable. Las jerarquías dinámicas y estáticas son soportadas. Cuando el nodo móvil (*Mobile Node, MN*)

recibe un anuncio de *router* (*Router Advertisement, RA*) con opción *MAP*, este debe desarrollar acciones de acuerdo a los mecanismos de detección de movimiento. En una red *HMIPv6* [1], el *MN* debe tener la prioridad de desarrollar nuevos *bindings* y a si mismo restarle importancia a la liberación de *bindings* existentes. Lo que significa que el *MN* debe registrarse con cualquier nuevo *MAP* anunciado por el *router* de acceso (*Access Router, AR*) que tenga una intensidad de señal mayor. El método por el cual el *MN* determina si el *MAP* es un nuevo *MAP* se describe en la sección 2.5.1. El *MN* no debe liberar *bindings* existentes hasta que este no reciba la opción *MAP*, o que reciba esta con un tiempo de vida de cero, o si el tiempo de vida de sus *bindings* existentes expira. Este enfoque provee un mecanismo de retraso en caso de una falla de uno de los *routers* de *MAP*, esto reducirá el tiempo que toma al *MN* informar a su nodo correspondiente (*Correspondent Node, CN*) y agente local (*Home Agent, HA*) acerca su nueva *Care-of Address (CoA)*. [1]

### 2.5.1 Selección del MAP en un ambiente de MAP distribuido

El *MN* necesita considerar varios factores para seleccionar óptimamente uno o más *MAPs*, donde varios *MAPs* están disponibles en el mismo dominio.

No hay beneficios previstos en seleccionar mas de un *MAP*, ni en forzar paquetes para ser enviados desde el más alto *MAP* bajando a través de una jerarquía de *MAPs*. Este enfoque puede adicionar retardos y eliminar la robustez del enrutamiento *IP* entre el más alto *MAP* y el *MN*, por lo tanto no se apropiado su implementación. Permitir más de un *MAP* dentro de una red no debería implicar que el *MN* tenga que forzar los paquetes a ser enrutados bajo la jerarquía de *MAPs*. Sin embargo, colocar más de un *MAP* arriba del *AR* sería redundante y podría ser usado como una optimización para los diversos escenarios de movilidad experimentados por *MNs*. Los *MAPs* son usados independientemente uno del otro, por el *MN* (cada *MAP* es usado para comunicar a un cierto grupo de *CNs*). En términos de la selección basada en la distancia en una red con varios *MAPs*, un *MN* puede escoger registrarse con el *MAP* más lejano para evitar re-registros frecuentes. Esto es particularmente importante para los *MNs* que desarrollaran *handovers* frecuentes. En este escenario, la elección de un *MAP* más distante reduce la probabilidad de tener que cambiar de un *MAP* a otro e informar a todos los *CN* y los *HA*.

En un escenario donde varios *MAPs* son descubiertos por el *MN* en un dominio, el *MN* necesita sofisticados algoritmos para poder seleccionar el apropiado *MAP*. Estos algoritmos tendrían la velocidad del *MN* como una entrada (para la selección basada en distancia) combinada con el campo de preferencia en la opción del



*MAP*. Sin embargo, se pone a consideración que el *MN* use el siguiente algoritmo por defecto, donde otros algoritmos optimizados no están disponibles. El siguiente algoritmo es simplemente basado sobre la selección del *MAP* que es más distante, teniendo en cuenta que su valor de preferencia no alcanza un valor de cero. La operación del *MN* es mostrada a continuación:

1. Recibir y analizar todas las opciones de *MAP*.
2. Disponer de los *MAPs* en un orden descendiente, comenzando con el *MAP* más lejano. (el *MAP* que tiene el campo de distancia más largo).
3. Seleccionar el primer *MAP* en la lista.
4. Si los campos de valor de preferencia o el tiempo de vida valido equivale a cero, seleccionar el siguiente *MAP* en la lista.
5. Repetir el paso (4) hasta que se encuentre un *MAP* con un valor de preferencia diferente de cero y un tiempo de vida diferente de cero.

Implementando estos pasos se obtiene como resultado que el *MN* selecciona por defecto el *MAP* más distante o más lejano disponible. Esto continuará hasta que el valor de preferencia o el tiempo de vida valido se reduzcan a cero. Seguido de esto, el *MN* comenzara seleccionando otro *MAP*. [1]

### **2.5.2 Selección del MAP en una arquitectura de gestión de movilidad plana**

Los operadores de redes pueden escoger una arquitectura plana en algunos casos donde un *handover* de *Mobile IPv6* [2] puede ser considerado un evento raro. En estos escenarios, los operadores pueden escoger incluir la función *MAP* en *ARs* solamente. La inclusión de una función *MAP* en los *ARs* puede aún ser útil para reducir el tiempo requerido para actualizar todos los *CNs* y el *HA*. En este escenario, un *MN* puede escoger un *MAP* como un punto de conexión cuando desarrolla un *handover*. Esta clase de jerarquía dinámica es solamente recomendada para casos donde los movimientos no son frecuentes entre *ARs*.

## **2.6 FAST HANDOVER PARA HMIPv6**

La combinación de *FMIPv6* (*Fast Handover for Mobile IPv6*) [8] y *HMIPv6* [1] proporciona un beneficio muy importante, facilitando la anticipación del *handover*, tal que el tráfico de datos puede ser eficientemente redireccionado a la nueva ubicación del *MN* antes de que este se mueva. Sin embargo, como no es fácil

determinar el tiempo correcto para empezar a enviar tráfico del MAP a la nueva ubicación del MN, no es posible precisar que tan suave podría ser el *handover*.

### 2.6.1 Mensajes

- ✓ **Router Solicitation for Proxy Advertisement (RtSolPr):** Este tipo de mensajes es enviado por el nodo móvil (*Node Mobile, MN*) solicitando un *Proxy Router Advertisement (PrRtAdv)* cuando inicia el *handover*.
- ✓ **Proxy Router Advertisement (PrRtAdv):** Mensaje enviado por el MAP por medio de los *routers* de acceso (*Access Routers, ARs*) al MN como respuesta al *Router Solicitation for Proxy Advertisement (RtSolPr)* con el fin de proporcionar la dirección del enlace, la dirección IP y el prefijo de subred de los *routers* cercanos.
- ✓ **Handover Initiate (HI):** Es un mensaje enviado por MAP al *router* de acceso próximo para iniciar el proceso de *fast handover* de los MNs. Se usa un código con valor de cero cuando se recibe un *Fast Binding Update (FBU)* con la dirección del *router* de acceso previo como dirección fuente, así mismo se usa un código con valor uno si la dirección no corresponde a la dirección del *router* de acceso previo.
- ✓ **Handover Acknowledge (HACK):** El *Handover Acknowledge* es un mensaje que debe ser enviado como respuesta al mensaje de *Handover Initiate (HI)*, este mensaje es enviado por el *router* de acceso próximo. El *router* de acceso próximo puede rechazar el *handover*, en este caso este debe indicar la razón, mediante alguno de los códigos disponibles.
- ✓ **Fast Binding Update (FBU):** Es un mensaje el cual es similar al mensaje de *Binding Update (BU)* descrito en el protocolo *Mobile IPv6* [2] sin embargo el proceso es un poco diferente.

El nodo móvil (*Mobile Node, MN*) envía un *FBU* después de recibir el mensaje *Proxy Router Advertisement (PrRtAdv)*. Si el MN se mueve antes de recibir el *PrRtAdv* este de igual manera debe enviar un *FBU* al MAP después de configurar la dirección del *router* de acceso próximo, de acuerdo con el proceso de descubrimiento de vecino (*Neighbor Discovery*).

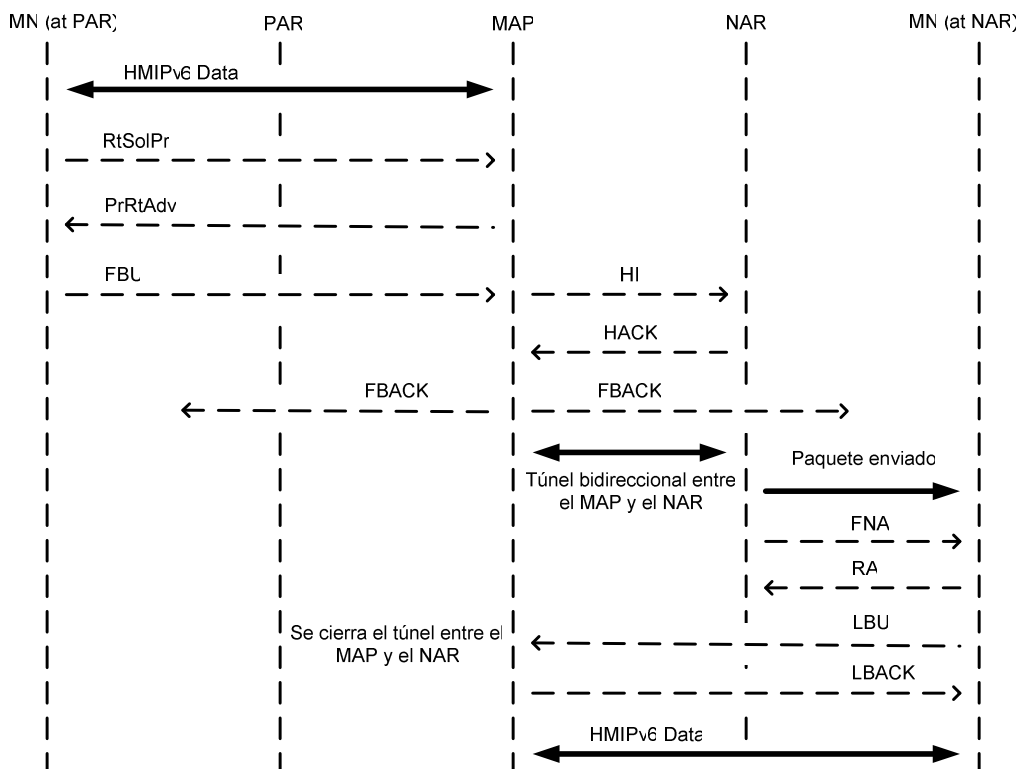
- ✓ **Fast Binding Acknowledgment (FBACK):** El *Fast Binding Acknowledgment* es un mensaje que es enviado por el MAP con el objetivo de confirmar la recepción del mensaje *Fast Binding Update (FBU)*. Este mensaje no debe ser enviado al nodo móvil (*Mobile Node, MN*) antes que el MAP reciba el mensaje *Handover Acknowledge (HACK)*, proveniente del *router* de acceso próximo.

- ✓ **Fast Neighbor Advertisement (FNA):** Es un mensaje que es enviado por el nodo móvil (*Mobile Node, MN*) al *router* de acceso próximo cuando ingresa a su dominio y recupera la conexión en el nuevo enlace. [8]

### 2.6.2 Operación de F-HMIPv6

Considerando que el *MAP* ya tiene la información necesaria para el soporte de *handover* acerca de la localización de los *ARs* en el dominio de *HMIPv6* [1], ya sea mediante configuración manual realizada por los operadores o con la ayuda de una operación de señalización entre los *ARs* y el *MAP*, teniendo en cuenta que esta información debe ser incluida en el enlace de la capa de direcciones y el prefijo de red de cada *AR*. En la figura 11 se muestra el flujo de mensajes que se presentan en la operación de *F-HMIPv6* [9] y se hace una breve descripción del diálogo de traspaso con *Fast handover* [8].

Figura 14: Diálogo de traspaso con Fast Handover.



Fuente: H. Jung, H. Soliman, S. Koh, N. Takamiya. *Fast Handover for Hierarchical MIPv6 (F-HMIPv6)*. draft-jung-mipshop-fhmipv6-00. Octubre 2005.

- ✓ El *MN* envía un mensaje *RtSolPr* (*Router Solicitation for Proxy*) al *MAP*. Este incluye información acerca del enlace de la capa de direcciones o el identificador del correspondiente *NAR*.
- ✓ En respuesta al mensaje *RtSolPr*, el *MAP* envía el mensaje *PrRtAdv* (*Proxy Router Advertisement*) al *MN*, el cual debe contener información acerca de *NLCoA* (*Next On-link Care-of Address*) o dirección del *NAR*.
- ✓ El *MN* envía un mensaje *FBU* (*Fast Binding Update*) al *MAP*. Este mensaje contiene la *PLCoA* (*Previous On-link Care-of Address*) o dirección del *PAR* y la dirección *IP* del *NAR*.
- ✓ Después de recibir el mensaje *FBU* del *MN*, el *MAP* enviara un mensaje *HI* (*Handover Initiate*) hacia el *NAR*, con el fin de establecer un túnel bidireccional entre el *MAP* y el *NAR*.

En respuesta al mensaje *HI*, el *NAR* fijara una ruta de entrada para la *PLCoA* del *MN* y envía un mensaje *HACK* (*Handover Acknowledge*). Como resultado se establece un túnel bidireccional entre el *MAP* y el *NAR*.

- ✓ El siguiente paso consiste en un mensaje *Fast Binding ACK* (*FBACK*) que es enviado por el *MAP* a las direcciones *PLCoA* del *PAR* y *NLCoA* del *NAR* con destino del *MN*. Después el *MAP* empezara a enviar los paquetes destinados al *MN* por medio del *NAR* usando el túnel establecido.
- ✓ El *MN* envía un mensaje *FNA* (*Fast Neighbor Advertisement*) al *NAR* una vez este detecta que se encuentra en el limite entre la región del *PAR* y el *NAR*, de igual manera recibe un mensaje de *RA* (*Router Advertisement*) enviado desde el *NAR*.
- ✓ Una vez realizados estos pasos el *MN* continuara su operación normal tal como se describe en *HMIPv6* [1], enviando un *LBU* (*Local Binding Update*) al *MAP*.

Cuando el *MAP* recibe el nuevo *LBU* con la *NLCoA* del el *MN*, este detiene el envío de paquetes mediante el *NAR* y cierra el túnel bidireccional establecido para el *Fast hadover* [9].

- ✓ El *MAP* envía un *LBACK* (*Local Binding ACK*) al *MN* como respuesta al *LBU*, por consiguiente los procedimientos posteriores se harán tal como se realizan en *HMIPv6* [1]. [9]

### 3. CONCEPTOS BÁSICOS DEL LENGUAJE DE PROGRAMACIÓN C++

#### 3.1 MATRICES

Una matriz es una estructura homogénea, compuesta por varios elementos, todos del mismo tipo y almacenados consecutivamente en memoria. Cada elemento puede ser accedido directamente por el nombre de la variable matriz seguido de uno o más subíndices encerrados entre corchetes.

En general, la representación de las matrices se hace mediante variables suscritas o de subíndices y pueden tener una o varias dimensiones (subíndices). A las matrices de una dimensión se les llama también listas y a las de dos dimensiones, tablas.

Desde el punto de vista matemático, es necesario utilizar variables subindicadas tales como:

$$v = [a_0, a_1, a_2, \dots, a_i, \dots, a_n]$$

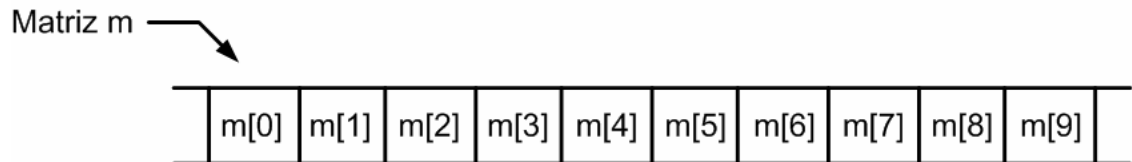
en el caso de un subíndice, o bien

$$m = \begin{bmatrix} a_{00} & a_{01} & a_{02} & \dots & a_{0j} & \dots & a_{0n} \\ a_{10} & a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i0} & a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \end{bmatrix}$$

si se utilizan dos subíndices. Esta misma representación se puede utilizar desde un lenguaje de programación recurriendo a las matrices antes mencionadas. [5]

Por ejemplo, suponiendo que se tiene una matriz unidimensional de enteros llamada  $m$ , la cual contiene 10 elementos. Estos elementos se identifican como se muestra en la figura 15:

Figura 15: Matriz unidimensional de 10 elementos.



Fuente: Ceballos, Javier. *Enciclopedia del lenguaje C++, Cuarta Edición*, Editorial Alfaomega, Mexico, Febrero 2004.

Los subíndices son enteros consecutivos, y en este caso el primer subíndice vale 0. Un subíndice puede ser cualquier expresión entera positiva.

Asimismo, una matriz de dos dimensiones se representa mediante una variable con dos subíndices (filas, columnas); una matriz de tres dimensiones se representa mediante una variable con tres subíndices etc. El número máximo de dimensiones o el número máximo de elementos para una matriz, dentro de los límites establecidos por el compilador, depende de la memoria disponible. [5]

### 3.1.1 Matrices Multidimensionales y de referencias.

La definición de una matriz numérica de varias dimensiones se hace de la siguiente forma:

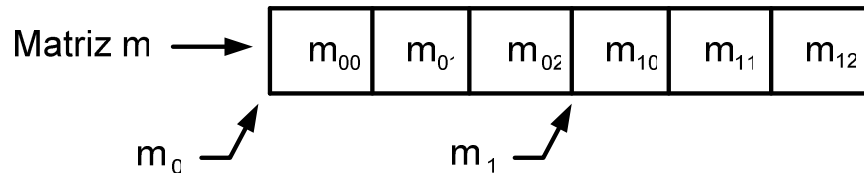
```
tipo nombre_matriz[expr-1][expr-2]
```

donde *tipo* es un tipo primitivo entero o real. El número de elementos de una matriz multidimensional es el producto de las dimensiones indicadas por *expr-1*, *expr-2*. Por ejemplo, se desea crear una matriz de dos dimensiones 2 x 3, la cual contiene 6 elementos tipo **int**. La línea de código sería la siguiente:

```
int m[2][3];
```

A partir de la línea de código anterior, C++ crea una matriz bidimensional *m* con dos filas *m[0]* y *m[1]* que hacen referencia a otras dos matrices unidimensionales de 3 elementos cada una, como se muestra en la figura 16:

Figura 16: Matriz bidimensional vista como dos matrices unidimensionales.



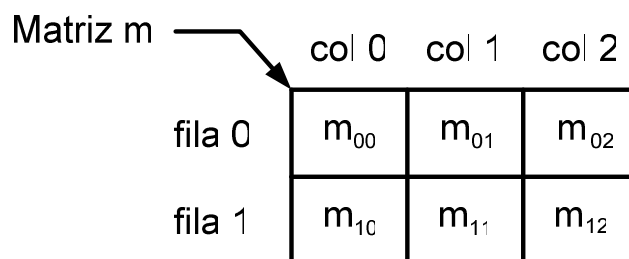
Fuente: Ceballos, Javier. *Enciclopedia del lenguaje C++*, Cuarta Edición, Editorial Alfaomega, Mexico, Febrero 2004.

De la figura anterior se deduce que los elementos de una matriz bidimensional son colocados por filas consecutivas en memoria. El nombre de la matriz representa la dirección donde se localiza la primera fila de la matriz, o el primer elemento de la matriz; esto es,  $m$ ,  $m[0]$  y  $\&m[0][0]$  son la misma dirección.

Evidentemente, el tipo de  $m[0]$  y  $m[1]$  es `int[ ]` y el tipo de los elementos de las matrices referenciadas por  $m[0]$  y  $m[1]$  es `int`.

Teniendo en cuenta que una matriz bidimensional es en realidad una matriz unidimensional cuyos elementos son a su vez matrices unidimensionales. Es recomendable que cuando se trata de matrices de dos dimensiones, es más fácil pensar en ellas como si se tratara de una tabla de  $f$  filas por  $c$  columnas. Tal como se representa en la figura 17. [5]

Figura 17: Matriz bidimensional vista como una tabla.



Fuente: Ceballos, Javier. *Enciclopedia del lenguaje C++*, Cuarta Edición, Editorial Alfaomega, Mexico, Febrero 2004.

Cuando se inicia una matriz multidimensional en el instante de su definición, el tamaño de su primera dimensión puede omitirse, ya que puede calcularse a partir del resto de las dimensiones y del número total de elementos. También puede

omitirse en los siguientes casos: cuando se declare como un parámetro formal en una función y cuando se haga una referencia a una matriz declarada en otra parte del programa.

```
int m[] [3] = {1, 2, 3, 4, 5, 6};
```

Para acceder a los elementos de la matriz  $m$ , puesto que se trata de una matriz de dos dimensiones, se utiliza dos subíndices, el primero indicará la fila y el segundo la columna donde se localiza el elemento, según se puede observar en la figura 17. Por ejemplo, la primera sentencia del ejemplo siguiente asigna el valor  $x$  al elemento que está en la fila 1, columna 2; y la segunda, asigna el valor de este elemento al elemento  $m[0][1]$ .

```
m[1][2] = x;  
m[0][1] = m[1][2];
```

El cálculo que hace el compilador para saber cuántos elementos tiene que avanzar desde  $m$  para acceder a un elemento cualquiera  $m[filas][cols]$  en la matriz anterior es: *fila x elementos por fila + col.*

Una matriz numérica de varias dimensiones puede también implementarse utilizando matrices de matrices. Desde esta perspectiva, la definición de una matriz numérica de dos dimensiones puede hacerse utilizando la clase **vector** (fichero de cabecera *vector*) de la siguiente forma:

```
vector< vector<tipo> > nombre(d1, vector <tipo>(d2));
```

donde *tipo*, en el caso de matrices numéricas, es un tipo primitivo entero o real, pero en general puede ser cualquier tipo predefinido o definido por el usuario.

El número de elementos de una matriz multidimensional en el caso de que sea rectangular es el producto de las dimensiones indicadas por  $d1$  y  $d2$ . Por ejemplo, la línea de código siguiente crea una matriz de dos dimensiones con  $2 \times 3 = 6$  elementos de tipo **int**:

```
// Matriz de 2 filas por 3 columnas representada por una  
// matriz de matrices. Cada elemento de m se inicia con  
// una matriz de 3 elementos de tipo int  
vector< vector<int> > m(2, vector<int>(3));  
// ...  
m[1][2] = x;  
m[0][1] = m[1][2];
```

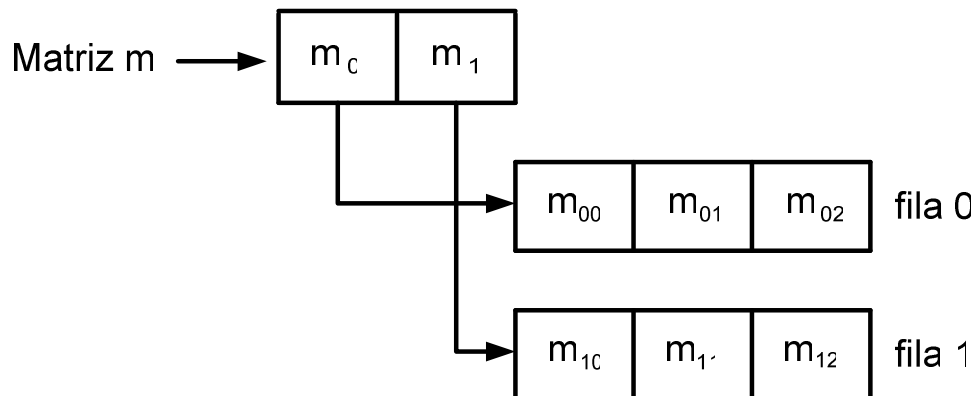


La definición anterior puede realizarse también así:

```
// Matriz de 2 filas por 3 columnas:  
// declarar una matriz de matrices de 2 elementos  
vector< vector<int> > m(2);  
// Inicia cada elemento con una matriz de 3 elementos  
m[0] = vector<int>(3);  
m[1] = vector<int>(3);
```

A partir del código anterior, C++ crea una matriz unidimensional  $m$  con 2 elementos  $m[0]$  y  $m[1]$  que son referencias a otras dos matrices unidimensionales de 3 elementos, tal como se observa en la figura 18.

Figura 18: Matriz unidimensional con 2 elementos.



Fuente: Ceballos, Javier. Enciclopedia del lenguaje C++, Cuarta Edición, Editorial Alfaomega, Mexico, Febrero 2004.

Asimismo, desde esta perspectiva, declarar una matriz de  $n$  dimensiones resulta igual de fácil (basta con seguir anidando el tipo `vector<tipo>`). [5]

### 3.2 FUNCIONES AMIGAS DE UNA CLASE

El hecho de que un método ordinario sea miembro de una clase implica tres cosas:

1. Tiene acceso al resto de los miembros de la clase, incluso a los miembros privados, lógicamente para manipular el estado de un objeto.
2. Pertenece al ámbito definido por la clase.

3. Debe invocarse para un objeto de su misma clase, al que se refiere por medio del puntero implícito **this**.

En cambio, cuando un método de una clase se declara **static**, sólo intervine en los ítems 1 y 2, con la excepción de que sólo puede acceder a los miembros **static**.

Por otra parte si se trata de una función externa esta no participa de ninguno de los puntos. No obstante, existen casos, en los que se hace necesario que una función externa participe del punto 1. Esto lo permite C++ declarando la función amiga de la clase.

Para declarar una función externa amiga de una clase, hay que incluir su declaración en el cuerpo de la clase, precedida por la palabra reservada **friend**.

Por ejemplo, se tiene una función *fnVisualizar* la cual no puede acceder a los miembros privados de la clase *CVector*. Según lo expuesto, este problema podría resolverse declarando la función *fnVisualizar* amiga (**friend**) de la clase *CVector*. Como se describe en las siguientes líneas de código:

```
class CVector
{
    friend void fnVisualizar(CVector& vector);
private:
    double *vector; // puntero al primer elemento de la matriz
    int nElementos; // número de elementos de la MATRIZ
protected:
    double *asignarMem(int);
public:
    CVector(); // crea un CVector con N elementos por omisión
    CVector(int ne); // crea un CVector con ne elementos
    CVector(double *, int); // crea un CVector desde una matriz
    CVector(const CVector&); // crea un CVector desde otro
    ~CVector(); // destructor
    double& elemento(int i);
    int longitud() const;
};
```

Ahora la definición de la función *fnVisualizar* podría escribirse como se indica a continuación:

```
void fnVisualizar (CVector& v)
{
    int ne = v.longitud();
    for (int i = 0; i < ne; i++)
```

```

        cout << setw(7) << v.vector[i];
    cout << "\n\n";
}

```

Ahora que la función *fnVisualizar*, en lugar de utilizar el método *elemento* para acceder a los datos de la matriz, puede acceder directamente al atributo *vector* del objeto *v* por ser amiga de *CVector*.

Una función **friend** puede declararse en cualquier parte de la clase, aunque generalmente se declaran al principio de la definición de la clase. Teniendo en cuenta que las palabras clave **public**, **protected** y **private** definen el nivel de protección de los miembros de la clase, y la función **friend** no es un miembro de la clase, razón por la cual no se ve afectada por estas palabras clave.

Una función amiga, puesto que tiene que aparecer en la declaración de la clase, es parte de la interfaz de la misma, tanto como lo es cualquier otro miembro. Por lo tanto, no se transgreden los mecanismos de protección. Es la clase quien concede la amistad, al igual que los demás accesos.

El mecanismo de la amistad es importante por dos razones:

- ✓ Una función puede ser amiga de más de una clase, lo que puede conducir a interfaces más claras.
- ✓ Una función amiga admite que se apliquen a su primer argumento conversiones implícitas o definidas por el usuario, donde los métodos no lo admiten.

También es posible declarar un método de una clase *C2* amigo de otra clase *C1*. En este caso, la definición de la clase *C2*, clase que aporta el método que va a ser amigo de la clase *C1*, debe preceder a la definición de la clase *C1*. Por ejemplo:

```

////////////////////////////////////
class C1; // declaración adelantada de C1

class C2
{
    private:
        int nc2;
    public:
        void AsignarDato(int n) { nc2 = n; }
        int ObtenerDato(const C1&);
};

```

```

class C1
{
    friend int C2::ObtenerDato(const C1&);
private:
    int nc1;
public:
    void AsignarDato(int n) { nc1 = n; }
};

int C2::ObtenerDato(const C1& obj)
{
    return obj.nc1 + nc2;
}
/////////////////////////////////////////////////////////////////

int main()
{
    C1 objeto1; // objeto de la clase C1
    C2 objeto2; // objeto de la clase C2
    int dato;

    cout << "N° entero: "; cin >> dato;
    objeto1.AsignarDato(dato);
    cout << "N° entero: "; cin >> dato;
    objeto2.AsignarDato(dato);
    cout << "\nResultado: ";
    cout << objeto2.ObtenerDato(objeto1) << endl;
}

```

De acuerdo al código se observa la declaración adelantada de *C1* antes de la definición de la clase *C2*; indica simplemente que la clase *C1* se define más adelante. Si no se realiza esta declaración anticipada, la declaración *int ObtenerDato(C1&)* daría lugar a un error por hacer referencia a una clase, *C1*, aún no declarada.

Por otra parte, la declaración del método *ObtenerDato* como amigo de *C1* se ha hecho especificando la clase a la que pertenece, ya que de omitir *C2::*, se analizaría como una función externa:

```
friend int C2::ObtenerDato(C1&);
```

Además, si *ObtenerDato* no fuera un método amigo de *C1* no podría acceder al atributo privado *nc1*; tendría que hacerlo a través de un método de la clase. [5]

Algunas veces puede ser también necesario que todos los métodos de una clase *C1* sean amigos de otra clase *C2*. Para permitir esto, hay que declarar a la clase *C1* amiga de la clase *C2* así:

```
class C1
{
    // ...
};

class C2
{
    friend class C1;
    // ...
};
```

## 4. METODOLOGÍA

El proyecto consiste en crear un modelo de simulación para el protocolo *Hierarchical Mobile IPv6 (HMIPv6)* [1] estipulado en la *RFC 4140*.

La metodología utilizada para cumplir los objetivos propuestos en este proyecto fue investigativa y experimental. Una primera etapa fue buscar e ilustrarse de los temas relacionados con el protocolo a implementar y documentación de *NS* [3] a través de libros, artículos, especificaciones de la *IETF (Internet Engineering Task Force)*, etc.

Se consultó información y simulaciones relacionadas sobre el tema y como resultado de esta búsqueda se tiene el trabajo desarrollado por *T. Kwon* y *M. Gerla* [10], los cuales presentan análisis de resultados para la latencia del handover bajo el uso del protocolo *Mobile IP* básico con la extensión *Smooth Handover*, el cual es básicamente una función de tunelización entre el *oFA* (Agente Foráneo Antiguo) y *nFA* (Agente Foráneo Nuevo); el estudio de *Y. Gwon* [11] que presenta el análisis de rendimiento de *MIP*, *HMIP*, *FMIP* y *FHMIP*; también el estudio de *I. Vivaldi* [12] que propone un esquema de *Handover* para macromovilidad para *HMIP* que esta basado en bi-casting. De igual manera se encontró la extensión creada por *Robert Hsieh* [4] realizada en el año 2004, tiempo en que el documento que define el protocolo experimental de *Hierarchical Mobile IPv6 Mobility Management* [1] (*RFC 4140*) no se había creado.

La extensión creada por *Robert Hsieh* [4] implementa el protocolo *HMIPv6* [1] con *Fast Handover* [8] para una topología específica. Ya que la extensión de *Robert Hsieh* [4] contenía algunas características del protocolo *HMIPv6* [1], se decide tomar esta como base para nuestro proyecto.

El paso a seguir fue instalar el *Network Simulator* [3] y familiarizarse con el entorno y lenguaje del simulador.

Luego de adaptarnos al lenguaje de programación utilizado en los script de *NS* [3]; se adapta y se instala la extensión de *Robert Hsieh* [4] a una versión de *NS* [3] más actual (v2.30), ya que la versión para la que fue diseñada dicha extensión ya ha quedado desactualizada.

Con el *NS* [3] y la extensión de *Robert* [4] funcionando, se evalúa y se experimenta lo aportado por *Robert* [4] para determinar que se necesita modificar y adicionar para que cumpla con las características generales del protocolo *HMIPv6* [1] descrito en la *RFC 4140*.

Teniendo claro las modificaciones que se necesitaban realizar y haciendo uso de los conocimientos del lenguaje de programación *C++* [5] y *TCL*, se procede a la implementación de dichas modificaciones en el programa, con el fin de obtener una extensión actualizada.

Luego se hacen pruebas con diferentes simulaciones para comprobar el funcionamiento de la herramienta *HMIPv6\_SM* y finalmente se procede a la elaboración del documento final.

## 5. DESARROLLO DEL PROYECTO

El protocolo *Hierarchical Mobile IPv6 (HMIPv6)* [1] es una extensión del protocolo *Mobile IPv6* [2] con el propósito de manejar la micro movilidad y reducir la cantidad de mensajes de señalización entre los *MNs (Mobile Nodes)* y los *HAs (Home Agents)*. En *Mobile IPv6* [2], cuando los *MNs* cambian su punto de acceso, debe actualizar a sus *HA* y *CN (Correspondent Node)* de su ubicación actual; de esta manera, el *HA* siempre conocen la posición del *MN*, pero la sobrecarga de señalización aumentara con el incremento de los usuarios móviles en Internet.

Para reducir la carga de señalización hacia el *CN* y el *HA*, *HMIPv6* [1] implementa el *MAP (Mobile Anchor Point)*, el cual maneja la micro movilidad del *MN*, esto hace que la movilidad dentro del dominio *MAP*, sea invisible para el *CN* y el *HA*.

Actualmente, hay pocas herramientas que den la opción de simular topologías pensadas para trabajar bajo el protocolo *Hierarchical Mobile IPv6* [1], de ahí la importancia de realizar una extensión para el *Network Simulator* [3] que cumpla las especificaciones mínimas de la *IETF (Internet Engineering Task Force)* descritas en *RFC 4140*.

Uno de los estudios mas cercanos al comportamiento del protocolo fue el realizado por *Robert Hsieh* [4], el cual es tomado como referencia para el desarrollo de este proyecto.

A partir del trabajo de *Robert Hsieh* [4] y haciendo un estudio comparativo entre la *RFC 4140* y este, se logra establecer las diferencias importantes para que la herramienta cumpla con las principales características del funcionamiento del protocolo *HMIPv6* [1]. Realizando las modificaciones pertinentes se obtiene una nueva herramienta la cual se ah denominado *HMIPv6\_SM*.

En la siguiente sección se explicara el procedimiento llevado a cabo para la obtención de *HMIPv6\_SM*.

### 5.1 HERRAMIENTA DE SIMULACIÓN HMIPv6\_SM

En la herramienta *HMIPv6\_Simulation Model (HMIPv6\_SM)* tomamos como base el estudio realizado por *Robert Hsieh* [4], que implementa el *MAP* con algunos de los mensajes de registro. Se realizaron cambios a la herramienta de *Hsieh* [4] tomando en cuenta las falencias que esta tiene, y se adiciona las características

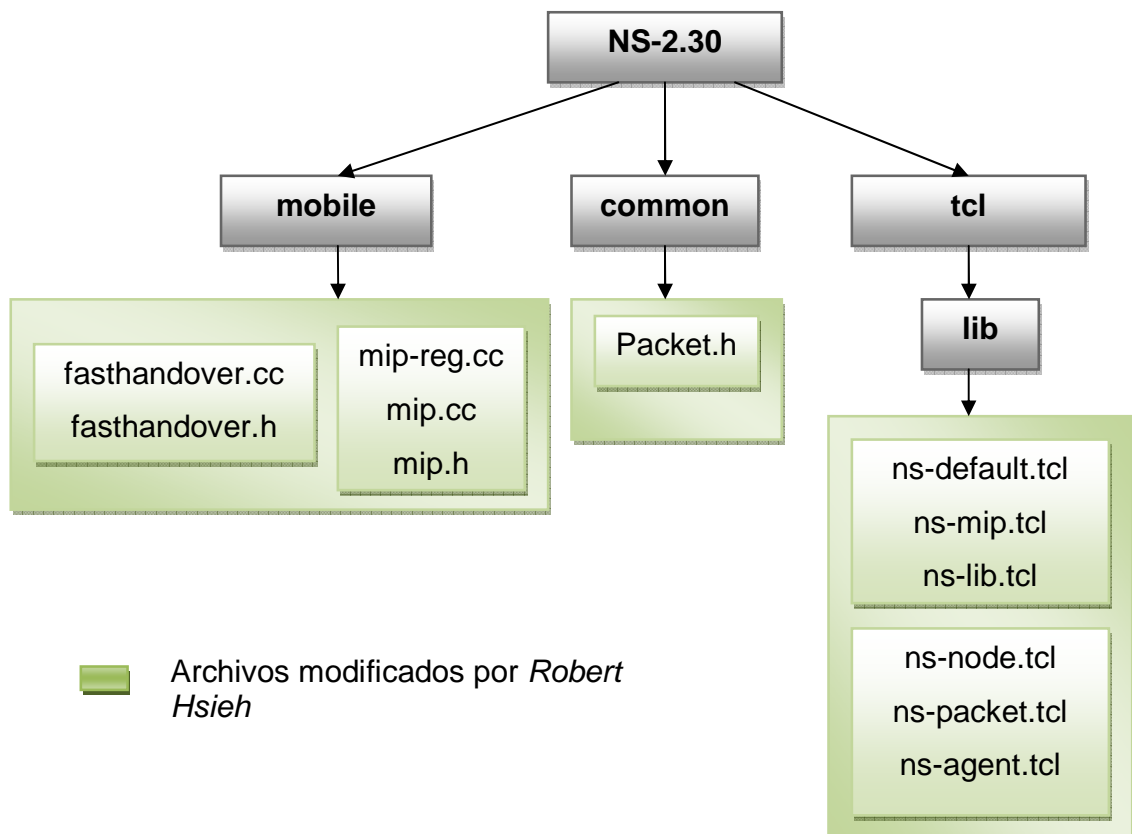


generales del protocolo *HMIPv6* [1]. Por este motivo se hace necesario mencionar los cambios que efectúa la herramienta de *Robert* [4] al código fuente de *NS* [3].

### 5.1.1 Estructura de la Herramienta FHMIPv6-ext de Robert Hsieh

La estructura de archivos que modifica la herramienta de *Robert Hsieh* [4] al *Network Simulator* [3] se muestra en la figura 19. A continuación se procede a explicar la función general de cada archivo y las modificaciones hechas por *Robert*.

Figura 19: Estructura de Archivos Modificados por Robert Hsieh

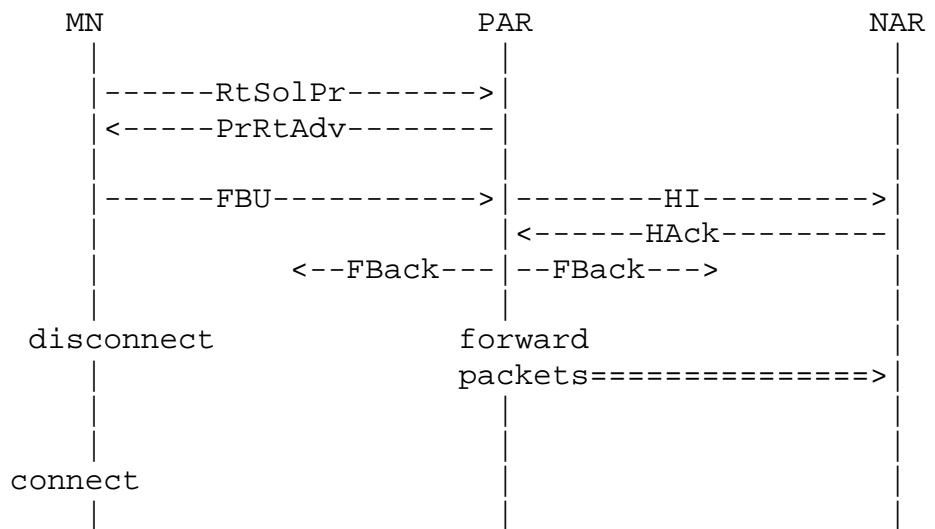


Fuente: Autores

**Carpeta *mobile*:** En la carpeta *mobile* se encuentran los principales archivos referentes a *Mobile IP*. Los archivos más significativos son (mip-reg.cc, fasthandover.cc, fasthandover.h, mip.cc y mip.h).

- ✓ El archivo (mip-reg.cc) se encarga de llevar a cabo los registros entre el nodo móvil y las estaciones base. Se definen principalmente 2 clases: la clase MIPBSAgent, que hace referencia al manejo de los registros en las Estaciones Base; y MIPMHAgent, clase que abarca el código para el manejo de los registros en el MN (Mobile Node). Robert Hsieh [4] crea una tercera clase llamada MAPAgent, la cual implementa el uso del MAP en las simulaciones.
- ✓ Los archivos (fasthandover.cc y fasthandover.h) fueron creados y adicionados por Robert [4], los cuales están encargados del mecanismo de Fast Handover [8]. El mecanismo de Fast Handover utilizado por Robert [4] es un mecanismo en el cual no interviene el MAP, ya que se establece un túnel directo entre el PAR y el NAR. Terminado el registro con el NAR, el CN se desconecta del PAR e inmediatamente los paquetes son enviados al NAR, lo cual es una característica propia del FMIPv6 [8] predictivo (ver figura 20).

Figura 20: Fast Handover Predictivo.



Fuente: R. Koodli. Fast Handovers for Mobile IPv6. RFC 4068. Julio 2005.

En el caso de Fast Handover for HMIPv6 [9], MAP tiene una participación más activa en el mecanismo de fast handover, ya que el túnel se realiza entre el MAP y NAR; y no entre PAR y NAR. Además en F-HMIPv6 [9] el MN realiza la petición de activación del mecanismo de fast handover directamente con el MAP ya que este posee la información necesaria para el soporte de handover acerca de la localización de los ARs en el dominio de HMIPv6 [1].

En la sección 2.6.2 se expone el mecanismo *F-HMIPv6* [9] de una manera mas clara.

- ✓ El archivo (`mip.cc`) se encarga de la parte de encapsulación y desencapsulación. En este se definen principalmente dos clases: la clase `MIPEncapsulator`, que se encarga de la encapsulación y la clase `MIPDesencapsulator`, la cual se encarga de la desencapsulación de los paquetes de datos. *Robert Hsieh* [4] incluye una nueva clase llamada `RecvVarifier` que se encarga de la comunicación entre los *AR* (*Access Router*) y el *MAP* cuando el *MN* (*Mobile Node*) realiza *Fast Handover* [8].
- ✓ En el archivo (`mip.h`) se define: la estructura `hdr_mip` que hace de cabecera *Mobile IP* y las clases que se encuentran en los archivos (`mip-reg.cc` y `mip.cc`).

**Carpeta *common*:** En la carpeta *common* se encuentran todos los archivos que almacenan las clases comunes para la simulación de redes inalámbricas y redes fijas. El archivo que se modifica es (`packet.h`)

- ✓ El archivo (`packet.h`) es un stack de cabeceras. *Robert* [4] adicio las cabeceras de los mensajes que intervienen en el mecanismo de *Fast Handover* [8].

**Carpeta *lib*:** En la carpeta *lib* se encuentran los archivos *tcl* que interviene en la simulación de *Mobile IP*. En esta carpeta podemos encontrar los archivos (`ns-default.tcl`, `ns-mip.tcl`, `ns-lib.tcl`, `ns-node.tcl`, `ns-packet.tcl` y `ns-agent.tcl`). A continuación se describirá brevemente la función y las modificaciones realizadas por *Robert* [4] para los archivos más significativos de la carpeta.

- ✓ El archivo (`ns-default.tcl`) se guarda los valores por defecto de las nuevas variables usadas en *NS* [3] por *Robert Hsieh* [4] como la variable `varify_` perteneciente a la clase `RecvVarifier`.
- ✓ El archivo (`ns-mip.tcl`) guarda los procesos *tcl* que intervienen en la simulación de *Mobile IP*, como los procesos de encapsulación y desencapsulación `encap-router` y `decap-router`, llamados en el archivo (`mip-reg.cc`). En (`ns-mip.tcl`), *Robert* [4] copia y modifica los procesos usados por `MIPBSAgent` y los asigna a la clase `MAPAgent`.
- ✓ En el archivo (`ns-node.tcl`) se encuentran los procesos *tcl* propios del nodo como los módulos de registros del nodo o los procesos para manipular los clasificadores de direcciones. En este archivo *Robert* [4] configura a todos los

nodos la posibilidad de encapsular y desencapsular, similar a lo que se ofrece en `Node/MIPBS`. [13]

### 5.1.2 Inconvenientes y Soluciones de la Herramienta F-HMIPv6-ext

Analizando la herramienta creada por *Robert Hsieh* [4] se encontró los siguientes inconvenientes y se les dio solución:

- ✓ La dificultad de realizar una topología diferente a la propuesta por él, ya que los cambios que se desean realizar en la topología se deben efectuar, en el archivo (`mip-reg.cc`) además que de los efectuados en el *script* (`simula.tcl`). Al tener que modificar el archivo `mip-reg.cc` cada vez que se quiera cambiar la topología, se hace necesario la compilación de *NS* [3].

Solución: Debido a que *Robert* [4] identifica directamente en el (`mip-reg.cc`) las direcciones de los *AR* (*Access Routers*) y el *MAP*, además de la asignación en el *script*, se decide crear una nueva clase, la cual obtendrá las direcciones asignadas desde el *script* a los *MAP*'s y los *AR*, para luego guardar esta información en una tabla, creada en la clase `MAPAgente`. Este método será llamado cada vez que se requiera saber las direcciones de cualquiera de los componentes del simulador (*AR*'s, *Nodos Móviles*, etc) dependiendo del *MAP* en donde se encuentren registrados.

- ✓ No tiene implementada la optimización de ruta, característica propia de *Mobile IPv6* [2].

Solución: Se adiciono la optimización de ruta mediante la creación de un nuevo proceso `tcl`, el cual hace uso de la nueva clase que permite el tratamiento de las cabeceras *IP* sin encapsular.

- ✓ Desencapsula los paquetes en los *ARs*, en los cuales el *MN* se encuentra conectado. Según las especificaciones de la *RFC 4140*, los paquetes van encapsulados del *MAP* hasta el *MN*.

Solución: El proceso de desencapsulación en cualquier nodo de la simulación se realiza con la misma clase en `C++`, `MIPDesencapsulator`. Se identifica la dirección del nodo que está llamando esta clase y dependiendo del nodo en que se encuentra (si es un *AR*, el *MAP* o un *Home Agent*) se realiza la desencapsulación o no. Así se podrá evitar la desencapsulación en los *ARs*.

### 5.1.3 Estructura de la Herramienta HMIPv6\_SM

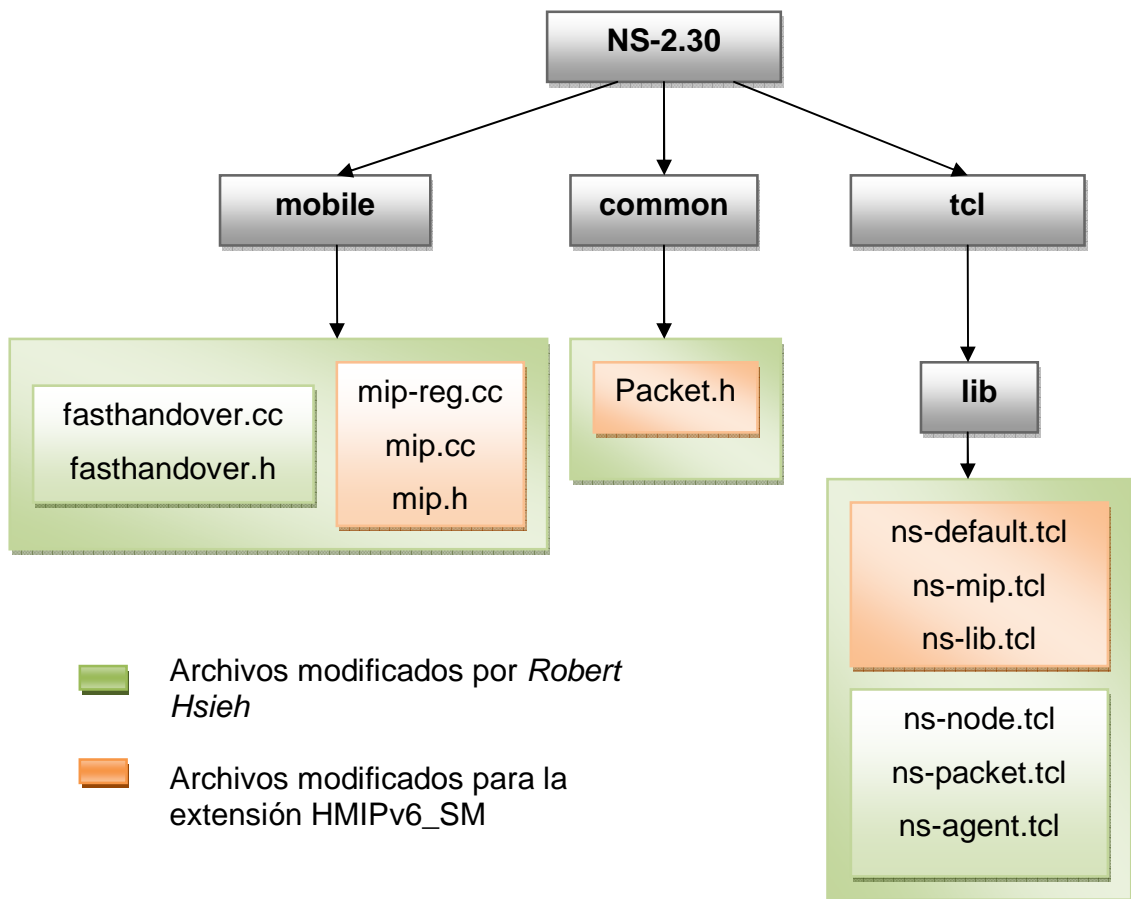
Teniendo en cuenta las deficiencias que presenta la herramienta de *Robert Hsieh* [4], se crea una herramienta propia que solucione las mencionadas falencias e implemente las principales características del protocolo *HMIPv6* [1] llamada *HMIPv6\_SM*.

La herramienta *HMIPv6\_SM* además de implementar la opción del *MAP*, permite la simulación de una gran variedad de topologías bajo el protocolo *HMIPv6* [1] desde el *script*, la tunelización entre el *MAP* y el *MN* y la optimización de ruta por defecto. La estructura de archivos modificados para la herramienta *HMIPv6\_SM* se muestra en la figura 21.

A continuación se menciona de una forma más general las modificaciones hechas para la creación de la herramienta *HMIPv6\_SM*; mas adelante se explicara los cambios y su funcionalidad de una forma mas detallada.

Los archivos modificados en el desarrollo de este proyecto, fueron principalmente los que involucran el protocolo *Mobile IP*, tanto los que están presentes en código C++ (*mip-reg.cc*, *mip.c* y *mip.h*) como en código *TCL* (*ns-mip.tcl*).

Figura 21: Estructura de archivos modificados para la herramienta HMIPv6\_SM y por Robert Hsieh.



Fuente: Autores.

**Carpeta *mobile*:** Los archivos de esta carpeta que fueron modificados en la creación de la herramienta HMIPv6\_SM fueron (mip-reg.cc, mip.cc y mip.h).

- ✓ Archivo (mip-reg.cc): Se adiciona una nueva clase llamada MAPA, la cual es la encargada de obtener las direcciones de los MAPs y los ARs, para luego guardarlas en una tabla que se localiza en el método registrar creado en la clase MAPAgent. Este nuevo método es el encargado del manejo de la tabla (adición, localización, limpieza, ordenamiento y entrega de direcciones).

- ✓ Archivo (mip.cc): Se crea una nueva clase llamada `OPTEncapsulator`, encargada del direccionamiento de los paquetes del *CN (Correspondent Node)* al *MAP*, sin la encapsulación de estos. Además, se modifica la clase `MIPDesencapsulator` para controlar la instancia en que el paquete tiene que ser desencapsulado y cuando no.
- ✓ Archivo (mip.h): En este archivo se definen las nuevas clases creadas para los archivos (mip-reg.cc y mip.cc); además se crea una nueva estructura llamada `mapa_datos`, la cual facilita enormemente la transferencia de datos entre el método `registrar` con otros métodos, tanto de la misma clase como los pertenecientes a otras clases.

**Carpeta *common*:** El archivo modificado en esta carpeta es (`packet.h`), en el cual se adiciona la variable `optimy_` a la estructura `hdr_cmn`.

**Carpeta *lib*:** Los archivos modificados para la creación de la herramienta fueron (`ns-default.tcl` y `ns-mip.tcl`).

- ✓ Archivo (`ns-mip.tcl`): Se adiciona el proceso `OPTEncapsulator` a los procesos de la clase `tcl Node/MIPBS` y el nuevo proceso `mencap-router` a `Agent/MIPBS`. Este proceso es el encargado del proceso `tcl` de direccionamiento de paquetes sin encapsular.
- ✓ Archivo (`ns-default.tcl`): Se agrega los valores por defecto de las nuevas variables usadas en las clases creadas.

Es importante resaltar que el mecanismo de *Fast Handover MIPv6* [8] predictivo implementado en la herramienta de *Robert* [4] se adecuó para ser usado en la herramienta *HMIPv6\_SM*, ya que la implementación de *F-HMIPv6* [9] a la nueva herramienta no es uno de los objetivos a cumplir en este proyecto.

## 5.2 MODIFICACIONES REALIZADAS PARA LA CREACIÓN DE *HMIPv6\_SM*

En esta sección se procederá a una explicación más explícita de las modificaciones hechas para la creación de *HMIPv6\_SM* como la creación de la clase `MAPA` y el método `registrar`, las cuales permiten la simulación de diferentes topologías creadas desde el *script*, la implementación de optimización de ruta y el envío del paquete encapsulado directamente al *MN*.





```

        return(TCL_OK);
    }
}

```

Esta parte del método `common`, nos permite adquirir la dirección del *MAP* definido en el script y las direcciones de los *AR* que pertenecerá su dominio (en esta ocasión el *MAP* tendrá 2 *AR*).

La línea (1) permite ver la palabra que activara el método de asignación de direcciones desde el script (`asignar`). Luego en (2) se adquiere la dirección del *MAP*, de una variable `string` (2.0.0) a una variable `integer` (8388608). De igual forma sucede con (3) y (4) pero en esta ocasión las direcciones que se adquirirán son las de los *ARs*.

Ya adquiridas las direcciones del *MAP* y sus dos *ARs*, se procede a llamar el método `registrar` de la clase `MAPAgent` en donde se almacenara los datos adquiridos en la matriz `map` (5).

El formato para suministrar las direcciones de un *MAP* y los *ARs* que pertenecen a su dominio en el script se presentara a continuación

```

set mapa [new Agent/MAPA]                (6)
$mapa asignar "2.0.0" "2.1.0" "2.2.0"    (7)
$ns_ attach-mapagent $MAP1                (8)

```

Primero se crea el agente `MAPA` (6), luego se asigna las direcciones del *MAP* con sus respectivos *ARs* (7) y por último se enlaza el nombre del *MAP* asignado anteriormente a `MAPAgent` (8).

### 5.2.1.2 Matriz “map”

Para almacenar todas las direcciones que se adquirieron en el método `common` de la clase `MAPA`, se crea la matriz `map`.

La matriz `map` está compuesta por 7 filas y 16 columnas. El número de filas representa el número máximo de *MAPs* que puede manejar la topología que se desea simular; en esta ocasión, esta posibilitado para 7 *MAPs*. La definición de la matriz `map` se hace mediante el código:

```

static int map[7][16];

```

Las columnas están distribuidas de la siguiente forma.

*Tabla 1: Distribución de las columnas de la matriz “map”*

<b>MAP</b>	<b>Node</b>	<b>Home Agent</b>	<b>Access Router</b>
0	1 – 5	6 – 10	11 – 15

En la columna 0, es asignada para las direcciones de los *MAPs*.

Las columnas de 1 - 5 están reservadas para los *MNs* que se registran con el *MAP*. El número máximo de *MNs* registrados por *MAP* es 5.

Las columnas de 6 – 10 están dispuestas para guardar el *HA* del *MN* que se registra.

Las columnas de 11 – 15 están destinadas a guardar las direcciones de los *AR* que son asignados al *MAP* de la columna 0.

Como la cantidad de datos que se pasan de una clase a otra es grande, se hace uso de una estructura llamada `mapa_datos`.

Las variables que componen la estructura `mapa_datos` se presentan a continuación:

```
struct mapa_datos
{
    int map_;
    int ARouter1_;
    int ARouter2_;
    int ARouter3_;
    int ARouter4_;
    int ARouter5_;
    int node_;
    int ha_;
    int BSpacket_;
};
```

### 5.2.1.3 Procesos del Método “registrar”

El método `registrar` fue creado con el objeto de guardar y manejar el contenido de la matriz `map`. Al ser llamado el método, le es enviado dos datos con los cuales este trabaja: un puntero a memoria y un entero. La definición del método `registrar` se realiza como se muestra a continuación:

```
int MAPAgent::registrar(mapa_datos& bili, int ah)
```

El puntero a memoria (`mapa_datos& bili`) indica los datos almacenados en la estructura (`mapa_datos`), esta es la información que maneja el método.

El entero `ah` indicara que clase de proceso se realizara; si es de registro o de petición de datos. Dependiendo del valor de `ah`, el método realiza uno de los siguientes procesos.

- ✓ Valor de `ah = 0`: Registra los *ARs* pertenecientes a un *MAP*. Este proceso se realiza al iniciar la simulación, ya que los *ARs* nunca cambiaran de *MAP*.

Al entrar al proceso, se busca una fila libre de la matriz `map` (que no tenga ningún *MAP* registrado), y limpia todas las celdas pertenecientes a dicha fila. Luego procede a llegar las celdas de *AR* con las respectivas direcciones.

```
if (ah == 0){
    int k = 1;
    while (k < 7) {
        if (map[k][0] == 0) {
            int j;
            for (j = 0; j <= 15; j++){
                map[k][j] = 0;
            }
            map[k][0] = bili.map_;
            map[k][11] = bili.ARrouter1_;
            map[k][12] = bili.ARrouter2_;
            map[k][13] = bili.ARrouter3_;
            map[k][14] = bili.ARrouter4_;
            map[k][15] = bili.ARrouter5_;
            k = 10;
        }
        else if (map[k][0] == bili.map_) {
            k = 10;
        }
        else {
            k++;
        }
    }
}
```

```

    }
}

```

- ✓ Valor de  $ah = 1$ : Envía la lista de *ARs* pertenecientes a un determinado *MAP*. Toma el número de la fila del *MAP* deseado y devuelve las direcciones de los *ARs*, y la del *MAP* al que pertenecen.

```

else if (ah == 1)
{
    int j;
    j = bili.map_;

    bili.map_ = map[j][0];
    bili.ARrouter1_ = map[j][11];
    bili.ARrouter2_ = map[j][12];
    bili.ARrouter3_ = map[j][13];
    bili.ARrouter4_ = map[j][14];
    bili.ARrouter5_ = map[j][15];
}

```

- ✓ Valor de  $ah = 2$ : Registrar *MN* y *HA*. Se ubica en la fila del *MAP* de la matriz *map*, en donde se encuentra el *MN*, registra la dirección del *MN* en uno de los *slot* vacíos dispuestos para esto y luego procede al registro de su correspondiente *HA*. Sabiendo que él *HA* siempre tendrá el mismo dominio y *clúster* del *MN*; y número de nodo será 0 (esto se puede observar en la tabla 2)

Tabla 2: Jerarquía de direcciones del *MN* y *HA*

	Domino	Clúster	Nº Nodo
Nodo Móvil	1	0	1
Home Agent	1	0	0

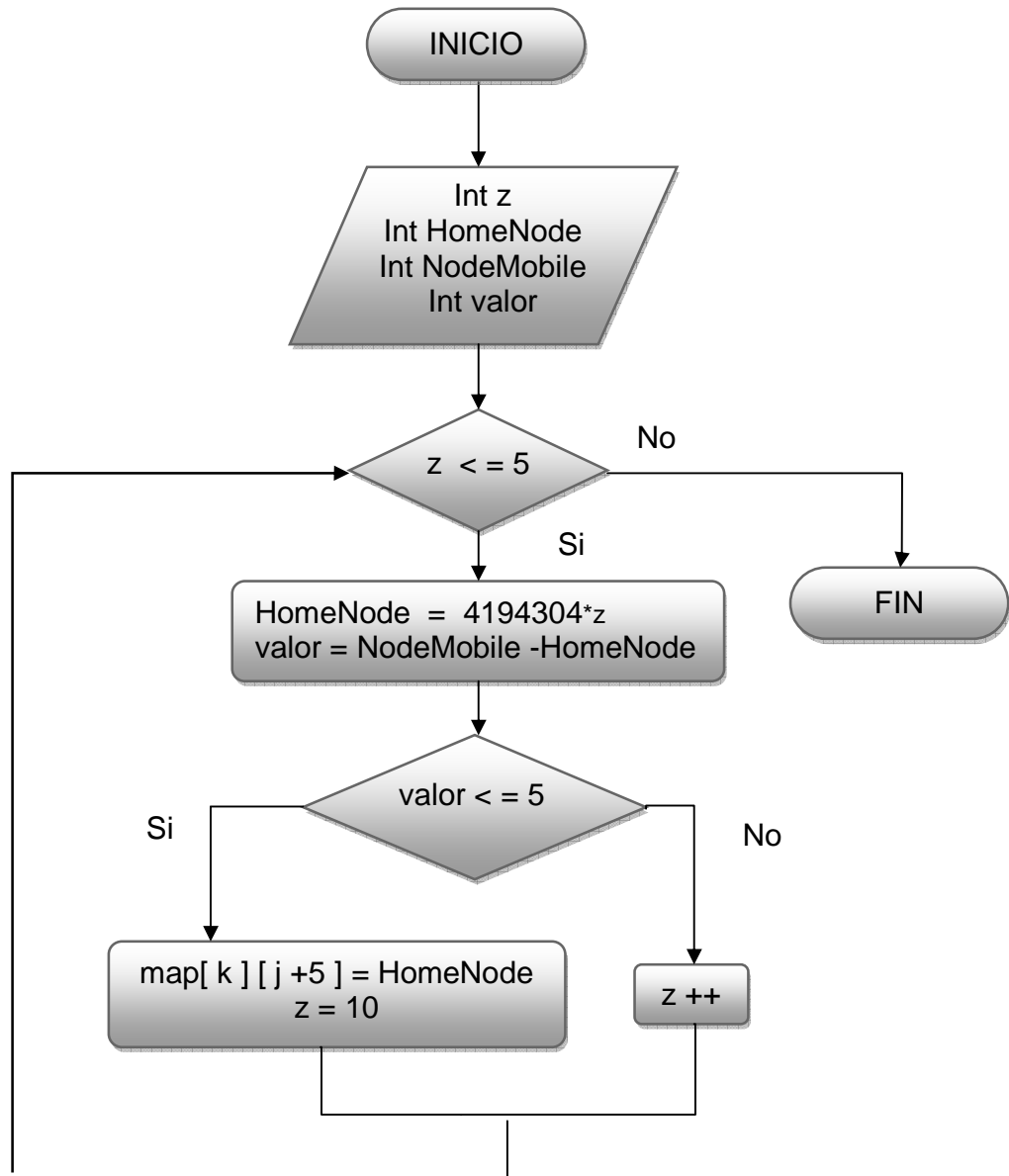
Se supone una dirección para él *HA* y se le resta la dirección del *MN*; si la diferencia entra estas dos direcciones es superior a 5, se aumenta el dominio de la dirección del *HA* en 1 y se vuelve a comparar. Este paso se repite hasta que la diferencia entre estas dos direcciones sea menor de 5.

Cuando ya se tiene la dirección del *HA*, se registra esta en los *slot* asignados para guardar estos datos.

En la figura 22 se puede ver de una forma más explícita el proceso de detección del *HA* a partir de la dirección del *MN* mediante un diagrama de bloques.

```
else if (ah == 2) {
    int j = 1;
    int k;
    k = bili.map_;
    while (j <= 5) {
        if (map[k][j] == bili.node_){
            j = 10;
            var = 10;
        }
        else if (map[k][j] == 0) {
            map[k][j] = bili.node_;
            int z = 0;
            while (5 >= z){
                int HomeNode = 4194304*z;
                int valor = map[k][j] - HomeNode;
                if (valor <= 5) {
                    map[k][j + 5] = HomeNode;
                    z = 10;
                }
                else {
                    z++;
                }
            }
            j = 10;
        }else {
            j++;
        }
    }
}
```

Figura 22: Detección del HA a partir del Nodo Móvil



Fuente: Autores.

- ✓ Valor de ah = 3: Busca y envía la dirección del *MAP* en que está registrado un *MN (Mobile Node)* con su respectivo *HA*. Con el número de la fila del *MAP* deseado, busca si *el MN* que se desea y luego su respectivo *HA*. Devuelve las direcciones del *MAP* y del *HA*

```

else if (ah == 3) {
    int j = 1;
    int k;
    k = bili.map_;
    while(j <= 5) {
        if (bili.node_ == map[k][j]) {
            bili.map_ = map[k][0];
            bili.ha_ = map[k][j+5];
            j = 10;
        }
        else {
            j++;
        }
    }
}

```

- ✓ Valor de ah = 4: Busca y envía el *MAP* en que se encuentra registrado un determinado *AR*. Devuelve el número de la fila del *MAP* en que se encuentra registrado un determinado *AR*.

```

else if (ah == 4) {
    int k = 0;
    while (k < 7){
        if ( (bili.node_ == map[k][11]) ||
            (bili.node_ == map[k][12]) ||
            (bili.node_ == map[k][13]) ||
            (bili.node_ == map[k][14]) ||
            (bili.node_ == map[k][15]))
        {
            var = k;
            k = 10;
        }
        else{
            k++;
            var = 10;
        }
    }
}

```

- ✓ Valor de ah = 5: Busca y envía el *MAP* en que está registrado un determinado *MN*. Devuelve el número de la fila del *MAP* en que se encuentra registrado un determinado *MN*.

```

else if (ah == 5) {
    int k = 0;
    while (k < 7){
        if ( (bili.node_ == map[k][1]) ||
            (bili.node_ == map[k][2]) ||
            (bili.node_ == map[k][3]) ||
            (bili.node_ == map[k][4]) ||
            (bili.node_ == map[k][5]))
        {
            var = k;
            k = 10;
        }
        else{
            k++;
            var = 10;
        }
    }
}

```

- ✓ Valor de ah = 6: Limpia el Registro de un *MN* que cambia de *MAP*

```

else if (ah == 6) {
    int j = 1;
    int k = bili.map_;
    while(j < 5) {

        if (bili.node_ == map[k][j]) {
            map[k][j] = 0;
            map[k][j+5] = 0;
            j = 10;
        }
        else {
            j++;
        }
    }
}

```



- ✓ Valor de *ah* = 7: Información general del registro del *MN*. Este proceso muestra, en consola, información de las direcciones de los componentes que se encuentran registrados en los *MAP* en el instante en que es llamado. Solo muestra las filas de los *MAP* que están siendo usadas.

```

else if (ah == 7) {
    int j;
    for (j = 1; j < 7; j++){
        if (map[j][0] != 0){
            printf("MAP [%s], ARouter %s %s \n",
                Address::instance().print_nodeaddr(map[j][0]),
                Address::instance().print_nodeaddr(map[j][11]),
                Address::instance().print_nodeaddr(map[j][12]));

            printf( "Mobile Node %s %s %s %s %s \n",
                Address::instance().print_nodeaddr(map[j][1]),
                Address::instance().print_nodeaddr(map[j][2]),
                Address::instance().print_nodeaddr(map[j][3]),
                Address::instance().print_nodeaddr(map[j][4]),
                Address::instance().print_nodeaddr(map[j][5]));
        }
    }
}

```

## 5.2.2 Optimización de Ruta

Una de las características importantes del protocolo *HMIPv6* [1] es la inclusión de la optimización de la ruta; la cual, ya no es opción como ocurría en el protocolo *MIPv4*, sino que por defecto ya se encuentra activada.

En la herramienta *HMIPv6\_SM* se utiliza por defecto la optimización de ruta, pero tiene la opción de desactivarla. Para la desactivación de la optimización de ruta, se crea dos variables, una en *TCL* (*Optimization\_*) y otra en *C++* (*opt\_*), las cuales se enlazan por medio de la instrucción *Otcl* (*bind*).

```
bind("Optimization_", &opt_);
```

Cuando el usuario quiere que un *MN* no cuente con optimización de la ruta, coloca la variable *Optimization\_* en 0 en el *script* después de definir el *MN* como se muestra a continuación:

```

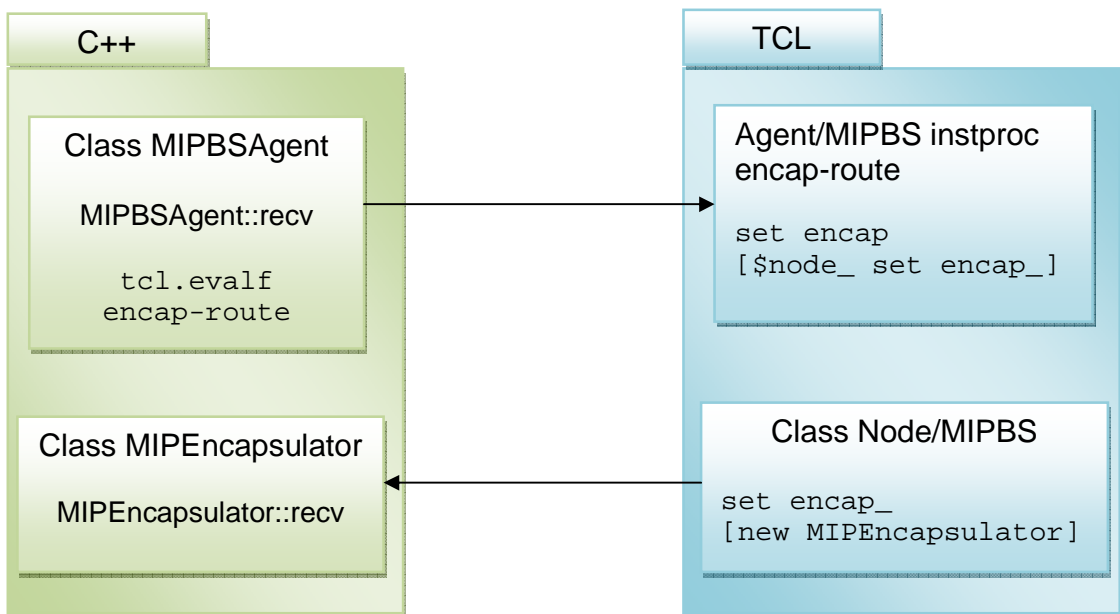
set MN [$ns_ node 1.0.1]
[$MN set regagent_] set Optimization_ 0

```

Al no estar la variable `Optimization_` definida desde el *script*, la optimización de ruta por defecto esta activa. Cuando esto ocurre, el *MAP* realiza el registro primero con el *HA*, de este modo, en los primeros paquetes enviados mostraran un enrutamiento triangulo normal mientras se realiza el registro con el *CN*.

Para la creación de la optimización de ruta, se toma el método *tcl/ encap-router*, el cual direcciona y tuneliza los paquetes entre el *HA* y el *MAP* en el enrutamiento triangulo (Ver figura 23). Tomando como base el método *encap-router*, se crea un nuevo método llamado *mencap-router*. Este es parecido al original, a excepción de que llama al método *recv* de la clase *OPTEncapsulate*; clase *C++* creada para realizar el procesamiento de las cabeceras *IP* de los paquetes y su envío al *MAP* sin realizar la encapsulación. (Ver figura 24).

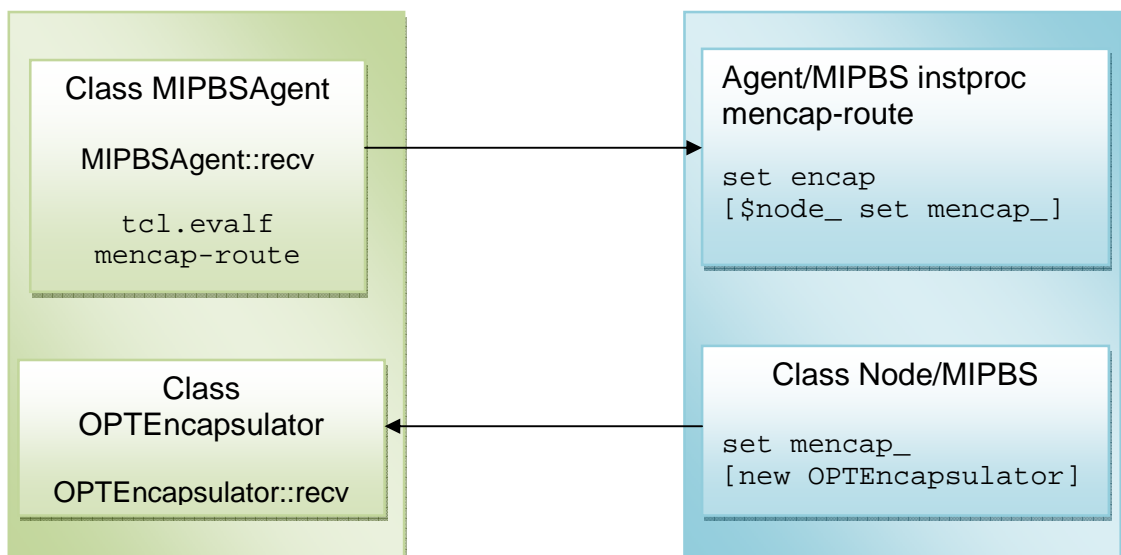
Figura 23: Registro y asignación del encapsulado en el Home Agent



Fuente: Autores.

Como el registro se hace desde *tcl*, se tiene que apartar un *CN* para los nodos móviles que tienen activado el mecanismo de optimización de ruta (en los primeros segundos de la comunicación entre *CN* y *MN*, se realiza un enrutamiento triangulo mientras se efectúa el registro para el direccionamiento directo del *CN* al *MAP*) y otro para los nodos móviles que realizaran un enrutamiento triangulo; ya que no se puede hacer que se ejecuten los procesos *mencap-router* y *encap-router* en un mismo *CN*. Para esto, se decide establecerle a los *MNs* con optimización de ruta el *CN* con dirección *0.0.0*; el cual será definida como estación base, ya que estos cuentan con el mecanismo de enrutamiento cableado, pero se le elimino la posibilidad de comunicación *wireless*, dejándolo como nodo fijo con posibilidad de registro, enrutamiento, encapsulamiento y desencapsulamiento.

Figura 24: Registro y asignación de proceso al Correspondent Node



Fuente: Autores.

### 5.2.3 Modificación del desencapsulamiento en los ARs

En *HMIP*, el *AR* desencapsulaba los paquetes que llegan a él para luego transmitirlos al *MN*. Pero esto no ocurre en el protocolo *HMIPv6* [1], el paquete es transmitido al *MN* encapsulado desde el *MAP*; es por eso que se implemento este aspecto a la Herramienta *HMIPv6\_SM*.

El proceso de desencapsulación en cualquier nodo de la simulación se realiza con la misma clase en C++, MIPDesencapsulator. Tomando en cuenta esta anotación, se decide crear una variable llamada (`optimy_`) para la estructura (`hdr_cmn`), la cual guarda la cabecera *common*.

```
struct hdr_cmn {
    enum dir_t { DOWN= -1, NONE= 0, UP= 1 };
    packet_t ptype_; // packet type (see above)
    int size_; // simulated packet size
    int uid_; // unique id
    int error_; // error flag
    int errbitcnt_; // # of corrupted bits jahn
    int fecsize_;
    int optimy_; // HMIPv6_SM
}
```

La variable (`optimy_`) nos servirá para determinar de qué nodo viene el paquete (*HA*, *MAP* o *AR*). Dependiendo del nodo que viene el paquete, se decide si se realiza la desencapsulación o no.

La razón por la que se resuelve colocarla en la cabecera *common* y no en otra cabecera (*mip*, *ip*, etc) es porque dicha cabecera está presente en todo el recorrido de los paquetes de datos, conservando sus valores en las clases de encapsulación y desencapsulación.

### 5.2.3.1 Valores de “`optimy_`” según el tipo de Enrutamiento.

Los valores de la variable `optimy_` varia dependiendo del nodo de donde es enviado el paquete. Es por eso que se mencionara los valores que esta variable toma cuando el envío de paquetes se hace a través de un enrutamiento triangulo y cuando se optimiza la ruta.

#### 5.2.3.1.1 Valores de “`optimy_`” en el Enrutamiento Triangulo

Inicialmente la variable `optimy_` esta en cero, indicando que el paquete viene del *CN*. Al llegar el paquete al *HA*, este llama un método que verifica si viene del *CN* y cambia la variable `optimy_` a 5, se trata del `MIPEncapsulatoreste::recv`:

```
if (ch->optimy() == 0){
    ch->optimy() = 5;
}
```

Luego el *HA* envía el paquete de datos al *MAP*, en donde se llama a la clase *MIPDesencapsulador*. Debido a que el paquete de datos pasó por el *HA* (Enrutamiento Triangulo), este viene encapsulado, es por eso que el *MAP* debe desencapsular el paquete. El método *MIPDesencapsulador::recv* comprueba el valor de *optimy\_* para verificar si el paquete viene del *HA*, realizar la desencapsulación y vuelve a cambiar el valor de *optimy\_* a 3.

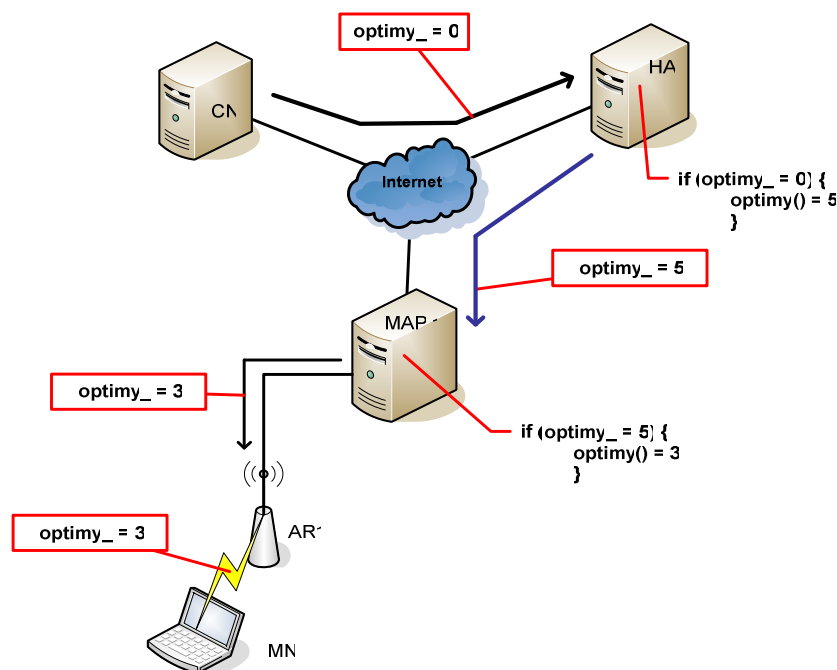
```

vartempol = ch->optimy();
if (vartempol == 5) {
    hdr_cmn::access(p)->size() -= IP_HEADER_SIZE;
    ch->optimy() = 3;
}

```

El *MAP*, encapsula el paquete (llama a la clase *MIPEncapsulador*) y lo envía a los *AR*. En los *ARs* (*Access Routers*), se llama al método de desencapsulamiento *MIPDesencapsulador::recv* el cual no quita la cabecera *IP\_HEADER\_SIZE* ya que el valor de la variable *optimy\_* es diferente a 5, transmitiendo el paquete encapsulado al *MN* (*Mobile Node*). Los valores de la variable *optimy\_* en un enrutamiento triangulo se describe en la figura 25.

Figura 25: Valores de la Variable “*optimy\_*” en el Enrutamiento Triangulo



Fuente: Autores.

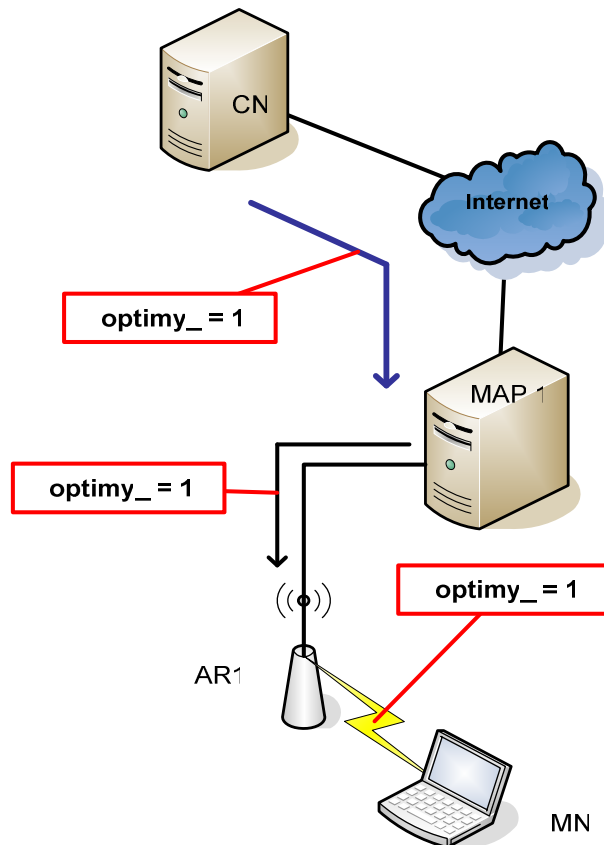
### 5.2.3.1.2 Valores de “optimy\_” en Optimización de Ruta

En el caso de que la optimización de la ruta este habilitada, el proceso descrito anteriormente varía levemente. Inicialmente valor de la variable `optimy_` es cero. Debido a que en la optimización de ruta, el *CN* y el *MAP* realizan un proceso de enrutamiento; el *CN* llama al método `OPTEncapsulator::recv` el cual cambia el valor de `optimy_` de cero a 1:

```
ch->optimy() = 1;
```

Esto le indica al *MAP*, que el paquete de datos no pasó por el *HA*, y no necesita desencapsularse. El proceso del *MAP* al *AR* y luego al *MN* se mantiene igual al descrito para enrutamiento triangulo. Los valores de la variable `optimy_` en la optimización de ruta se describen en la figura 26.

Figura 26: Valor de la Variable “optimy\_” en la Optimización de Ruta



Fuente: Autores.

## 6. SIMULACIONES CON HMIPv6\_SM

Con el propósito de demostrar el funcionamiento de la herramienta *HMIPv6\_SM*, se realizarán pruebas para varias topologías y distintas configuraciones de estas; en las cuales se mostrarán la configuración de red de cada simulación, su comportamiento con la visualización de los registros *traffic\_Simulacion.tr* y los mensajes que se despliegan en pantalla cuando se ejecuta el *script* para una explicación más clara; y por último las gráficas de paquetes recibidos y perdidos durante la transmisión.

Para quienes no estén familiarizados con el formato de las trazas, en el anexo C se explica con más detenimiento cada una de los campos en los que está dividido este formato.

### 6.1 SIMULACIÓN 1

En esta simulación se analizará el funcionamiento de una topología simple y, con la ayuda de los archivos *trace* (*traffic\_Simulacion1.tr*) y los mensajes que se despliegan en pantalla durante la simulación, se demostrará el correcto funcionamiento de la herramienta *HMIPv6\_SM* bajo un enrutamiento triángulo y con optimización de ruta.

#### 6.1.1 Configuración de la Simulación

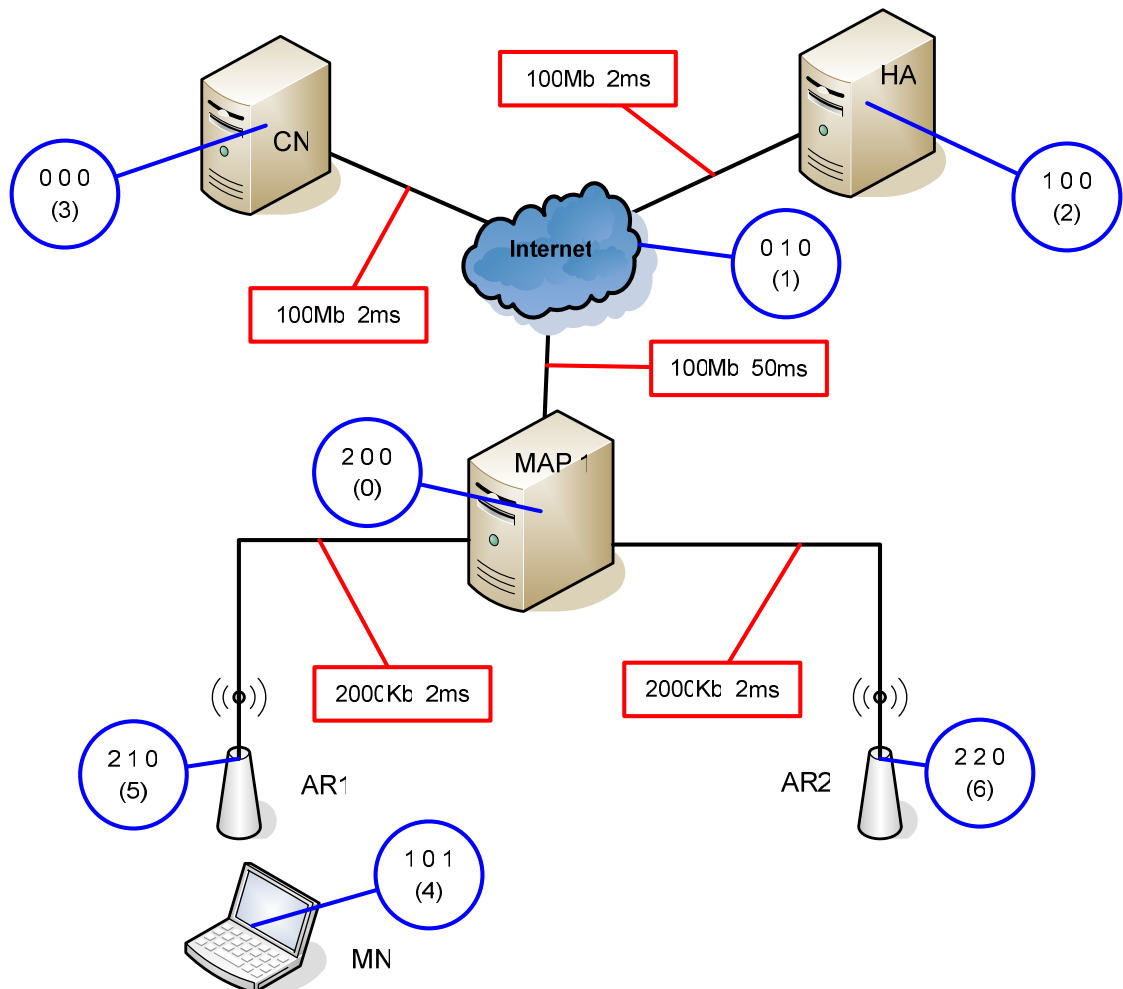
Esta simulación consta de un *CN* (*Correspondent Node*), un *Nodo Internet*, un *MAP* con 2 *ARs* (*Access Routers*), un *MN* (*Mobile Node*) con su respectivo *HA*. La dirección y el número correspondiente en la simulación con el *NAM* [3], se presentan en la tabla 3.

Tabla 3: Direcciones y número de nodo de los componentes de Simulación1

	<b>Home Agent</b>	<b>Nodo Móvil</b>	<b>MAP</b>	<b>Access Router 1</b>	<b>Access Router 2</b>	<b>Nodo Internet</b>	<b>Nodo Correspondiente</b>
<b>Dirección</b>	1.0.0	1.0.1	2.0.0	2.1.0	2.2.0	0.1.0	0.0.0
<b>Nodo</b>	2	4	0	5	6	1	3

En la figura 27 se muestra la configuración de la topología de Simulación1 con las direcciones y número de nodo de cada componente de la simulación, y el ancho de banda y retardo de los enlaces entre estos.

Figura 27: Topología de Simulación 1



Fuente: Autores

### 6.1.1.1 Configuración de la Red Fija

La configuración de la red fija está dispuesta así:

El enlace del el NI (*Nodo Internet*) con el CN (*Correspondent Node*) y HA (*Home Agent*) es un enlace tipo *duplex* con un ancho de banda de 100Mb, con tipo de cola RED (*Random Early Detection*) y con un tiempo de retardo máximo del paquete de 2ms.



```
$ns_ duplex-link $CN $N1 100Mb 2ms RED
$ns_ duplex-link $HA $N1 100Mb 2ms RED
```

El enlace del *NI* al *MAP* es un enlace tipo *duplex*, con ancho de banda de 100Mb, tipo de cola *RED* y tiempo de retardo máximo del paquete de 50ms

```
$ns_ duplex-link $MAP $N1 100Mb 50ms RED
```

El enlace del *MAP* a los *ARs*, al igual que los anteriores es de tipo *duplex*, con ancho de banda de 2000Kb, tiempo de retardo máximo del paquete de 2ms y tipo de cola *DropTail (FIFO)*.

```
$ns_ duplex-link $AR1 $MAP 2000Kb 2ms DropTail
$ns_ duplex-link $AR2 $MAP 2000Kb 2ms DropTail
```

Se usa un agente de transporte *UDP* atado al *CN*. Este se conecta a un generador de tráfico *CBR (Constant Bit Rate)* con un tamaño de paquete de 1000 *bytes* y con envío de paquete cada 0.05 segundos.

```
set udp [new Agent/UDP]
$ns_ attach-agent $CN $udp
set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 1000
$cbr set interval_ 0.05
$cbr attach-agent $udp
```

Al *MN* se ata un agente *LossMonitor* el cual nos servirá para poder monitorear el comportamiento de *MN* gracias las variables que posee dicho agente.

```
set null [new Agent/LossMonitor]
$ns_ attach-agent $MN $null
$ns_ connect $udp $null
```

El envío de datos comienza al segundo de haber comenzado la simulación.

```
$ns_ at 1.0 "$cbr start"
```

### 6.1.1.2 Configuración de la Red Inalámbrica

La configuración del comportamiento inalámbrico de la topología se mostrara a continuación:

```
set chan_ [new Channel/WirelessChannel]
```

```

$ns_ node-config -mobileIP ON \
    -adhocRouting AODV \
    -llType LL \
    -macType Mac/802_11 \
    -ifqType Queue/DropTail/PriQueue \
    -ifqLen 50 \
    -antType Antenna/OmniAntenna \
    -propType Propagation/TwoRayGround \
    -phyType Phy/WirelessPhy \
    -channel $chan_ \
    -topoInstance $topo \
    -wiredRouting ON \
    -agentTrace ON \
    -routerTrace ON \
    -macTrace OFF

```

Para esta topología se usa un protocolo de enrutamiento *adhoc AODV*, una antena *Omni-direccional*, modelo de propagación *TwoRayGround*, el protocolo *MAC 802.11* y un tipo de cola *DropTail / PriQueue*. También se activa las trazas de *router*, agente y *wiredRouting*

Los *ARs* tienen una cobertura inalámbrica de 100m, y el espacio entre ellos es de 70m, dejando así una intersección de las dos coberturas de 30m.

### 6.1.2 Configuración de Movimiento

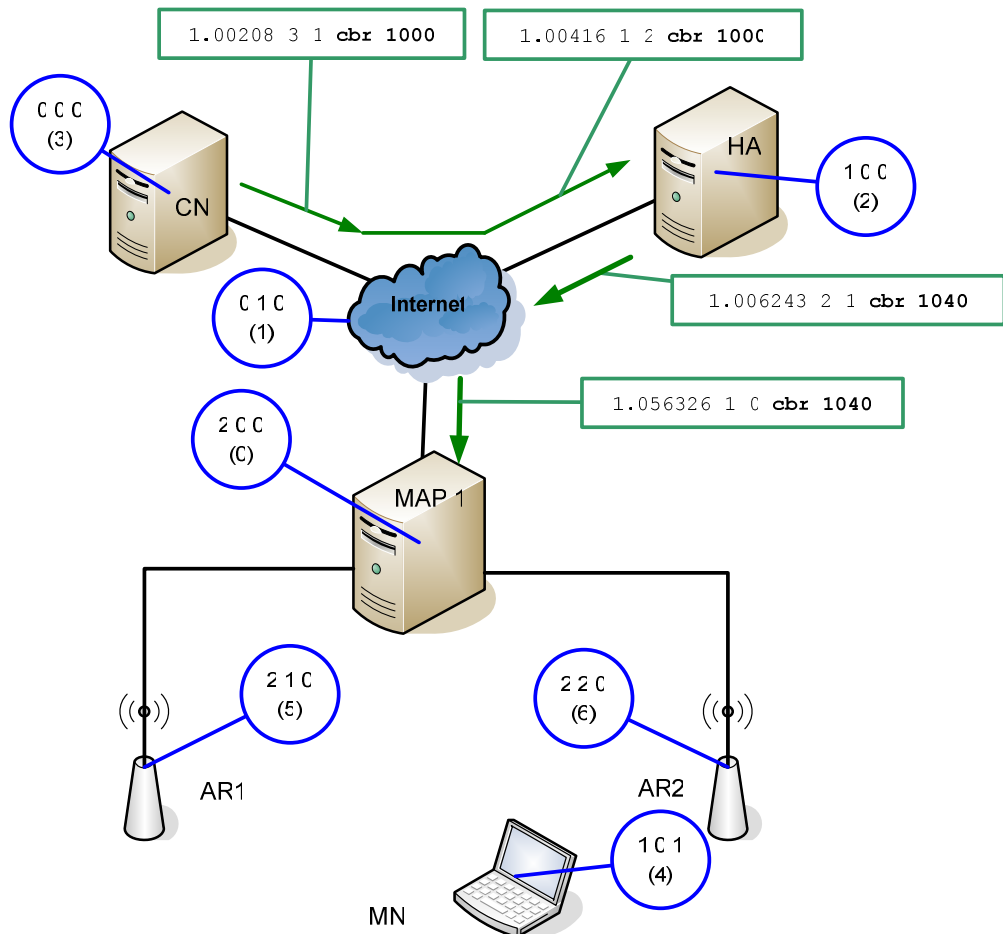
El *MN* se ubica en el *AR2* al comienzo de la simulación; transcurridos 20 segundos de inicializada la simulación, se mueve linealmente del *AR2* al *AR1* con una velocidad constante de 5m/s.

#### 6.1.2.1 Comportamiento de la Simulación

En los primeros momentos de simulación, se realizan reconocimiento de nodos y los pertinentes registros. Para mostrar cómo se visualiza en el archivo donde quedan registradas todas la trazas (*traffic\_Simulacion1.tr*) le hacemos seguimiento al paquete 7, eliminando cualquier otra línea que no tuviese relación con dicho paquete.

Cuando se inicia el envío de paquetes del *CN (Correspondent Node)* al *MN (Mobile Node)*; debido a que no hay optimización de ruta, *CN* envía el paquete hacia el *HA (Home Agent)*, este lo encapsula hacia el *MAP*. En la figura 28 se puede observar el recorrido del paquete en el envío de datos.

*Figura 28: Envío de paquete del CN al MAP en Enrutamiento Triangulo*



Fuente: Autores

Desde el archivo trace (traffic\_Simulacion1.tr) se puede hacer seguimiento al paquete. El paquete 7 es enviado de CN (3) a Internet (1) y de Internet (1) a HA (2) con 1000 bytes de tamaño.

```
+ 1 3 1 cbr 1000 ----- 0 0.0.0.2 1.0.1.2 0 7
- 1 3 1 cbr 1000 ----- 0 0.0.0.2 1.0.1.2 0 7
r 1.00208 3 1 cbr 1000 ----- 0 0.0.0.2 1.0.1.2 0 7
+ 1.00208 1 2 cbr 1000 ----- 0 0.0.0.2 1.0.1.2 0 7
- 1.00208 1 2 cbr 1000 ----- 0 0.0.0.2 1.0.1.2 0 7
r 1.00416 1 2 cbr 1000 ----- 0 0.0.0.2 1.0.1.2 0 7
```

El *HA* encapsula el paquete con una cabecera de 40 bytes para luego enviarla a Internet (1) y de este al MAP (0).

```
+ 1.00416 2 1 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7
- 1.00416 2 1 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7
r 1.006243 2 1 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7
+ 1.006243 1 0 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7
- 1.006243 1 0 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7
r 1.056326 1 0 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7
```

El *MAP* desencapsula y lo vuelve a encapsular el paquete hacia su *CoA* (en *NS*, es la dirección del *AR*) para luego ser enviado al *MN*. Ver figura 29.

```
+ 1.056326 0 6 cbr 1040 ----- 0 0.0.0.1 2.2.0.1 0 7
- 1.056326 0 6 cbr 1040 ----- 0 0.0.0.1 2.2.0.1 0 7
r 1.062486 0 6 cbr 1040 ----- 0 0.0.0.1 2.2.0.1 0 7
```

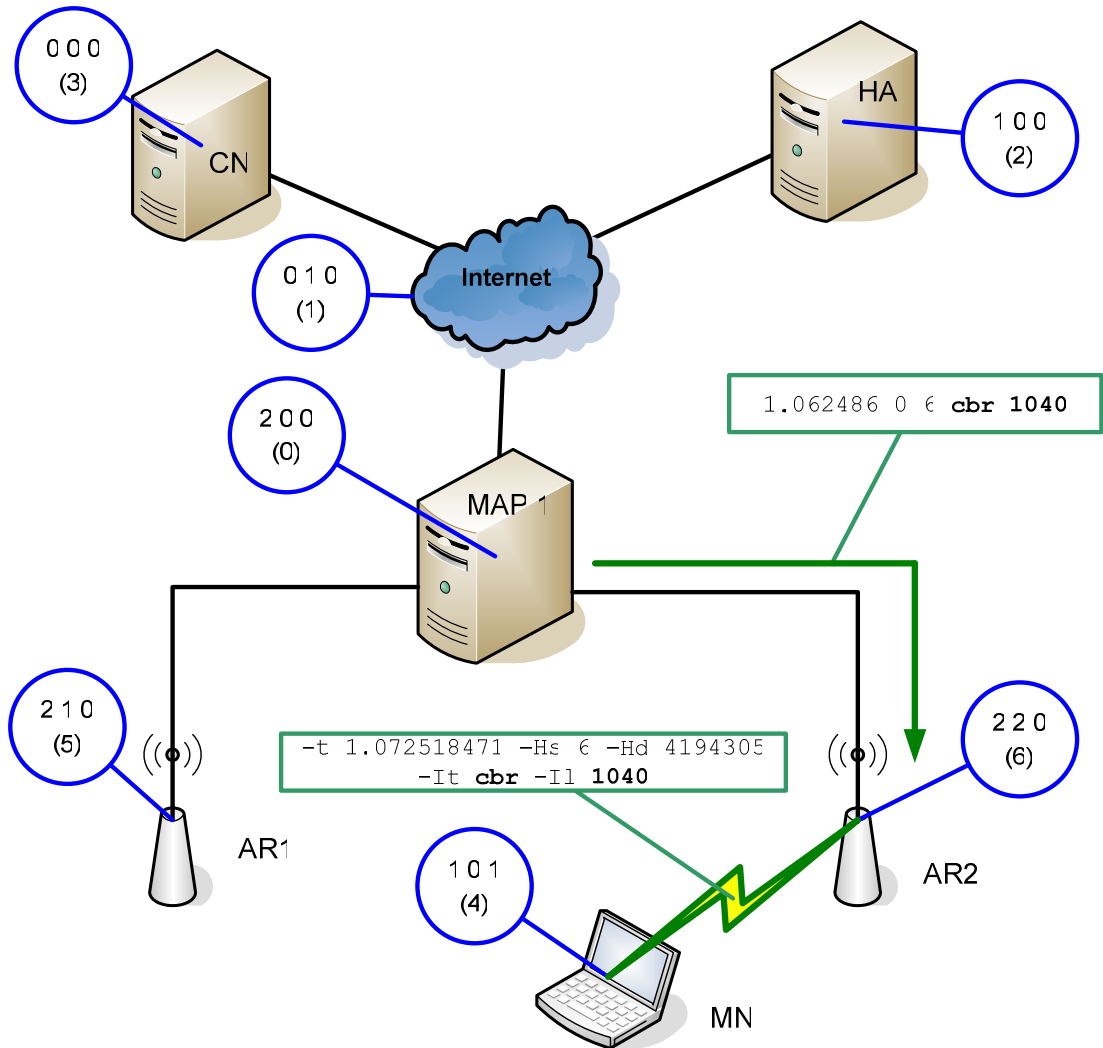
El paquete llega al *AR2* (6) a los 1.062486 segundos de iniciada la simulación y de ahí es transmitido a *MN* (Hd 4194305)

```
f -t 1.062486400 -Hs 6 -Hd 4194305 -Ni 6 -Nx 155.00 -Ny
135.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms
0 -Mt 0 -Is 0.2 -Id 4194305.2 -It cbr -Il 1040 -If 0 -Ii 7 -
Iv 27 -Pn cbr -Pi 0 -Pf 0 -Po 0
```

El paquete llega a *MN* a los 1.07219 segundos de iniciada la simulación.

```
r -t 1.072518471 -Hs 4 -Hd 4194305 -Ni 4 -Nx 150.00 -Ny
130.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 13a -Md 2 -
Ms 4 -Mt 800 -Is 0.2 -Id 4194305.2 -It cbr -Il 1040 -If 0 -Ii
7 -Iv 27 -Pn cbr -Pi 0 -Pf 1 -Po 0
```

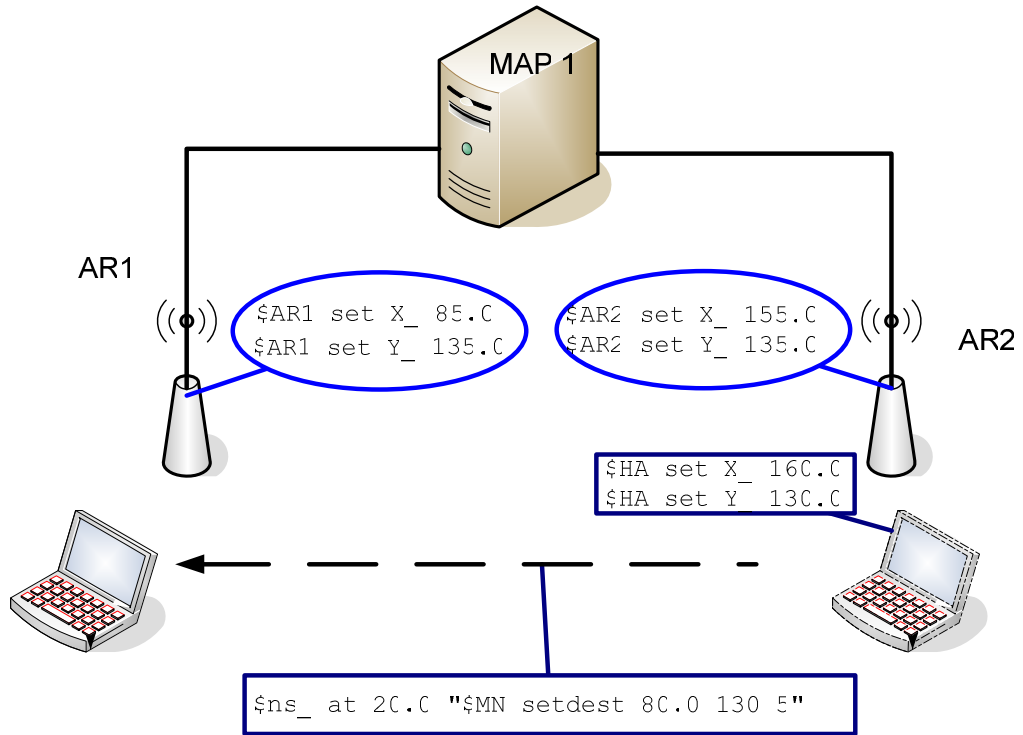
Figura 29: Envío de paquete del MAP al MN en Enrutamiento Triangulo



Fuente: Autores

A los 20 segundos de simulación, el MN que se encuentra conectado en el dominio de AR2, comienza a moverse linealmente al AR1 (ver figura 30).

Figura 30: Movimiento del Nodo Móvil de Simulación 1



Fuente: Autores

El MN cuando entra en la intersección de los campos de cobertura de AR1 y AR2, este mantiene recibiendo mensajes ADS de los dos, pero hace caso omiso de los que sean diferentes al AR actual hasta que su CoA caduque (1).

```
28.088123 BS send ADS 2.1.0
28.093458 MH 1.0.1 received ADS with coa 2.1.0
28.441173 BS send ADS 2.2.0
(1) 29.000000 current coa (2.2.0) timed out
```

Al caducar la CoA, este envía un *Binding Update* al MAP para registrar la nueva CoA, solo hasta que el registro con el MAP este completado, el MN no realizara el *handover* al otro Access Router (AR1). Mientras el MN hace sus respectivos registros con el MAP, este deja de recibir paquetes.

El nodo móvil (1.0.1) envía una petición de registro al MAP (2.0.0)

```
dst of request 2.0.0 (MAP), packet 704
29.000000 MH 1.0.1 sends request to 2.1.0
29.000340 BS send ADS 1.0.0
29.001788 BS 2.1.0 forwarding map-reg-request from 1.0.1 to
    MAP 2.0.0, packet 704
```

El *MAP* recibe la petición y actualiza la nueva *COA*

```
MAP [2.0.0], ARouter 2.1.0 2.2.0
Mobile Node 1.0.1 0.0.0 0.0.0 0.0.0 0.0.0
MAP[2.0.0] received MAP_REG_REQUEST
MAP Setting up encapsulate mechanism [1.0.1]->[2.1.0]
29.004076 MAP 2.0.0 updated (AR 2.1.0), packet 704
```

El *MAP* envía un mensaje de confirmación del registro al nodo móvil.

```
send MAP_REG_REPLY
BS recv MAP_REG_REPLY
29.008472 MH 1.0.1 received reply from MAP via BS 2.1.0,
    packet 704
Executing L2 handoff (MAP Mode)
29.028472 MH 1.0.1 has new coa (2.1.0) and new map (2.0.0)
29.028472 Complete L2 handoff
```

### 6.1.2.2 Resultados Obtenidos

Como se puede observar en las graficas, el handover de *AR2* a *AR1* se ve marcado por una disminución en los paquetes recibidos. Se comienza a perder paquetes desde el segundo 29 (ver Figura 31), tiempo en el cual, la *COA* actual del *MN* caduca (en ese instante 2.2.0). Al caducar la *COA* anterior, el *MN* busca y registra una nueva *COA*. Mientras *MN* realiza el proceso de búsqueda y registro, este deja de recibir paquetes provenientes del *CN*. (Ver Figura 32 y 33)

Figura 31: Pérdida de paquetes y paquetes recibidos de Simulación 1



Figura 32: Paquetes recibidos de Simulación 1.

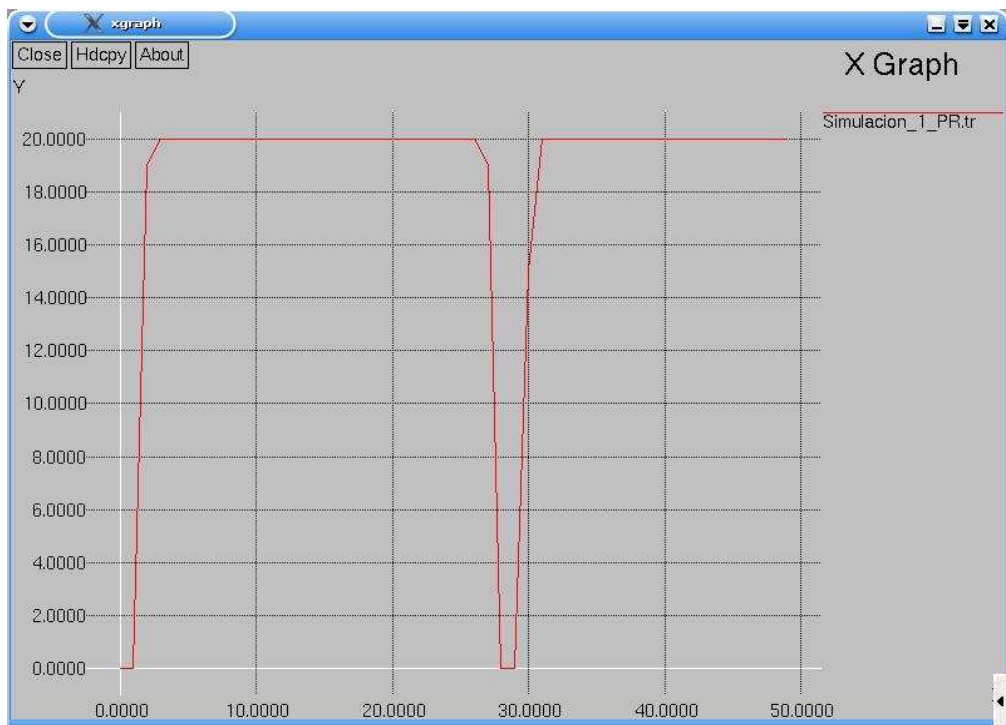
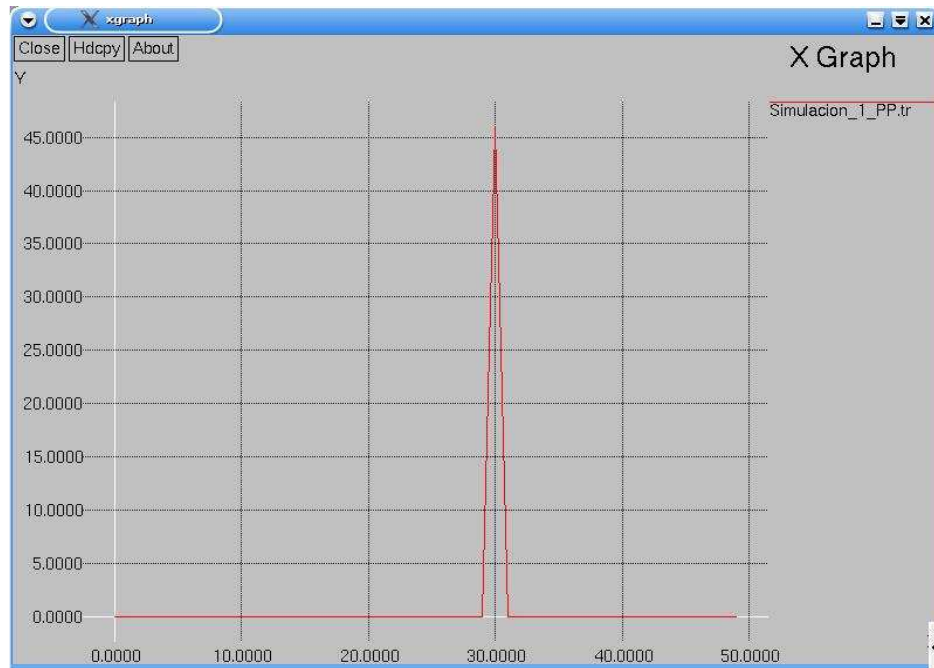




Figura 33: Pérdida de paquetes Simulación 1.



### 6.1.3 Configuración de Movimiento con *Optimización de Ruta*

Se utiliza la misma configuración de nodos y de movimiento del *MN* que en la configuración de movimiento anterior, a excepción de la activación de la optimización de ruta.

#### 6.1.3.1 Comportamiento de la Simulación

En el inicio de la simulación, el *MN* realiza el registro normalmente, se registra con el *MAP* 2.0.0 y luego con el *HA* 1.0.0, dando como resultado un enrutamiento triangulo.

Cuando comienza el flujo de datos y esta activada la optimización de ruta, vuelve a registrarse con el *MAP* 2.0.0, para luego enviar un *BU* (*Binding Update*) al *CN* (*Correspondent Node*) 0.0.0, que establece un mecanismo de direccionamiento del *CN* al *MAP*.

El nodo móvil envía una petición de registro al *CN* (0.0.0).

```
dst of request 0.0.0, packet 13
1.107249 MH 1.0.1 sends request to 2.1.0
1.109621 BS 2.1.0 forwarding reg-request from 1.0.1 to
    0.0.0, packet 13
```

CN recibe la petición y establece un mecanismo de direccionamiento con el MAP

CN Setting up addressing mechanism to [2.0.0]

```
1.163921 CN 0.0.0 updated (MAP 2.0.0), packet 13
1.218220 BS 2.1.0 received reply from 0.0.0, packet 13
```

Mientras el MN efectúa el pertinente registro con el CN, el envío de paquetes se hace por enrutamiento triangulo. Ver figura 34.

El paquete sale de CN al nodo Internet (1), con destino a HA (2). Ahí el paquete es tunelizado al MAP (4) pasando primero por el nodo internet. Al agregarle la cabecera MIPv6 [2] de 40 Bytes al paquete en el HA, aumenta el tamaño del paquete de 1000 Bytes a 1040 Bytes. En el MAP, el paquete es desencapsulado y vuelto a encapsular para ser enviado al AR (5), que transmitirá el paquete de 1040 Bytes al nodo móvil (6).

```
+ 1 3 1 cbr 1000 ----- 0 0.0.0.2 1.0.1.2 0 7
- 1 3 1 cbr 1000 ----- 0 0.0.0.2 1.0.1.2 0 7
(1) r 1.00208 3 1 cbr 1000 ----- 0 0.0.0.2 1.0.1.2 0 7

+ 1.00208 1 2 cbr 1000 ----- 0 0.0.0.2 1.0.1.2 0 7
- 1.00208 1 2 cbr 1000 ----- 0 0.0.0.2 1.0.1.2 0 7
(2) r 1.00416 1 2 cbr 1000 ----- 0 0.0.0.2 1.0.1.2 0 7

+ 1.00416 2 1 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7
- 1.00416 2 1 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7
(3) r 1.006243 2 1 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7

+ 1.006243 1 0 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7
- 1.006243 1 0 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7
(4) r 1.056326 1 0 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7

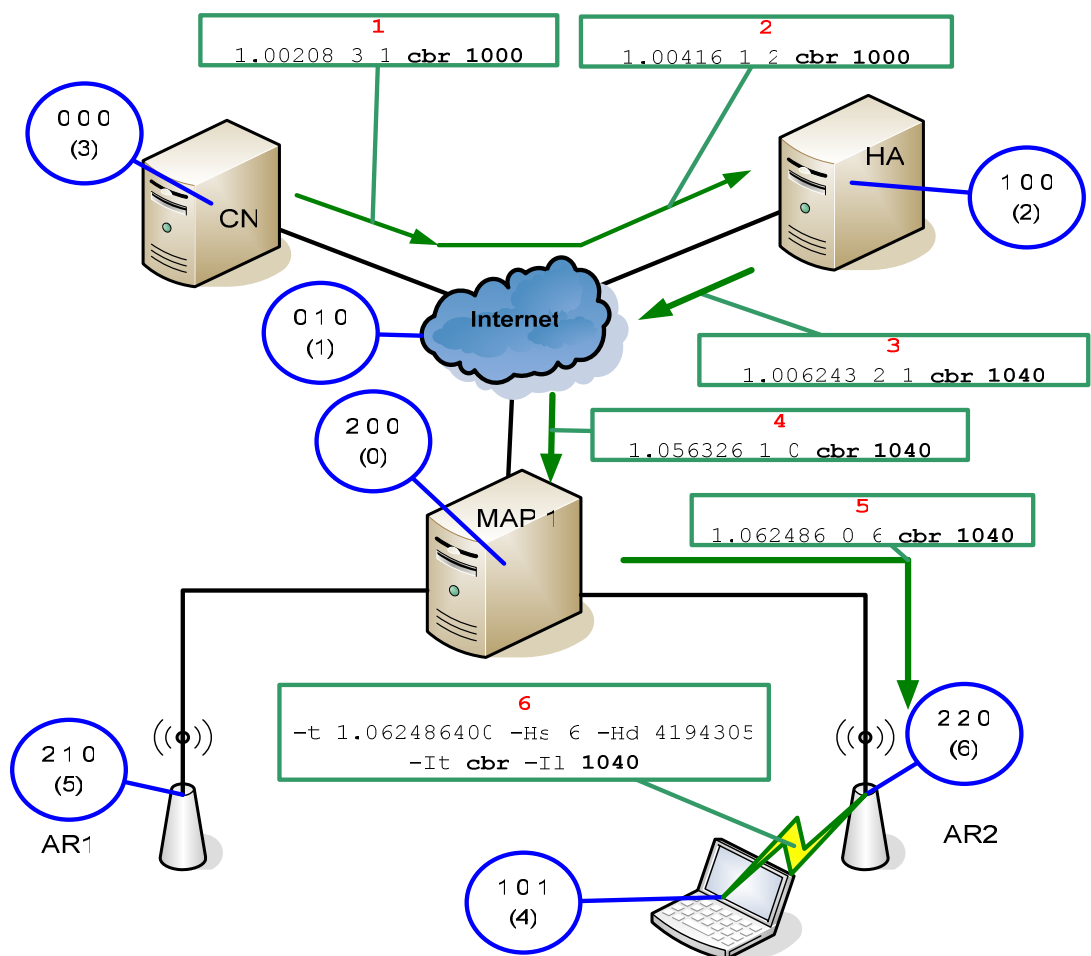
+ 1.056326 0 6 cbr 1040 ----- 0 0.0.0.1 2.1.0.1 0 7
- 1.056326 0 6 cbr 1040 ----- 0 0.0.0.1 2.1.0.1 0 7
(5) r 1.062486 0 6 cbr 1040 ----- 0 0.0.0.1 2.1.0.1 0 7

(6) f -t 1.062486400 -Hs 6 -Hd 4194305 -Ni 6 -Nx 155.00 -Ny
135.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms
```

```
0 -Mt 0 -Is 0.2 -Id 4194305.2 -It cbr -Il 1040 -If 0 -Ii 7 -
Iv 27 -Pn cbr -Pi 0 -Pf 0 -Po 0
```

```
r -t 1.072198471 -Hs 4 -Hd 4194305 -Ni 4 -Nx 150.00 -Ny
130.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 13a -Md 2 -
Ms 3 -Mt 800 -Is 0.2 -Id 4194305.2 -It cbr -Il 1040 -If 0 -Ii
7 -Iv 27 -Pn cbr -Pi 0 -Pf 1 -Po 0
```

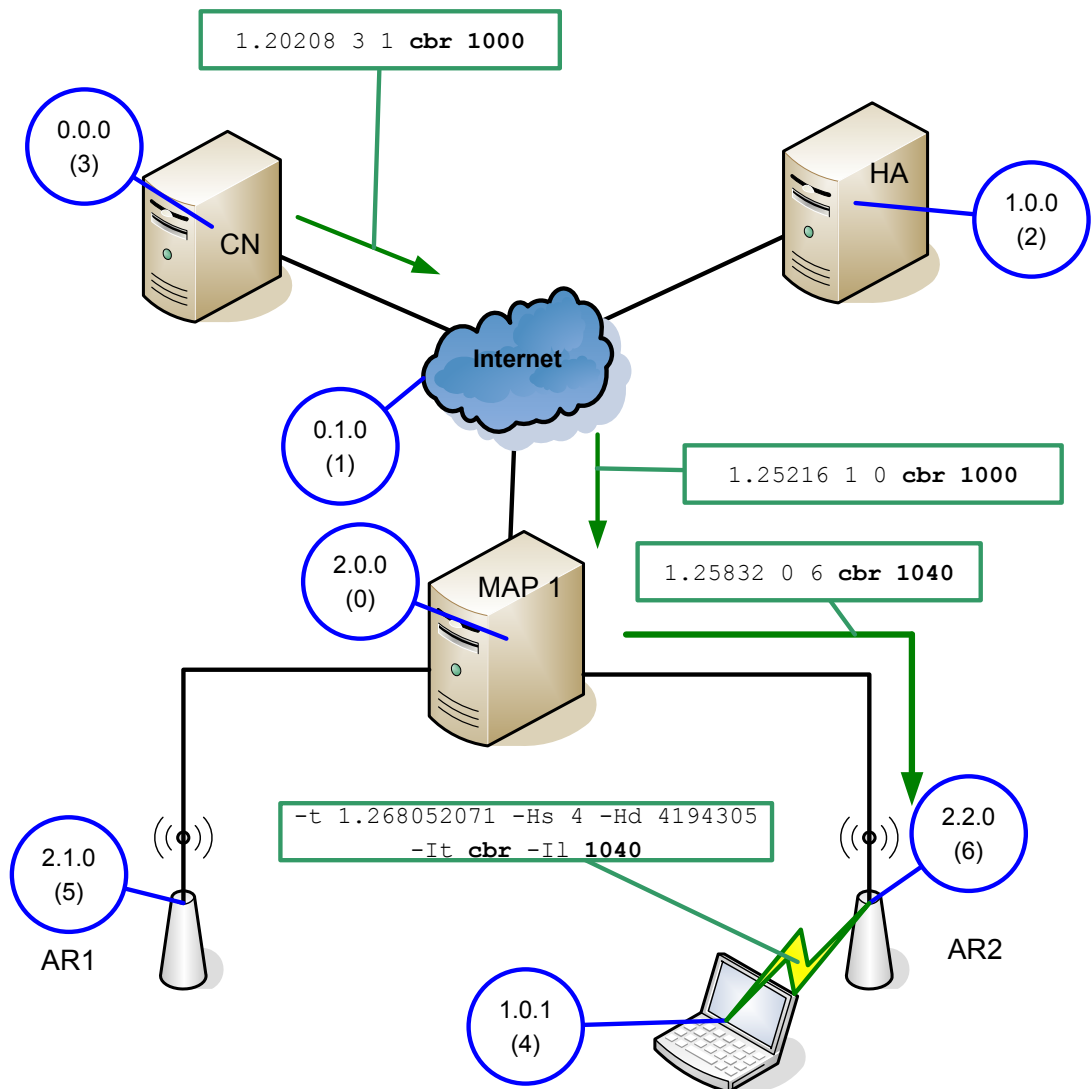
Figura 34: Enrutamiento Triangulo en Simulación1



Fuente: Autores.

Terminado el registro con el CN (Correspondent Node), los paquetes son enviados directamente al MAP. Ver figura 35.

Figura 35: Optimización de Ruta en Simulación1.



Fuentes: Autores.

A los 1.2 segundos de iniciada la simulación sale el paquete 15 de CN (3) a Internet (1) con destino al MAP (0), el tamaño del paquete es de 1000 bytes.

Como se observa, el paquete llega directamente al MAP sin estar encapsulado.

```

+ 1.2 3 1 cbr 1000 ----- 0 0.0.0.1 2.0.0.1 4 15
- 1.2 3 1 cbr 1000 ----- 0 0.0.0.1 2.0.0.1 4 15
r 1.20208 3 1 cbr 1000 ----- 0 0.0.0.1 2.0.0.1 4 15
+ 1.20208 1 0 cbr 1000 ----- 0 0.0.0.1 2.0.0.1 4 15
- 1.20208 1 0 cbr 1000 ----- 0 0.0.0.1 2.0.0.1 4 15
r 1.25216 1 0 cbr 1000 ----- 0 0.0.0.1 2.0.0.1 4 15

```

En el *MAP* (0) encapsula el paquete y lo envía al *AR* (6)

```

+ 1.25216 0 6 cbr 1040 ----- 0 0.0.0.1 2.1.0.1 4 15
- 1.25216 0 6 cbr 1040 ----- 0 0.0.0.1 2.1.0.1 4 15
r 1.25832 0 6 cbr 1040 ----- 0 0.0.0.1 2.1.0.1 4 15

```

Para luego transmitir el paquete al Nodo Móvil. Como se puede apreciar en los registros del archivo trace, el paquete no es desencapsulado en el *AR*; al no suceder la desencapsulación, el paquete es enviado al nodo móvil con la cabecera *MIPv6* [2] de 40 Bytes.

```

f -t 1.258320000 -Hs 6 -Hd 4194305 -Ni 6 -Nx 155.00 -Ny
135.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms
0 -Mt 0 -Is 0.2 -Id 4194305.2 -It cbr -Il 1040 -If 0 -Ii 15 -
Iv 29 -Pn cbr -Pi 4 -Pf 0 -Po 0

```

```

r -t 1.268052071 -Hs 4 -Hd 4194305 -Ni 4 -Nx 150.00 -Ny
130.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 13a -Md 2 -
Ms 3 -Mt 800 -Is 0.2 -Id 4194305.2 -It cbr -Il 1040 -If 0 -Ii
15 -Iv 29 -Pn cbr -Pi 4 -Pf 1 -Po 0

```

### 6.1.3.2 Resultados Obtenidos

Como la optimización de ruta no interviene en el *handover* del *MN*, la grafica de pérdida de paquetes y paquetes recibidos es la misma mostrada en la simulación anterior.

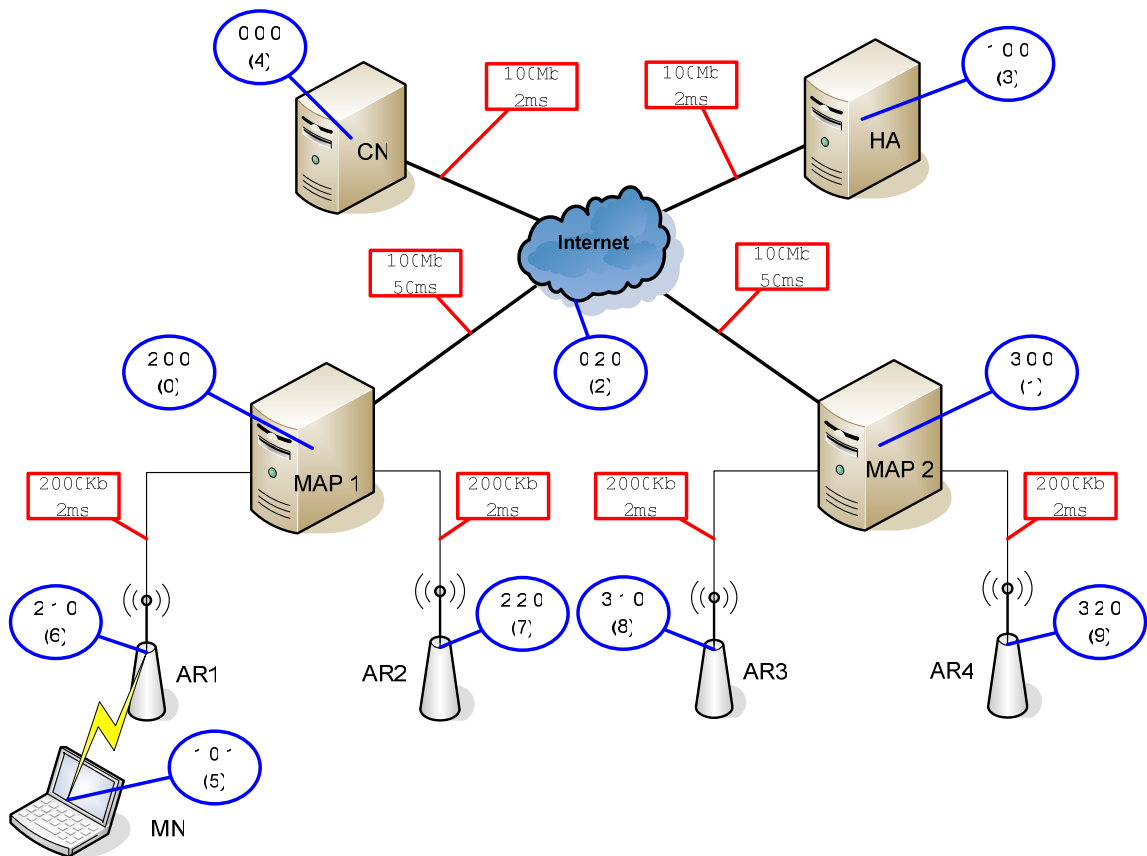
## 6.2 SIMULACIÓN 2

En esta simulación, se pretende comprobar el funcionamiento de la herramienta *HMIPv6\_SM* con la simulación de una topología de dos *MAPs*. Se evaluara la topología para un enrutamiento triangulo y optimización de ruta.

## 6.2.1 Configuración de Red

Esta simulación consta de un CN, un *Nodo Internet*, 2 MAPs con 2 ARs por cada MAP, un MN con su respectivo HA. Ver figura 36.

Figura 36: Topología de Simulación 2



Fuente: Autores.

La dirección y el número correspondiente en la simulación con el NAM [3], se presentan en la tabla 4.

Tabla 4: Direcciones y número de nodo de los componentes de Simulación 2

	HA	MN	MAP1	AR 1	AR 2	MAP2	AR 3	AR 4	NI	CN
<b>Dirección</b>	1.0.0	1.0.1	2.0.0	2.1.0	2.2.0	3.0.0	3.1.0	3.2.0	0.1.0	0.0.0
<b>Nodo</b>	3	5	0	6	7	1	8	9	2	4

### 6.2.1.1 Configuración de Red Fija

La configuración de la red fija está dispuesta así:

El enlace del el *Nodo Internet (NI)* con el *CN (Correspondent Node)* y *HA (Home Agent)* es un enlace tipo *duplex* con un ancho de banda de 100Mb, con tipo de cola *RED (Random Early Detection)* y con un tiempo de retardo máximo del paquete de 2ms.

```
$ns_ duplex-link $CN $N1 100Mb 2ms RED
$ns_ duplex-link $HA $N1 100Mb 2ms RED
```

El enlace del *Nodo Internet* al *MAP1* y el *Nodo Internet* al *MAP2* es un enlace tipo *duplex*, con ancho de banda de 100Mb, tipo de cola *RED* y tiempo de retarde máximo del paquete de 50ms

```
$ns_ duplex-link $MAP1 $N1 100Mb 50ms RED
$ns_ duplex-link $MAP2 $N1 100Mb 50ms RED
```

El enlace de los *MAPs* a sus respectivos *ARs*, es de tipo *duplex*, con ancho de banda de 2000Kb, tiempo de retardo máximo del paquete de 2ms y tipo de cola *DropTail (FIFO)*.

```
$ns_ duplex-link $AR1 $MAP1 2000Kb 2ms DropTail
$ns_ duplex-link $AR2 $MAP1 2000Kb 2ms DropTail
$ns_ duplex-link $AR3 $MAP2 2000Kb 2ms DropTail
$ns_ duplex-link $AR4 $MAP2 2000Kb 2ms DropTail
```

Se usa un agente de transporte *UDP* atado al nodo *CN*. Este se conecta a un generador de trafico *CBR (Constant Bit Rate)* con un tamaño de paquete de 1000 *bytes* y con envío de paquete cada 0.05 segundos.

```
set udp [new Agent/UDP]
$ns_ attach-agent $CN $udp
set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 1000
$cbr set interval_ 0.05
$cbr attach-agent $udp
```

Al *MN* se enlaza un agente *LossMonitor* el cual nos servirá para poder monitorear el comportamiento de *MN* gracias las variables que posee dicho agente.

```

set null [new Agent/LossMonitor]
$ns_ attach-agent $MN $null
$ns_ connect $udp $null

```

El envío de datos comienza al segundo de haber comenzado la simulación.

```

$ns_ at 1.0 "$cbr start"

```

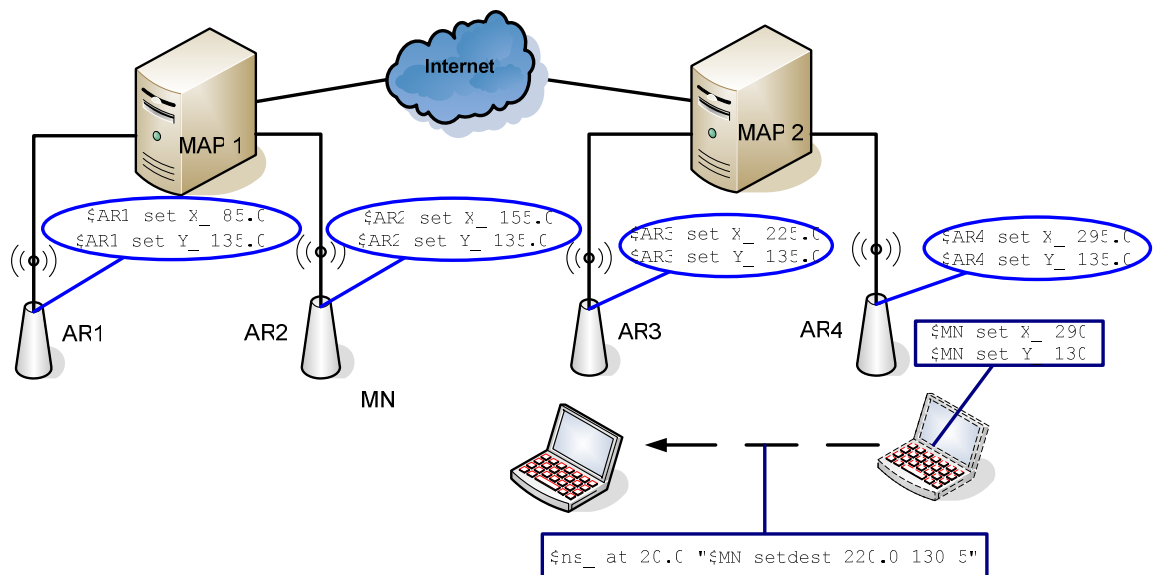
### 6.2.1.2 Configuración de Red Inalámbrica

La configuración del comportamiento inalámbrico es la misma configuración usada en la simulación 1

### 6.2.2 Configuración de Movimiento

El MN se ubica en el AR4 al comienzo de la simulación; transcurridos 20 segundos de inicializada la simulación, se mueve linealmente del AR4 al AR3 con una velocidad constante de 5m/s (ver figura 37). Luego, a los 40 segundos, se traslada al AR2 (ver figura 38). Espera en este hasta los 60 segundos cuando el MN viaja al AR1 (ver figura 39).

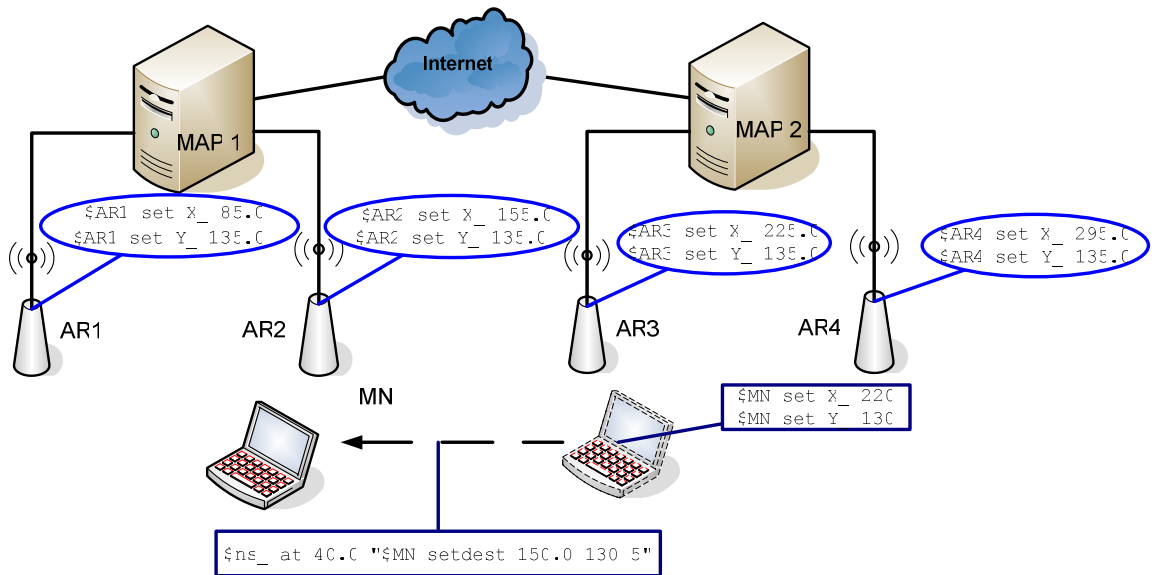
Figura 37: Movimiento del Nodo Móvil en Simulacion2 de AR4 a AR3



Fuente: Autores.

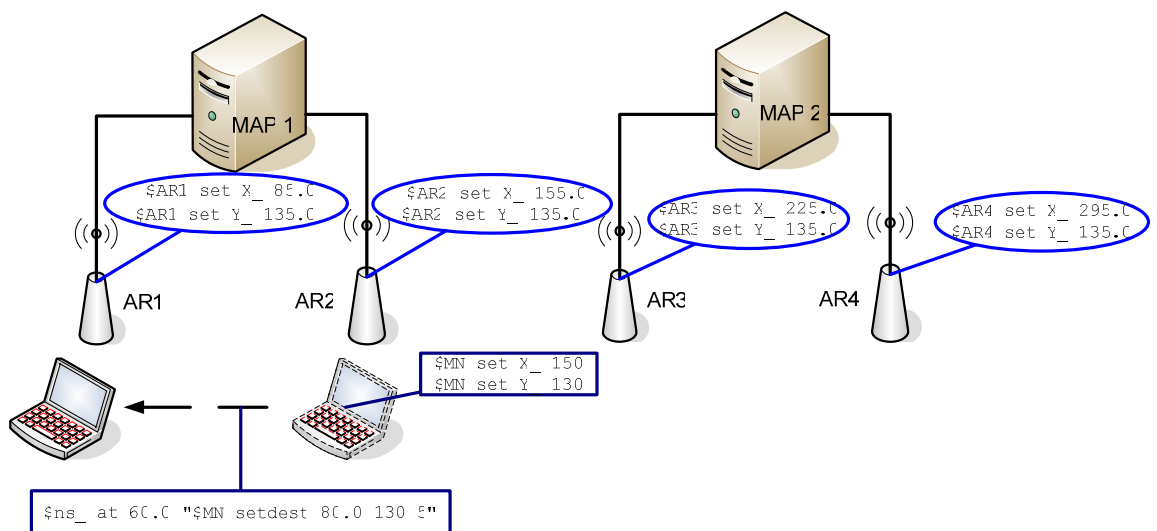
Figura 38: Movimiento del Nodo Móvil en Simulacion2 de AR3 a AR2





Fuente: Autores

Figura 39: Movimiento del Nodo Móvil en Simulacion2 de AR3 a AR2



Fuente: Autores

### 6.2.2.1 Comportamiento de la Simulación

Cuando el nodo móvil realiza un handover entre ARs pertenecientes al mismo dominio MAP (handover entre AR4 al AR3 y AR2 al AR1), el comportamiento de la simulación es exactamente igual al presentado en la Simulación1.

Cuando el nodo móvil pasa de un AR a otro AR pertenecientes a diferentes MAP (handover entre AR3 al AR2), el nodo móvil envía una petición de registro MAP al nuevo MAP. Cuando el registro con el nuevo MAP es satisfactorio, el nodo móvil envía un BU al HA para darle a conocer a este del cambio del dominio MAP.

El nodo móvil se encuentra en AR3 (3.1.0) moviéndose hacia AR2.

```
MAP [2.0.0], ARouter 2.1.0 2.2.0
Mobile Node 0.0.0 0.0.0 0.0.0 0.0.0 0.0.0
MAP [3.0.0], ARouter 3.1.0 3.2.0
Mobile Node 1.0.1 0.0.0 0.0.0 0.0.0 0.0.0
MAP[3.0.0] received MAP_REG_REQUEST
```

Cuando el nodo móvil entra al dominio del AR2 (2.2.0), este detecta el mensaje ADS de AR2 con CoA 2.2.0

```
45.441173 BS send ADS 2.2.0
45.444002 MH 1.0.1 received ADS with coa 2.2.0
```

Inmediatamente el nodo móvil envía una petición de registro al nuevo MAP

```
dst of request 2.0.0 (MAP), packet 1208
45.444002 MH 1.0.1 sends request to 2.2.0
45.452364 BS 2.2.0 forwarding map-reg-request from 1.0.1 to
    MAP 2.0.0, packet 1208
```

Cuando el MAP recibe la petición del nodo móvil, este realiza el registro del nuevo nodo móvil y envía un mensaje de respuesta al nodo móvil (1)

```
MAP [2.0.0], ARouter 2.1.0 2.2.0
Mobile Node 1.0.1 0.0.0 0.0.0 0.0.0 0.0.0
MAP [3.0.0], ARouter 3.1.0 3.2.0
Mobile Node 0.0.0 0.0.0 0.0.0 0.0.0 0.0.0
```

```
MAP[2.0.0] received MAP_REG_REQUEST
MAP Setting up encapsulate mechanism [1.0.1]->[2.2.0]
45.454652 MAP 2.0.0 updated (AR 2.2.0), packet 1208
send MAP_REG_REPLY (1)
```

EL nodo móvil recibe el mensaje de respuesta de registro por parte del *MAP*, e inmediatamente envía un *BU* al *HA* para actualizar la nueva *CoA* (2.2.0)

```
BS recv MAP_REG_REPLY
45.459389 MH 1.0.1 received reply from MAP via BS 2.2.0
      [COA: 2.2.0], packet 1208
dst of request 1.0.0 (HA), packet 1210
45.459389 MH 1.0.1 sends request to 3.1.0
45.461501 BS 3.1.0 forwarding reg-request from 1.0.1 to
      HA 1.0.0, packet 1210
```

El *HA* actualiza la *RCoA*, envía un mensaje de respuesta de registro al *MN* (2) y de una vez envía los paquetes hacia el nuevo *MAP* (3)

```
Normal rego
BS Setting up encapsulate mechanism [1.0.1]->[2.0.0]
45.515801 HA 1.0.0 updated (RCOA 2.0.0), packet 1210 (2)

45.554160 ENCAP TunnelExit:[2.0.0/1] OrigPkt:[1.0.1],
      packet 1212 (3)
45.606326 DECAP[2.0.0] Remove outer header [2.0.0] Restore
      inner header [1.0.1:0.0.0], packet 1212
45.606326 ENCAP TunnelExit:[2.2.0/1] OrigPkt:[1.0.1],
      packet 1212
45.612486 DECAP[2.2.0] Remove outer header [2.2.0] Restore
      inner header [1.0.1:0.0.0], packet 1212
```

### 6.2.2.2 Resultados Obtenidos

A los 28 segundos de simulación, el nodo móvil comienza a disminuir los paquetes recibidos (ver figura 42) debido a que se está produciendo el handover entre AR4 y AR3 (ver figura 41). Luego a los 45.4 segundos de simulación, el nodo móvil inicia el handover, pero esta vez entre AR3 y AR2, los cuales pertenecen a distinto MAP.

Figura 40: Pérdida de paquetes y paquetes recibidos de Simulación 2

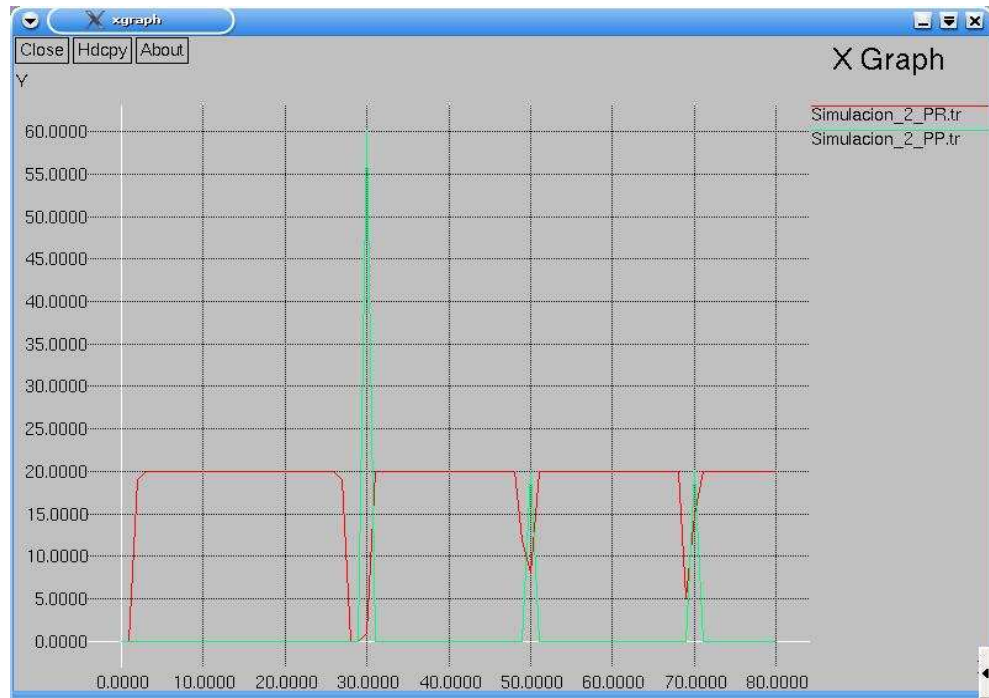


Figura 41: Pérdida de paquetes de Simulación 2

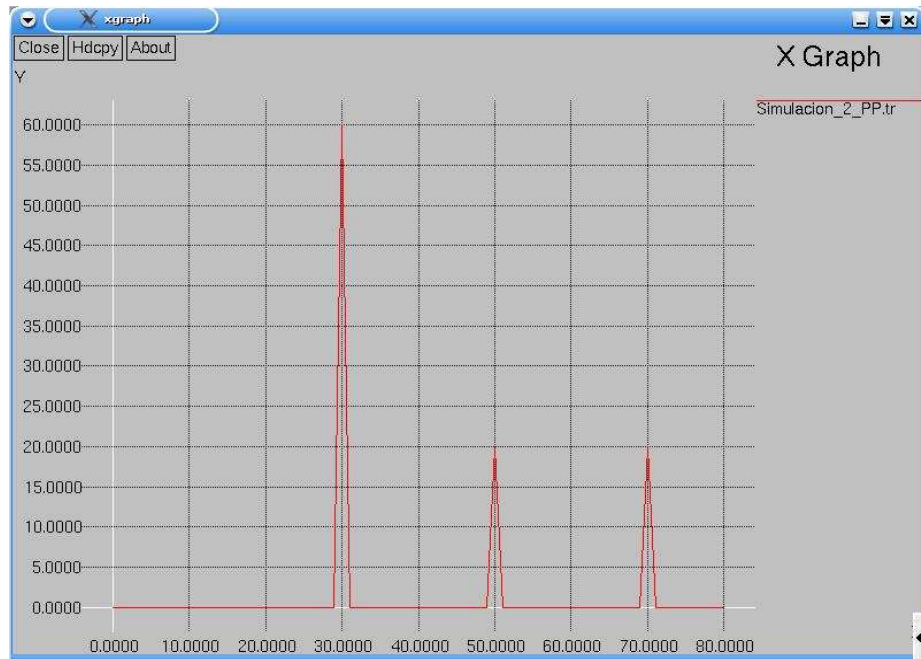
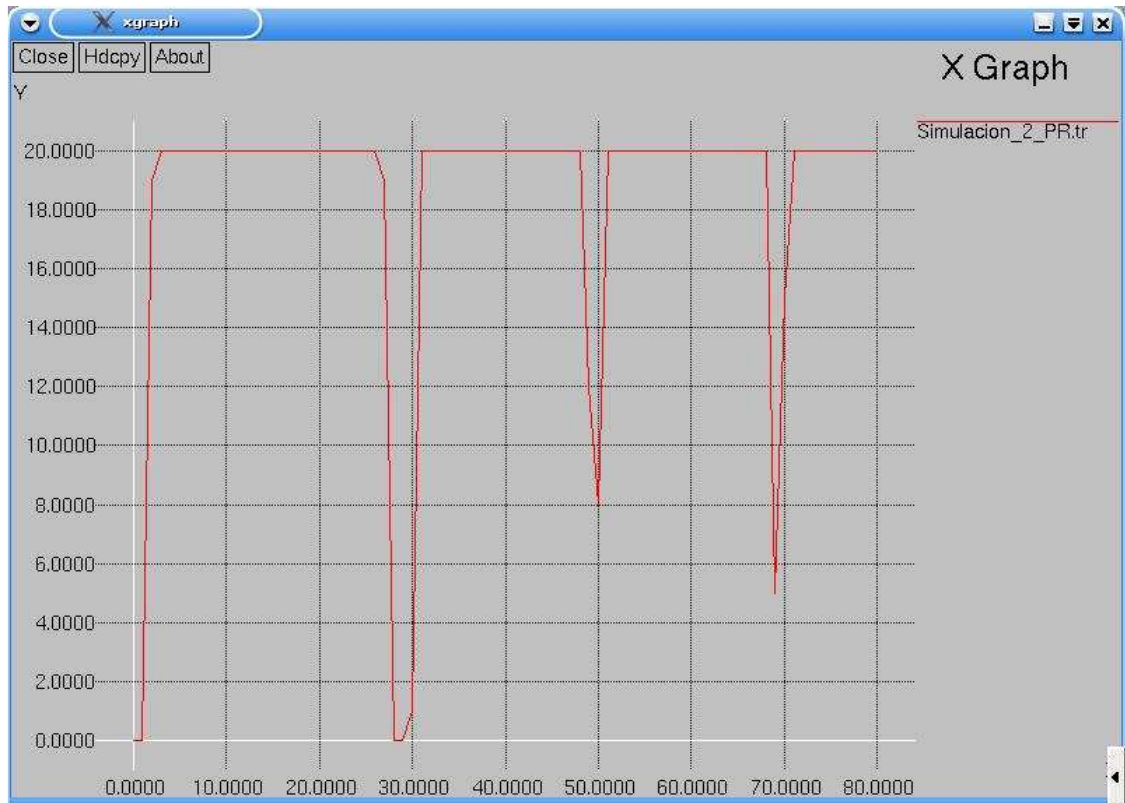


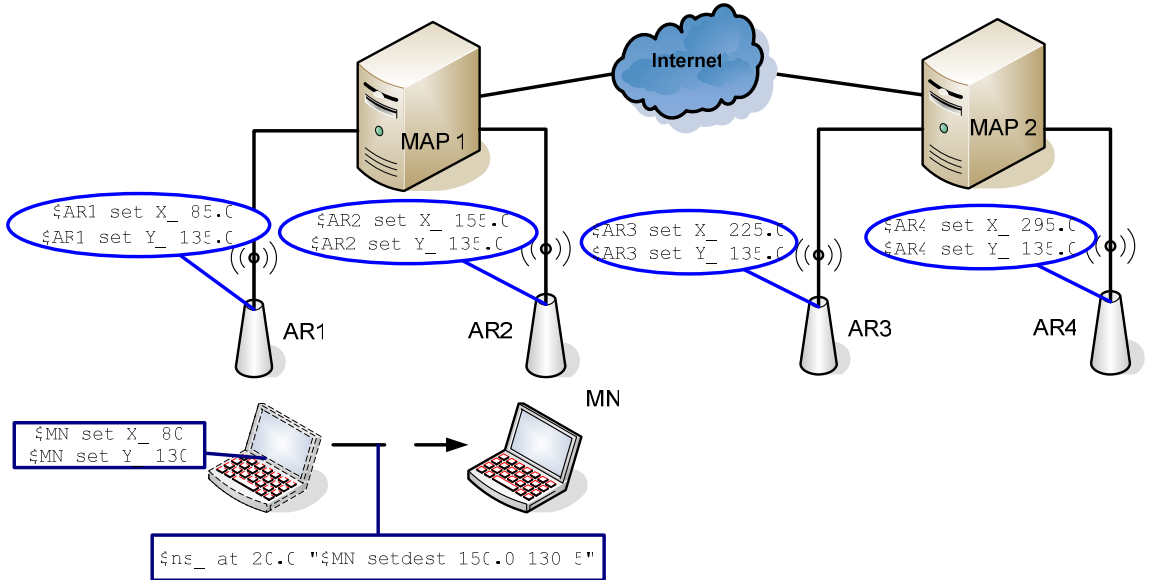
Figura 42: Paquetes recibidos de Simulación 2



### 6.2.3 Configuración de Movimiento con *Optimización de Ruta*

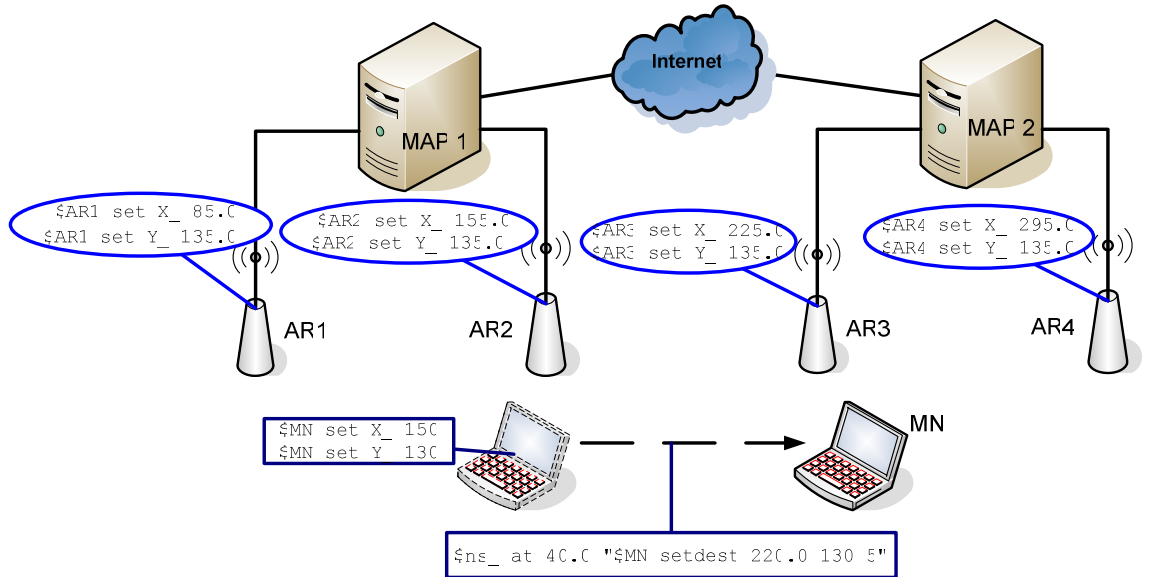
Al inicio de la simulación, el *MN* se sitúa en el *AR1*; a los 20 segundos de simulación, el *MN* se traslada linealmente al *AR2* con una velocidad constante de 5m/s (ver figura 43). Al llegar a la posición del *AR2*, espera hasta los 40 segundos de simulación para trasladarse al *AR3* (ver figura 44). En la última instancia, el *MN* se traslada al *AR4* a los 60 segundos y espera en esta hasta el fin de la simulación (ver figura 45).

Figura 43: Movimiento del Nodo Móvil en Simulación2 de AR1 a AR2



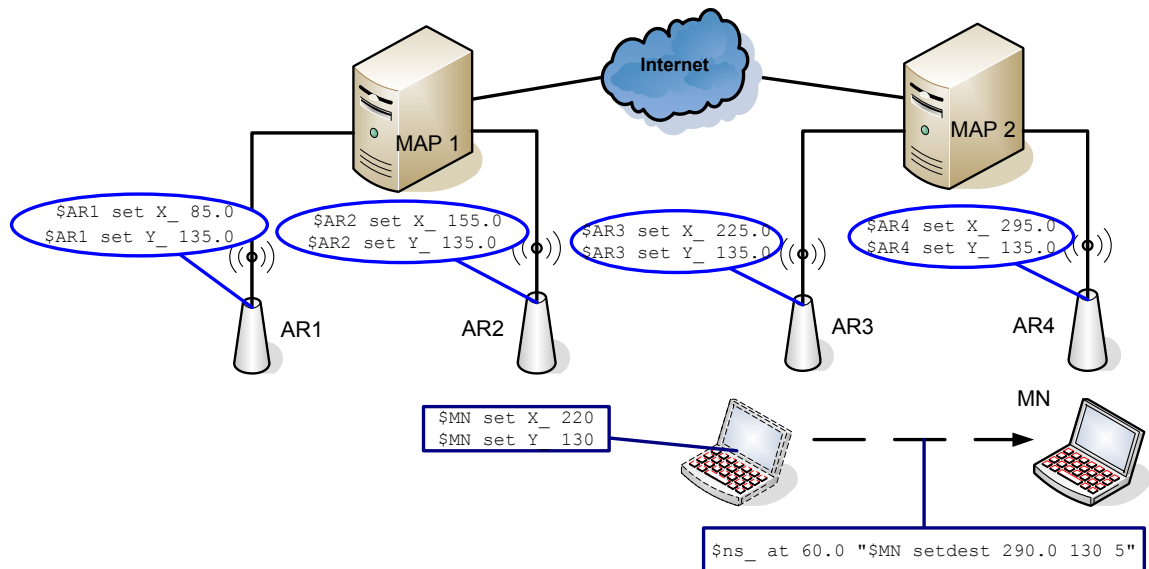
Fuente: Autores.

Figura 44: Movimiento del Nodo Móvil en Simulación2 de AR2 a AR3



Fuente: Autores.

Figura 45: Movimiento del Nodo Móvil en Simulación2 de AR3 a AR4



Fuente: Autores.

### 6.2.3.1 Comportamiento de la Simulación

El MN realiza el registro normalmente, se registra con el MAP 2.0.0 y luego con el HA 1.0.0, dando como resultado un enrutamiento triángulo.

Cuando CN comienza el envío de paquetes de datos al MN (después de un segundo de iniciada la simulación), se efectúa un enrutamiento triángulo mientras MN hace el *Binding Update* con CN. Realizado el registro, el CN direcciona todos los paquetes hacia el MAP.

### 6.2.3.2 Resultados Obtenidos

Como la optimización de ruta no interviene en el *handover* del MN, la grafica de pérdida de paquetes y paquetes recibidos es la misma con optimización de ruta activada o no.

## 6.3 SIMULACIÓN 3

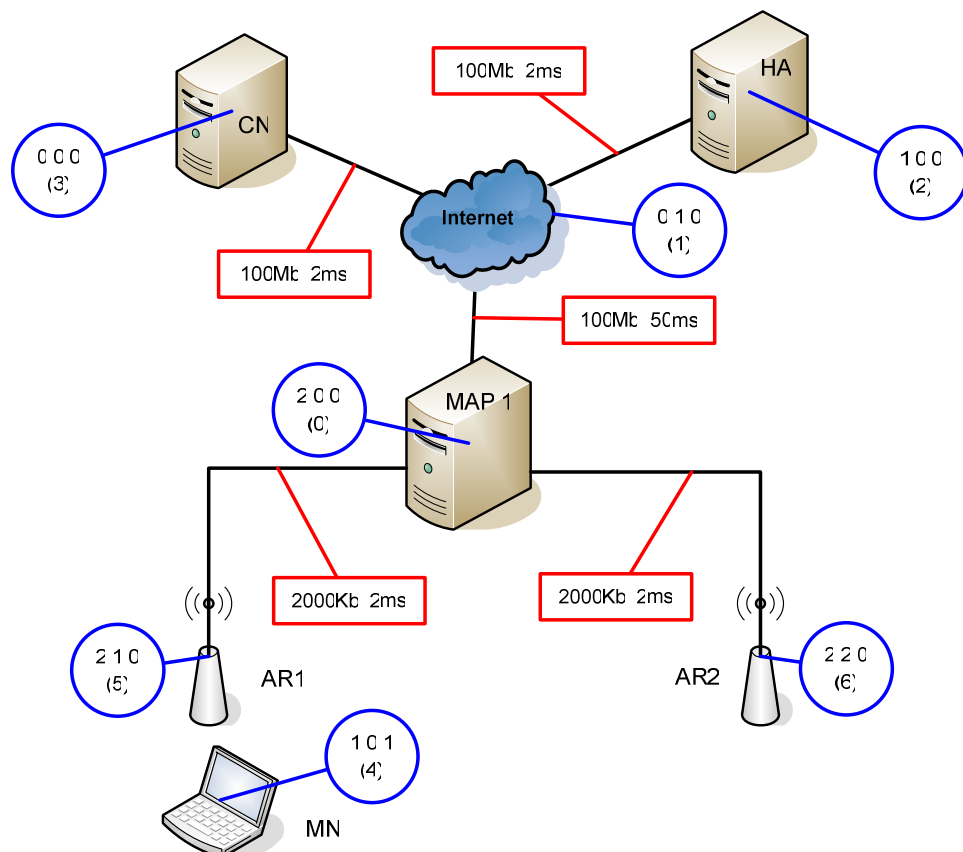
La Simulación 3 pretende demostrar el funcionamiento de la Herramienta HMIPv6\_SM con un tipo de tráfico distinto al CBR, el cual fue usado en las simulaciones anteriores.

Esta simulación utiliza la misma configuración de Simulación 1, tanto en la red fija como en la red inalámbrica, la diferencia radica en el generador de tráfico, se usa tráfico *UDP* con distribución exponencial. La configuración de este Agente se presentara a continuación.

```
set udp [new Agent/UDP]
$ns_ attach-agent $CN $udp

set expo [new Application/Traffic/Exponential]
$expo set packetSize_ 210
$expo set burst_time_ 500ms
$expo set idle_time_ 500ms
$expo set rate_ 100k
$expo attach-agent $udp
```

Figura 46: Topología de Simulación 3



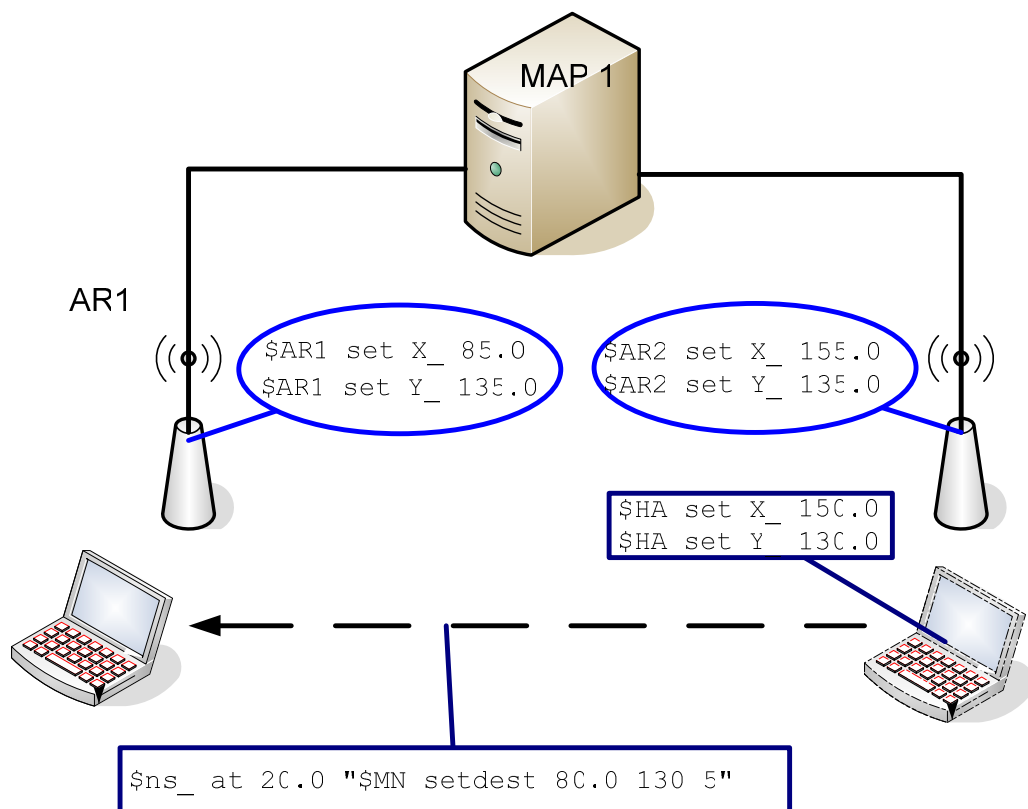
Fuente: Autores.



### 6.3.1 Configuración de Movimiento

El *MN* (*Mobile Node*) se ubica en el *AR2* al comienzo de la simulación; transcurridos 20 segundos de inicializada la simulación, se mueve linealmente del *AR1* al *AR2* con una velocidad constante de 5m/s (ver figura 47).

Figura 47: Movimiento del nodo móvil en Simulación3 de *AR2* a *AR1*



Fuente: Autores.

#### 6.3.1.1 Resultados Obtenidos

La grafica de paquetes recibidos oscila drásticamente (ver figura 49) debido a la utilización de una fuente de trafico UDP con distribución exponencial, el cual realiza un envío de paquetes sin una tasa de transferencia constante.

La figura de la perdida de paquetes (ver figura 50) es igual a la de Simulación 1 presentando pérdida de paquetes en el handover de *AR2* al *AR1*.

Figura 48: Pérdida de paquetes y paquetes recibidos de Simulación 3



Figura 49: Paquetes recibidos de Simulación 3

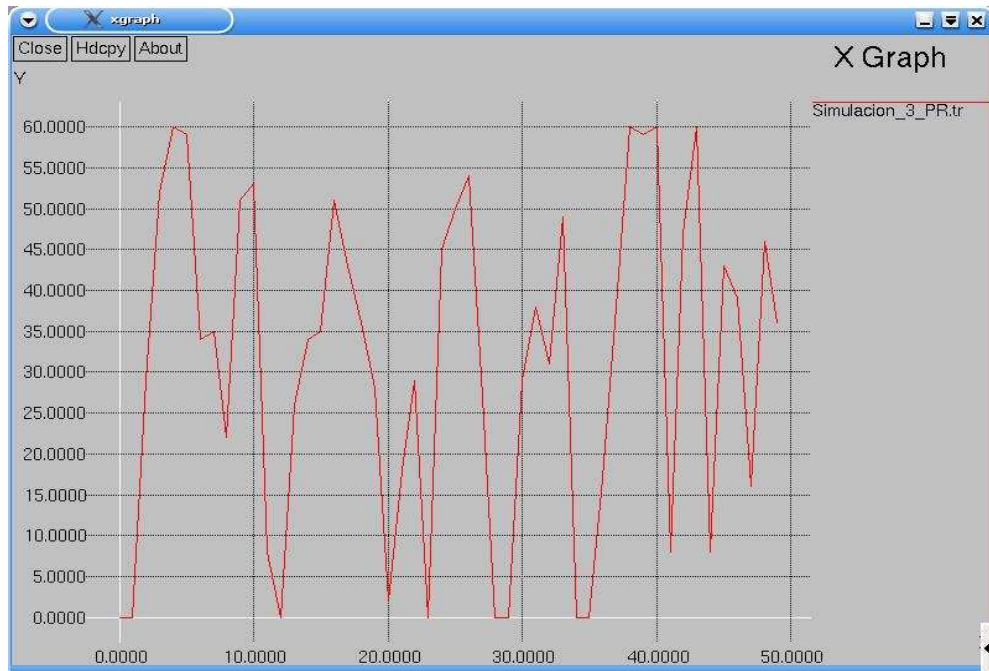
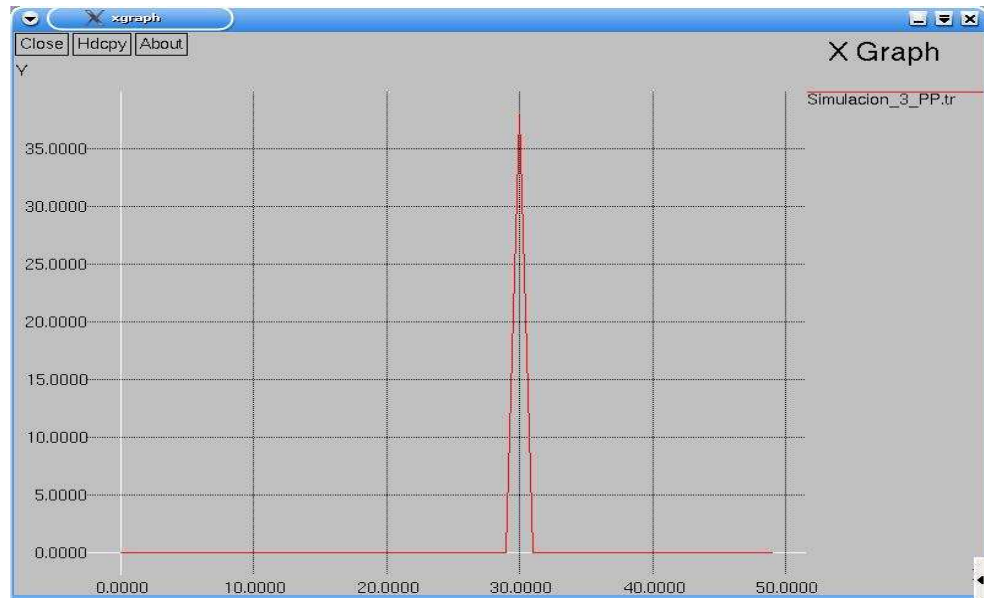


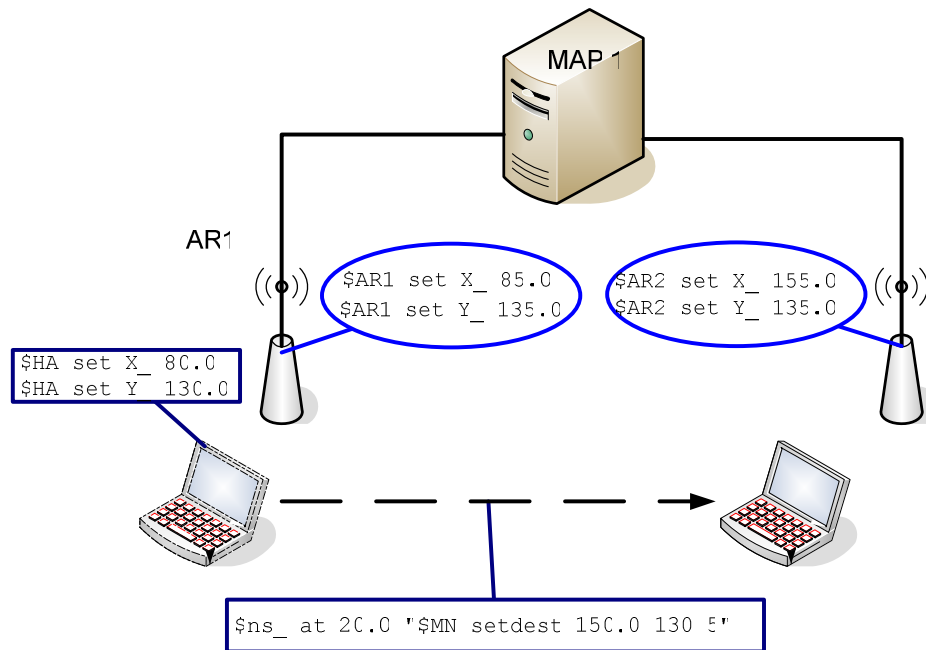
Figura 50: Paquetes Perdidos de Simulación 3



### 6.3.2 Configuración de Movimiento Optimización de Ruta

El *MN (Mobile Node)* se ubica en el *AR1* al comienzo de la simulación; transcurridos 20 segundos de inicializada la simulación, se mueve linealmente del *AR1* al *AR2* con una velocidad constante de 5m/s. La posición y el movimiento del *MN* entre los *AR* se pueden visualizar en la figura 51.

Figura 51: Movimiento del nodo móvil en Simulación3 de AR1 a AR2



Fuente: Autores.

### 6.3.2.1 Resultados Obtenidos

Como la optimización de ruta no interviene en el *handover* del MN, la grafica de pérdida de paquetes y paquetes recibidos es la misma con optimización de ruta activada o no.

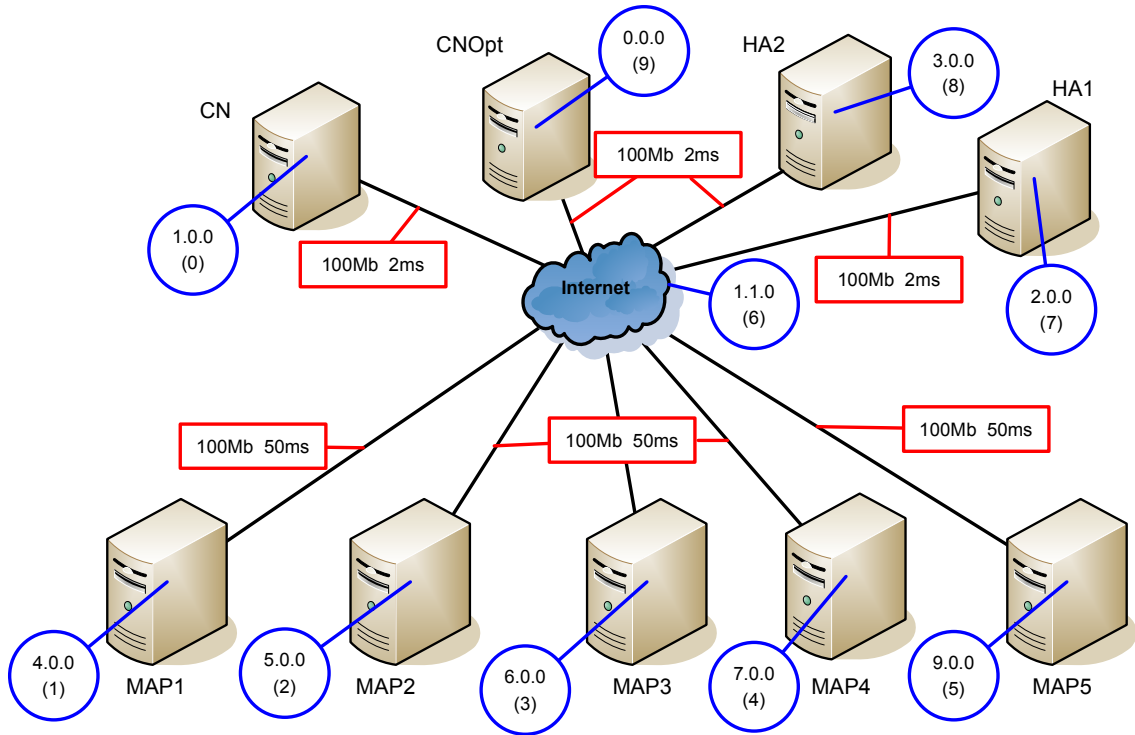
## 6.4 Simulación 4

Esta simulación pretende comprobar el funcionamiento de la herramienta *HMIPv6\_SM* en topologías grandes con varios nodos móviles, algunos con optimización de ruta activada y otros sin esta.

### 6.4.1 Configuración de Red

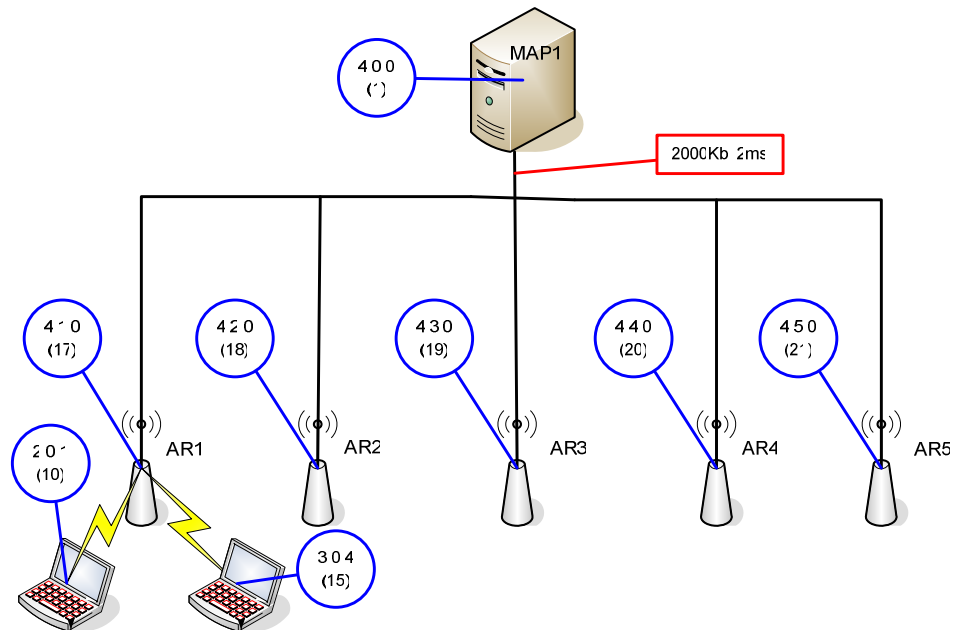
La topología de la simulación consta de dos CN, un *Nodo Internet*, 5 MAPs con 5 ARs por cada MAP, 7 MN y 2 HA. Ver figuras 52 al 57

Figura 52: Topología Simulación 4.



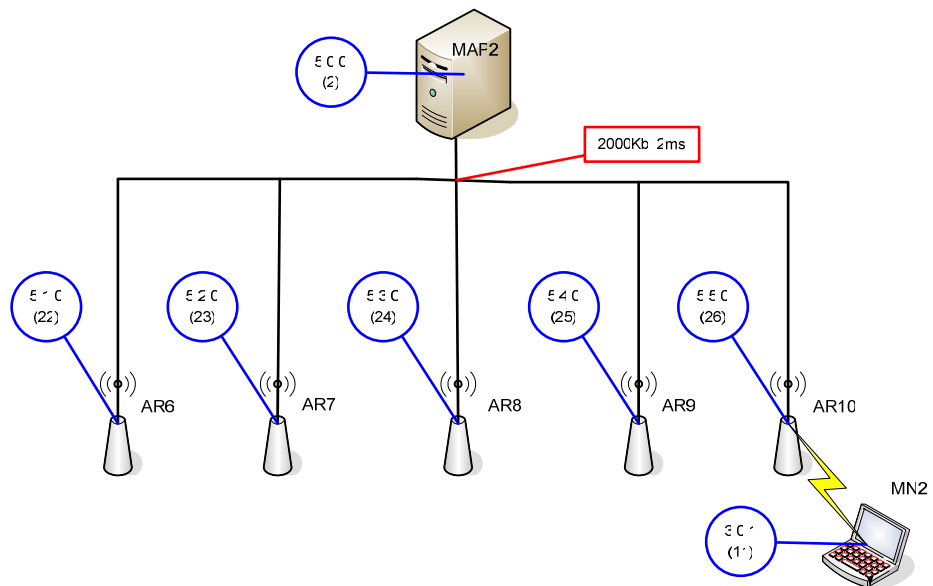
Fuente: Autores.

Figura 53: Topología Simulación 4 (MAP1).



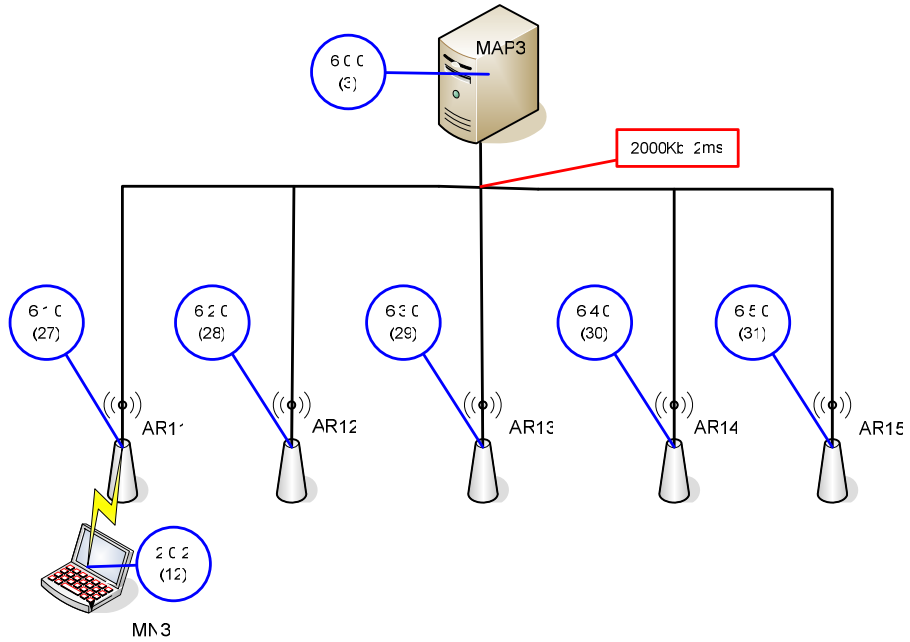
Fuente: Autores

Figura 54: Topología Simulación 4 (MAP2).



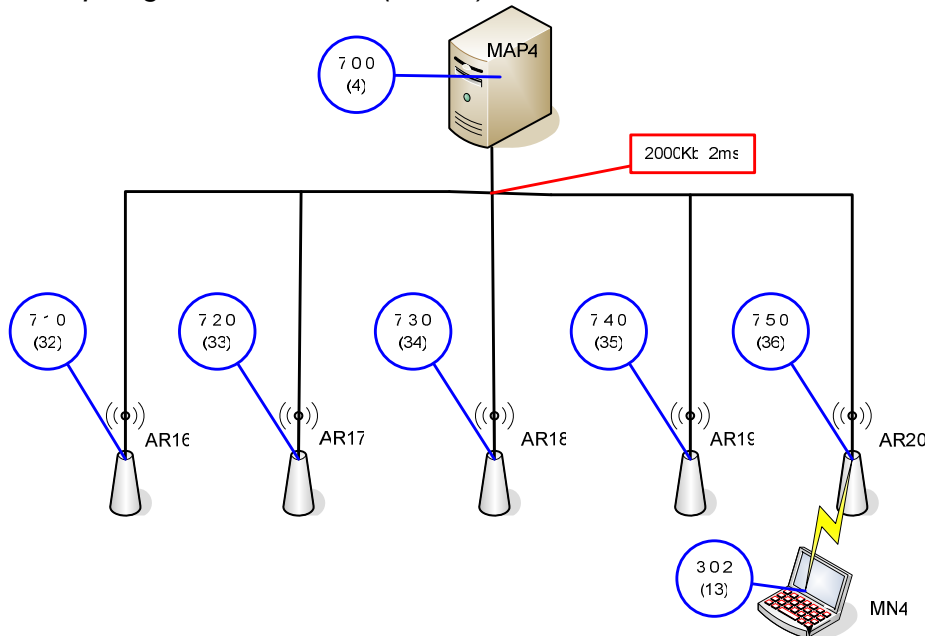
Fuente: Autores

Figura 55: Topología Simulación 4 (MAP3).



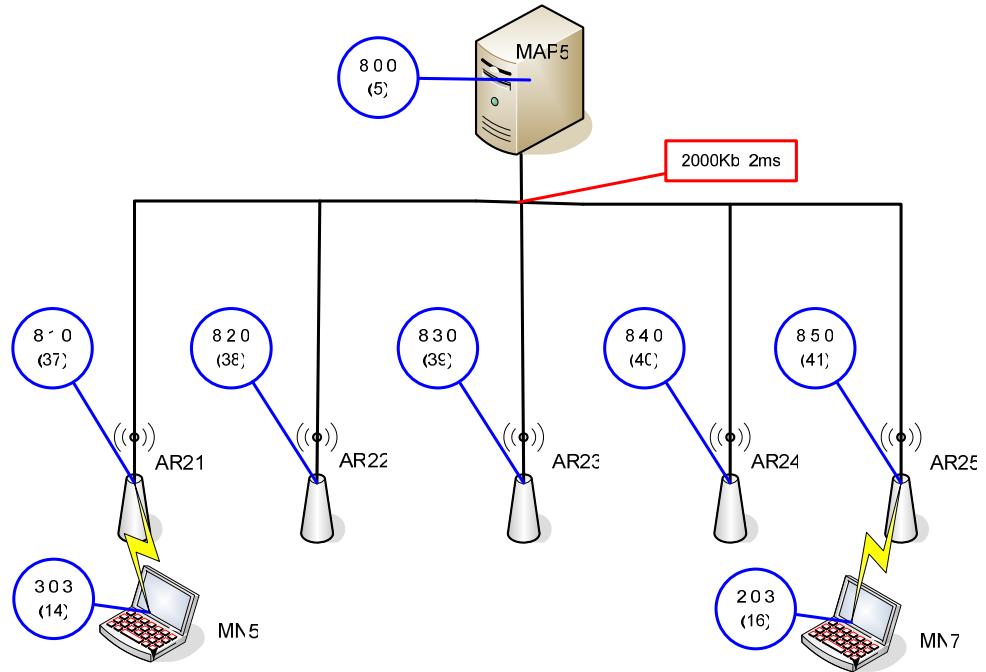
Fuente: Autores.

Figura 56: Topología Simulación 4 (MAP4).



Fuente: Autores.

Figura 57: Topología Simulación 4 (MAP5)



Fuente: Autores.

La dirección y el número correspondiente en la simulación con el NAM [3], se presentan en la tabla 5.

Tabla 5: Direcciones y número de nodo de los componentes de Simulación2

	<b>CNOpt</b>	<b>CN</b>	<b>NI</b>	<b>HA1</b>	<b>HA2</b>	<b>MN1</b>	<b>MN2</b>	<b>MN3</b>	<b>MN4</b>	<b>MN5</b>
<b>Dirección</b>	0.0.0	1.0.0	1.1.0	2.0.0	3.0.0	2.0.1	3.0.1	2.0.2	3.0.2	3.0.3
<b>Nodo</b>	9	0	6	7	8	10	11	12	13	14
	<b>MN6</b>	<b>MN7</b>	<b>MAP1</b>	<b>AR1</b>	<b>AR2</b>	<b>AR3</b>	<b>AR4</b>	<b>AR5</b>	<b>MAP2</b>	<b>AR6</b>
<b>Dirección</b>	3.0.4	2.0.3	4.0.0	4.1.0	4.2.0	4.3.0	4.4.0	4.5.0	5.0.0	5.1.0
<b>Nodo</b>	15	16	1	17	18	19	20	21	2	22
	<b>AR7</b>	<b>AR8</b>	<b>AR9</b>	<b>AR10</b>	<b>MAP3</b>	<b>AR11</b>	<b>AR12</b>	<b>AR13</b>	<b>AR14</b>	<b>AR15</b>
<b>Dirección</b>	5.2.0	5.3.0	5.4.0	5.5.0	6.0.0	6.1.0	6.2.0	6.3.0	6.4.0	6.5.0
<b>Nodo</b>	23	24	25	26	3	27	28	29	30	31
	<b>MAP4</b>	<b>AR16</b>	<b>AR17</b>	<b>AR18</b>	<b>AR19</b>	<b>AR20</b>	<b>MAP5</b>	<b>AR21</b>	<b>AR22</b>	<b>AR23</b>



<b>Dirección</b>	7.0.0	7.1.0	7.2.0	7.3.0	7.4.0	7.5.0	8.0.0	8.1.0	8.2.0	8.3.0
<b>Nodo</b>	4	32	33	34	35	5	36	37	38	39
					<b>AR24</b>	<b>AR25</b>				
					<b>Dirección</b>	8.4.0	8.5.0			
					<b>Nodo</b>	40	41			

#### 6.4.1.1 Configuración de la Red fija

La configuración de la red fija está dispuesta así:

El enlace del *Nodo Internet (NI)* con el *CN*, el *HA1*, *HA2* y *CNOpt* es un enlace tipo *duplex* con un ancho de banda de 100Mb, con tipo de cola *RED (Random Early Detection)* y con un tiempo de retardo máximo del paquete de 2ms.

```
$ns_ duplex-link $CNOpt $N1 100Mb 2ms RED
$ns_ duplex-link $CN $N1 100Mb 2ms RED
$ns_ duplex-link $HA1 $N1 100Mb 2ms RED
$ns_ duplex-link $HA2 $N1 100Mb 2ms RED
```

El enlace del *Nodo Internet* los *MAPs* es un enlace tipo *duplex*, con ancho de banda de 100Mb, tipo de cola *RED* y tiempo de retarde máximo del paquete de 50ms

```
$ns_ duplex-link $MAP1 $N1 100Mb 50ms RED
$ns_ duplex-link $MAP2 $N1 100Mb 50ms RED
$ns_ duplex-link $MAP3 $N1 100Mb 50ms RED
$ns_ duplex-link $MAP4 $N1 100Mb 50ms RED
$ns_ duplex-link $MAP5 $N1 100Mb 50ms RED
```

El enlace de los *MAPs* a sus respectivos *ARs*, es de tipo *duplex*, con ancho de banda de 2000Kb, tiempo de retardo máximo del paquete de 2ms y tipo de cola *DropTail (FIFO)*.

*ARs* pertenecientes al dominio *MAP1*: *AR1*, *AR2*, *AR3*, *AR4* y *AR5*

```
$ns_ duplex-link $AR1 $MAP1 2000Kb 2ms DropTail
$ns_ duplex-link $AR2 $MAP1 2000Kb 2ms DropTail
$ns_ duplex-link $AR3 $MAP1 2000Kb 2ms DropTail
$ns_ duplex-link $AR4 $MAP1 2000Kb 2ms DropTail
$ns_ duplex-link $AR5 $MAP1 2000Kb 2ms DropTail
```

**ARs pertenecientes al dominio MAP2: AR6, AR7, AR8, AR9 y AR10**

```
$ns_ duplex-link $AR6 $MAP2 2000Kb 2ms DropTail
$ns_ duplex-link $AR7 $MAP2 2000Kb 2ms DropTail
$ns_ duplex-link $AR8 $MAP2 2000Kb 2ms DropTail
$ns_ duplex-link $AR9 $MAP2 2000Kb 2ms DropTail
$ns_ duplex-link $AR10 $MAP2 2000Kb 2ms DropTail
```

**ARs pertenecientes al dominio MAP3: AR11, AR12, AR13, AR14 y AR15**

```
$ns_ duplex-link $AR11 $MAP3 2000Kb 2ms DropTail
$ns_ duplex-link $AR12 $MAP3 2000Kb 2ms DropTail
$ns_ duplex-link $AR13 $MAP3 2000Kb 2ms DropTail
$ns_ duplex-link $AR14 $MAP3 2000Kb 2ms DropTail
$ns_ duplex-link $AR15 $MAP3 2000Kb 2ms DropTail
```

**ARs pertenecientes al dominio MAP4: AR16, AR17, AR18, AR19 y AR20**

```
$ns_ duplex-link $AR16 $MAP4 2000Kb 2ms DropTail
$ns_ duplex-link $AR17 $MAP4 2000Kb 2ms DropTail
$ns_ duplex-link $AR18 $MAP4 2000Kb 2ms DropTail
$ns_ duplex-link $AR19 $MAP4 2000Kb 2ms DropTail
$ns_ duplex-link $AR20 $MAP4 2000Kb 2ms DropTail
```

**ARs pertenecientes al dominio MAP5: AR21, AR22, AR23, AR24 y AR25**

```
$ns_ duplex-link $AR21 $MAP5 2000Kb 2ms DropTail
$ns_ duplex-link $AR22 $MAP5 2000Kb 2ms DropTail
$ns_ duplex-link $AR23 $MAP5 2000Kb 2ms DropTail
$ns_ duplex-link $AR24 $MAP5 2000Kb 2ms DropTail
$ns_ duplex-link $AR25 $MAP5 2000Kb 2ms DropTail
```

Los nodos móviles *MN1*, *MN4* y *MN5* tienen activada la optimización de ruta, es por eso que a estos nodos móviles se les asignara a cada uno un agente *UDP* unido a *CNOpt*. Estos se conectaran a un generador de trafico *CBR* (*Constant Bit Rate*) con un tamaño de paquete de 1000 bytes y con envío de paquete cada 0.05 segundos.

```
set udp1 [new Agent/UDP]
$ns_ attach-agent $CNOpt $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 1000
$cbr1 set interval_ 0.05
```

```
$cbr1 attach-agent $udp1
```

```
set udp4 [new Agent/UDP]  
$ns_ attach-agent $CNOpt $udp4  
set cbr4 [new Application/Traffic/CBR]  
$cbr4 set packetSize_ 1000  
$cbr4 set interval_ 0.05  
$cbr4 attach-agent $udp4  
set udp5 [new Agent/UDP]  
$ns_ attach-agent $CNOpt $udp5  
set cbr5 [new Application/Traffic/CBR]  
$cbr5 set packetSize_ 1000  
$cbr5 set interval_ 0.05  
$cbr5 attach-agent $udp5
```

Los nodos móviles *MN2*, *MN3*, *MN6* y *MN7* tiene la misma configuración de fuente y generador de tráfico que los nodos móviles anteriores, solo que sus fuentes de tráfico *UDP* van unidas al nodo *CN* debido a que estos nodos móviles no tiene activada la optimización de ruta.

```
set udp2 [new Agent/UDP]  
$ns_ attach-agent $CN $udp2  
set cbr2 [new Application/Traffic/CBR]  
$cbr2 set packetSize_ 1000  
$cbr2 set interval_ 0.05  
$cbr2 attach-agent $udp2
```

```
set udp3 [new Agent/UDP]  
$ns_ attach-agent $CN $udp3  
set cbr3 [new Application/Traffic/CBR]  
$cbr3 set packetSize_ 1000  
$cbr3 set interval_ 0.05  
$cbr3 attach-agent $udp3
```

```
set udp6 [new Agent/UDP]  
$ns_ attach-agent $CN $udp6  
set cbr6 [new Application/Traffic/CBR]  
$cbr6 set packetSize_ 1000  
$cbr6 set interval_ 0.05  
$cbr6 attach-agent $udp6
```

```
set udp7 [new Agent/UDP]  
$ns_ attach-agent $CN $udp7
```

```
set cbr7 [new Application/Traffic/CBR]
$cbr7 set packetSize_ 1000
$cbr7 set interval_ 0.05
$cbr7 attach-agent $udp7
```

Para poder monitorear el flujo de datos que le llega a los nodos móviles, se enlaza un agente *LossMonitor* a cada *MN*. Luego este agente se conecta con su correspondiente agente *UDP* creado anteriormente.

```
set null1 [new Agent/LossMonitor]
$ns_ attach-agent $MN1 $null1
$ns_ connect $udp1 $null1
$ns_ at 1.0 "$cbr1 start"
```

```
set null2 [new Agent/LossMonitor]
$ns_ attach-agent $MN2 $null2
$ns_ connect $udp2 $null2
$ns_ at 1.0 "$cbr2 start"
```

```
set null3 [new Agent/LossMonitor]
$ns_ attach-agent $MN3 $null3
$ns_ connect $udp3 $null3
$ns_ at 1.0 "$cbr3 start"
```

```
set null4 [new Agent/LossMonitor]
$ns_ attach-agent $MN4 $null4
$ns_ connect $udp4 $null4
$ns_ at 1.0 "$cbr4 start"
```

```
set null5 [new Agent/LossMonitor]
$ns_ attach-agent $MN5 $null5
$ns_ connect $udp5 $null5
$ns_ at 1.0 "$cbr5 start"
```

```
set null6 [new Agent/LossMonitor]
$ns_ attach-agent $MN6 $null6
$ns_ connect $udp6 $null6
$ns_ at 1.0 "$cbr6 start"
```

```
set null7 [new Agent/LossMonitor]
$ns_ attach-agent $MN7 $null7
$ns_ connect $udp7 $null7
$ns_ at 1.0 "$cbr7 start"
```

### 6.4.1.2 Configuración de la Red Inalámbrica.

La configuración inalámbrica de la topología es la misma usada en las anteriores simulaciones:

```
set chan_ [new Channel/WirelessChannel]
$ns_ node-config -mobileIP ON \
    -adhocRouting AODV \
    -llType LL \
    -macType Mac/802_11 \
    -ifqType Queue/DropTail/PriQueue \
    -ifqLen 50 \
    -antType Antenna/OmniAntenna \
    -propType Propagation/TwoRayGround \
    -phyType Phy/WirelessPhy \
    -channel $chan_ \
    -topoInstance $topo \
    -wiredRouting ON \
    -agentTrace ON \
    -routerTrace ON \
    -macTrace OFF
```

Se usa un protocolo de enrutamiento *adhoc AODV*, una antena *Omni-direccional*, modelo de propagación *TwoRayGround*, el protocolo *MAC 802.11* y un tipo de cola *DropTail / PriQueue*. También se activa las trazas de *router*, agente y *wiredRouting*.

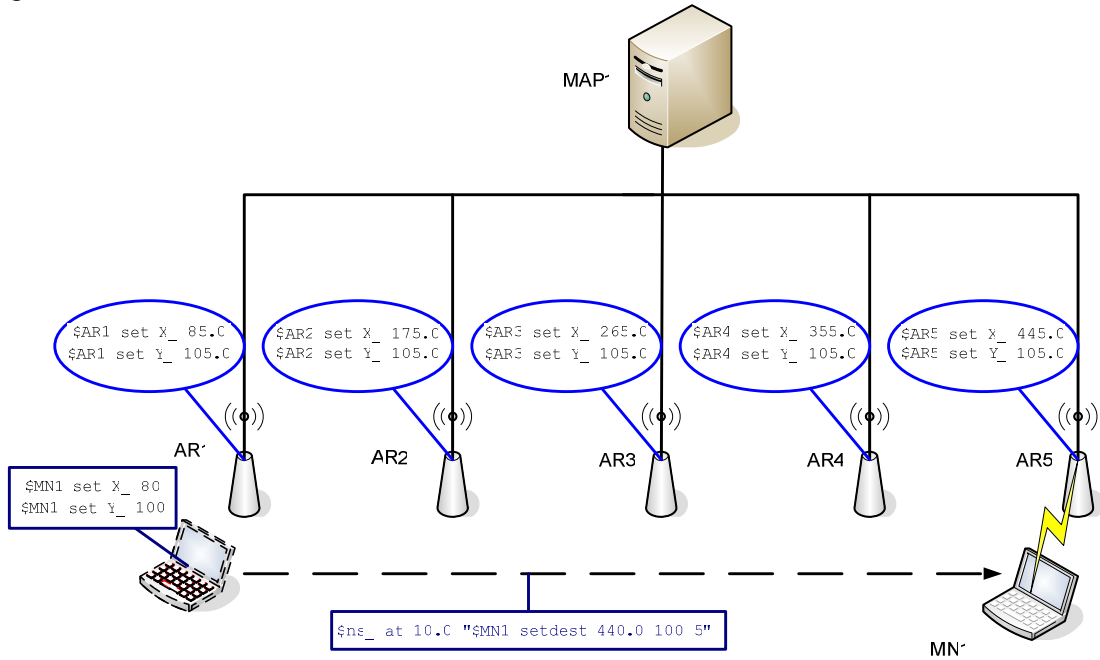
### 6.4.2 Configuración de Movimiento

A continuación se mostrara como es el movimiento de los 7 nodos móviles durante la simulación.

El nodo móvil 1 (*MN1*) inicialmente esta situado en el *AR1*, el cual pertenece al dominio del *MAP1*. A los 10 segundos de simulación, *MN1* comienza a moverse hacia *AR5* que también pertenece al dominio del *MAP1* (ver figura 58). *MN1* recorre todos los ARs pertenecientes al dominio del *MAP1* (*AR1*, *AR2*, *AR3*, *AR4* y *AR5*).

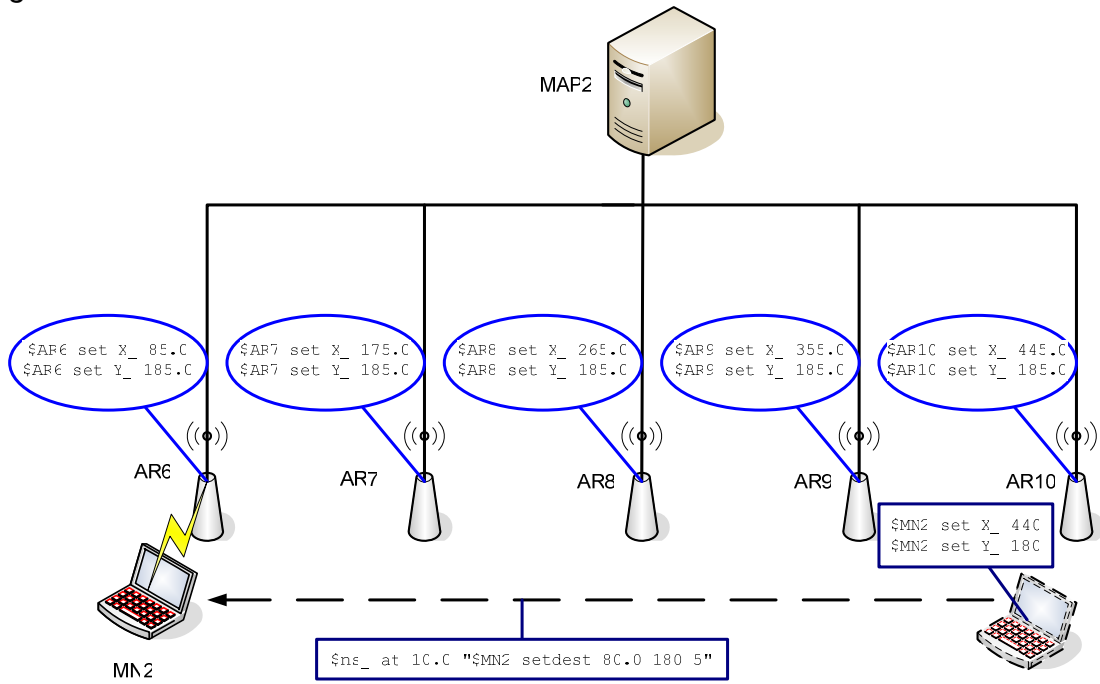
El nodo móvil 2 (*MN2*) esta ubicado en el dominio del *MAP2*. Este nodo inicialmente se sitúa en *AR10*, luego a los 10 segundos de simulación comienza a moverse linealmente hacia *AR6* (ver figura 59), que también pertenece al dominio *MAP2*.

Figura 58: Movimiento de MN1 en Simulación 4



Fuente: Autores.

Figura 59: Movimiento de MN2 en Simulación 4

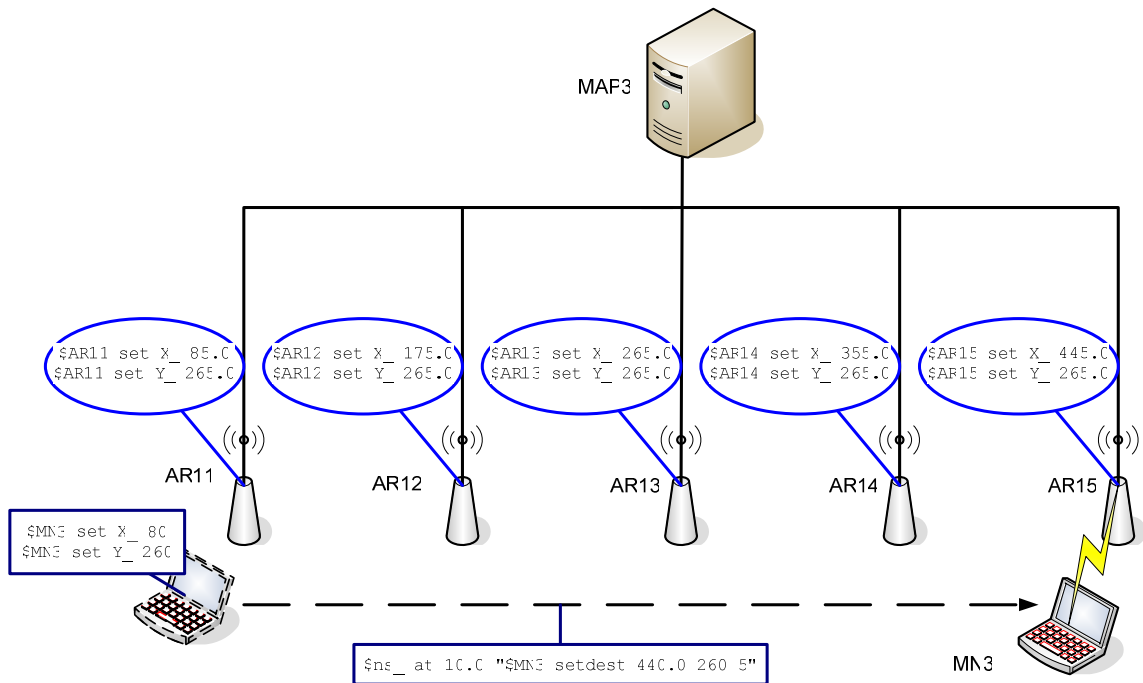


Fuente: Autores.

El nodo móvil 3 (MN3) inicialmente está ubicado en el AR11, el cual pertenece al dominio MAP3. MN3 comienza su desplazamiento a través del dominio MAP3 (AR11, AR12, AR13, AR14 y AR15) a los 10 segundos de haber comenzado la simulación (ver figura 60).

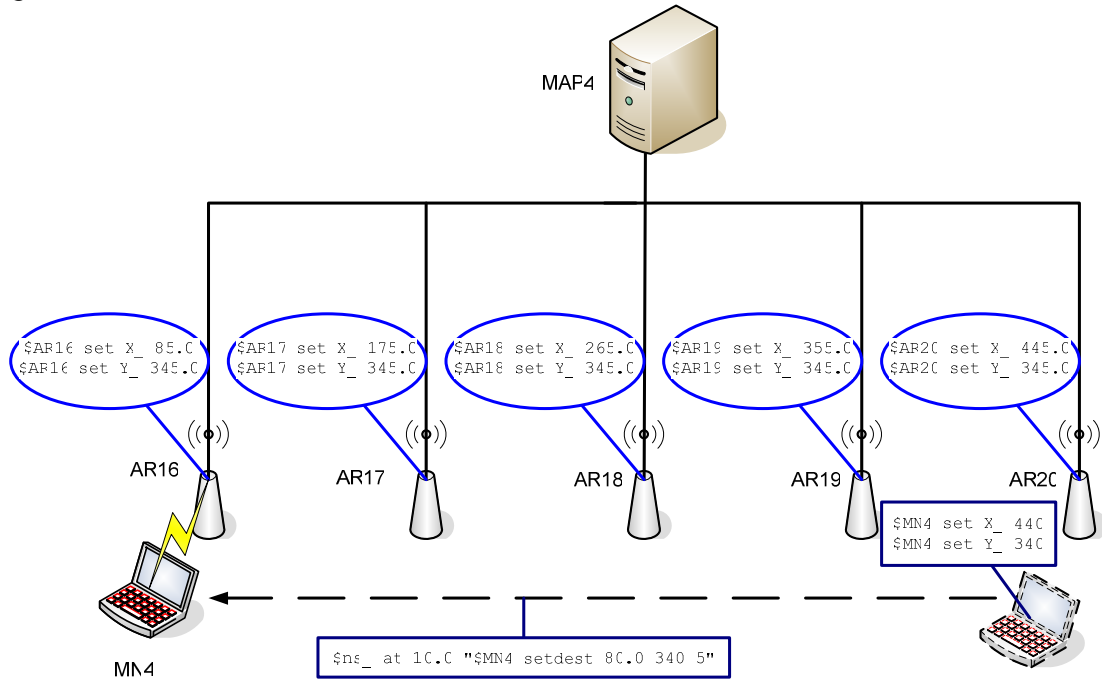
El nodo móvil 4 (MN4) y 5 (MN5) se desplazan linealmente a través de los ARs pertenecientes a los dominios MAP4 (ver figura 61) y MAP5 (ver figura 62) respectivamente.

Figura 60: Movimiento de MN3 en Simulación 4



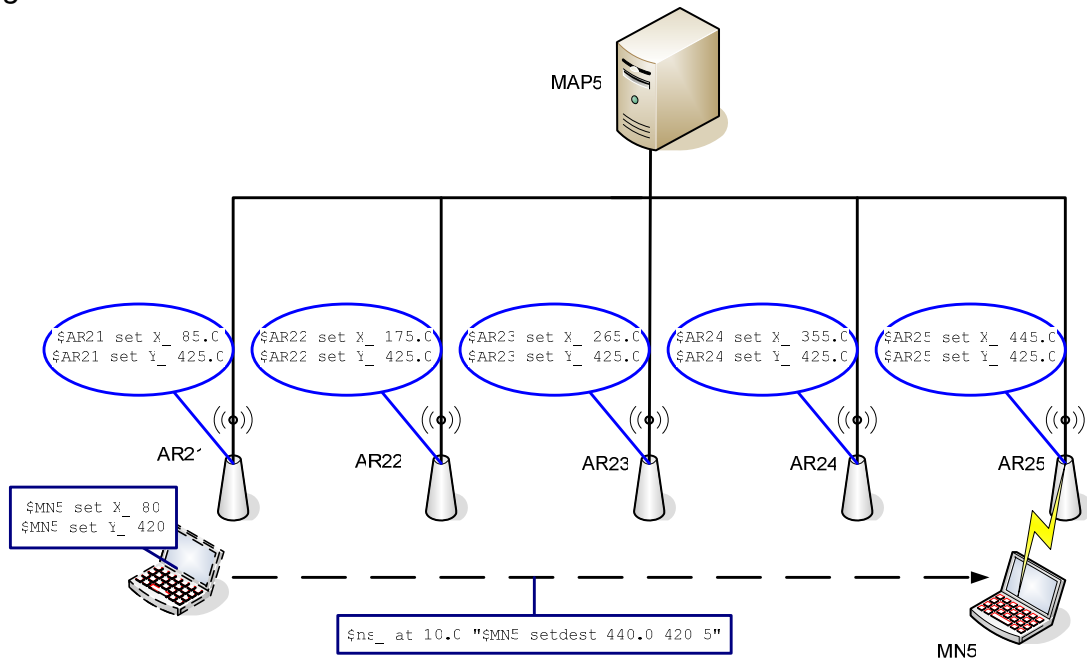
Fuente: Autores.

Figura 61: Movimiento de MN4 en Simulación 4.



Fuente: Autores.

Figura 62: Movimiento de MN5 en Simulación 4



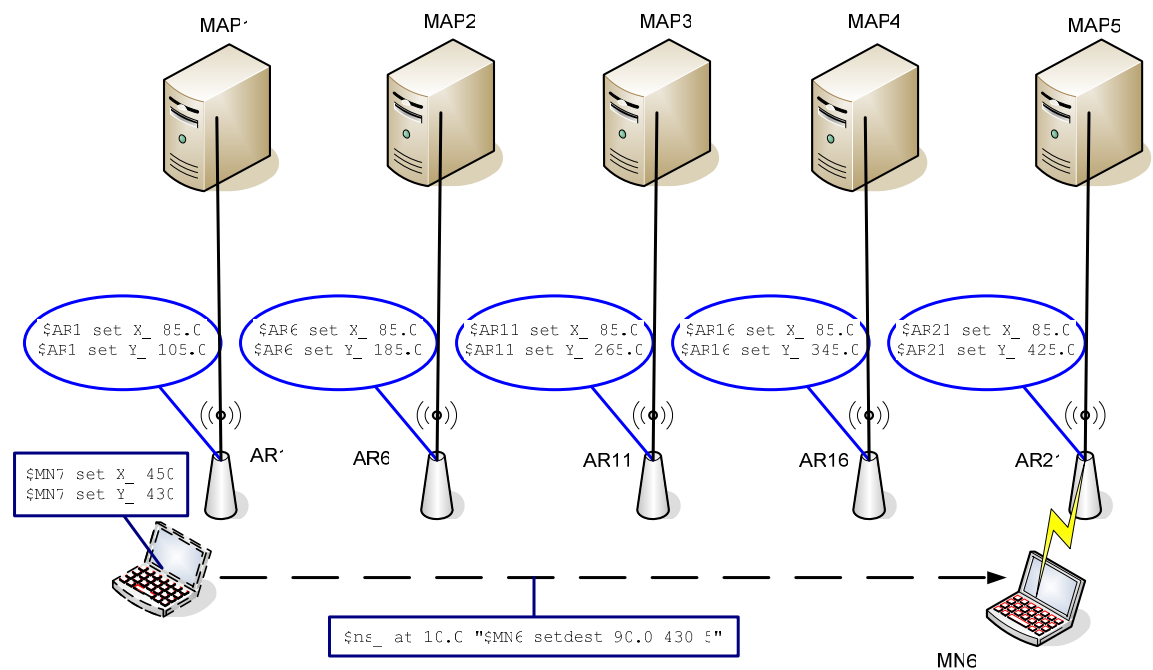
Fuente: Autores.



El nodo móvil 6 (MN6) inicialmente está ubicado en el AR1 que pertenece al dominio MAP1. A los 10 segundos de simulación, comienza a moverse linealmente hacia AR21, perteneciente al dominio MAP5, atravesando los routers de acceso AR6, AR11 y AR16. Como se puede ver en la figura 63, el nodo móvil MN6 pasa por 5 ARs pertenecientes a distintos dominios MAP.

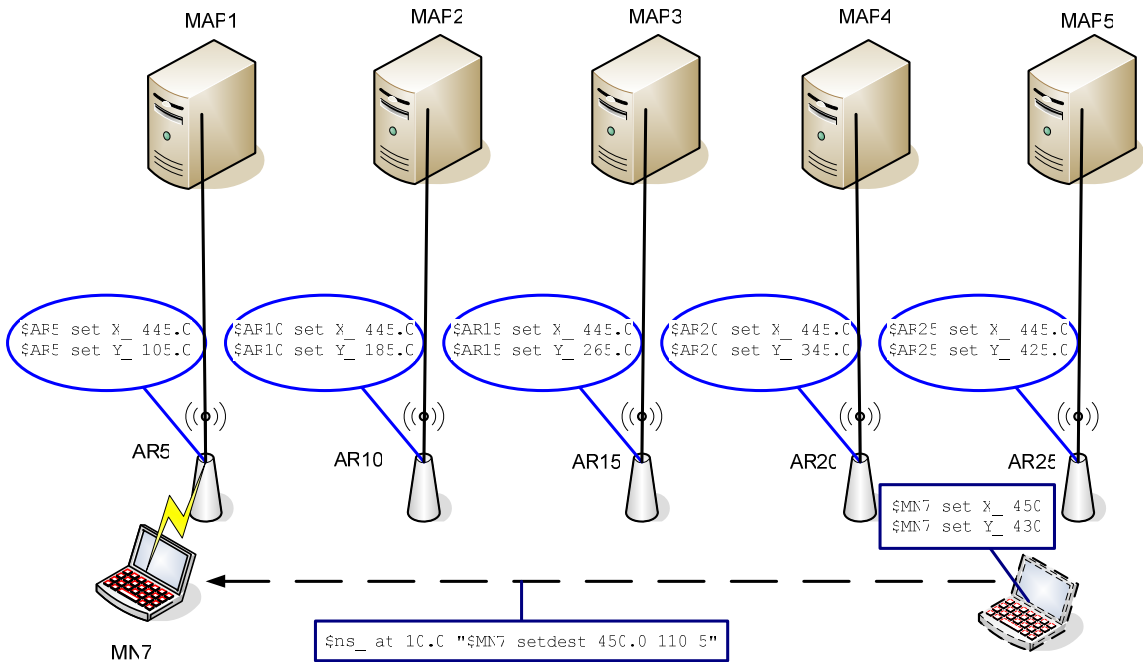
De igual forma, el nodo móvil 7 (MN7) cruza a través de ARs pertenecientes a diferentes dominios MAP, pero este hace su movimiento de AR25, que pertenece al dominio MAP5; al AR5 que pertenece al dominio MAP1 (ver figura 64).

Figura 63: Movimiento de MN6 en Simulación 4



Fuente: Autores.

Figura 64: Movimiento de MN7 en Simulación 4

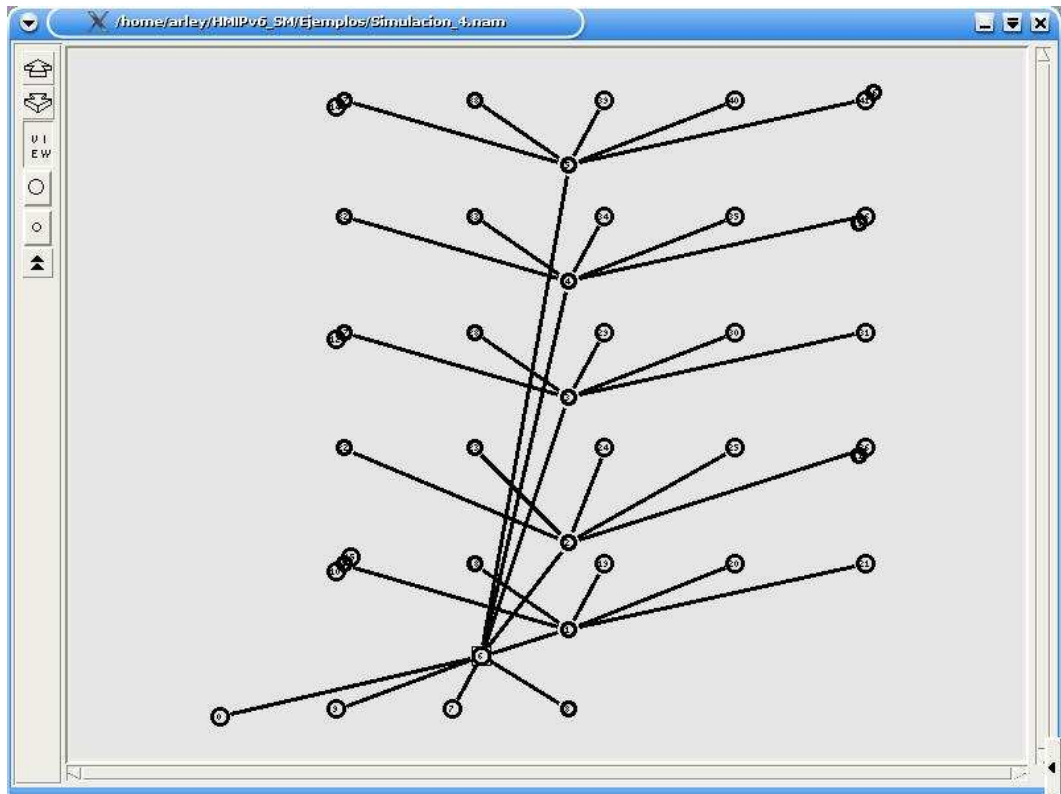


Fuente: Autores.

### 6.4.3 Resultados Obtenidos

La Topología se visualiza en el *NAM* como se muestra en la figura 65. Debido a que es una topología muy extensa, la visualización de la topología en el *NAM* resulta muy pequeña y no se logra apreciar muy bien cada nodo, pero se ve perfectamente la distribución completa de los ARs, MAPs, HAs y CNs en la grilla de simulación.

Figura 65: Visualización de Simulación 4 en el NAM.



El comportamiento de los nodos *MN1*, *MN2*, *MN3*, *MN4* y *MN5* es igual al evaluado en simulación 1 debido a que el movimiento de estos nodos se efectúa a través de un único dominio *MAP*. Por este motivo se prescindirá de analizar las graficas de estos nodos.

La grafica de paquetes perdidos y paquetes recibidos de *MN7* se muestra en la figura 67 y 68.

Como se puede apreciar en las graficas, el nodo *MN7* presenta variaciones en los paquetes recibidos y perdidos, cada vez que realiza el *handover* de un *AR* a otro.

Figura 66: Pérdida de paquetes y paquetes recibidos de MN7 en Simulación 4

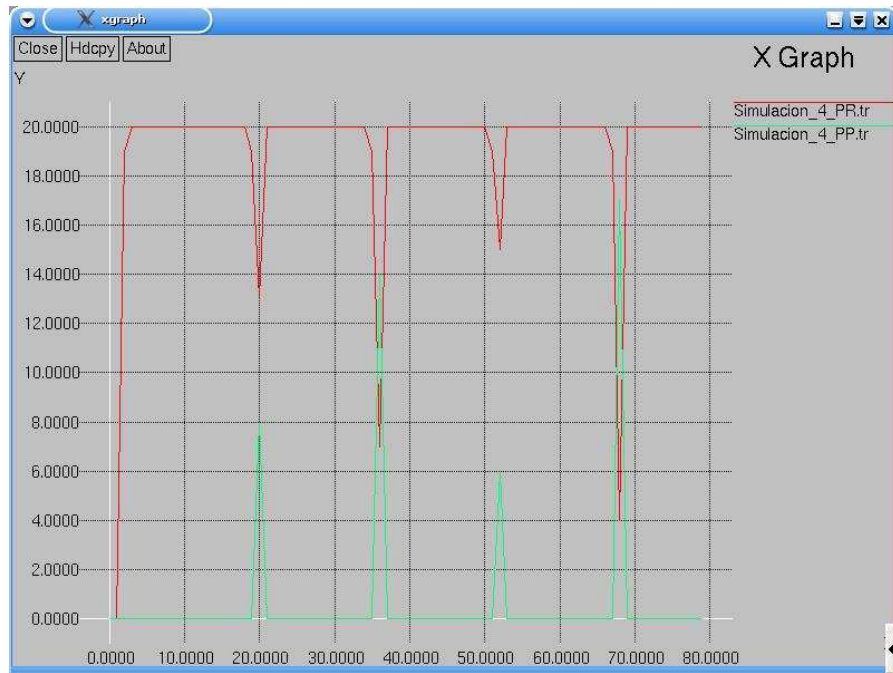


Figura 67: Paquetes perdidos de MN7 en Simulación 4

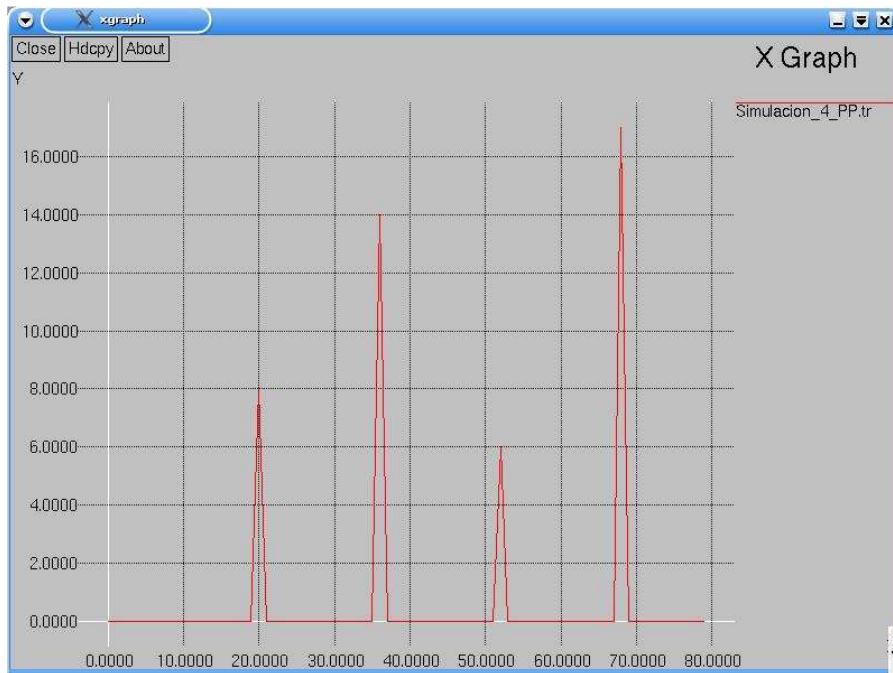
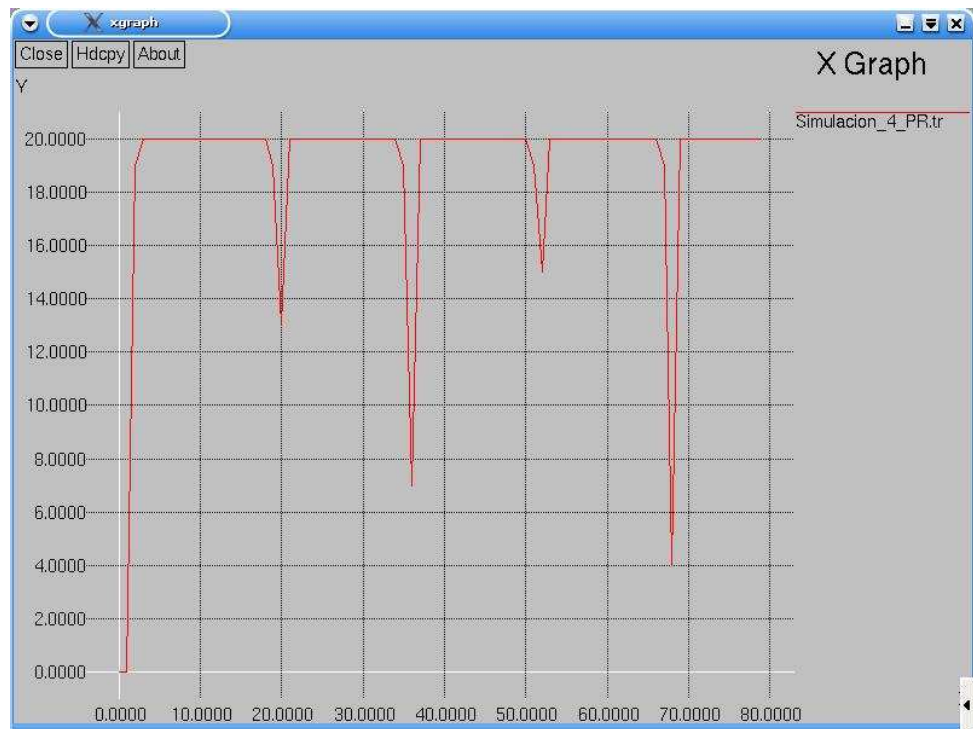


Figura 68: Paquetes recibidos por MN7 en Simulación 4



## 7. CONCLUSIONES

Se desarrollo un modelo de simulación denominado *HMIPv6\_SM* para simular el protocolo *HMIPv6* [1] con las especificaciones mínimas descritas en la *RFC 4140*. El cual permite simular diferentes topologías de red, también se tiene en cuenta la inclusión del proceso de la optimización de ruta y el encapsulamiento desde el MAP hacia el *MN*. El código del programa es invisible al usuario, las simulaciones se realizan directamente desde el *script* sin necesidad de que el usuario tenga que modificar el código fuente de *NS* [3] y mucho menos compilar cada vez que se modifique la topología.

El modelo *HMIPv6\_SM* permite la simulación de topologías en las cuales se implementen hasta 6 *MAPs*, cada *MAP* tiene la posibilidad de manejar 5 *ARs* (*Access Routers*); y cada *AR* permite la conexión de hasta 5 *MNs* (*Mobile Nodes*).

En *HMIPv6\_SM* se implementó el proceso de optimización de la ruta habilitada por defecto, permitiendo así el direccionamiento directo del nodo correspondiente al *MAP*, con la posibilidad de desactivarla desde el *script*.

El modelo *HMIPv6\_SM* es accesible a los usuarios brindándole la posibilidad de estudiar las características básicas de una topología pensada o propuesta para funcionar bajo el protocolo *HMIPv6* [1].

Durante el desarrollo de este proyecto se adquirieron conocimientos acerca de los protocolos *Mobile IPv6* [2] y *Hierarchical Mobile IPv6* [1], la operación de cada uno y la importancia que representan en la innovación y mejoramiento de las comunicaciones para gestionar la movilidad cuando se quiere acceder a una gran fuente de información y medio de comunicación como lo es Internet.

Debido a que el *Network Simulator* [3] se compone de dos lenguajes de programación, fue indispensable el estudio de los lenguajes de programación C++ y TCL; permitiéndonos descubrir la capacidad y rapidez del lenguaje C++, y de la versatilidad del lenguaje TCL.

Uno de los aportes sobresalientes de este proyecto es resaltar la importancia del desarrollo de software para la simulación de los diferentes protocolos existentes, que permitirán experimentar y a su vez comprender el uso y operación de estos protocolos. De igual manera comprobar la funcionalidad y las ventajas que ofrece cada protocolo, con el propósito de lograr un mejor desempeño de los mismos y por consiguiente la aparición de ideas para el desarrollo a futuro de nuevos protocolos.

## 8. RECOMENDACIONES

Para un adecuado desempeño de la herramienta se debe tener en cuenta no exceder el número de elementos de red permitidos tanto para los *MAPs*, los Router de Acceso y la cantidad de nodos móviles a usar en las simulaciones, debido a que los elementos adicionales no son tomados en cuenta. Si se desea una simulación con una topología mayor a la permitida por este modelo de simulación, se debe modificar el código de programación.

Cuando en la simulación se maneja varios nodos móviles que se están comunicando con un mismo *CN*, deben tener todos configurada la opción de optimización de la ruta o por el contrario todos deben estar configurados para operar sin la optimización de la ruta. Esto se debe a que el *CN* no puede ejecutar los métodos *encap-router* y *mencap-router* simultáneamente, esto se puede observar de forma mas detallada en la sección 5.2.2.

En la herramienta *HMIPv6\_SM* se deja la opción de realizar simulaciones con el mecanismo de *fast handover*, pero es un mecanismo *fast handover* para *MIPv6* [2] y no para *HMIPv6* [1]. Además se mencionan las algunas de las diferencias existentes entre los protocolos de fast handover para *HMIPv6* [1] y *MIPv6* [2] con el fin de dejar una base para el inicio de un proyecto futuro; ya que la implementación del protocolo Fast Handover para *HMIPv6* [9] no es parte de los objetivos del proyecto.

## GLOSARIO

- ✓ **Access Router (AR):** Un *router* de acceso es la interfaz mediante la cual el nodo móvil (*Mobile Node, MN*) establece la comunicación, son los encargados de enviar los anuncios de *router* (*Router Advertisement, RA*) hacia los *MNs*.
- ✓ **Binding:** Un *binding* equivale a la asociación de la dirección local (*Home Address*) de un nodo móvil (*Mobile Node, MN*) con una *Care-of address (CoA)* para ese *MN*.
- ✓ **Binding Acknowledgement (BACK):** Esta opción es usada para confirmar la recepción de un *Binding Update (BU)*, si el reconocimiento fuese pedido.
- ✓ **Binding Cache:** Cada nodo *IPv6* [6] tiene un *Binding Caché* o listado de ubicaciones de las *CoAs* que se usa para almacenar las ubicaciones de otros nodos. Si un nodo recibe un mensaje *BU*, agregará la nueva ubicación *CoA* a su *Binding Caché*. Cuando quiere enviar un paquete, buscará la *CoA* en el *Binding Caché* y envía el paquete a la *CoA* del nodo, usando una cabecera de enrutamiento.
- ✓ **Binding Request (BR):** Esta opción es usada por cualquier nodo para solicitar a un nodo móvil (*Mobile Node, MN*) el envío de un *BU* con la actual *CoA*.
- ✓ **Binding Update (BU):** Esta opción es usada por un nodo móvil (*Mobile Node, MN*) para informar a su agente local (*Home Agent, HA*) o a cualquier otro nodo correspondiente (*Correspondent Node, CN*) a cerca de su actual *Care-of Address (CoA)*.
- ✓ **Binding Update List:** Cada *MN* tiene una tabla llamada *Binding Update List* que se usa para guardar la información sobre cada actualización enviada por este *MN* cuando el tiempo de vida (*lifetime*) no ha expirado aún. Contiene todas los *BU* enviados a su *HA* y a cualquier *CN* (móvil o estacionario), es decir nodos que se están comunicando o se han comunicado en un tiempo corto con el *MN*.
- ✓ **Care-of Address (CoA):** Es la dirección asociada con un nodo móvil (*Mobile Node, MN*) mientras visita un enlace foráneo; el prefijo de subred de esta dirección IP es el prefijo de subred foráneo. Entre las múltiples direcciones *Care-of address (CoA)* que un nodo móvil (*Mobile Node, MN*) puede tener en algún momento, la primera dirección registrada con el agente local (*Home Agent, HA*) del nodo móvil (*Mobile Node, MN*) para una dirección local (*Home Address*) dada es llamada su *Care-of address* primaria.



- ✓ **Correspondente Node (CN):** Es un nodo con el cual el nodo móvil (*Mobile Node, MN*) se esta comunicando. El nodo correspondiente (*Correspondent Node, CN*) puede ser cualquier nodo estacionario o móvil.
- ✓ **Fast Binding Acknowledgment (FBACK):** El *Fast Binding Acknowledgment* es un mensaje que es enviado por el *MAP* con el objetivo de confirmar la recepción del mensaje *Fast Binding Update (FBU)*. Este mensaje no debe ser enviado al nodo móvil (*Mobile Node, MN*) antes que el *MAP* reciba el mensaje *Handover Acknowledge (HACK)*, proveniente del *router* de acceso próximo.
- ✓ **Fast Binding Update (FBU):** Es un mensaje el cual es similar al mensaje de *Binding Update (BU)* descrito en el protocolo *Mobile IPv6* [2] sin embargo el proceso es un poco diferente.

El nodo móvil (*Mobile Node, MN*) envía un *FBU* después de recibir el mensaje *Proxy Router Advertisement (PrRtAdv)*. Si el *MN* se mueve antes de recibir el *PrRtAdv* este de igual manera debe enviar un *FBU* al *MAP* después de configurar la dirección del *router* de acceso próximo, de acuerdo con el proceso de descubrimiento de vecino (*Neighbor Discovery*).

- ✓ **Fast Neighbor Advertisement (FNA):** Es un mensaje que es enviado por el nodo móvil (*Mobile Node, MN*) al *router* de acceso próximo cuando ingresa a su dominio y recupera la conexión en el nuevo enlace.
- ✓ **Foreign Link:** El enlace foráneo es cualquier enlace diferente al enlace local del nodo móvil (*Mobile Node, MN*).
- ✓ **Foreign Subnet Prefix:** Es cualquier prefijo de subred IP diferente al prefijo de subred local del nodo móvil (*Mobile Node, MN*).
- ✓ **Handover Acknowledge (HACK):** El *Handover Acknowledge* es un mensaje que debe ser enviado como respuesta al mensaje de *Handover Initiate (HI)*, este mensaje es enviado por el *router* de acceso próximo.

El *router* de acceso próximo puede rechazar el handover, en este caso este debe indicar la razón, mediante alguno de los códigos disponibles.

- ✓ **Handover Initiate (HI):** Es un mensaje enviado por *MAP* al *router* de acceso próximo para iniciar el proceso de *fast handover* de los *MNs*. Se usa un código con valor de cero cuando se recibe un *Fast Binding Update (FBU)* con la dirección del *router* de acceso previo como dirección fuente, así mismo se usa un código con valor uno si la dirección no corresponde a la dirección del *router* de acceso previo.

- ✓ **Home Address:** Es la dirección asignada a un nodo móvil (*Mobile Node, MN*), usada como la dirección permanente del nodo móvil (*Mobile Node, MN*). Esta dirección se encuentra dentro del enlace local del *MN*. Los mecanismos de enrutamiento IP Standard entregarán los paquetes destinados para la dirección local del *MN* a su enlace local. Los *MNs* pueden tener múltiples direcciones locales cuando existen múltiples prefijos en un enlace local.
- ✓ **Home Agent (HA):** Es un *router* en el enlace local del nodo móvil (*Mobile Node, MN*) con el cual el *MN* tiene registrada su actual *Care-of address (CoA)*. Mientras el *MN* está lejos de su enlace local, el agente local (*Home Agent, HA*) intercepta los paquetes en el enlace local destinados a la dirección local (*Home Address*) del *MN*, los encapsula y los tuneliza a la *CoA* registrada del *MN*.
- ✓ **Home Agents List:** Cada *HA* genera una lista, la cual contiene información sobre otros *HAs* en su subred. La información en esta lista es tomada de los anuncios de router multicast no solicitados (*Unsolicited Multicast Router Advertisements*), los cuales son enviados por todos los *HAs*. La información acerca de todos los *HAs* es usada por el mecanismo de descubrimiento de agente local dinámico (*Dynamic Home Agent Discovery Mechanism*).
- ✓ **Home Link:** Es el enlace en el cual se define el prefijo de subred local del nodo móvil (*Mobile Node, MN*).
- ✓ **Home Subnet Prefix:** Es el prefijo de subred IP correspondiente a la dirección local (*Home Address*) de un nodo móvil (*Mobile Node, MN*).
- ✓ **Interface:** Es una conexión de un nodo móvil (*Mobile Node, MN*) a un enlace.
- ✓ **Interface Identifier:** Es un número usado para identificar la interfase de un nodo en un enlace. El identificador de interfase es el resto de bits de bajo orden de la dirección IP del nodo, que están después del prefijo de subred.
- ✓ **Local Binding Update (LBU):** Mensaje de actualización enviado por el nodo móvil (*Mobile Node, MN*) al MAP, con el objetivo de establecer un enlace entre la *On-link Care-of Address (LCoA)* y la *Regional Care-of Address (RCoA)*.
- ✓ **Mobile Node (MN):** Es un nodo que puede cambiar su punto de conexión de un enlace a otro, manteniendo la comunicación por medio de su dirección local (*Home Address*).
- ✓ **Mobility Anchor Point (MAP):** Es un *router* localizado en la red visitada por el nodo móvil (*Mobile Node, MN*), se encarga de la movilidad local y tiene funciones de agente local.

- ✓ **On-link Care-of Address (LCoA):** Es la dirección configurada, basada en el prefijo del *router* de acceso (*Access Router, AR*) defecto del nodo móvil (*Mobile Node, MN*) y enlazada con la *Regional Care-of Address (RCoA)*.
- ✓ **Proxy Router Advertisement (PrRtAdv):** Mensaje enviado por el *MAP* por medio de los *routers* de acceso (*Access Routers, ARs*) al *MN* como respuesta al *Router Solicitation for Proxy Advertisement (RtSolPr)* con el fin de proporcionar la dirección del enlace, la dirección *IP* y el prefijo de subred de los *routers* cercanos.
- ✓ **Regional Care-of Address (RCoA):** Es la dirección configurada con el prefijo de subred del *MAP*.
- ✓ **Router:** Es un nodo que envía paquetes *IP*, los paquetes no necesariamente deben estar direccionados a él.
- ✓ **Router Advertisement (RA):** Anuncio de router que contienen información acerca de la red visitada por el nodo móvil (*Mobile Node, MN*).
- ✓ **Router Solicitation for Proxy Advertisement (RtSolPr):** Este tipo de mensajes es enviado por el nodo móvil (*Node Mobile, MN*) solicitando un *Proxy Router Advertisement (PrRtAdv)* cuando inicia el *handover*.
- ✓ **Subnet Prefix:** El prefijo de subred es una cadena de bits que consiste en los bits iniciales de la dirección *IP*.

## BIBLIOGRAFÍA

- [1] H. Soliman, C. Castelluccia, K. El Malki, L. Bellver. Hierarchical Mobile IPv6 Mobility Management (HMIPv6). RFC 4140. Agosto 2005.
- [2] D. Johnson, C. Perkins, J. Arkko. Mobility Support in IPv6. RFC 3775. Junio 2004.
- [3] K. Fall, K. Varadhan. The ns Manual. Noviembre 16 de 2006.
- [4] Robert Hsieh, The unofficial ns tutorial on implementing HMIPv6 with fast-handover. School of Electrical Engineering & Telecommunication. 2003
- [5] Ceballos, Javier. Enciclopedia del Lenguaje C++, Editorial Alfaomega, Mexico, Febrero 2004, Paginas (210, 211, 240-246, 467-470).
- [6] S. Deering, R. Hinden. Internet Protocol, Version 6 (IPv6). RFC 2460. Diciembre 1998.
- [7] Becerra S, Line Y, Amaya S, Cecilia. Proyecto de Especialización. Simulación del protocolo Mobile IPv6 mediante la herramienta MOBIWAN bajo el simulador Network Simulator. Universidad Pontificia Bolivariana – Seccional Bucaramanga.
- [8] R. Koodli. Fast Handovers for Mobile IPv6. RFC 4068. Julio 2005.
- [9] H. Jung, H. Soliman, S. Koh, N. Takamiya. Fast Handover for Hierarchical MIPv6 (F-HMIPv6). draft-jung-mipshop-fhmipv6-00. Octubre 2005.
- [10] T. Kwon, M. Gerla, and S. Das. "Mobility management for VoIP service: Mobile IP vs. SIP". IEEE Wireless Communications, Volume 9, Issue 5:66-75, Oct. 2002.
- [11] Y. Gwon, J. Kempf, and A Yegin. "Scalability and Robustness Analysis of Mobile IPv6, Fast Mobile IPv6, Hierarchical Mobile IPv6, and Hybrid IPv6 Mobility Protocols using a Large-Scale Simulation". In 2004 IEEE International Conference on Communications, volume 7, pages 4087-4091, 20 March-24 June 2004.
- [12] I. Vivaldi, M.H. Habaebi, B.M. Ali, and V. Prakesh. "Fast Handover Algorithm for Hierarchical Mobile IPv6 Macro-Mobility Management". In The 9th Asia-

Pacific Conference on Communications. APCC 2003. Volume 2, pages 630-634, 21-24 Sept 2003.

- [13] Miranda P, Diana. Tesis de Grado. Estudios de los protocolos Mobile IPv6 y Hierarchical Mobile IPv6 implementados con el mecanismo Fast-Handoff utilizando la herramienta Network Simulator. Universidad Pontificia Bolivariana – Seccional Bucaramanga. 2005.

## ANEXOS

### ANEXO A

#### A.1 Instalación del Programa NS y la Herramienta HMIPv6\_SM

El proceso de instalación se divide en dos partes, la instalación del programa *Network Simulator* versión 2.30 y la instalación de la herramienta *HMIPv6\_SM*.

Antes de iniciar la instalación del NS-2.30 y la herramienta *HMIPv6\_SM*, se debe tener instalado el sistema Linux de su agrado.

##### A.1.1 Instalación de ns-2.30

Teniendo instalado la distribución Linux, procedemos a la instalación del ns-3.0 (el lugar de instalación mencionado en este tutorial es opcional)

1. Se descomprime el archivo `ns-allinone-3.0.tar.gz` en el directorio

```
file: /home/"nombre_de_cuenta"
```

2. Desde terminal, se ubica en el directorio

```
/home/"nombre_de_cuenta"/ns-allinone-3.0/
```

Y digitando `./install` se inicia el proceso de instalación del instalación del programa.

3. Al terminar la instalación, desde el mismo terminal, digita `cd ns-3.0` ubicándonos en el directorio

```
/home/"nombre_de_cuenta"/ns-allinone-3.0/ns-3.0
```

Donde escribimos `./validate` para iniciar la ejecución de las pruebas pertinentes.

4. En este momento el ns ya está instalado, ahora procedemos a la ejecución del programa desde el terminal.

Nos volvemos a ubicar en el directorio de la cuenta donde se instaló el ns pero ahora desde el escritorio. Nos vamos al escritorio, y entramos a Personal, saldrá la siguiente línea en la barra de direcciones.

*file: /home/"nombre\_de\_cuenta"*

En este caso, "nombre\_de\_cuenta" es arley.

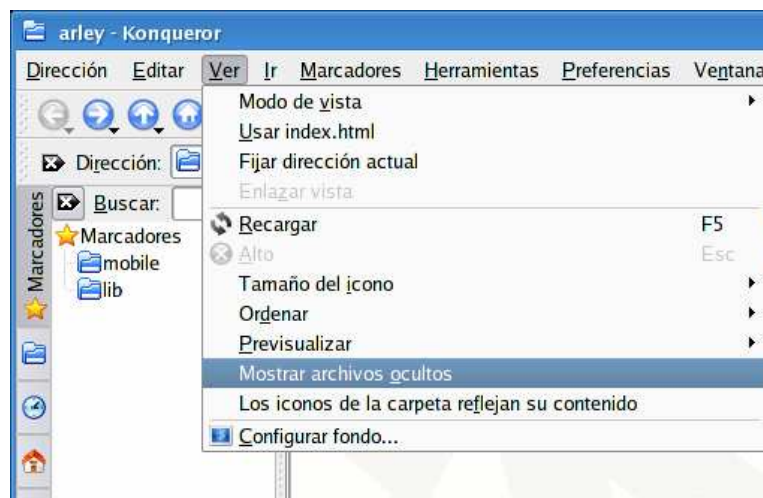
*Figura A.1: Instalación de NS "nombre\_de\_cuenta".*



Habilitamos la opción de ver archivos ocultos, en la barra de Herramientas

*Ver -> Mostrar archivos ocultos*

*Figura A.2: Instalación de NS "mostrar archivos ocultos".*



Cuando se visualicen todos los archivos ocultos, abrimos el `.bashrc` con un editor de texto y agregamos las siguientes líneas

```
# .bashrc

PATH=$PATH:/home/arley/ns-allinone-2.30/ns-2.30:/home/arley/ns-allinone-2.30/bin:/home/arley/ns-allinone-2.30/tcl8.4.13/unix:/home/arley/ns-allinone-2.30/tk8.4.13/unix

export LD_LIBRARY_PATH=/home/arley/ns-allinone-2.30/lib:/home/arley/ns-allinone-2.30/otcl-1.12

export TCL_LIBRARY=/home/arley/ns-allinone-2.30/tcl8.4.13/library

export PATH

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
```

5. Con el ns instalado y con el bash modificado, nos disponemos a hacer ha probar el programa. Nos situamos desde el terminal en la carpeta donde tengamos el script `example.tcl` y digitamos `ns example.tcl`. Si el programa ns está bien instalado, se visualizara el archivo `nam` y el archivo `xgraph` al terminar la ejecución del script.

### **A.1.2 Instalación de la Herramienta HMIPv6\_SM**

Con el *Network Simulator* totalmente funcional, procedemos a la instalación de la herramienta HMIPv6\_SM

1. Se descomprime el archivo `HMIPv6_SM.zip`
2. Se copian los archivos de la carpeta `mobile` que se encuentran en la carpeta HMIPv6\_SM

```
HMIPv6_SM/mobile
```

A la carpeta `mobile` del ns sin modificar.

```
/home/arley/ns-allinone-2.30/ns-2.30/mobile
```

De igual forma se hace con la carpeta



*HMIPv6\_SM/tcl/lib*

A la carpeta lib del ns sin modificar

*/home/arley/ns-allinone-2.30/ns-2.30/tcl/lib*

3. Se abre el archivo Makefile.ini con un editor de texto para agregar al programa los archivos *fast-handover.cc* y *fast-handover.h* que se encuentran en la carpeta *mobile*

```
OBJ_CC = \  
    common/scheduler.o common/object.o common/packet.o \  
    common/ip.o routing/route.o common/connector.o common/ttl.o \  
    trace/trace.o trace/trace-ip.o \  
    classifier/classifier.o classifier/classifier-addr.o \  
    classifier/classifier-hash.o \  
    mobile/fasthandover.o \  
@V_STLOBJ@
```

Se debe dejar @V\_STLOBJ@ como última línea de código para  
OBJ\_CC = \

4. Desde terminal, vamos a la dirección

*/home/arley/ns-allinone-2.30/ns-2.30*

Y digitamos *./configure*

5. Por último escribimos *make*

Si la compilación termina sin ningún problema, puede pasar a probar algunos de los script disponibles en la carpeta Ejemplos que se encuentra dentro de HMIPv6\_SM.zip para confirmar la correcta instalación de la herramienta.

## ANEXO B

### B.1 Tutorial de HMIPv6\_SM

A fin de hacer mas explicito la creación de un script para ns, describiremos la programación de la topología presente en Simula 1: un nodo móvil con su respectivo HA, un MAP con dos ARs, un nodo internet y un nodo correspondiente.

Para la creación de este tutorial se tomo como base el tutorial creado por Marc Greis agregándole los cambios en el script para el funcionamiento de la Herramienta HMIPv6\_SM.

#### B.1.1 Creación del Objeto Simulador

En primer instancia hay que crear el archivo, lo cual se hace en cualquier editor de texto de Linux y guardándolo con una extensión .tcl. Por ejemplo Simulacion\_1.tcl.

Creado ya el archivo, se crea el objeto simulador

```
set ns_ [new Simulator]
```

#### B.1.2 Estructura Jerárquica

Para que los paquetes se enruten entre el dominio inalámbrico y el dominio cableado se necesita usar un enrutamiento jerárquico. Para esto se necesita que los nodos sean configurados para usar direcciones tipo jerárquico.

```
$ns_ node-config -addressType hierarchical
```

Los nodos cableados e inalámbricos se dividen en distintos dominios. Los dominios y subdominios son definidos por medio de la estructura jerárquica

```
AddrParams set domain_num_ 3 #Numero de Dominios.  
lappend cluster_num 2 1 3 #Numero de Clúster por Dominio.  
AddrParams set cluster_num_ $cluster_num  
lappend eilastlevel 1 1 2 1 1 1 #Numero de Nodos por clúster.  
AddrParams set nodes_num_ $eilastlevel
```

#### B.1.3 Creación e Inicialización de los Archivos de Salida

Se crean los archivos y se abren los archivos de salida:

*Simulacion\_1.nam* para la simulación visual de la topología.

*Simulacion\_1\_PP.tr* y *Simulacion\_1\_PR.tr* donde se guardaran datos de paquetes perdidos y paquetes recibidos.

*traffic\_Simulacion1.tr* donde se guardara las trazas tanto para la red inalámbrica como para la red fija.

```
set tracefd [open traffic_Simulacion1.tr w]
set f0 [open Simulacion_1_PP.tr w]
set f1 [open Simulacion_1_PR.tr w]
set namf [open Simulacion_1.nam w]
```

Indicamos a los objetos simuladores creados anteriormente para escribir todos los datos de la simulación pertinentes para cada archivo.

```
$ns_ use-newtrace
$ns_ trace-all $tracefd
$ns_ namtrace-all $namf
```

#### **B.1.4 Asignación de Grilla**

Se crea la grilla en la cual en se efectuara la simulación

```
set topo [new Topography]
$topo load_flatgrid 1000 1000
set god_ [create-god 1]
```

#### **B.1.5 Red Fija**

Ahora procedemos a la creación de los nodos de la red fija. Se crean los nodos MAP e Internet, y se le asigna una dirección. La dirección asignada depende directamente al dominio al que pertenece.

```
#MAP's
set enc_ 0
set MAP [$ns_ node 2.0.0]

# Nodo Internet
set N1 [$ns_ node 0.1.0]
```

#### **B.1.6 Configuración de la Red Inalámbrica**

Ahora para crear los nodos estación base (HA y ARs) necesitamos configurar su estructura. Esto hace parte de un nuevo nodo API, el cual consiste primero en su configuración y luego en la creación de los nodos. Como la estación base es una

compuerta entre las dos redes, esta necesita tener los mecanismos de enrutamiento cableado encendido, lo cual se hace programando *node-config option-wiredRouting ON*. Luego de crear la estación base la reconfiguramos nodo inalámbrico cambiando a *wiredRouting OFF*. Las otras opciones de configuración usadas por la estación base permanecen iguales a las usadas por un nodo móvil.

En el caso del Nodo Móvil, este debe tener el mismo dominio que el HA

```
set chan_ [new Channel/WirelessChannel]
$ns_ node-config -mobileIP ON \
    -adhocRouting AODV \
    -llType LL \
    -macType Mac/802_11 \
    -ifqType Queue/DropTail/PriQueue \
    -ifqLen 50 \
    -antType Antenna/OmniAntenna \
    -propType Propagation/TwoRayGround \
    -phyType Phy/WirelessPhy \
    -channel $chan_ \
    -topoInstance $topo \
    -wiredRouting ON \
    -agentTrace ON \
    -routerTrace ON \
    -macTrace OFF

# Agentes Locales
set HA [$ns_ node 1.0.0]

# Nodos Correspondientes
set CN [$ns_ node 0.0.0]

$ns_ node-config -wiredRouting OFF

# Nodos Móviles
set MN [$ns_ node 1.0.1]
[$MN set regagent_] set home_agent_ [AddrParams addr2id [$HA
node-addr]]
```

La optimización de Ruta por defecto esta activada, pero para esta simulación se desactivara insertando la siguiente line a de código.

```
[$MN set regagent_] set Optimization_ 0
```

```
$ns_ node-config -wiredRouting ON
# Routers de Acceso

set AR1 [$ns_ node 2.1.0 2.0.0]

set AR2 [$ns_ node 2.2.0 2.0.0]
```

### **B.1.7 Ubicación de los Nodos en la Grilla**

Creados todos los nodos, tanto los pertenecientes a la red fija como a la red inalámbrica, procedemos a darles su ubicación dentro de la grilla de simulación. Es de gran importancia que el usuario haya hecho la distribución de los nodos en la grilla con anterioridad, así se prevén errores y agiliza la programación del script.

```
$CN set X_ 80.0
$CN set Y_ 5.0

$N1 set X_ 120.0
$N1 set Y_ 10.0

$HA set X_ 160.0
$HA set Y_ 5.0

$MN set X_ 150
$MN set Y_ 130

$MAP set X_ 120.0
$MAP set Y_ 15.0

$AR1 set X_ 85.0
$AR1 set Y_ 135.0

$AR2 set X_ 155.0
$AR2 set Y_ 135.0
```

### **B.1.8 Enlaces de la Red Fija**

Creados y posicionados los nodos, se realizan los enlaces de los nodos de la red fija. Primero se asigna el tipo de enlace, el ancho de banda del enlace, tipo de cola y el tiempo de retardo máximo del paquete. Por ejemplo, el enlace del el *Nodo Internet (NI)* con el *Nodo Correspondiente (CN)* es un enlace tipo duplex con un ancho de banda de 100Mb, con tipo de cola RED (Random Early Detection) y con un tiempo de retardo máximo del paquete de 2ms.

```

$ns_ duplex-link $CN $N1 100Mb 2ms RED
$ns_ duplex-link $HA $N1 100Mb 2ms RED
$ns_ duplex-link $MAP $N1 100Mb 50ms RED
$ns_ duplex-link $AR1 $MAP 2000Kb 2ms DropTail
$ns_ duplex-link $AR2 $MAP 2000Kb 2ms DropTail

```

### B.1.9 Configuración del Nodo MAP

Ahora nos disponemos a la configuración del MAP, elemento esencial en el protocolo *HMIPv6*. Se crea el agente MAPA y se le asigna la dirección del MAP (2.0.0) y sus respectivos ARs (2.1.0 y 2.2.0). Siempre se tiene que asignar la dirección del MAP primero que las direcciones de los ARs. Y por último se ata el agente al nodo MAP.

Se puede asignar hasta 5 ARs por MAP.

```

set mapa [new Agent/MAPA]
$mapa asignar "2.0.0" "2.1.0" "2.2.0"
$ns_ attach-mapagent $MAP

```

### B.1.10 Creación de los Agentes Transporte y las Fuentes de Trafico

Se establece el agente de transporte y de tráfico. Para esta simulación, se escoge un agente UDP y se ata al Nodo Correspondiente.

```

set udp [new Agent/UDP]
$ns_ attach-agent $CN $udp

```

Como agente de tráfico se elije un agente CBR que se ata al agente UDP, con tamaño de paquete de 1000 e intervalo de 0.05 segundos entre paquetes.

```

set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 1000
$cbr set interval_ 0.05
$cbr attach-agent $udp

```

Se crea un Agente LossMonitor como receptor de tráfico, que nos permite monitorear el comportamiento del Nodo Móvil gracias a las variables que posee este agente (nlost\_, npkts\_, entre otras).

```

set null [new Agent/LossMonitor]
$ns_ attach-agent $MN $null
$ns_ connect $udp $null

```

Inicia la transmisión de paquetes al segundo de simulación.

```
$ns_ at 1.0 "$cbr start"
```

### B.1.11 Movimiento del Nodo Móvil

Ahora se procede a la programación del movimiento del nodo móvil. El nodo móvil comienza a trasladarse a los 20 segundo de simulación, a la posición  $x= 80.0$   $y= 130$  a una velocidad de 5m/s.

```
$ns_ at 20.0 "$MN setdest 80.0 130 5"
```

### B.1.12 Creación de los Procesos de Grabación y Finalización

El procedimiento de grabación varía según los datos que el usuario desee registrar para un posterior análisis; por este motivo, el procedimiento de grabación que se mencionara a continuación puede estar sujeto a una modificación parcial o total, dependiendo de las necesidades del usuario.

Este procedimiento permite realizar la graficas de los paquetes perdidos y los paquetes recibidos, que consiste en tomar una instancia del simulador y fijar el tiempo después del cual debe ser llamado el proceso nuevamente, ya que una grafica es una serie de puntos que se deben hallar constantemente, en este caso se escogió que fuera cada segundo (manejado con la variable *time*). Luego se captura en *bw0* y *bw1* cuantos datos se han perdido (*nlost\_*) y cuantos paquetes se han recibido (*npkts\_*) en el receptor de trafico *null*. Se toma el tiempo actual de simulación (*set now*) para ponerlo en los archivos de salida *f0* y *f1* junto con el valor de las variables *bw0* y *bw1* para ir guardando los datos.

```
proc record {} {
    global null f0 f1
    set ns_ [Simulator instance]
    set time 1.0
    set bw0 [$null set nlost_]
    set bw1 [$null set npkts_]
    set now [$ns_ now]
    puts $f0 "$now [expr $bw0/1.0]"
    puts $f1 "$now [expr $bw1/1.0]"
    $null set nlost_ 0
    $null set npkts_ 0
    $ns_ at [expr $now+$time] "record"
}
```

El procedimiento *finish* cierra los archivos de salida mencionados al inicio de la simulación; además nos permite llamar el Xgraph para mostrar la grafica y la ejecución del archivo nam.

```
proc finish {} {
    global f0 f1 tracefd ns_ namf
    $ns_ flush-trace
    flush $tracefd
    close $tracefd
    close $f0
    close $f1
    close $namf
    exec nam Simulacion_1.nam &
    exec xgraph Simulacion_1_PR.tr Simulacion_1_PP.tr -
    geometry 1000x1000&
    exit 0
}
```

Por último, se le asigna el tiempo de inicio de los procesos *record* y *finish*.

```
$ns_ at 0.0 "record"
$ns_ at 50.0 "finish"
$ns_ run
```

Terminado el script, se procede a guardar los cambios realizados en un archivo con extensión tcl; para este caso se le asignara el nombre de Simulacion\_1.tcl

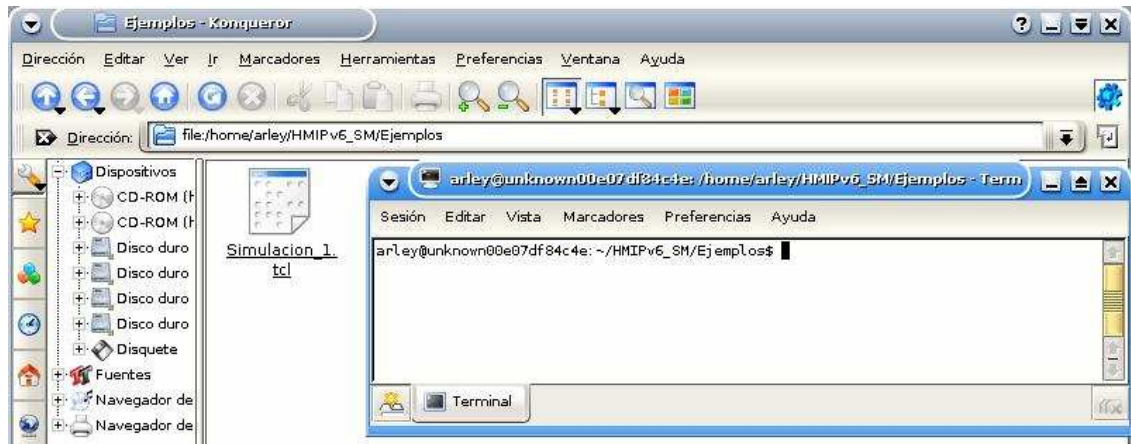
## B.2 Ejecución del Script

Con el script ya creado (Simulacion\_1.tcl), procedemos a la ejecución del archivo Simulacion1.tcl. Para esto se abre un terminal en el lugar donde se encuentra el archivo. En esta ocasión hemos guardado el script en la carpeta

```
:/home/arley/HMIPv6_SM/Ejemplos
```



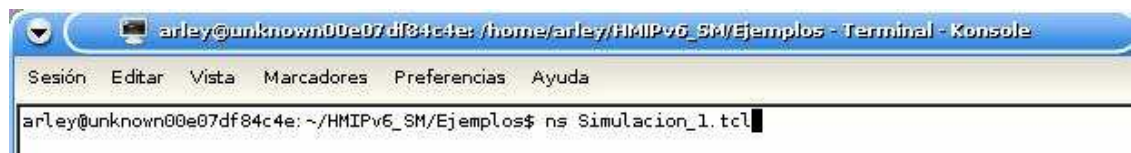
Figura B.1: Ejecución del Script “La terminal”.



Ya posicionado en la dirección donde se encuentra el archivo tcl, se ejecuta el programa escribiendo:

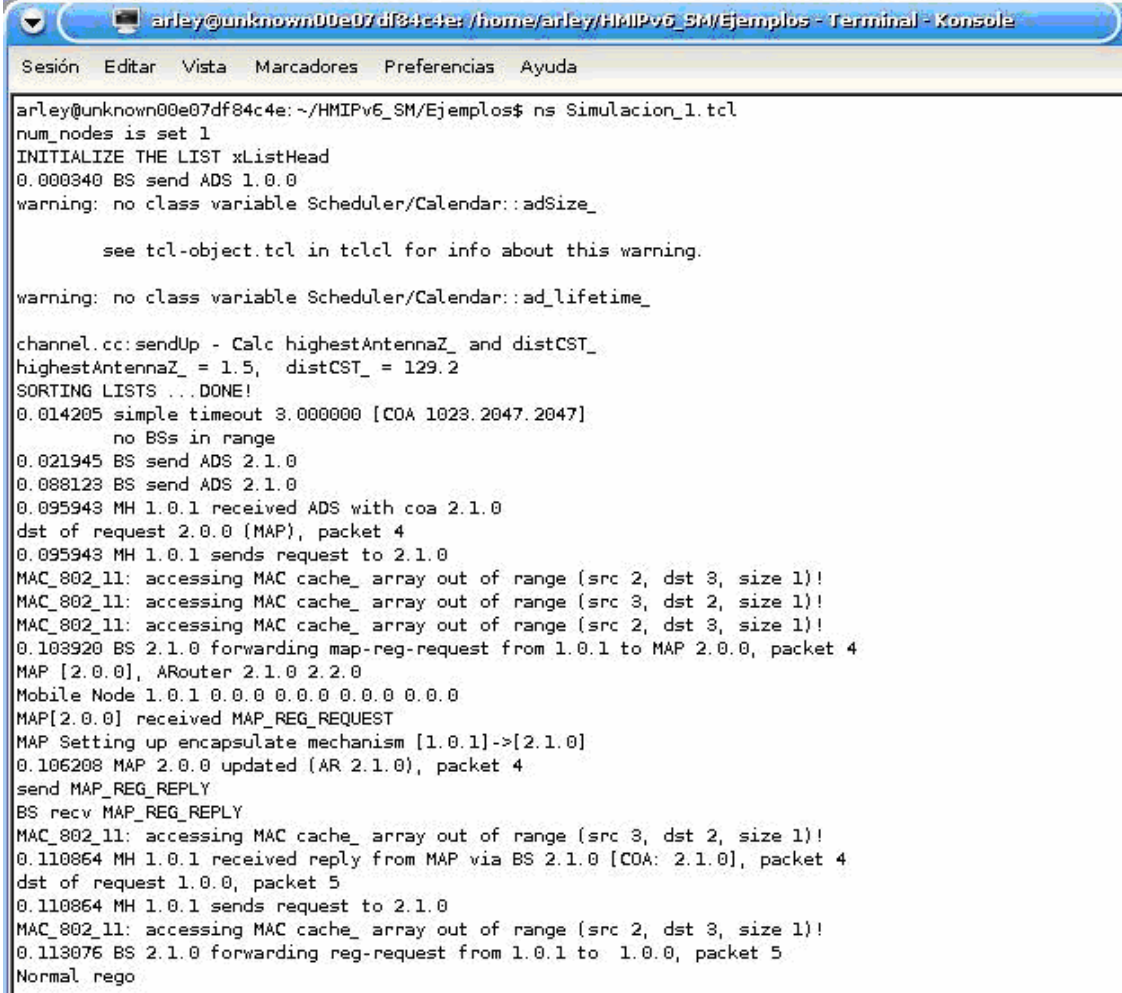
```
ns Simulacion_1.tcl
```

Figura B.2: Ejecución del programa “Simulacion\_1.tcl”.



Inmediatamente comienza la simulación. Durante la ejecución de la simulación se visualiza unos mensajes en pantalla que nos muestra de una forma mas clara los métodos de registros de los diferentes elementos del protocolo *HMIPv6*, ya que en el archivo trace que se nos presenta al final de la simulación, no se puede distinguir con claridad estos métodos de registros.

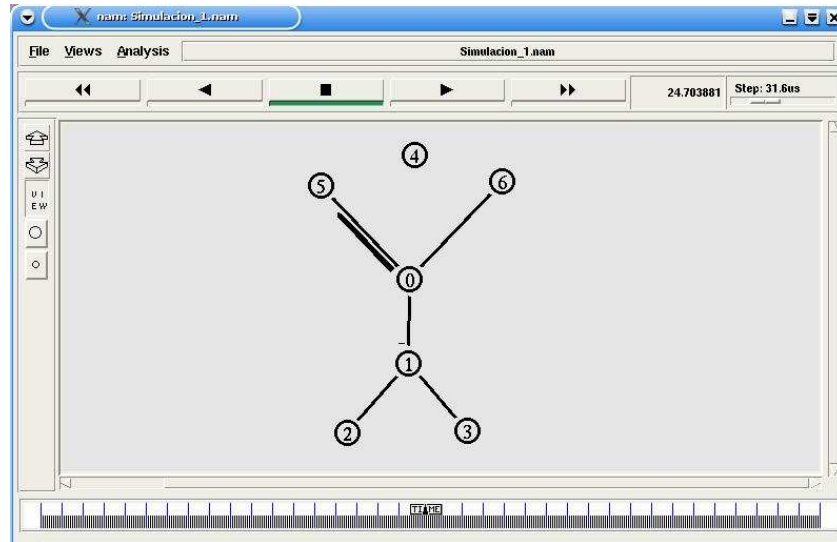
Figura B.3: Visualización de la ejecución del programa desde la terminal.



```
arley@unknown00e07df84c4e: ~/HMIPv6_SM/Ejemplos - Terminal - Konsole
Sesión Editar Vista Marcadores Preferencias Ayuda
arley@unknown00e07df84c4e:~/HMIPv6_SM/Ejemplos$ ns Simulacion_1.tcl
num_nodes is set 1
INITIALIZE THE LIST xListHead
0.000340 BS send ADS 1.0.0
warning: no class variable Scheduler/Calendar::adSize_
        see tcl-object.tcl in tclcl for info about this warning.
warning: no class variable Scheduler/Calendar::ad_lifetime_
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 129.2
SORTING LISTS ... DONE!
0.014205 simple timeout 3.000000 [COA 1023.2047.2047]
        no BSs in range
0.021945 BS send ADS 2.1.0
0.088123 BS send ADS 2.1.0
0.095943 MH 1.0.1 received ADS with coa 2.1.0
dst of request 2.0.0 (MAP), packet 4
0.095943 MH 1.0.1 sends request to 2.1.0
MAC_802_11: accessing MAC cache_array out of range (src 2, dst 3, size 1)!
MAC_802_11: accessing MAC cache_array out of range (src 3, dst 2, size 1)!
MAC_802_11: accessing MAC cache_array out of range (src 2, dst 3, size 1)!
0.103920 BS 2.1.0 forwarding map-reg-request from 1.0.1 to MAP 2.0.0, packet 4
MAP [2.0.0], ARouter 2.1.0 2.2.0
Mobile Node 1.0.1 0.0.0 0.0.0 0.0.0 0.0.0
MAP[2.0.0] received MAP_REG_REQUEST
MAP Setting up encapsulate mechanism [1.0.1]->[2.1.0]
0.106208 MAP 2.0.0 updated (AR 2.1.0), packet 4
send MAP_REG_REPLY
BS recv MAP_REG_REPLY
MAC_802_11: accessing MAC cache_array out of range (src 3, dst 2, size 1)!
0.110864 MH 1.0.1 received reply from MAP via BS 2.1.0 [COA: 2.1.0], packet 4
dst of request 1.0.0, packet 5
0.110864 MH 1.0.1 sends request to 2.1.0
MAC_802_11: accessing MAC cache_array out of range (src 2, dst 3, size 1)!
0.113076 BS 2.1.0 forwarding reg-request from 1.0.1 to 1.0.0, packet 5
Normal rego
```

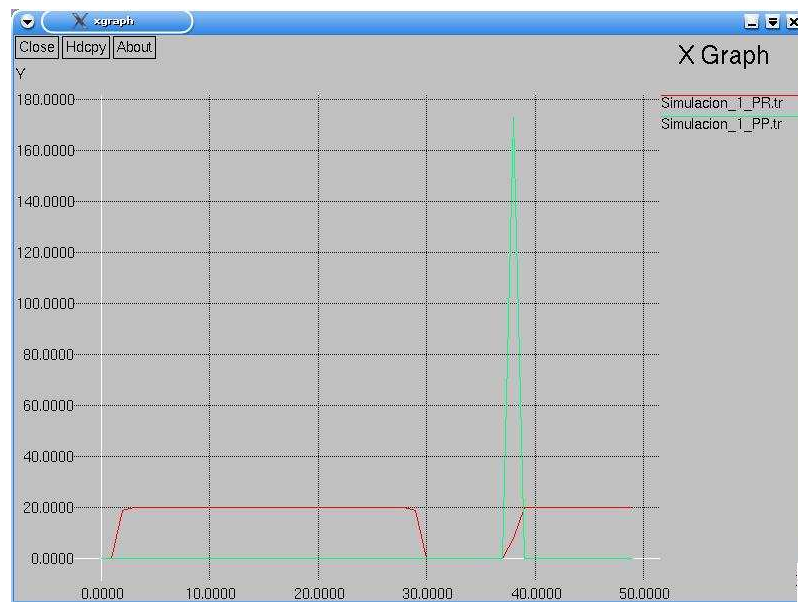
Al terminar la simulación, se abre el archivo nam que nos permite hacer el seguimiento a la simulación de una forma visual.

Figura B.4: Visualización grafica del funcionamiento del programa mediante el NAM.



Y también se despliega las graficas de los archivos .tr

Figura B.5: Visualización grafica de los archivos (.tr) mediante el X Graph.



Los archivos nam y xgraph creados por Simulacion\_1.tcl quedan guardados en la misma carpeta desde donde se ejecuto el script.

## ANEXO C

A fin de facilitar el entendimiento del archivo trace, se procede a explicar los campos en que se divide el formato de trazas tanto para la traza de una red fija como para la de una red inalámbrica.

### C.1 Formato para Trazas de una Red Fija

Ejemplo de una traza de una Red Fija se muestra a continuación:

```
+ 1.006243 1 0 cbr 1040 ----- 0 0.0.0.1 2.0.0.1 0 7
```

1	2	3	4	5	6	7	8	9	10	11	12
+	1.006243	1	0	cbr	1040	-----	0	0.0.0.1	2.0.0.1	0	7

El formato de la traza anterior se divide en los siguientes campos:

#### C.1.1 Tipo de Evento (1)

El primer campo describe el tipo de evento que esta tomando lugar en el nodo y puede ser uno de los siguientes 5 tipos:

- + : Entrando en Cola
- - : Saliendo de la Cola
- r : Recibiendo paquete
- d : Paquete perdido
- e: Error

#### C.1.2 Tiempo de Simulacion (2)

El segundo campo muestra el tiempo de simulación en segundo, en que transcurre el evento.

#### C.1.3 Nodo Fuente y Nodo Destino (3 y 4)

En estos campos se puede visualizar el nodo desde donde sale el paquete y luego el nodo que tiene como destino el paquete.

### **C.1.5 Nombre del Paquete (5)**

En el quinto campo se muestra el tipo del paquete.

### **C.1.6 Tamaño del Paquete (6)**

Este campo muestra el tamaño del paquete en Bytes

### **C.1.7 Banderas (7)**

El siguiente campo se visualiza las banderas activas que lleva el paquete. Si no hay ninguna activa se muestra una línea punteada. Las banderas pueden ser:

- E: Experimento congestión (CE)
- N: ECN-Capable-Transport (ECT) indicado en la cabecera IP
- C: ECN-Echo
- A: Congestión en ventana reducida (CWR) en la cabecera TCP
- P: Priotidad
- F: Rápido inicio en TCP.

### **C.1.8 Flow ID (8)**

Este campo es utilizado cuando se especifica la secuencia de colores para visualizar en el NAM.

### **C.1.9 Dirección Fuente y Dirección Destino (9 y 10)**

En estos campos se puede observar la dirección fuente y la dirección destino del paquete de la forma

Dirección. Puerto

### **C.1.10 Número de Secuencia (11)**

En este campo se muestra el número de secuencia del paquete en la capa de red del protocolo.

### **C.1.11 Unique Packet ID (12)**

EN este ultimo campo se visualiza la ID del paquete en la simulación.

## C.2 Formato para Trazas de una Red Inalámbrica

Un ejemplo de una red inalámbrica se muestra a continuación:

```
f -t 1.258320000 -Hs 6 -Hd 4194305 -Ni 6 -Nx 155.00 -Ny
135.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms
0 -Mt 0 -Is 0.2 -Id 4194305.2 -It cbr -Il 1040 -If 0 -Ii 15 -
Iv 29 -Pn cbr -Pi 4 -Pf 0 -Po 0
```

1	2	3	4	5	6	7
F	-t 1.2583 20000	Ni 6	-Is 0.2	-Hs 6	-Ma 0	-Pn cbr
		-Nx 155.00	-Id 419430 5.2	-Hd 4194305	-Md 0	-Pi 4
		-Ny 135.00	-It cbr		-Ms 0	-Pf 0
		-Nz 0.00	-Il 1040		-Mt 0	-Po 0
		-Ne - 1.000000	-If 0			
		-Nl RTR	-Ii 15			
		-Nw ---	-Iv 29			

El formato de la traza anterior se divide en los siguientes campos

### C.2.1 Tipo de Evento (1)

El primer campo, al igual que en la traza de una red fija, muestra el tipo de eventos que esta tomando lugar en el nodo y puede ser uno de los siguientes 4 tipos:

- s: Enviar.
- r: Recibir.
- p: Dejar.
- f: Transmitir.

### C.2.2 Tag Generales (2)

El segundo campo inicia con una “-t” y se coloca para señalar tiempo o configuración global

- -t time
- -t \* (Configuración Global)

### **C.2.3 Tags de Propiedades del Nodo (3)**

Este campo denota las propiedades del nodo como id del nodo, router o MAC. Los tags inician con “-N” y se describirán a continuación.

- -Ni: id del nodo.
- -Nx: Coordenada x del nodo
- -Ny: Coordenda y del nodo
- -Nz: Coordenada z del nodo.
- -Ne: Nivel de energía del nodo.
- -NI: Nivel de la traza, tal como AGT, RTR, MAC.
- -Nw: Razón para el evento. Las diferentes razones para dejar un paquete se muestran a continuación.
  - “END” DROP\_END\_OF\_SIMULATION
  - “COL” DROP\_MAC\_COLLISION
  - “DUP” DROP\_MAC\_DUPLICATE
  - “ERR” DROP\_MAC\_PACKET\_ERROR
  - “RET” DROP\_MAC\_RETRY\_COUNT\_EXCEEDED
  - “STA” DROP\_MAC\_INVALID\_STATE
  - “BSY” DROP\_MAC\_BUSY
  - “NRTE” DROP\_RTR\_NO\_ROUTE. Ej: Router no valido

- “*LOOP*” DROP\_RTR\_ROUTE\_LOOP.
- “*TTL*” DROP\_RTR\_TTL. Ej: TTL a rechazado el cero
- “*TOUT*” DROP\_RTR\_QTIMEOUT. Ej: El paquete a expirado
- “*CBK*” DROP\_RTR\_MAC\_CALLBACK
- “*IFQ*” DROP\_IFQ\_QFULL. Ej: IFQ no tiene espacio en el buffer.
- “*ARP*” DROP\_IFQ\_ARP\_FULL
- “*OUT*” DROP\_OUTSIDE\_SUBNET. Ej: Dejado porque la estación base esta recibiendo actualización de enrutamiento de nodos fuera del dominio.

#### **C.2.4 Información del Paquete de nivel IP (4)**

Los tags de este campo inicia con “-I” y son enumeradas a continuación.

- -Is: Dirección fuente. Número de puerto de la fuente
- -Id: Dirección destino. Número de puesto del destino
- -It: Tipo de paquete.
- -Il: Tamaño del paquete.
- -If: ID de flujo
- -Ii: unique ID
- -Iv: valor ttl.

#### **C.2.5 Información Siguiendo Destino (5)**

Este campo provee información del próximo destino y el tag inicia con “-H”

- -Hs: ID de este nodo.
- -Hd: ID para el próximo destino.



### C.2.6 Información del paquete en el nivel MAC (6)

Este campo da información de la capa MAC e inicia con “-M”

- -Ma: Duración.
- -Md: Dirección Ethernet del destino
- -Ms: Dirección Ethernet de la fuente
- -Mt: tipo de Ethernet.

### C.2.7 Información del paquete del nivel de Aplicación. (7)

La información del paquete del nivel de aplicación consiste del tipo de aplicación como ARP, TCP, tipo de enrutamiento adhoc como DSDV, DSR, AODV etc. Este campo inicia con “-P”

- -P arp: Protocolo de Resolución de Dirección.
  - -Po: Petición/Respuesta de ARP.
  - -Pm: Dirección MAC de la fuente.
  - -Ps: Dirección fuente.
  - -Pa: Dirección MAC de destino.
  - -Pd: Dirección de destino.
- -P dsr: Este indica el protocolo de enrutamiento adhoc llamado *Dynamic Source Routing*. Información de DSR es representada como sigue:
  - -Pn: Número de nodos recorridos.
  - -Pq: Bandera de petición de enrutamiento.
  - -Pi: Número de secuencia de petición de enrutamiento.
  - -Pp: Bandera de respuesta de enrutamiento.
  - -Pe: Scr de fuente de enrutamiento->Dst de fuente de enrutamiento.
  - -Pw: Bandera de reporte de error.
  - -Pm: Número de errores.
  - -Pc: Reporte de whom.
  - -Pb: Enlace de error del enlace A -> enlace B.
- -P cbr: Rata de Bit Constante. Información acerca de la aplicación CBR
  - -Pi: Número de secuencia.
  - -Pf: Número de veces que se transmitió el paquete.
  - -Po: Número opcional de transmisión.