Capítulo 5: Simulaciones y resultados

En este capítulo se presentan las simulaciones del modelado de los mecanismos de transición. Una vez creados los modelos de las redes a simular, se prosigue con la creación del código que describirá su comportamiento. Debido a que estos códigos son muy extensos, no se mencionarán directamente en este capítulo; por consiguiente, si se es necesario se referirá a ellos directamente en el apéndice F; ahí se incluirán sólo las funciones más importantes de cada módulo, el resto se anexará en el disco adjunto.

5.1 Modelo de Pila-dual

Este modelo se basa en el RFC 2893 y consiste en el hecho de que los dispositivos de red utilizan dos pilas en su arquitectura que les permite dar servicio tanto a IPv4 como a IPv6. Estas dos pilas trabajan de manera independiente una de la otra, pero no de forma simultánea, lo que lleva a desactivar una de ellas para utilizar la otra. Esta característica permite tener dispositivos, como ruteadores *o host*, enteramente del tipo IPv6 que pueden interactuar con IPv4 y viceversa.

Para la realización de este modelo, se utilizaron parámetros como indicadores para cada uno de los módulos. Estos parámetros diferencian entre el tipo de pila de los dispositivos, pudiendo ser 4 para IPv4 y 6 para IPv6, y el tipo de *host* que inicia la transmisión durante la simulación. Estos parámetros regularmente son pedidos al inicio de la simulación por OMNeT++, pero para efectos prácticos de este trabajo se definieron previamente en el archivo .ini del modelo. La Tabla 5.2 muestra el nombre del parámetro asignado a cada uno de los dispositivos utilizados en el modelo de la Pila-dual (ver Figura 5.2). Como se puede notar unos son de tipo entero, como los utilizados para diferenciar las pilas, y otros de tipo *string*, los que diferencian entre *host*.

Primera simulación

Modelo: Pila-dual (*Dual-stack*).

Características de inicio (*especificadas en el archivo .ini*).

La Tabla anterior significa que la simulación va a comenzar en el sistema cuyo parámetro es *41*, en este caso *Network.IPv4_end_system1,* el nodo origen del datagrama será "*H1*" del *Network.IPv4_end_system1*. El tipo de pila que se va a seleccionar va a ser tipo *4*, *stackv4*, de "*H1*" del *Network.IPv4_end_system1*. El nodo destino de la simulación será el sistema cuyo parámetro es *61*, *Network.IPv6_end_system1*, el nodo destino de este sistema será "*H5*". Por deducción tratándose del mecanismo de pila doble, la pila destino tendrá que ser tipo *4 (ver Tabla 5.2).*

Tabla 5.1: Parámetros de inicio para simulación 1.

Parámetro	Valor	Descripción
Network.type	41	Nodo de inicio
Network.host	"H1"	*Host* origen
Network.stack	4	Tipo de pila
Network.type_dest	61	Nodo destino
Network.host_dest	"H5"	*Host* destino

Tabla 5.2: Parámetros de inicio para el modelo de pila-dual.(a) muestra los valores probables para el tipo de subred y nodo de inicio.(b) Valor para el tipo de pila.

Network.IPv4_end_system1.type=41
Network.IPv6_end_system1.type=61

Network.IPv4_end_system1.host="H1"
Network.IPv4_end_system1.host="H2"
Network.IPv4_end_system1.host="H3"
Network.IPv4_end_system1.host="H4"
Network.IPv4_end_system1.host="H5"

Network.IPv6_end_system1.host="H1"
Network.IPv6_end_system1.host="H2"
Network.IPv6_end_system1.host="H3"
Network.IPv6_end_system1.host="H4"
Network.IPv6_end_system1.host="H5"

(a)

Network.IPv4_end_system1.H1.stackv4.stack=4
Network.IPv4_end_system1.H2.stackv4.stack=4
Network.IPv4_end_system1.H3.stackv4.stack=4
Network.IPv4_end_system1.H4.stackv4.stack=4
Network.IPv4_end_system1.H5.stackv4.stack=4

Network.IPv6_end_system1.H1.stackv6.stack=6
Network.IPv6_end_system1.H2.stackv6.stack=6
Network.IPv6_end_system1.H3.stackv6.stack=6
Network.IPv6_end_system1.H4.stackv6.stack=6
Network.IPv6_end_system1.H5.stackv6.stack=6

(b)

Así pues, lo anterior lo podemos resumir de la siguiente manera:

Ruta de inicio: "*Network.IPv4_end_system1.H1.stackv4*"

Ruta de destino: "*Network.IPv6_end_system1.H5.stackv4*"

Para efectos prácticos, se evitará el volver a explicar este proceso de asignación y se dará automáticamente la ruta de inicio y la de destino.

Cuando se inicia la simulación, una serie de eventos son impresos en la interfaz gráfica de OMNeT++: Tkenv. Estos eventos denotan el proceso de simulación. Conforme se vaya explicando la simulación de los modelos, se anexará en un recuadro este código de salida así como la función en C++ a la que pertenece esta salida (los códigos detallados se encuentran en el en el disco adjunto).

1. **Inicio de simulación:**

Se origina la transmisión de un datagrama en un *host (ver Figura 5.2*):

```
Route to initialize: Network.IPv4_end_system1.H1.stackv4
IP version: 4
Source host: H1: 223.1.1.1
Destination host: H5: 223.1.5.5
`Network.IPv4_end_system1.H1.stackv4'
Actual Module: Dual_Host
Start transmission
```

El datagrama llega al discriminador, encapsula y manda al switch:

```
Network.IPv4_end_system1.H1.ipd'
Discriminator...
Datagram coming from IPv4 stack
Encapsulating Datagram into Frame...
```

El switch recibe el datagrama y lo manda fuera de la subred:

```
``Network.IPv4_end_system1.switch.linklLayer[1]'
Handle Message at Link Layer
Packet received
Decapsulating Frame...
`Network.IPv4_end_system1.switch.networkLayer'
Handle Message...
Actual Module: Switch
Accessing Switch
 Message out the switch
```

La Figura 5.1 muestra el contenido del datagrama original creado por el *Host* H1:

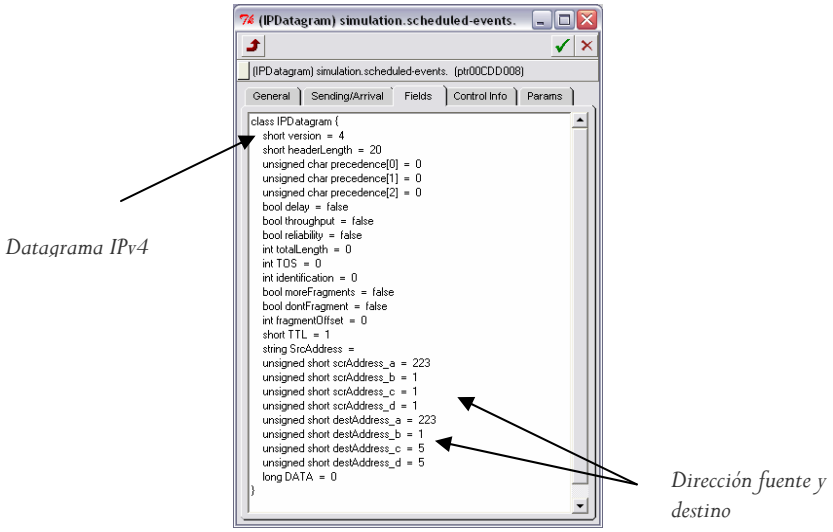


Figura 5.1:Datagrama generado1.

2. Ruteo:

Una vez fuera de la subred, el datagrama llega al primer ruteador, que se va a encargar de buscar el camino para el destino final del datagrama (ver Figura 5.2).

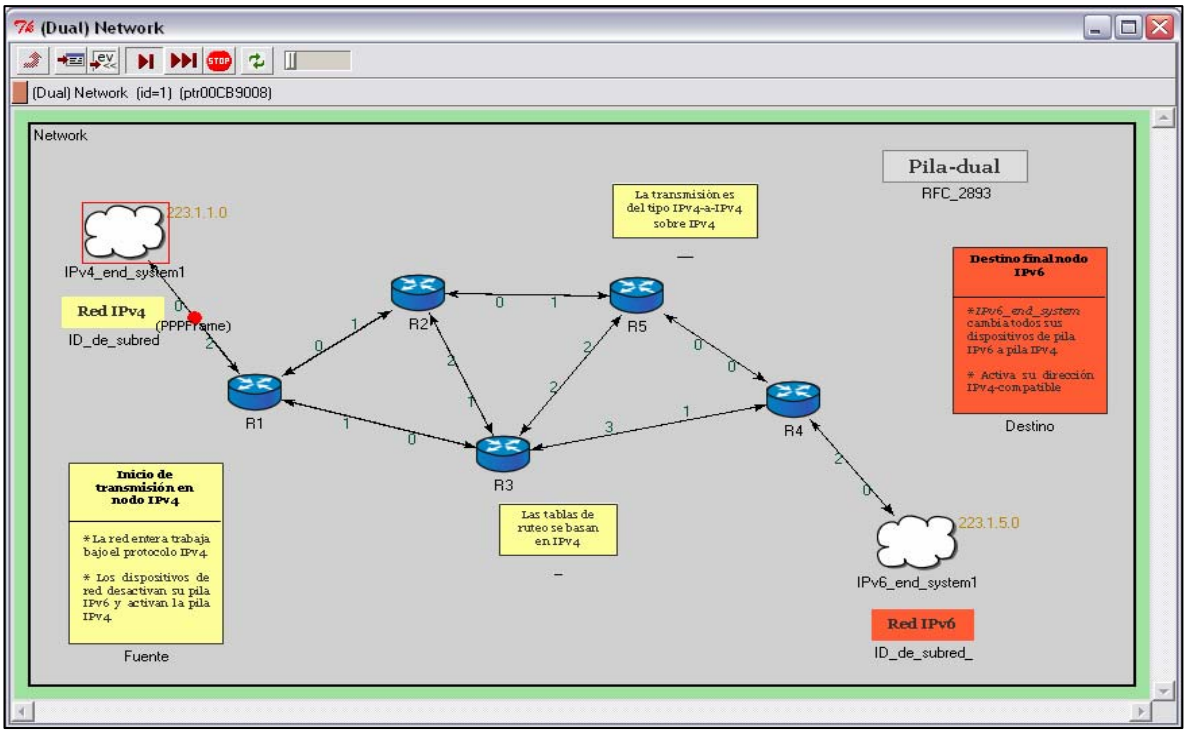


Figura 5.2: Ruteo gráfico.

Entra a ruteador de tipo IPv4/IPv6 (ver Figura 5.3):

Entering IPv4/IPv6 node
Decapsulating Frame...
Handle Message...
Actual Module: Dual_Router
Accessing Router

Revisa tabla de ruteo para buscar la interfaz de salida:

Routing Table

Subnet ID:	*Mask:*	*Gateway:*	*Interfase out:*
223.1.1.0	*255.255.255.0*	*R1*	*2*
223.1.2.0	*255.255.255.0*	*R1*	*0*
223.1.3.0	*255.255.255.0*	*R1*	*1*
223.1.4.0	*255.255.255.0*	*R1*	*2*
223.1.5.0	*255.255.255.0*	*R1*	*1*
223.1.6.0	*255.255.255.0*	*R1*	*0*
223.1.7.0	*255.255.255.0*	*R1*	*0*
223.1.8.0	*255.255.255.0*	*R1*	*0*

Después de hacer una operación lógica AND entre dirección destino y la mascara de red, determina la subred a la que pertenece la dirección y elige la interfaz de salida.

Routing Table
Router Name: R1
Routing table found...
Subnet ID generated: 223.1.5.0
RouterName: R1
Interface out:1

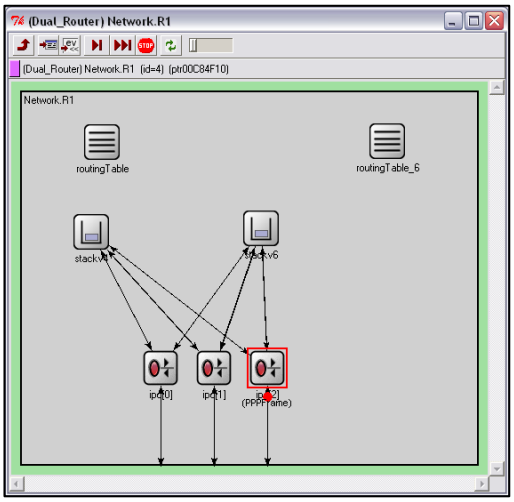


Figura 5.3: Ruteador de pila doble.

El proceso de ruteo se repite en cada ruteador al que se llega. Se busca en las tablas el ID de subred y la interfaz de salida y se manda. La Figura 5.4 muestra el camino para llegar al nodo destino (ver disco ajunto para ver simulación completa).

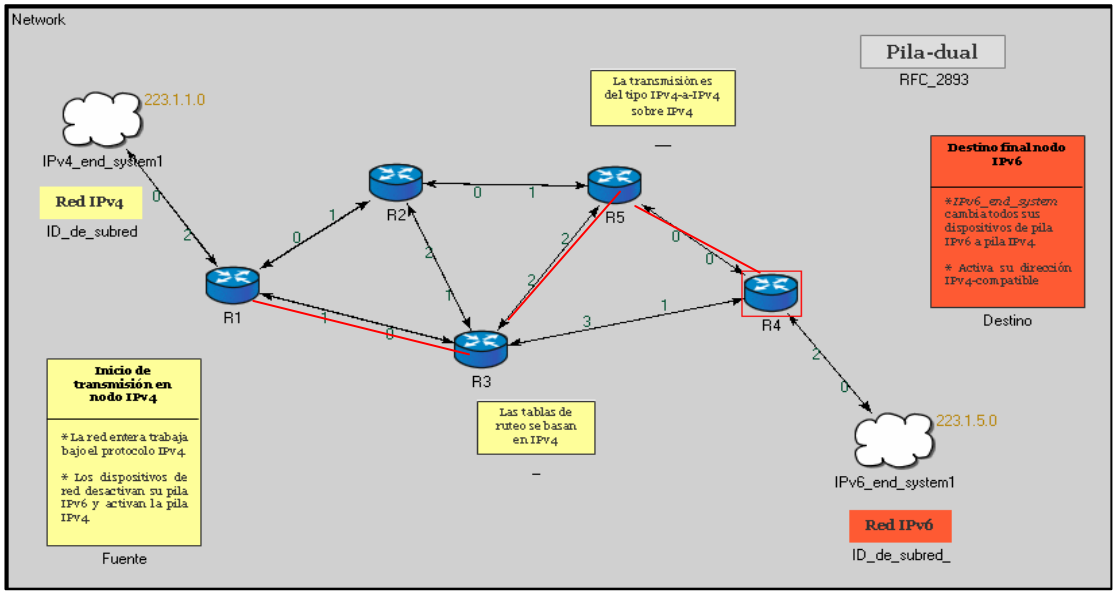


Figura 5.4: Ruteo a través de red IPv4 (R1-R3-R5-R4).

Debido a que este protocolo es de pila-dual, podemos observar que al ser de tipo 4 la versión de datagrama que viaja a través de la red, ésta a la vez activa sus pilas de tipo versión 4 para dar soporte a este tipo de datagrama. Con esto se puede decir que la red es de tipo IPv4.

3. Fin de simulación:

Una vez que el datagrama fue enrutado hacia la dirección probablemente correcta (puede haber alguna falta de actualización de tablas de ruteo) éste entra a la subred en la que se encuentra la ruta destino del datagrama.Entra a la subred. El switch se encarga de decidir por que interfaz transmitir el datagrama.

Discriminator...
Datagram coming from IPv4 stack
Encapsulating Datagram into Frame...
Handle Message at Link Layer
Packet received
Decapsulating Frame...
Handle Message...
Actual Module: Switch
Accessing Switch
Message comes from a subnet
Host name: Network.IPv6_end_system1.switch.networkLayer: 223.1.5.5
Port: 4

Al llegar el datagrama al *host*, este se desencapsula en el discriminador y se decide qué pila activar para procesar el paquete. En esta simulación, se hace el supuesto que esta subred a la que entra es de tipo IPv6, ya que tiene activada su pila IPv6, pero al detectar un paquete de versión diferente a su pila activa, inmediatamente hace cambio de ésta.

```
Handle Message at Link Layer
Datagram coming from Network Layer
Encapsulating Frame into Datagram...
Network.IPv6_end_system1.H5.ipd'
Discriminator...
Entering IPv4/IPv6 node
Decapsulating Frame...
Handle Message...
```

Al llegar el datagrama, se comprueba la dirección destino con la actual del nodo:

```
Actual Module: Dual_Host
Accessing Host IPv4...
Host IP: Network.IPv6_end_system1.H5.stackv4 223.1.5.5
Accesing host:
Arrived
Datagram has arrived sucessfully
```

Segunda simulación

Modelo: Pila-dual (*Dual-stack*).

Características de inicio:

Tabla 5.3: Parámetros de inicio para simulación 2.

Parámetro	Valor	Descripción
Network.type	61	Nodo de inicio
Network.host	"H3"	*Host* orígen
Network.stack	6	Tipo de pila
Network.type_dest	41	Nodo destino
Network.host_dest	"H2"	*Host* destino

Ruta de inicio: "*Network.IPv6_end_system1.H3.stackv6*"

Ruta de destino: "*Network.IPv4_end_system1.H2.stackv6*"

1. Inicio de simulación:

Se origina la transmisión de un datagrama (ver figura 5.5) en un *host:*

```
Route to initialize: Network.IPv6_end_system1.H3.stackv6
IP version: 6
Source host: H3: 0:0:0:0:0:0:223.1:5.3
Destination host: H2: 0:0:0:0:0:0:223.1:1.2
** Event #0.  T=0.0000000  ( 0.00s).  Module #79
`Network.IPv6_end_system1.H3.stackv6'
Handle Message...
Actual Node: Dual_Host
Start transmission
```

El datagrama llega al discriminador, encapsula y manda al switch:

```
Discriminator...
Datagram coming from IPv6  stack
Encapsulating Datagram into Frame...
```

El switch procesa el datagrama y lo manda fuera de la subred:

```
Handle Message at Link Layer
Packet received
Decapsulating Frame...
Network.IPv6_end_system1.switch.networkLayer'
Handle Message...
Actual Module: Switch
Accessing Switch
 Message out the switch
```

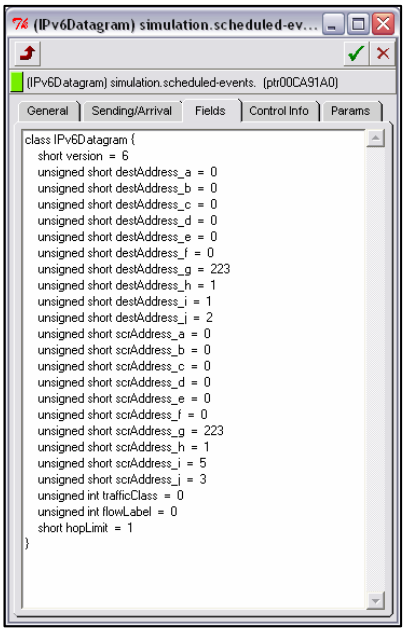


Figura 5.5: Datagrama generado en simulación 2. Tipo Ipv6.

2. Ruteo:

Una vez fuera de la subred, el datagrama llega al primer ruteador, que se va a encargar de buscar el camino para el destino final del datagrama (ver Figura 5.6).

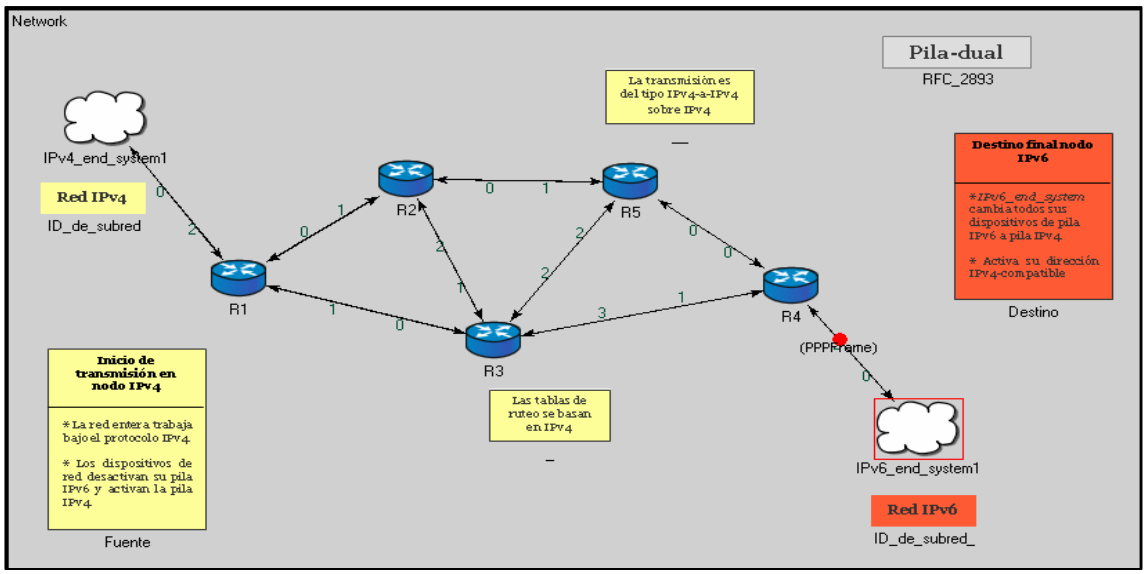


Figura 5.6: Ruteo gráfico tipo IPv6.

El datagrama entra a un ruteador de tipo IPv4/IPv6 (ver Figura 5.7):

```
Entering IPv4/IPv6 node
Decapsulating Frame...
Network.R4.stackv6'
Handle Message...
Actual Node: Dual_Router
Accessing Router
```

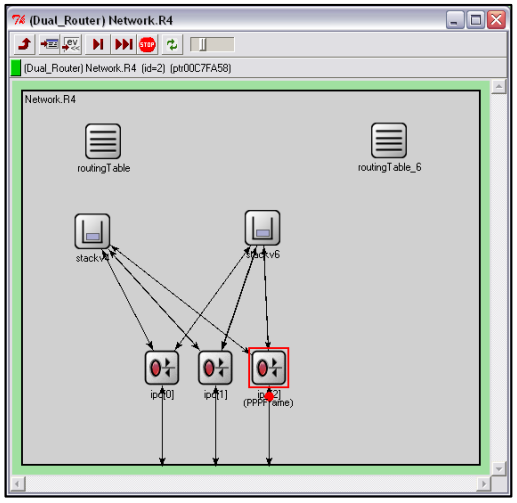


Figura 5.7: Ruteador de pila doble.

Revisa tabla de ruteo para buscar la interfaz de salida:

Routing Table

Subnet ID:	*Mask:*	*Gateway:*	*Interfase out:*
::223.1.1.0	*255.255.255.0*	*R4*	*0*
::223.1.2.0	*255.255.255.0*	*R4*	*2*
::223.1.3.0	*255.255.255.0*	*R4*	*2*
::223.1.4.0	*255.255.255.0*	*R4*	*2*
::223.1.5.0	*255.255.255.0*	*R4*	*2*
::223.1.6.0	*255.255.255.0*	*R4*	*2*
::223.1.7.0	*255.255.255.0*	*R4*	*2*
::223.1.8.0	*255.255.255.0*	*R4*	*2*

Se supone que IPv6 ya no realiza operaciones lógicas debido a que su formato de 128 bits utiliza prefijos en vez de máscaras. Debido a que esto no está muy bien desarrollado, se decidió utilizar el mismo de ruteo para IPv6 dado que las direcciones utilizadas son compatibles (ver capítulo 2). Interfaz de salida:

Routing Table
Table found...
interfase: 0

El proceso de ruteo se vuelve a repetir en cada ruteador al que se llega. La Figura 5.8 muestra el camino para llegar el nodo destino (ver disco ajunto para ver simulación completa).

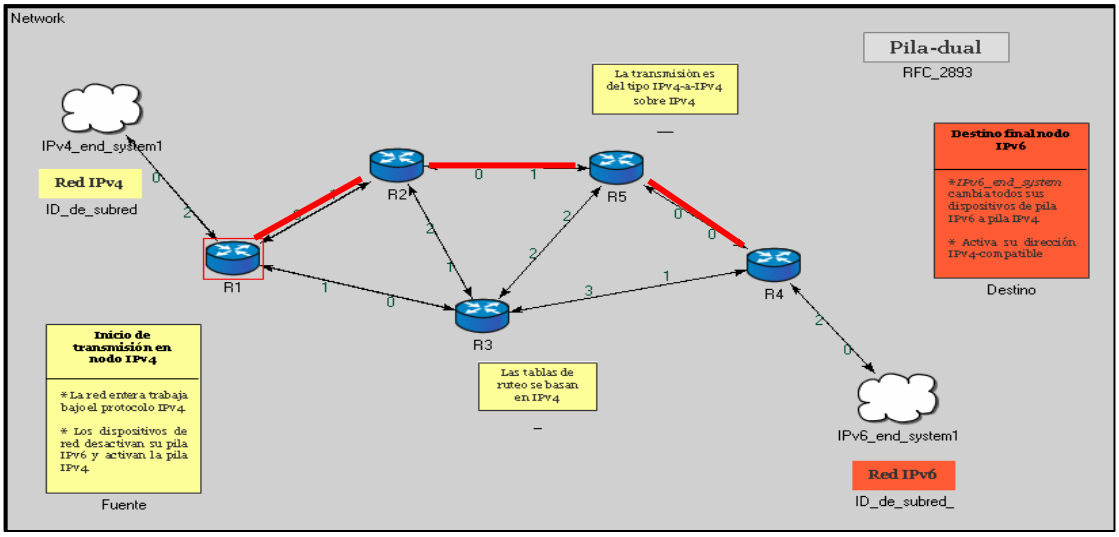


Figura 5.8: Ruteo a través de red IPv6 (R4-R5-R2-R1).

Podemos observar que al ser de tipo 6 la versión del datagrama ésta hace que el resto de la red active su pila a la versión 6.

3. Fin de simulación:

Una vez que el datagrama encuentra la ruta destino, sigue el camino marcado. Entra a la subred. El switch se encarga de decidir por qué interfaz transmitir el datagrama.

```
Packet received
Decapsulating Frame...
Network.IPv4_end_system1.switch.networkLayer'
Handle Message...
Actual Module: Switch
Accessing Switch
Message comes from a subnet
Accesing host:
Port: 1
Network.IPv4_end_system1.switch.linklLayer[1]'
Handle Message at Link Layer
Datagram coming from Network Layer
Encapsulating Frame into Datagram...
```

Se selecciona qué pila activar y se envía el datagrama:

```
Discriminator...
Entering IPv4/IPv6 node
Decapsulating Frame...
`Network.IPv4_end_system1.H2.stackv6'
Handle Message...
```

Al llegar el datagrama, se comprueba la dirección destino con la actual del nodo:

```
Actual Node: Dual_Host
Accessing Host IPv6...
ParentModule: Network.IPv4_end_system1.H2.stackv6
Host name: Network.IPv4_end_system1.H2.stackv6 0:0:0:0:0:0:223.1:1.2
Accesing host version 6:
Arrived
Datagram has arrived sucessfully
```

5.2 Modelo de SIIT o de Traducción

Este modelo se basa en la traducción de encabezados. Éstos se realizan con base a lo contenido en el RFC 2765. La estructura general de la Traducción consta de dos *end_systems* uno IPv6 y otro IPv4, separados por una red tipo IPv4. Para que la comunicación entre estos extremos se lleve a cabo, se

hace uso de un traductor que se encarga de cambiar la versión origen del datagrama por el del destino.

Para que se pueda apreciar los dos tipos de traducción, se mostrarán dos simulaciones del mismo modelo. El primero de ellos (que corresponde a la tercera simulación del proyecto) será del tipo IPv6/Ipv4. El segundo es de tipo IPv4/IPv6.

Igualmente, para efectos prácticos de simulación cada *end_system* sólo representa a un usuario debido a que se interesa mostrar más el proceso de migración que el comportamiento de una red. Por lo que al elegirse el valor de parámetro en 6 con nodo inicio, automáticamente se inferirá que el destino es de tipo 4 y viceversa.

Tercera simulación

Modelo: SIIT o Traducción.

Características de inicio:

Tabla 5.4: Parámetros de inicio para simulación 1.

Parámetro	Valor	Descripción
Network.IPv6_end_system.type	6	Nodo de inicio version 6

Ruta de inicio: "*IPv6_end_system*"

Ruta de destino: "*IPv4_end_system*"

1. Inicio de simulación:

Se origina la transmisión en un *end_system IPv6*:

```
Type:6
Network.IPv6_end_system.networkLayer'
IPv6_end_system
Start transmission
Network.IPv6_end_system.linkLayer'
Datagram coming from Network Layer version 6
```

2. Traducción IPv6/IPv4:

Se envía el datagrama fuera de la subred y llega hasta un traductor de borde de red que se encarga de hacer la traducción (ver Figura 5.9):

Datagram coming from IPv6 network
Decapsulating frame into IPv6 Datagram
Sending IPv6 Datagram...
Network.translator.queue_6'
Datagram coming from discriminator
Performing translation...

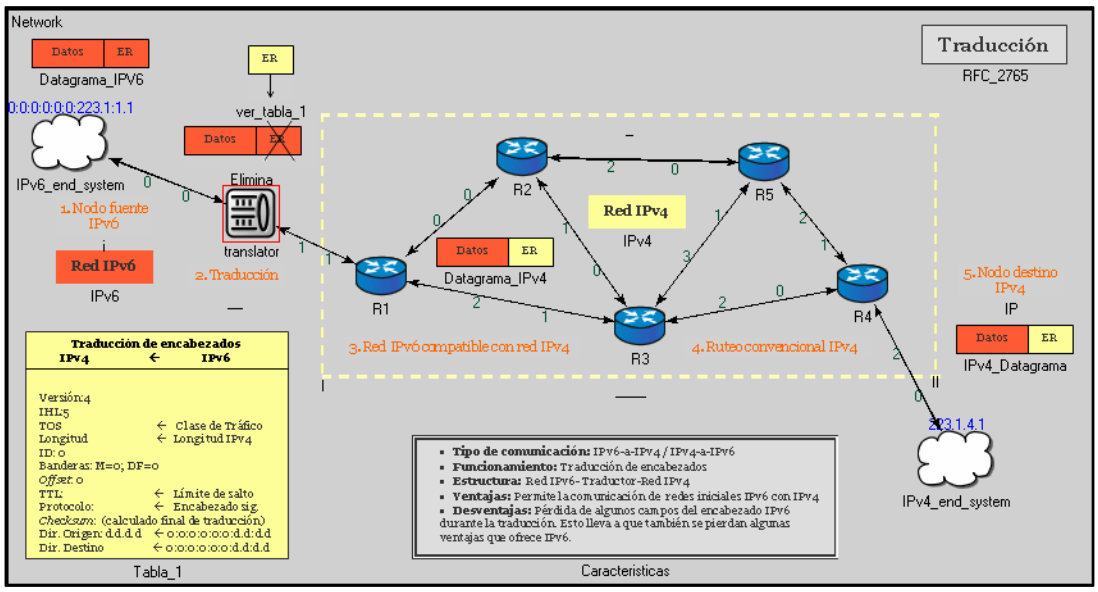


Figura 5.9: Traductor de borde de red.

El proceso que se lleva a cabo dentro del traductor es el siguiente: primero se entra a un discriminador que desencapsula el paquete y compara la versión de éste (6 en este caso). Seguidamente lo manda a la pila que le corresponde según la versión (*queue_6*), realiza la traducción y se manda fuera del traductor como un datagrama de versión 4.

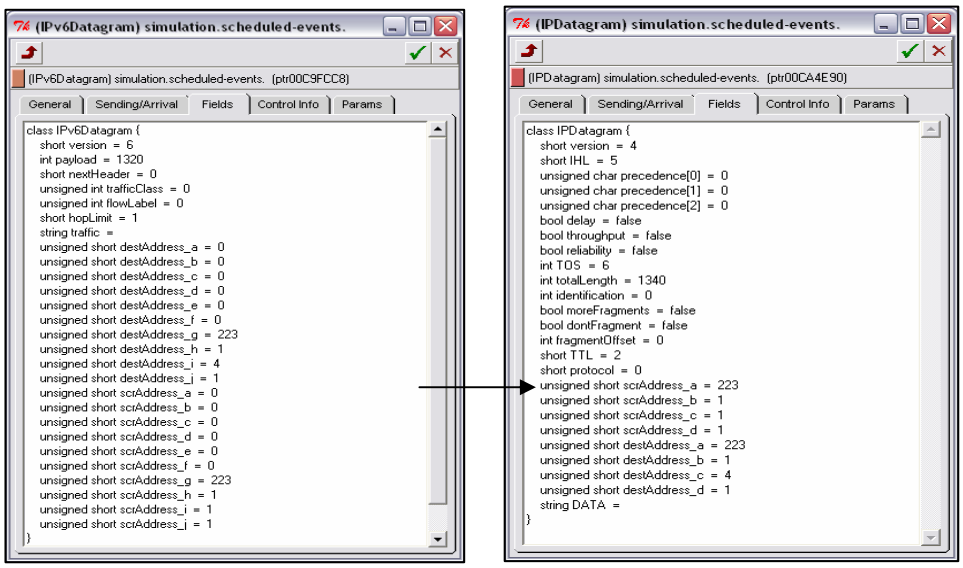


Figura 5.10: Traducción de encabezados IPv6 / IPv4.

Como se mencionó en el capítulo 3, en el proceso de traducción de IPv6 a IPv4 se crea un encabezado IPv4 que sustituye al del datagrama original IPv6. Algunos campos se copian en el nuevo encabezado, mientras que otros se eliminan, RFC 2765 (ver Figura 5.10).

Version:6	*-----> Version: 4*
	-----> Internet Header Length [IHL]: 5
Traffic Class: 6	*-----> Type of Service [TOS]: 6*
Payload Length: 1320	*-----> Total Length: 1340*
	-----> Identification: 0
	-----> Flags: MF: 0DF: 1
	-----> Fragment Offset:0
Hop Limit: 2	*-----> Time to Live [TTL]: 2*
Next Header:0	*-----> Protocol: 0*
	-----> Header Checksum:
Source Address: ::223.1:1.1	*----->223.1.1.1*
Destination Address: ::223.1:4.1	*----->223.1.4.1*

3. Ruteo de IPv6 sobre IPv4:

Una vez hecha la traducción el datagrama está listo para transportarse dentro de una red IPv4. Este proceso de ruteo IPv4 es el que se ha venido manejando mediante el uso de tablas e interfaces de salida.

Búsqueda de la interfaz de salida en la tabla de ruteo. Debido a que el siguiente proceso es el mismo, se omiten estas salidas ya que solo corresponden a otras tablas de ruteo e interfaces de salida.

Module Name: R1
Destination Address: 223.1.4.1
Source Address: 223.1.1.1
Accesing Router:

RoutingTable

Subnet ID:	*Mask:*	*Gateway:*	*Interfase out:*
223.1.4.0	*255.255.255.0*	*R1*	*2*
223.1.1.0	*255.255.255.0*	*R1*	*1*

Generating subnet ID...
Destination IP: 223.1.4.1
Mask: 255.255.255.0.
Subnet ID generated: 223.1.4.0
Interfase out: 2
Datagram coming from Network Layer
Sending Frame out
Encapsulating IPv4 Datagram into Frame

El datagrama es ruteado a través de toda la red hasta que llega a su destino final (ver figura 5.11).

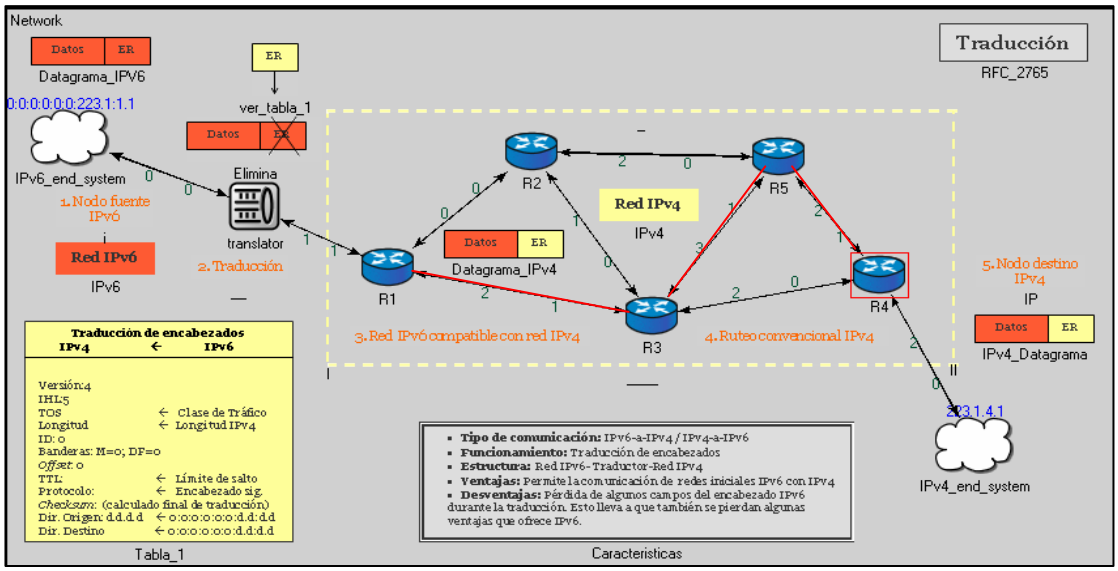


Figura 5.11: Ruteo convencional IPv4 para simulación 3.

4. Fin de simulación:

Al llegar a su destino final, el datagrama se procesa en el *host*. Como se puede observar, el mecanismo de Traducción permitió que una red IPv6 pudiera entablar comunicación con una red IPv4.

```
Sending IPv4 Datagram
Network.IPv4_end_system.networkLayer'
Module Name: IPv4_end_system
Destination Address: 223.1.4.1
Source Address: 223.1.1.1
Accesing Host:IPv4_end_system
ParentModule: IPv4_end_system
Host name: IPv4_end_system 223.1.4.1
Accesing host:
Arrived
Datagram has arrived sucessfully
```

Cuarta simulación

Modelo: SIIT o Traducción.

Características de inicio:

Tabla 5.5: Parámetros de inicio para simulación 4.

Parámetro	Valor	Descripción
Network.IPv4_end_system.type	4	Nodo de inicio version 4

Ruta de inicio: "*IPv4_end_system*"

Ruta de destino: "*IPv6_end_system*"

1. Inicio de simulación:

Se origina la transmisión en un *end_system IPv4*:

```
Type:4
Network.IPv4_end_system.networkLayer'
Module Name: IPv4_end_system
Destination Address: 223.1.1.1
Source Address: 223.1.4.1
Starts transmission
Network.IPv4_end_system.linkLayer'
Datagram coming from Network Layer
Sending Frame out
Encapsulating IPv4 Datagram into Frame
```

2. Ruteo IPv4:

Una vez originada la transmisión, este datagrama IPv4 se enruta a través de la red en búsqueda de su destino final. Este ruteo es el que ya se conoce para IPv4 (ver Figura 5.12).

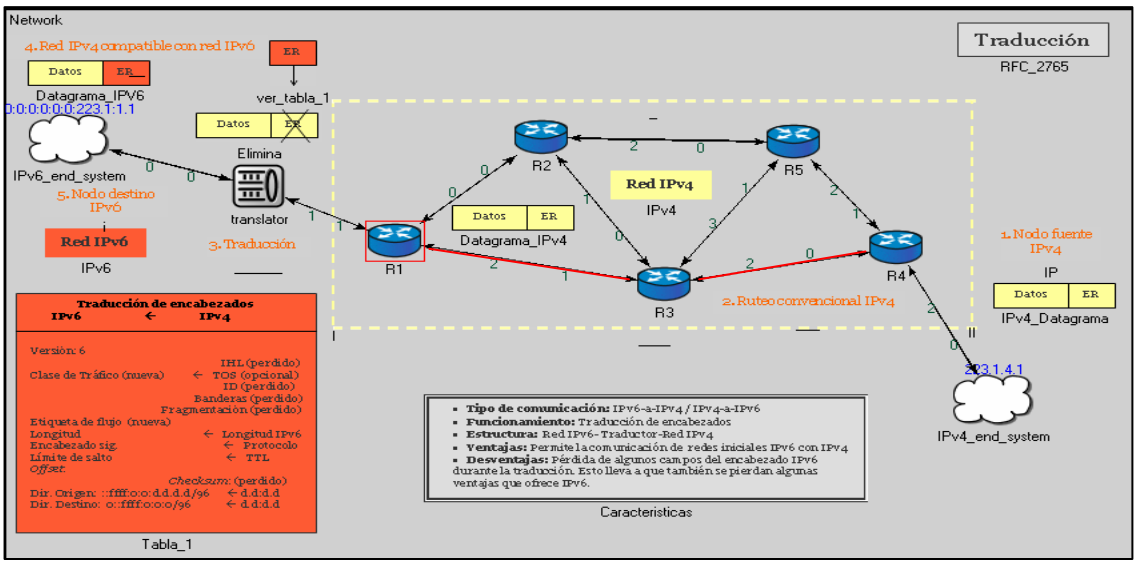


Figura 5.12: Ruteo convencional IPv4 para simulación 4.

```
Sending IPv4 Datagram
Network.R4.networkLayer'
Module Name: R4
Destination Address: 223.1.1.1
Source Address: 223.1.4.1

Accesing Router:
223.1.1.0    255.255.255.0      R4          0
223.1.4.0    255.255.255.0      R4          2

Generating subnet ID...
Destination IP: 223.1.1.1
Mask: 255.255.255.0.
Subnet ID generated: 223.1.1.0
Interfase out: 0
```

3. Traducción IPv4/IPv6:

Cuando el datagrama se topa con un traductor, éste hace la traducción de IP4 a IPv6 (ver Figura 5.13). Esta traducción se realiza creando un nuevo encabezado de red, en este caso de IPv6, y reemplazándolo por el encabezado IPv4 mediante la encapsulamiento de los encabezados.

Datagram coming from Network Layer
Sending Frame out
Encapsulating IPv4 Datagram into Frame
Network.translator.ipd'
Datagram coming from IPv4 network
Decapsulating frame into IPv4 Datagram
Sending IPv4 Datagram

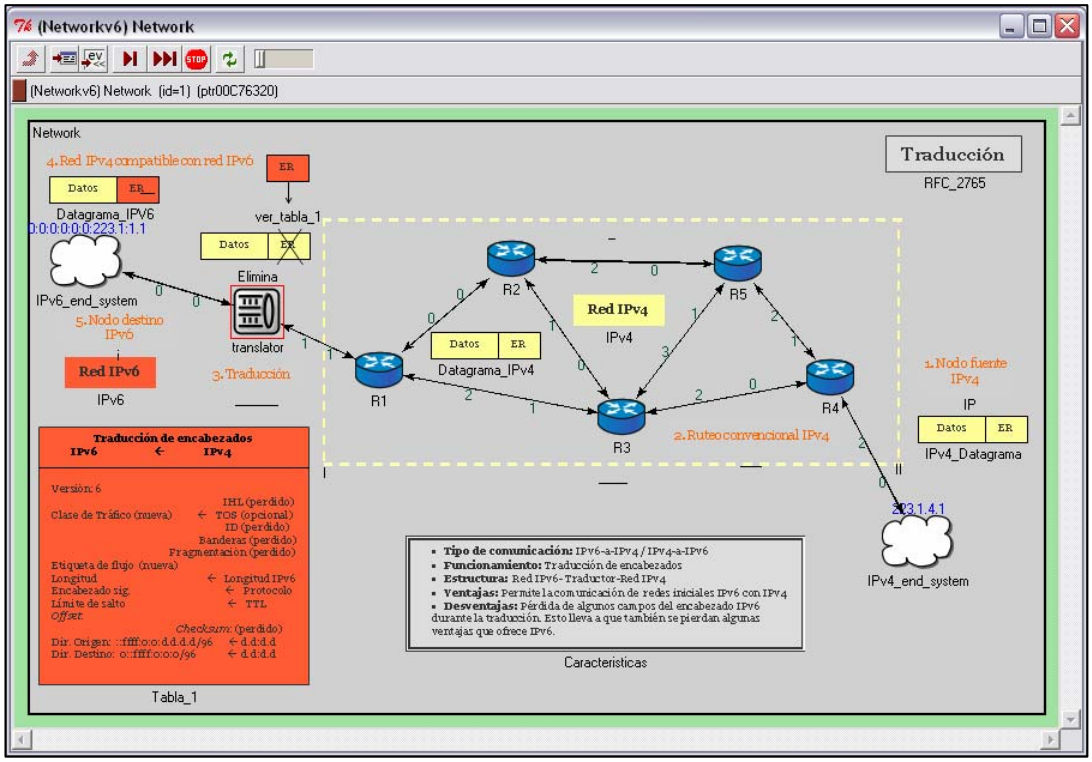


Figura 5.13: Traductor de borde de red.

Algunos campos se copian en el nuevo encabezado, mientras que otros se eliminan, RFC 2765 (ver Figura 5.14).

Performing translation...

Version:4	*-----> Version: 6*
Internet Header Length [IHL]:	*-----> LOST in translation*
Type of Service [TOS]: 0	*-----> Traffic Class: 0*
Total Length:	*-----> Payload Length:*
Identification:	*-----> LOST in translation*
Flags:	*-----> LOST in translation*
Fragment Offset:	*-----> LOST in translation*
Time to Live [TTL]: 4	*-----> Hop Limit: 4*
Protocol:	*-----> Next Header:*
Header Checksum:	*-----> LOST in translation*
Source Address: 223.1.4.1	*----->0:0:0:0:0:0:223.1:4.1*
Destination Address: 223.1.1.1	*----->0:0:0:0:0:0:223.1:1.1*

Figura 5.14: Traducción de encabezados IPv6/IPv4.

4. Fin de simulación:

Al llegar a su destino final, el datagrama se procesa en el *host*. En esta ocasión, la comunicación permitida fue de una red IPv6 a una red IPv4.

```
IPv6_end_system
entering Ipv6 network
ParentModule: IPv6_end_system
Accesing host version 6:
IPv6_end_system000000223111Arrived
Datagram has arrived sucessfully
```

5.3 *Tunneling*

Este último bloque de simulaciones pertenece al mecanismo de traducción que se basa en la creación de "túneles" para lograr comunicación IPv6-a-IPv6 sobre IPv4. *Tunneling* basa su estructura general en dos nodos que funcionan como la entrada y salida, respectivamente, del túnel creado. Estos nodos encapsulan y desencapsulan los paquetes a enviar. Funciona de cierta manera como la traducción ya que crea un encabezado nuevo, la diferencia reside en que este mecanismo no elimina el encabezado original.

Las simulaciones correspondientes pertenecen a dos tipos de *tunneling* llevados a la práctica. El primero de ellos es conocido como *tunneling* configurado y se basa en tablas para la creación de los túneles. El segundo, *tunneling* automático, basa su funcionamiento y creación de túneles en la dirección destino contenida en el encabezado de red (direcciones compatibles).

Quinta simulación

Modelo: *Tunneling* configurado.

Características de inicio: Ninguna.

Ruta de inicio: "*Network.end_system2.IPv6_end_system1.H1.networkLayer*"

Ruta de destino: "*Network.end_system2.IPv6_end_system2.H1.netwrokLayer"*

1. Inicio de simulación:

Se origina la transmisión en un *end_system IPv6*. Primero sale del *host* a un switch que lo manda fuera de la subred.

```
`Network.end_system2.IPv6_end_system1.H1.networkLayer'
dest: 0.0.0.0.223.3.1.2
(Host_6)H1
Start transmission
`Network.end_system2.IPv6_end_system1.H1.linkLayer'
Datagram coming from Network Layer
`Network.end_system2.IPv6_end_system1.switch.linklLayer[1]'
Packet received
`Network.end_system2.IPv6_end_system1.switch.networkLayer'
 entering Switch
`Network.end_system2.IPv6_end_system1.switch.linklLayer[3]'
Datagram coming from Network Layer
```

2. Creación del túnel:

El datagrama llega a un ruteador de borde que está en el límite de IPv6 con IPv4. Este dispositivo es de pila doble y se encarga de realizar el encapsulado del datagrama IPv6 en uno IPv4, para enviarlo con seguridad a una red IPv4. De esta manera, el ruteador comprende el inicio del túnel que se va a asignar al proceso.

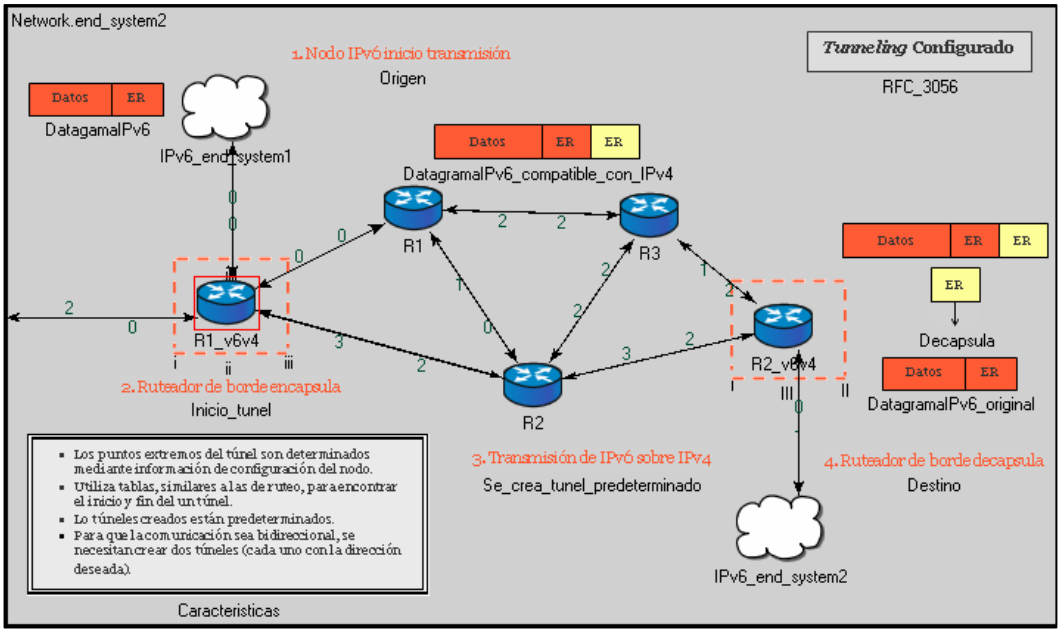


Figura 5.15: Ruteador de borde que realiza el encapsulado de IPv6.

Debido a que este mecanismo determina el inicio y fin del túnel mediante el uso de tablas, la dirección actual del nodo corresponde el inicio del túnel. A esta dirección de inicio de túnel le corresponde un fin que se establece por medio de una tabla. Las entradas de esta tabla son modificadas manualmente, no existe algún tipo de generador de túneles

```
Packet received
Entering Discriminator
Version: 6
des: 0.0.0.223.3.1.2
(Router_6to4)R1_v6v4

Creating tunnel…
Encapsulates into IPv4
(RouterIP)routerIP

      Node IP        Entry node   Exit node
0.0.0.223.3.1.2   223 1 1 1    223 1 3 1 0
0.0.0.223.3.1.3   223 1 1 1    223 1 3 1 0
0.0.0.223.3.1.4   223 1 1 1    223 1 3 1 0
0.0.0.223.3.2.2   223 1 3 1    223 1 1 1 0
0.0.0.223.3.2.3   223 1 3 1    223 1 1 1 0
0.0.0.223.3.2.4   223 1 3 1    223 1 1 1 0

Router Name: R1_v6v4
      Node IP        Entry node   Exit node
0.0.0.223.3.1.2   223 1 1 1    223 1 3 1 0
```

Dentro de este ruteador de borde, se realiza el encapsulado del datagrama IPv6 dentro de uno IPv4 (ver Figura 5.16).

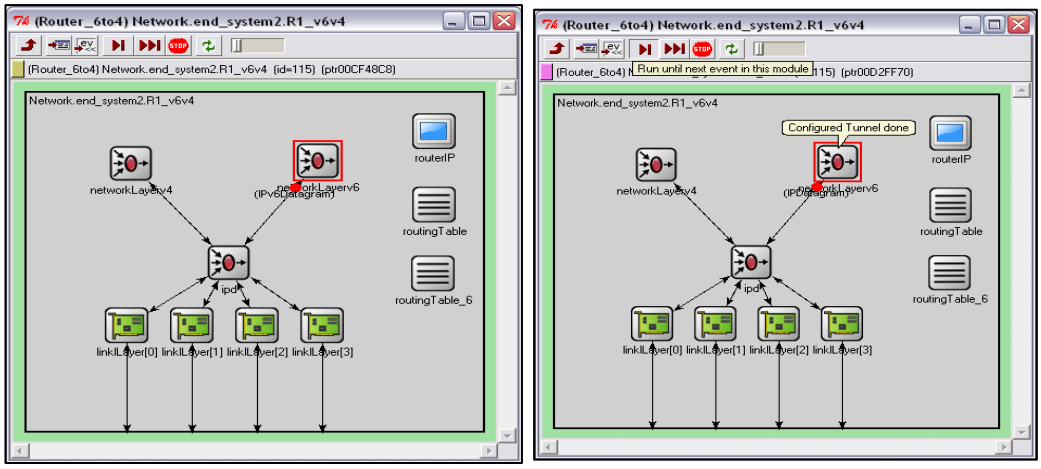


Figura 5.16: Ruteador de borde que realiza el encapsulado de IPv6/IPv4. Inicio del túnel.

En este proceso de encapsulado, se crea un encabezado nuevo de red que es agregado al datagrama original [RFC 2893] y se considera que el túnel es creado. La Figura 5.17 muestra el datagrama IPv6 original contenido dentro de un datagrama IPv4.

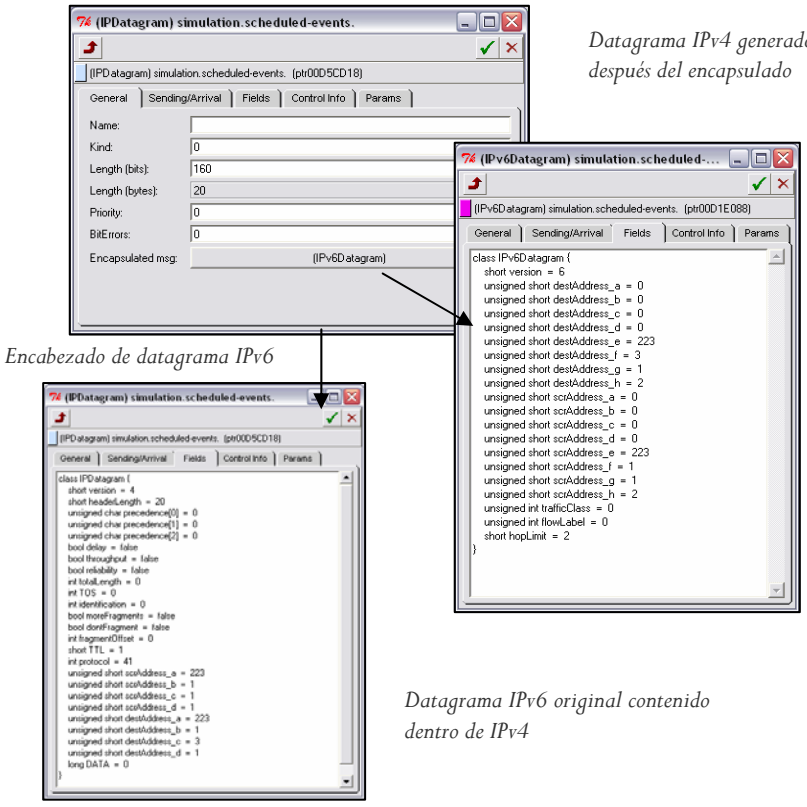


Figura 5.17: Datagramas encapsulados.

3. Ruteo del datagrama:

Una vez "convertido" el datagrama IPv6 a uno IPv4 para permitir que éste se maneje sin problemas sobre una red IPv4, el resto del proceso de ruteo es el convencional hecho para IPv4.

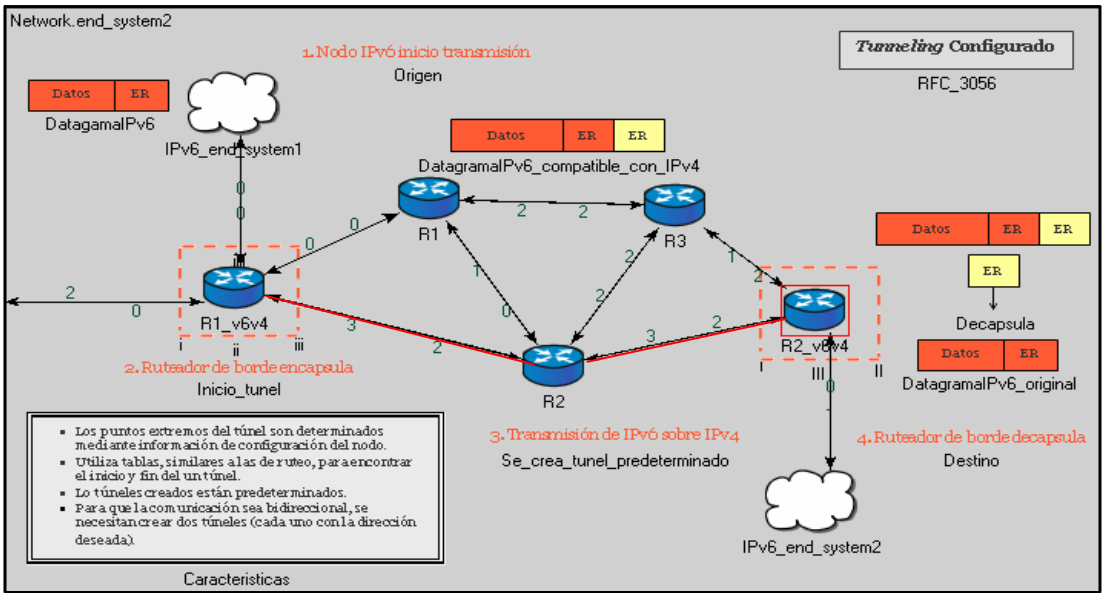


Figura 5.18: Ruteo de datagrama IPv4 en una red IPv4, Rv6v2-R2-R2v6v4.

```
Datagram coming from Network Layer
Packet received
(Router)R2:
Start tunnel: 223.1.1.1

Routing Table:
        223.1.3.0 R1 2
        223.1.1.0 R2 0
        223.1.3.0 R2 3
        223.1.1.1 R3 1
        223.1.3.0 R3 0
Finding corresponding routing table
Routing table found
Subnet ID generated: 223.1.3.0
Interface out:3
`Network.end_system2.R2.linklLayer[3]'
Datagram coming from Network Layer
```

4. Desencapsulado:

Una vez enviado el paquete a través del túnel establecido, el nodo final de túnel, es encargado de desencapsularlo para obtener el datagrama original (ver figura 5.17).

Packet received
Entering Discriminator
Version: 4
(Router_6to4)R2_v6v4
Header conversion: IPv4 into IPv6
Datagram coming from IPv4 stack
`Network.end_system2.R2_v6v4.linklLayer[2]'
Datagram coming from Network Layer

5. Fin de simulación:

Una vez que el datagrama llega a la subred IPv6, entra al switch y éste lo manda a su destino correspondiente.

Packet received
dest: 0.0.0.0.223.3.1.2
entering Switch
Accesing host:
Port: 0
Datagram coming from Network Layer
Packet received
entering IPv6 network
Host name: H1
Arrived
Datagram has arrived sucessfully
*** Calling finish() methods of modules*

Sexta simulación

Modelo: *Tunneling* automático.

Características de inicio: Ninguna.

Ruta de inicio: "*Network.end_system1.IPv6_end_system1.networkLayer*"

Ruta de destino: "*Network.end_system2.IPv6_end_system2.H1.netwrokLr*"

1. Inicio de simulación:

Se origina la transmisión en un *end_system IPv6 (ver Figura 5.19*). Sale del *host* y se manda fuera de la subred.

```
dest: 0.0.0.223.1.7.5
 (Host_6)IPv6_end_system1
Start transmission
`Network.end_system1.IPv6_end_system1.linkLayer'
Datagram coming from Network Layer
```

2. Creación del túnel:

El datagrama llega a un ruteador de borde que realiza el encapsulado del datagrama IPv6 en uno IPv4, para enviarlo a través de una red IPv4.

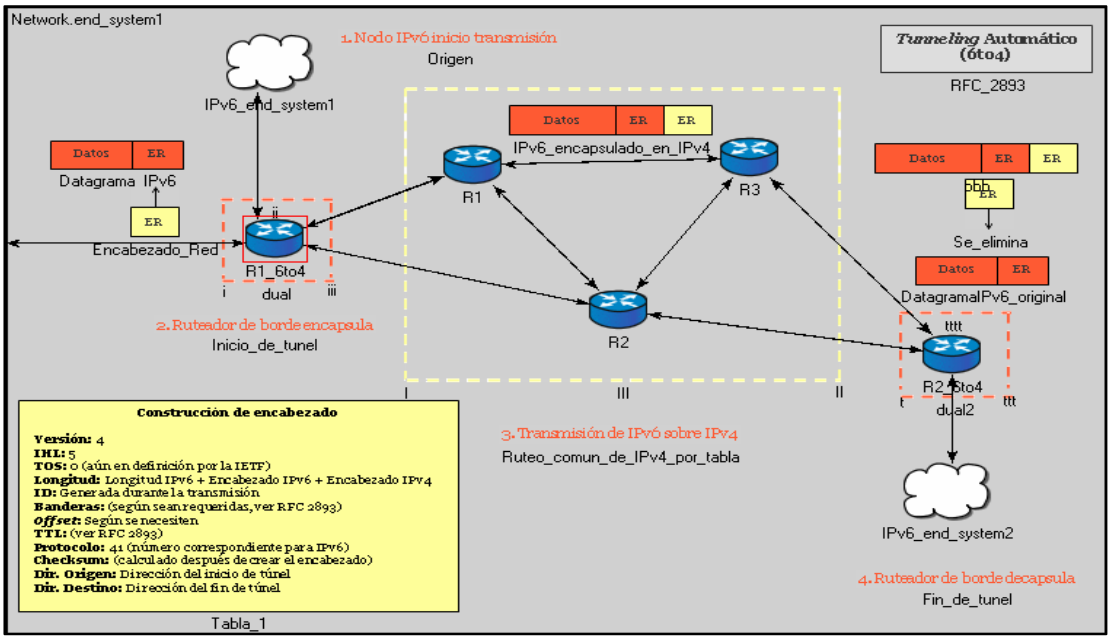


Figura 5.19: Ruteador de borde que realiza el encapsulado de IPv6 simulación 6.

En este mecanismo, se determina el fin del túnel mediante la dirección compatible IPv4 contenida en la dirección destino IPv6.

```
Packet received
Entering Discriminator
Version: 6
dest: 0.0.0.223.1.7.5
Header conversion: IPv6 into IPv4
encapsulates a IPv4
(RouterIP)routerIP
Router NameR1_6to4
Datagram coming from IPv6  stack
(Automatic)end_system1
Datagram coming from Network Layer
```

Dentro de este ruteador de borde, se realiza el encapsulado del datagrama IPv6 dentro de uno IPv4 (ver Figura 5.20).

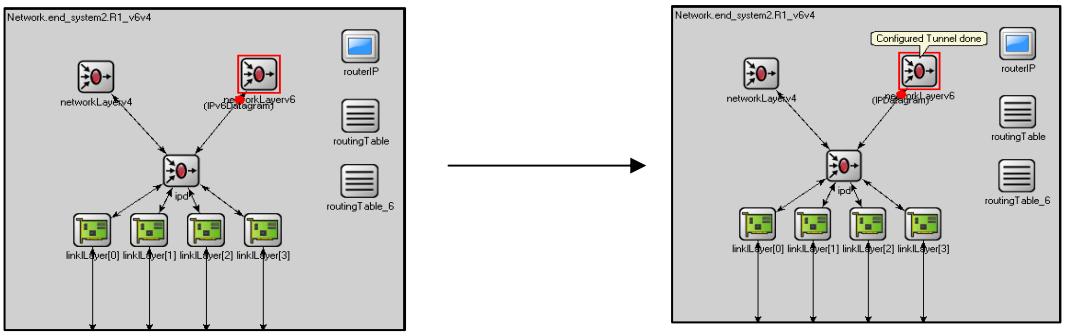
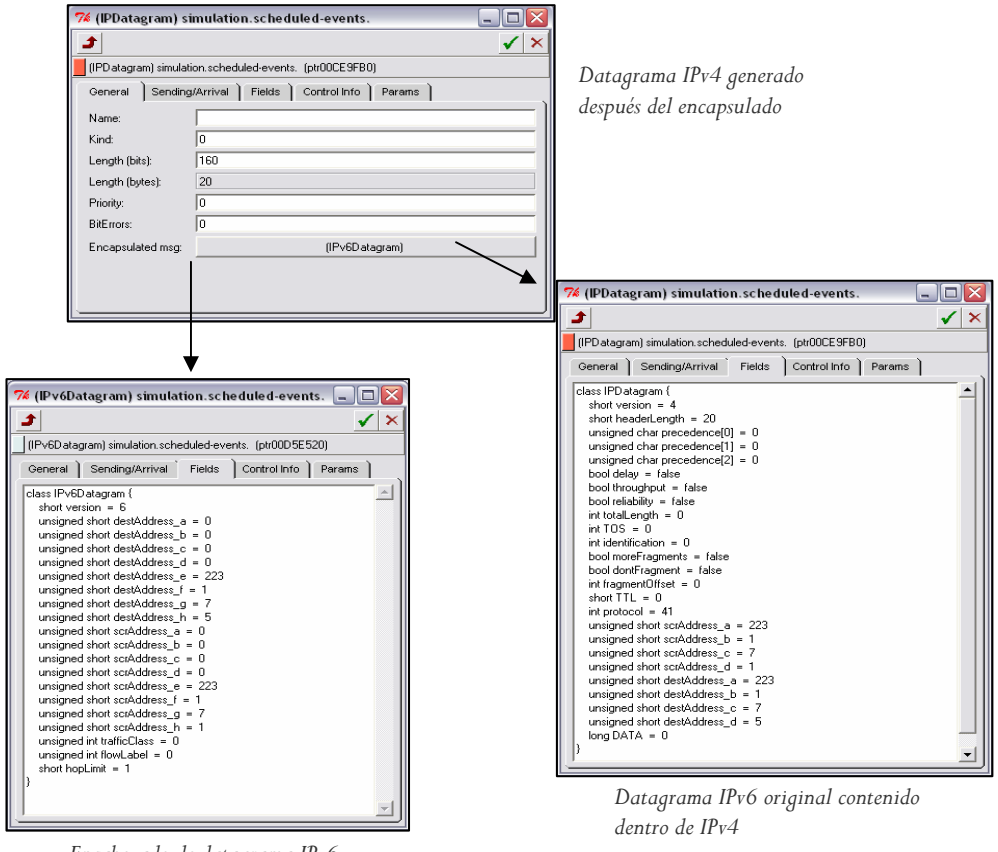


Figura 5.20: Ruteador de borde que realiza el encapsulado de IPv6/IPv4. Inicio del túnel para simulación 6.

En este proceso de encapsulado, se crea un encabezado nuevo de red que es agregado al datagrama original [RFC 2893] y se considera que el túnel es creado. La Figura 5.21 muestra el datagrama IPv6 original contenido dentro de un datagrama IPv4.



Datagrama IPv4 generado después del encapsulado

Encabezado de datagrama IPv6.

Datagrama IPv6 original contenido dentro de IPv4

Figura 5.21: Datagrama IPv6 encapsulado en IPv4, simulación 6.

3. Ruteo del datagrama:

En este proceso, se revisa la dirección actual del nodo, y se envía a través de la interfaz de salida que tiene asignada, lo que representa el siguiente nodo del túnel a seguir (el fin de túnel depende de la dirección IPv6 original).

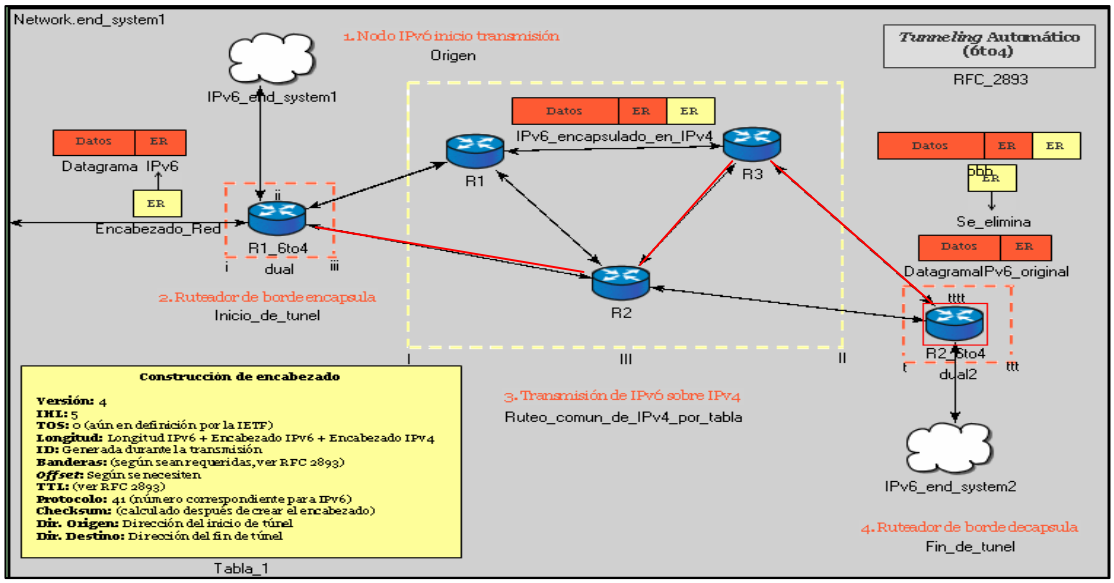


Figura 5.22: Ruteo de datagrama IPv4 en una red IPv4, Rv6v2-R2-R2v6v4.

Tabla contenida en el ruteador. Muestra los extremos del túnel (*Start* y *End* túnel), y el salto siguiente a partir del nodo actual. De cierta forma, esta ruta preestablecida por el ruteador, genera el túnel por que cual se va conducir el paquete.

Packet received
(Router)R1
Scr: 223.1.7.1
Dest: 223.1.7.5
G Start tunnel End tunnel..Actual…..Netx……Iout
R1 | 223.1.7.1 | 223.1.7.5 | 223.1.7.2 | 223.1.7.3 | 1
R2 | 223.1.7.1 | 223.1.7.5 | 223.1.7.3 | 223.1.7.4 | 1
R3 | 223.1.7.1 | 223.1.7.5 | 223.1.7.4 | 223.1.7.5 | 1
R1 | 223.1.7.5 | 223.1.7.1 | 223.1.7.2 | 223.1.7.1 | 0
R2 | 223.1.7.5 | 223.1.7.1 | 223.1.7.3 | 223.1.7.2 | 0
R3 | 223.1.7.5 | 223.1.7.1 | 223.1.7.4 | 233.1.7.3 | 2
This router: R1
End node : 223.1.7.5
Entry: node 223.1.7.1
Address: 223.1.7.5
Actual node: 223.1.7.2
Next node in tunnel: 223.1.7.3
Interfase out: 1
Datagram coming from Network Layer

4. Desencapsulado:

En el nodo final de túnel, éste se encarga de desencapsular el paquete (ver Figura 5.23)

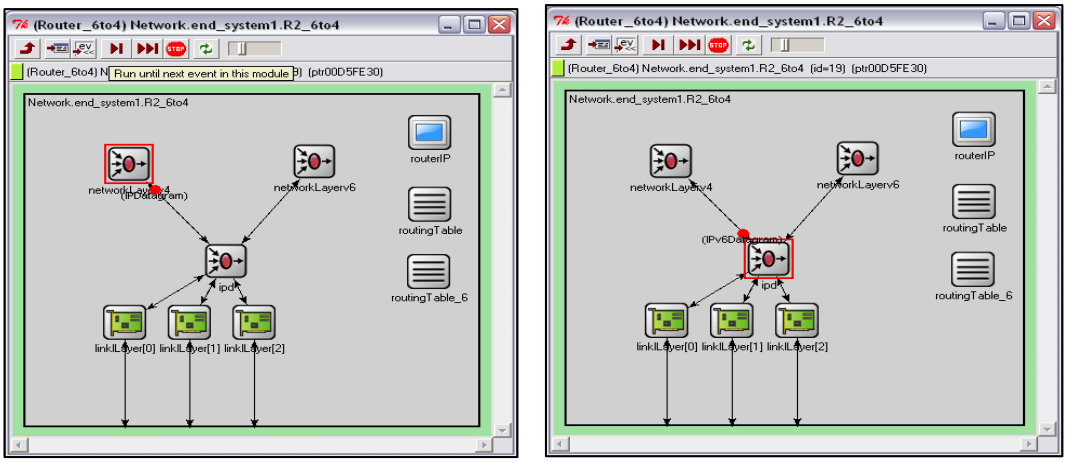


Figura 5.23: Proceso de desencapsulado de IPv4/IPv6 se obtiene el datagrama original, simulación 6.

```
Packet received
Entering Discriminator
Version: 4
(Router_6to4)R2_6to4
Dest: 223.1.7.1
Router_6to4
Header conversion: IPv4 into IPv6
```

5. Fin de simulación:

Una vez que el datagrama llega a la subred IPv6, entra al switch y éste lo manda a su destino correspondiente.

```
Datagram coming from IPv4 stack
name: R2_6to4
Datagram coming from Network Layer
Packet received
dest: 0.0.0.0.223.1.7.5
entering IPv6 network
Host name: IPv6_end_system2
Arrived
Datagram has arrived sucessfully
** Calling finish() methods of modules
```

5.4 Resultados y Evaluación

Durante la simulación de cada uno de los modelos de traducción se hizo uso de una variable, en el código C++, necesaria para utilizarla como comparador entre ellos. Ésta variable, de nombre: *time*, es utilizada para contar los eventos ocurridos dentro de ciertos módulos, como el que pertenece a la capa de red (*networkLayer*).

Como podemos apreciar en el siguiente bloque de figuras, se muestran las arquitecturas internas de cada uno de los dos dispositivos más importantes de los modelos propuestos. La Figura 5.24 corresponde a los dispositivos del modelo de pila-doble. Sus componentes básicos son: una pila IPv4 y otra tipo IPv6, y un módulo discriminador que hace las veces también de la capa de enlace.

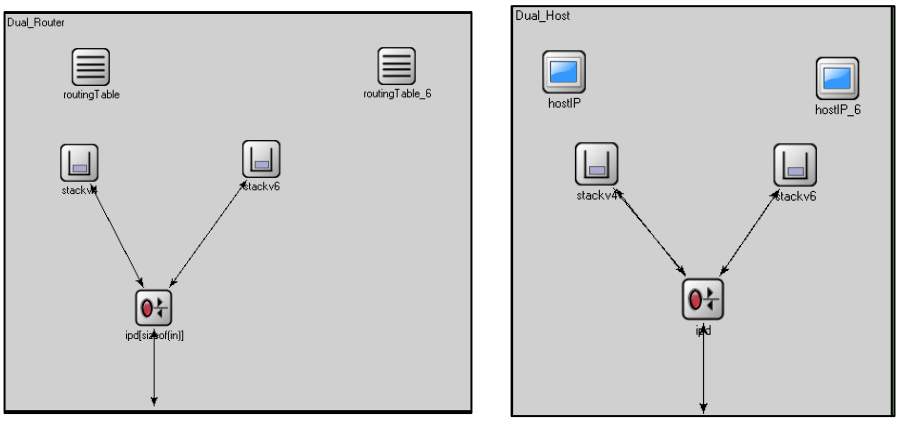


Figura 5.24: Arquitectura de un ruteador (a) y un host (b) para el mecanismo de Pila-dual.

Al mismo tiempo tenemos en la Figura 5.25 y 5.26, se tiene la arquitectura de estos mismos dispositivos en el modelado del mecanismo de Traducción y *Tunneling*.

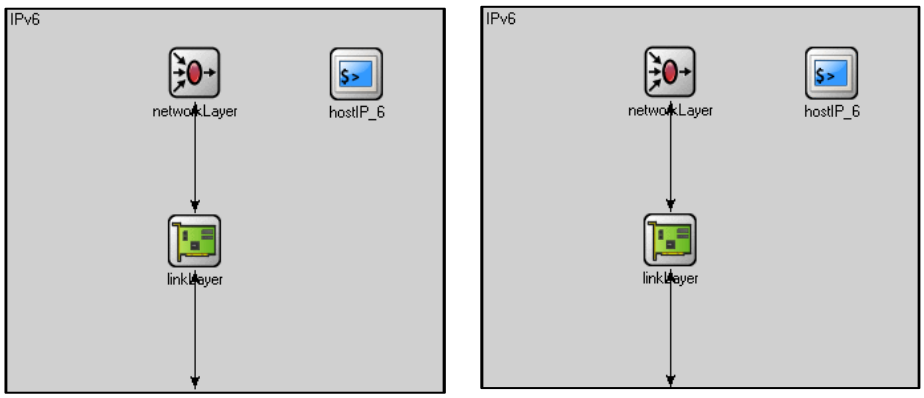


Figura 5.25: Arquitectura de un ruteador (a) y un host (b) para el mecanismo de Traducción.

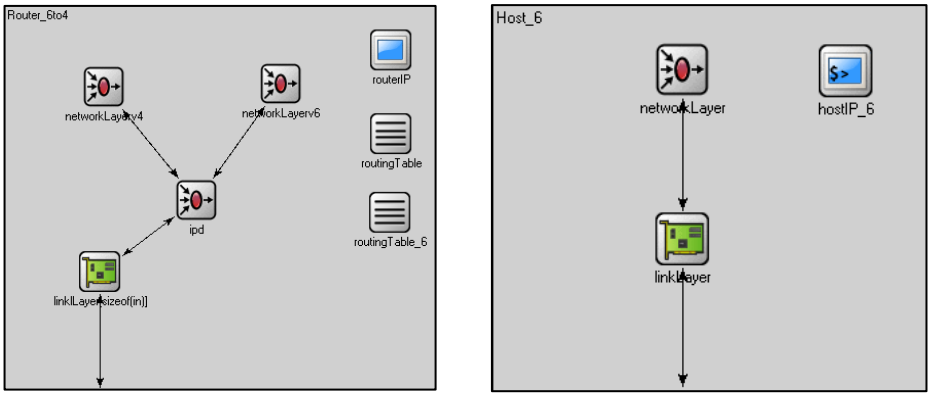


Figura 5.26: Arquitectura de un ruteador (a) y un host (b) para el mecanismo de Tunneling.

Utilizando la herramienta de Scalars de OMNeT++, se obtuvieron los valores del contador de eventos *time*, mencionado anteriormente. Estos valores se filtraron en 6 categorías diferentes con la finalidad de poder hacer comparaciones entre dispositivos o alguna de las capas de éstos. La Tabla 5.6 muestra estas categorías.

La columna de Caso refiere al número de simulación hecha. Como se puede ver, se omitió la sexta simulación ya que las dos primeras simulaciones de la Tabla 5.6 pertenecen al mismo mecanismo de Pila-doble, por lo que está presente sólo un mecanismo de este tipo. Lo mismo sucede con el mecanismo de traducción; por esto, se decidió omitir una de las simulaciones hechas de *Tunneling* ya que éstos representaban dos variantes del mismo mecanismo.

La columna de *Host* representa los eventos de simulación en un *host* a nivel capa red (uno activo) igualmente sucede para *Router*, ya que representa a un ruteador activo a nivel capa de red. Por otro lado, la columna *networkLayer* representa los eventos ocurridos a nivel capa de red de todos los dispositivos del sistema. *Router** representa a los eventos ocurridos en todos lo ruteadores de la red a nivel de capa de red. Por último *Network* refiere a todos los eventos generados en la red.

Si hacemos una comparación de las estructuras de los ruteadores y *host* en cada uno de los modelos simulados, podemos observar qué composición define el comportamiento del número de eventos generados.

En el primer caso, Pila-dual, se tiene que la arquitectura interna de estos dispositivos parece no ser tan compleja al incluir dos pilas de ruteo, ya que también comparten el uso del resto de las capas.

El punto a notar en ese primer caso es el uso periódico del discriminador en capa de enlace; ya que, se utiliza constantemente para diferenciar entre protocolos. Este discriminador o diferenciador está incluido en cada uno del resto de los dispositivos utilizados en la topología, lo que eleva el número de eventos generados por el proceso.

Si observamos en la tabla 5.6, el mayor número de eventos generados dentro de un ruteador es el del caso de la Pila-dual. Esto se debe a lo mencionado previamente acerca del uso del diferenciador en cada dispositivo. Así pues, este elevado número de eventos en el ruteador provoca que el número de eventos general de la red también aumente.

En cuanto al número de eventos obtenidos por el mecanismo de Traducción, se tiene la impresión de que es uno de los mecanismos más eficientes. Se puede observar que la arquitectura de sus dispositivos es simple y el factor que agrega "complejidad" a este mecanismo es el traductor puesto de por medio. Si se considera que este traductor sólo se utiliza una sola vez en el proceso de traducción, IPv6/IPv4 o IPv4/IPv6, podemos observar que es más efectivo realizar esto en vez de utilizar un discriminador para dispositivo de pila doble, ya que este traductor opera fuera de los dispositivos de red y se presenta una sola vez.

Tabla 5.6: Parámetros de simulación obtenidos del contador de eventos time.

Caso	Host	Router	Router*	Network Layer	Network*
I Primera simulación (Pila-dual-pila-v4)	3	39	156	6	170
II Segunda simulación (Pila-dual-v6)	3	14	56	6	70
III Tercera simulación (Traducción-IPv6/IPv4)	1	1	4	6	23
IV Cuarta simulación (Traducción-IPv4/IPv6)	1	1	3	5	9
V Quinta simulación (*Tunneling*)	6	11	66	18	78

Es posible pensar que uno de los mecanismos de transición que más eventos genera durante la migración, es el de traducción. Esto debido a que primero se tiene que crear un nuevo encabezado de red, hacer la traducción de éste, encapsularlo dentro del datagrama original y enviarlo. Aunque *tunneling* también utiliza este principio al usar dos ruteadores para encapsular y desencapsular, este mecanismo parece situarse en el medio de las comparaciones ya que sino es el más efectivo de ellos,

tampoco demuestra todo lo contrario. Así pues, el número de eventos generados en la capa de red se puede justificar diciendo que no es sólo debido a la creación de un nuevo encabezado sino también al proceso de encapsulado y desencapsulado lo que lleva a que este conteo de eventos no sea muy bajo pero tampoco muy elevado.

Observando el número de eventos en capa de red de cada uno de los modelos podemos concluir lo siguiente:

Tabla 5.7: Tabla comparativa de eficiencia entre mecanismos de traducción

Mecanismo	Ventaja	Desventaja
Pila-dual	El número de eventos en capa de red es relativamente pequeño, comparado con otro mecanismo como *tunneling*.	El número de eventos en ruteo es elevado, lo que eleva el número total de eventos de la red, al utilizar su mecanismo de pila doble en cada uno de los nodos de la red.
Traducción	Su número de eventos en el ruteador y el *host,* es relativamente pequeño. Igualmente lo es el número de eventos en capa de red.	Generalmente sus desventajas se pueden observar en cuanto a eficiencia y seguridad, ya que alguno de los efectos de esta traducción es la pérdida de ciertos campos importantes para alguno de los protocolos.
Tunneling	Es una aproximación intermedia entre la traducción y la pila-dual, ya que su número de eventos por dispositivo es moderado, lo mismo que en cuanto a tiempo de procesamiento en capa de red.	El número de eventos por ruteo es ligeramente elevado.

5.5 Resumen

El cambio tan esperado de IPv4 a IPv6 se ha comenzado. Para que este cambio se lleve a cabo de manera completa, es necesario que la migración sea gradual. Lo anterior es posible gracias a la presencia de mecanismos de transición propuestos por organismos como la IETF, que se encargan de que la convivencia entre ambos protocolos sea afable.

Estos mecanismos se pueden agrupar de tres maneras: Pila-dual, Traducción y *Tunnneling*. Ya que cada uno de estos depende de una infraestructura en particular, no es tan común encontrar estos mecanismos trabajando en la misma red; sin embargo, se pretende que esta situación cambie y que funcionen mutuamente como mecanismos de apoyo.

La simulación de estos mecanismos mediante OMNeT++, permite aclarar el panorama acerca de cómo éstos trabajan e interactúan con el protocolo IPv6. De esta forma se puede dar una aproximación de qué método es más eficiente y mostrar, mediante su simulación, las ventajas y desventajas de su implementación.