Estudio y análisis del funcionamiento de ANTop sobre IPv6

Tesis de grado

Ingeniería Electrónica

Abril 2012

Tesista: Matías Campolo

Tutor: Dr. José Ignacio Alvarez-Hamelin

Índice general

Índice de figuras

Capítulo 1

Introducción

En este capítulo se presenta una reseña general de este trabajo de Tesis. Aquí se pueden encontrar dos secciones, la primera de ellas discute la motivación que da lugar a este trabajo, mientras que la segunda describe brevemente el modo en que se presentan los temas tratados a lo largo del documento.

1.1. Objetivos

El presente trabajo de Tesis responde a la motivación de embeber el protocolo ANTop *(Adjacent Network Topologies)* [1], [2] y [3], en un sistema operativo Linux basado en la distribución Debian, en forma flexible y eficiente. Las versiones actuales trabajan sobre un simulador llamado QUENAS *(QUeued Event Network Automatic Simulator)*[1], y por lo tanto no permiten evaluar el desempeño de este diseño en el campo real de las redes inalámbricas. El diseño de dicho simulador comenzó en la Tesis de Alejandro Marcu [1], y continuó en la de Gaston Tejía [2].

Para lograr cumplir con la meta, se estudiarán diferentes herramientas que permitan interactuar con el sistema operativo y sus funciones de red, buscando embeber allí las características que ANTop presenta para redes inalámbricas ad-hoc.

Una vez definido el esquema más eficiente de mapeo de las funciones de ANTop al sistema operativo, se realizarán pruebas de funcionamiento y se analizarán los resultados.

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

1.2. Organización del documento

En el capítulo 2 se presentan algunos protocolos de ruteo reactivo para redes *ad hoc*. Los discutidos son aquellos de mayor difusión en la bibliografía y que cuentan con un *RFC*. También se mencionan brevemente las características de ANTop.

El capítulo 3 presenta las herramientas disponibles para interactuar con las funciones de red del sistema operativo. Allí se analizan los beneficios y desventajas de cada una de ellas al tiempo que comienza a definirse el esquema final de implementación.

Más adelante, en el capítulo 4 se discute en profundidad el uso de las herramientas elegidas en el capítulo anterior. Se analizan las dificultades surgidas en el proceso de integrar las funcionalidades de red de ANTop al sistema operativo, de un modo transparente para las aplicaciones y para el usuario. Se presentan, además, las soluciones adoptadas para superar dichas dificultades.

Por último, el capítulo 5 presenta las pruebas realizadas para corroborar que el funcionamiento real de la implementación obtenida se corresponda con el comportamiento proyectado a partir del diseño del protocolo ANTop.

Capítulo 2

Estado del arte

En este capítulo se presentan los protocolos de ruteo para redes ad-hoc más tratados en la bibliografía, y que son respaldados por un *RFC*. Para cada uno de ellos se esboza la teoría de funcionamiento.

2.1. AODV (Ad-hoc On-demand Distance Vector)

Este es uno de los protocolos de ruteo reactivo más desarrollados e implementados actualmente. El diseño del mismo brinda la posibilidad de rutear tanto paquetes unicast como multicast. Como su nombre lo indica, está basado en un vector de distancias [4].

Como se explicará en breve, una de sus características principales es el uso de números de secuencia para sus rutas, los cual permite mantener una topología libre de bucles. Para su funcionamiento, AODV define tres tipos de paquetes, *Route Requests* (RREQs), *Route Replies* (RREPs), y *Route Errors* (RERRs). Estos paquetes se envían a través de UDP.

A diferencia de ANTop, al momento de obtener una ruta que no se tenía previamente definida en la tabla de ruteo, AODV envía un paquete en modo broadcast. Como se explica en 2.3, ANTop utiliza el mismo paquete que dispara la búsqueda de ruta, para encontrarla. Dicho paquete podrá ser, típicamente, un TCP SYN.

2.1.1. Descubrimiento de rutas

Aquí los nodos que pueden comunicarse son considerados vecinos. Un nodo lleva registro de sus vecinos, escuchando mensajes de tipo *HELLO*, que son enviados en modo broadcast por otros nodos cada cierto intervalo de tiempo. Si al cabo de cierto tiempo no se reciben paquetes *HELLO* de un vecino, se considera que el nodo abandono la red.

Cuando un nodo necesita comunicarse con otro que no es su vecino, envía en modo broadcast un mensaje RREQ. Dicho mensaje contiene algunos campos clave como ser el origen, el destino, el tiempo de vida del mismo y el número de secuencia que sirve como un identificador universal.

En la topología de la figura 2.1, el nodo 1 quiere enviar un mensaje al nodo 3. Sus vecinos son 2 y 4. Al no poder comunicarse directamente con 3, envía un RREQ, que es escuchado por 2 y 4. Estos nodos tienen dos opciones; si tienen una ruta al destino, o lo son, puede enviar un mensaje RREP al nodo 1, de otra forma retransmitirán, también en modo broadcast, el RREQ a sus vecinos. De esa forma el mensaje continua viajando hasta que se agota el campo de vida del mismo *(lifespan)*. Si luego de un cierto tiempo, el nodo 1 no recibe una respuesta, volverá a transmitir el mensaje, pero esta vez con un lifespan mayor, y un ID distinto. En este caso en particular, el nodo 2 tiene una ruta al 3, con lo cual devuelve un RREP al nodo 1, mientras que el nodo 4 retransmite el RREQ, ya que no tiene ninguna ruta al 3. Todos los nodos utilizan el número de secuencia para asegurarse de no reenviar los RREQ.

2.1.2. Mantenimiento de rutas

Los números de secuencia sirven como marcas de tiempo. Les permiten a los nodos tener noción de que tan "fresca" es la información que tienen acerca de la red. Cada vez que un nodo envía cualquier tipo de mensaje, incrementa este valor. A su vez, se guarda el número de secuencia de todos los nodos con los que se habla. Cuando se recibe un número de secuencia mayor al que se tiene guardado, significa que la información que se tiene acerca de ese nodo está desactualizada.

En el siguiente ejemplo, el nodo 1 está reenviando un mensaje RREP al nodo 4. Luego, como el número de secuencia del mensaje es mayor al que tenía guardado para esta ruta, se reemplaza la misma por la contenida en el paquete.

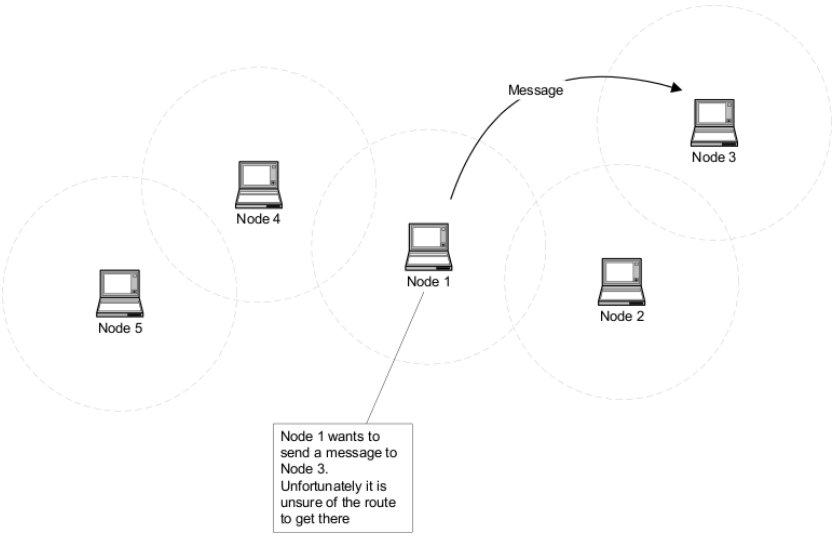


Figura 2.1: Red de computadoras corriendo AODV

El envío de mensajes RERR conforma otro mecanismo de mantenimiento de rutas. La figura 2.3 presenta los tres casos en los que puede darse esta situación.

En el primero de ellos, un nodo recibe un paquete de datos, el cual se supone que debe reenviar. Sin embargo, el mismo no tiene una ruta válida hacia el destino. Esto implica que algún otro nodo tiene una ruta equivocada hacia el destino del paquete.

En el segundo caso, un nodo recibe un RERR que invalida al menos una de sus rutas. Esto causa que el nodo deba enviar RERRs por todos los destinos que el ya no es capaz de alcanzar.

En el tercer caso, un nodo deja de recibir paquetes de *HELLO* de un vecino. Inmediatamente chequea en su tabla de ruteo cuales son las entradas que utilizan dicho vecino como próximo salto y las marca como inválidas. Luego envía un RERR informando acerca de estas rutas.

2.2. DSR (Dinamic Source Routing)

Al igual que AODV, DSR es un protocolo de ruteo reactivo para redes ad-hoc, se define en el RFC4728 [5], y es el resultado de los trabajos de Johnson [6] y Maltz [7]

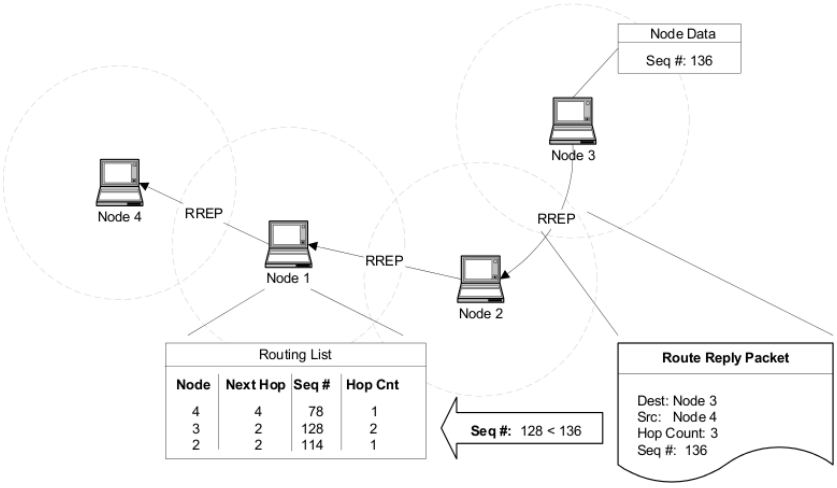


Figura 2.2: Búsqueda de rutas en una red AODV

A grandes rasgos, el funcionamiento del protocolo responde al siguiente mecanismo. Cuando el nodo origen genera un paquete para un cierto destino, el primero coloca una "ruta de origen" en el encabezado del paquete, la cual se conforma por una secuencia de saltos que el paquete deberá seguir. Normalmente el nodo encontrará esta ruta en su tabla *Route Cache*, la cual contiene las rutas previamente halladas. Si la misma no se encuentra allí, se inicia el procedimiento de descubrimiento de ruta.

2.2.1. Descubrimiento de rutas

En el siguiente esquema se supone que el nodo A desea enviar un paquete al nodo B.

En este caso se transmite en primer medida un paquete de tipo *Route Request*, en modo broadcast, el cual es recibido por el nodo B. Cada paquete de este tipo identifica al origen y al destino del proceso de descubrimiento de ruta. También contiene un identificador unívoco del pedido. En este caso su valor es 2. Conforme el mismo transita por la red, se va populando un registro con las direcciones intermedias de los nodos que recibieron el pedido.

Cuando un nodo recibe el *Route Request* para el cual es el destino, devuelve un *Route Reply* al origen. Esta respuesta contiene una copia del registro con las direcciones por las que el pedido transitó. Cuando el origen recibe esta respuesta, guarda la ruta obtenida en su tabla Route Cache, para rutear los paquetes subsiguientes al mismo destino. De otro modo, si el nodo que recibe

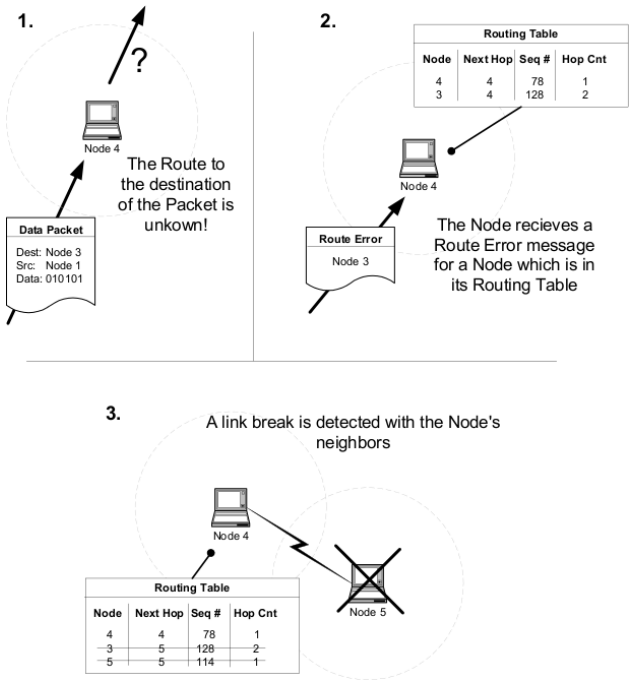


Figura 2.3: Casos de generación de paquetes RERR

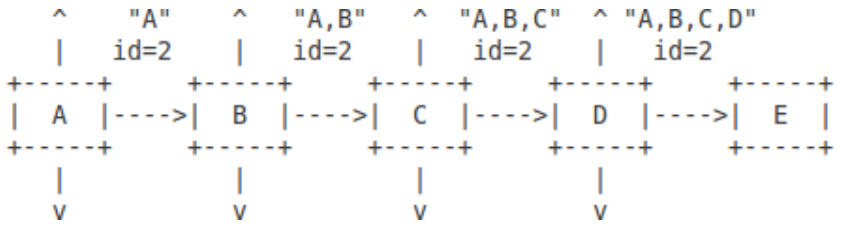


Figura 2.4: Búsqueda de rutas en una red DSR

el paquete ha visto recientemente otro Route Request del mismo origen, con el mismo identificador de pedido y el mismo destino, o si la dirección propia ya está listada en el registro contenido en el paquete, entonces el mismo es descartado. Si ninguno de los dos casos recién mencionados se cumple, entonces se agrega la dirección al registro y el mismo se retransmite en forma de broadcast. En este ejemplo, el nodo B reenvía el paquete en modo broadcast, el cual es recibido por C, quien a su vez hace lo mismo, al igual que D. Finalmente el pedido llega a E, quien con este mismo mecanismo enviará un *Route Reply* al nodo A.

Al iniciar el proceso de descubrimiento de ruta, el nodo origen guarda una copia del paquete original que debe ser ruteado, en un buffer local llamado

Send Buffer. A este paquete se le asocia el tiempo en el que fue guardado, y al agotarse un timer, el mismo se descarta si no se encuentra una ruta.

2.2.2. Mantenimiento de rutas

Cuando se genera o se retransmite un paquete usando una ruta de origen, cada nodo en el camino es responsable de confirmar que el tráfico puede ser ruteado a través del vinculo desde ese nodo al siguiente.

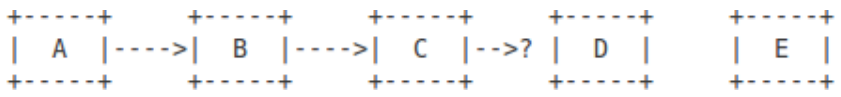


Figura 2.5: Mantenimiento de rutas en una red DSR

En este caso, el nodo A es responsable del vínculo entre A y B, el nodo B es responsable por el vínculo entre B y C, y así sucesivamente. Para cumplir esta función se utiliza un acknowledgement.

En el caso de que al cabo de un tiempo no se reciba un ACK, entonces el nodo considera que el vínculo al siguiente salto dejó de ser válido. Cuando esto sucede, el nodo debe quitar las rutas que contengan al vínculo caído de la tabla Route Cache, y debe devolver un paquete de tipo "Route Error" a cada nodo que haya enviado paquetes a ser ruteado a través de dicho vínculo.

2.2.3. Implementación del protocolo

A continuación se presentará el diseño original de la implementación del protocolo sobre redes IPv4 [8], y luego se analizará un modelo que permiten superar las dificultades de migración a IPv6.

En la siguiente figura, se muestra un paquete IPv4 que lleva un campo de opción *Source Route*.

En el caso de que exista un enlace directo entre el origen y el destino, entonces la lista de direcciones en el campo de opciones estará vacía y su tamaño será 3 bytes. De forma alternativa, el campo de opciones puede ser omitido. De no haber vínculo directo, entonces se listan todas las direcciones intermedias y la de destino. El campo de opciones es seguido por un *end of list* (EOL). Luego se ubica el payload del paquete.

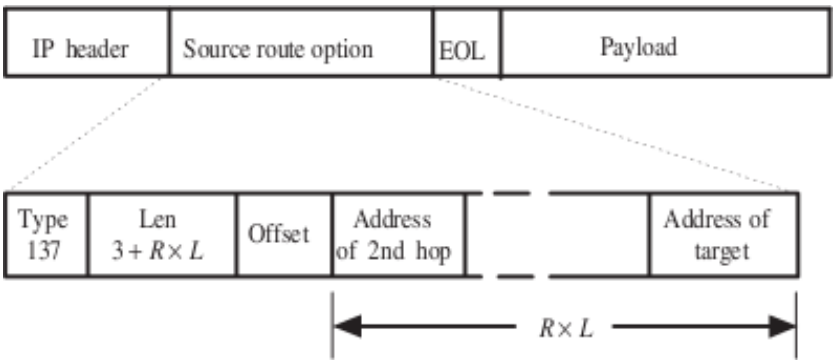


Figura 2.6: Paquete IPv4, con opción Source Route

El campo *Type* tiene un tamaño de 1 byte y su valor es 137. El campo *Len* también tiene un tamaño de 1 byte, y corresponde al tamaño de toda la opción. Si R es el número de direcciones listadas y L el tamaño de de una dirección IP, entonces *Len* vale 3 + R X L. El campo *Offset* tiene un tamaño de 1 byte y devuelve el índice del byte de la próxima dirección IP a ser usada.

En este punto es importante destacar que este modelo de ruteo desde el origen, que utiliza DSR, presenta un incremento considerable en el overhead asociado al tráfico de datos, haciendo que el protocolo sea aplicable en redes cuyas rutas consistan de entre cinco y diez saltos.

A continuación se muestra un paquete Route Request bajo este esquema,

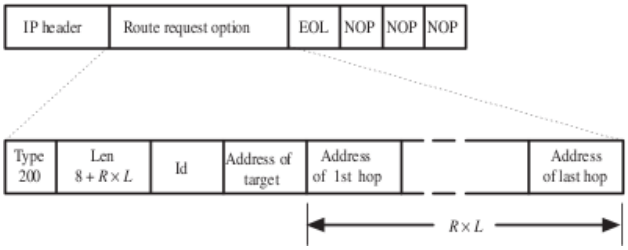


Figura 2.7: Paquete Route Request en IPv4

Luego del listado de direcciones, se incluye un EOL y tres NOP *No operation*, de modo tal que la opción en su conjunto tenga un tamaño múltiplo de 4 bytes. El campo *Type* cobra un valor de 200.

Según lo visto hasta aquí, este diseño presenta dos problemas fundamentales,

- *Limitaciones de escalabilidad:* Como se vio antes para redes IPv4 con más de diez saltos, el protocolo no es eficiente. Según las simulaciones realizadas en el trabajo [9], el *overhead* causado por paquetes de control puede llegar a utilizar un 95 % del ancho de banda del vínculo, cuando la red tiene 100 saltos.

- *No apto para IPv6:* Bajo el mismo razonamiento que el ítem anterior, se puede ver que al utilizar direcciones IPv6 (128 bits), el rendimiento cae críticamente. Esto es de vital importancia, debido a la creciente importancia de las redes IPv6.

El modelo basado en *Hash* [9], pretende extender el diseño para poder superar estas dos dificultades. La idea detrás de *Hash Based DSR* es comprimir la lista de direcciones usando un filtro Bloom [10].

2.3. ANTop (Adjacent Network Topology)

El protocolo ANTop es un protocolo diseñado para redes *ad hoc* (ver [1], [2] y [3]), con un esquema descentralizado, ya que todos los nodos conectados tienen igual jerarquía, pudiendo cumplir las mismas funciones, como ser asignación de direcciones a los nuevos vecinos, resolución de direcciones a partir de un identificador universal o efectuar el ruteo de paquetes.

Las redes *ad hoc* se caracterizan por ser de creación espontánea. En ellas, los nodos gozan de una gran movilidad, con lo cual su estructura cambia permanentemente. Es por esta razón que la dirección dependiente de la ubicación en la red cambiará frecuentemente, volviéndose imprescindible contar con un identificador universal del nodo. Este identificador, sin embargo, no será suficiente para poder rutear un paquete hacia su destino. En respuesta a este requerimiento, surge el concepto de *Distributed Hash Tables*, DHT [3]. Incorporando esta abstracción a la capa de red, ANTop define una técnica de ruteo indirecto, en la cual nodos llamados *Rendez-Vous* son responsables de guardar la dirección de red (y por ende la ubicación) de otros nodos, asociándola con su identificador universal. Como se dijo anteriormente, cualquier nodo que se conecte a la red *ad hoc* podrá actuar como *Rendez-Vous*.

2.3.1. Ruteo Indirecto

Cada nodo cuenta con tres direcciones. La primera es la dirección universal, la cual es conocida por cualquier otro nodo que quiera establecer una comunicación. Al ser totalmente independiente de la capa de red, puede ser

cualquier tipo de identificador. La segunda dirección de un nodo, la relativa, es aquella dependiente de la ubicación en la topología de red y la que nos permitirá rutear los paquetes. Si el nodo se mueve, este identificador cambiará. Finalmente la tercer dirección, la virtual, se obtiene aplicando una función de hash lineal a la universal y será la dirección relativa del nodo rendez-vous que está a cargo de proveer el identificador relativo asociado a esa dirección universal.

El esquema de ruteo indirecto funciona en la siguiente forma. El nodo origen s, quiere comunicarse con el destino d, pero no sabe su dirección relativa, tan solo la universal. Por esta razón, s contactará al nodo rendez-vous T quien conoce la dirección relativa de d, ya que este último se registró con T al conectarse a la red. A continuación T envía un mensaje a s, que contiene la dirección relativa de d, de modo que los paquetes desde s a d ya pueden ser ruteados.

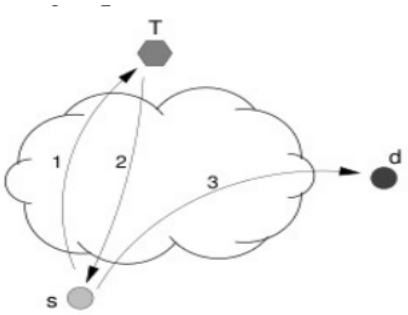


Figura 2.8: Modelo de ruteo indirecto

2.3.2. Direccionamiento en un Hipercubo

Una vez que se tiene la dirección relativa del destino, se debe rutear el paquete hacia el mismo. Aquí se presentan, dos esquemas posibles, ruteo proactivo y ruteo reactivo. En el primero de ellos las tablas de ruteo se construyen y se mantienen actualizadas constantemente, asegurando una ruta a cada nodo de la red, en todo momento. En el segundo esquema las rutas se generan bajo demanda y se mantienen durante un determinado lapso de tiempo. Dada la naturaleza de las redes *ad hoc*, donde la posición de cada nodo puede variar con frecuencia, el segundo esquema (reactivo) es el que mejor se adapta. Para redes cuya topología cambia frecuentemente, las rutas obtenidas proactivamente pueden ser obsoletas al momento de tener que

rutear un paquete hacia un destino en particular, haciendo que los recursos insumidos para conseguirlas hayan sido desperdiciados.

ANTop, define un esquema de direccionamiento basado en hipercubos. Estos son una generalización de un cubo de tres dimensiones, a un número d. Cada nodo tiene una coordenada de valor 0 ó 1 en cada dimensión. Esto implica que el número total de nodos es 2 d. Cada nodo está conectado a aquellos cuyas coordenadas varían solo en una dimensión. A continuación se muestra un hipercubo de dimensión 4. ANTop utiliza las coordenadas de cada nodo en el hipercubo, como la dirección relativa de los mismos.

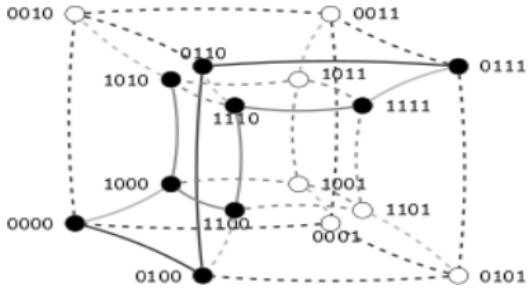


Figura 2.9: Direccionamiento basado en hipercubo

En la figura anterior, los vértices negros representan nodos que físicamente están presentes en la red, mientras que los blancos, representan direcciones que aún no fueron asignadas. Se dice en este caso que el hipercubo es incompleto, ya que no todas las direcciones están siendo utilizadas por nodos conectados a la red. Por el contrario, algunas de ellas están disponibles para ser asignadas a nuevos nodos que deseen conectarse a la red. Cuando todas hayan sido asignadas, la red no admitirá la conexión de nuevos nodos.

En este esquema, es fácil ver que la distancia entre dos nodos, definida como la cantidad de dimensiones con coordenadas distintas, se puede obtener aplicando una función XOR a las dos direcciones. Por ejemplo la distancia entre los nodos 0100 y 0111 es 2.

Cuando un nuevo nodo esté en el radio de cobertura de alguno de los ya conectados a la red, hará un pedido en modo broadcast, para que le sea asignada una dirección de hipercubo. Este pedido será escuchado por los posibles vecinos del nuevo nodo los cuales le ofrecerán una dirección que variará en solo una coordenada (un bit) respecto de la propia.

Cada nodo de la red contará con, además de una dirección relativa, una máscara que permitirá definir un espacio de direcciones que ese nodo administra. Tomando como ejemplo un hipercubo de dimensión d=4, el nodo A con dirección y máscara 1000/3, estará encargado de administrar las direcciones 1000 y 1001. Cuando un nuevo nodo desee conectarse a la red, A podrá cederle la dirección 1001/4. En caso de que esto suceda, es decir que efectivamente el nuevo nodo acepte esta propuesta como su dirección relativa, A aumentará su máscara a un valor de 4. Es en esta forma que A delega parte de su espacio de direcciones con su correspondiente administración a un nuevo nodo que se conecta a la red.

Este esquema de direccionamiento basado en un hipercubo, presenta un aporte que simplifica el proceso de ruteo. Esto puede verse claramente si se considera el caso de un hipercubo completo, es decir aquel en el que todas las direcciones están siendo usadas por nodos. Para ilustrar esto se puede considerar como ejemplo el caso en que el nodo 0001, quiere enviar un mensaje al nodo 1101 en un hipercubo de d=4. Es posible ver que la distancia entre ambos es 2, ya que esa cantidad de bits difieren entre las direcciones de dichos nodos. Al existir todos los nodos en la red, es posible saber *a priori* que el camino 0001 -> 1001 -> 1101 será válido para rutear el paquete, y la cantidad de saltos que el mismo transitará es igual a la distancia entre los nodos. Como puede verse, el nodo origen sin disponer aún de ninguna ruta, ya sabe un posible camino al destino, con distancia mínima.

En un hipercubo incompleto, en cambio, algunas direcciones no estarán siendo efectivamente utilizadas por nodos, con lo cual no es posible asegurar un camino a priori. Sin embargo, es posible que algunos caminos con distancia mínima si existan y por lo tanto vale la pena intentar rutear el paquete en esa dirección. Esto mismo es lo que hace ANTop, ya que bajo su esquema de ruteo reactivo, los paquetes se envían en primer instancia a aquel vecino cuya distancia al destino sea menor, al tiempo que este último repetirá el procedimiento para lograr rutear el paquete. Si durante esta exploración, se llega a un callejón sin salida, el paquete será devuelto por donde vino para que el nodo que originalmente envió el paquete en esa dirección pueda enviarlo a su próximo vecino, eligiéndolo por el mismo criterio de distancia al destino.

Capítulo 3

Interacción con el Sistema Operativo

En trabajos anteriores [1] [2] [3] se abordó la especificación de ANTop, así como el diseño de un simulador, tanto para la versión reactiva del protocolo, como para la proactiva.

Como ya se mencionó en la introducción, la presente Tesis se encarga de estudiar las distintas opciones de diseño para llevar ANTop hacia una red inalámbrica de computadoras, que funcione con direcciones IPv6. Para encarar dicho trabajo, se utilizó un sistema operativo Linux, más específicamente la distribución Ubuntu basada en Debian. Esta elección se justifica por la madurez y flexibilidad del software, así como también por las posibilidades de desarrollo que presenta.

En este capitulo, se analiza como mapear cada una de las características de funcionamiento que presenta ANTop a las funciones de red de un SO Ubuntu. Para hacerlo se adopta un criterio que prioriza escalabilidad de la solución en términos del tamaño de la red, portabilidad y simplicidad de diseño.

De aquí en adelante, se presentan los diferentes puntos característicos de un esquema de adaptación del diseño del protocolo al campo real de las redes de computadoras.

La memoria de sistema en Linux se divide en dos regiones: *Espacio de Kernel* y *Espacio de Usuario*. La primera es donde se ejecuta el núcleo del sistema operativo, es decir el *Kernel*. Desde allí, este último provee sus servicios. Por otro lado el Espacio de Usuario es la región de memoria del sistema

operativo en la que corren los *Procesos de usuario*, es decir las instancias de los programas.

En el Kernel corren servicios de bajo nivel, los cuales interactúan directamente con el hardware. Es allí donde, por ejemplo, se ejecutan los drivers de los dispositivos. Por esta razón, este código debe compilarse para cada computadora en forma particular.

Es en este punto donde se encuentra la primera decisión que tomar. La misma refiere a donde alojar el software que hará que el sistema operativo corra ANTop como una de sus funciones de red. Si bien colocar código en el Kernel del permite trabajar a bajo nivel, y por ende de un modo muy eficiente, se resta portabilidad a la solución debido a la necesidad de compilar el Kernel para cada computadora. Además dada la importancia de los servicios que allí corren, interactuar con ellos directamente puede hacer que todo el SO se vuelva inestable. Por estas razones, se decide que el código de ANTop corra en el Espacio de Usuario y que interactúe con las funcionalidades del Kernel mediante interfaces creadas para tal fin.

La interacción con el Kernel se da en tres puntos básicos,

- Tabla de ruteo
- Interfaz de red
- Manejo de paquetes

En el proceso de enrutamiento, ANTop consulta la tabla de ruteo del nodo que corre el protocolo, y si encuentra alguna ruta para el destino del paquete, toma esa decisión. En el caso en que no se tenga una ruta definida previamente, se dispara el mecanismo de obtención de la misma. A lo largo de este algoritmo, se vuelve imprescindible la interacción con la tabla. Aquí se debe recordar que ANTop trabaja en el espacio de usuario del sistema operativo, con lo cual en primer instancia se debe realizar un volcado de la misma a dicho espacio, para ser recorrida y analizada. Luego, se deberá agregar una nueva ruta en caso de que no exista una previamente. Nuevamente se necesita un mecanismo de interacción entre el kernel y el espacio de usuarios que en este caso en particular nos permita leer y escribir la tabla de ruteo.

En cuanto al segundo ítem, se debe tener en cuenta que el protocolo estará asociado a una interfaz de red, del total con el que puede contar el nodo. Es por esta razón que el funcionamiento tanto de los procesos de ruteo, como de manejo de paquetes, debe estar restringido a dicho universo. Por último, recordando que ANTop establece un esquema de asignación de direcciones relativas a la topología de la red, es posible ver que se necesitará poder definir desde la capa de usuario dicho parámetro de configuración determinado por el protocolo en base a las respuestas que reciba de nodos vecinos. Por esta razón y las anteriores, también se vuelve un requerimiento el contar con una herramienta que permita obtener y modificar la configuración

Para analizar el último ítem, se debe tener en cuenta que ANTop, en este caso, procesa los paquetes en la capa de usuario. Es por esto que se requiere de una herramienta que permita transportar dichos paquetes desde el espacio de kernel, para que luego el mismo pueda ser procesado y eventualmente modificado.

Finalmente, se analizará el funcionamiento de la capa de enlace sobre la que correrá ANTop.

3.1. IOCTL (Input / Ouput Control)

Una de las posibles opciones para cubrir la interacción con la tabla de ruteo y la configuración de la interfaz de red es IOCTL. El nombre de esta herramienta hace referencia a *Input / Output Control*. Se trata básicamente de una interfaz que permite comunicarse con el controlador del dispositivo a controlar mediante un conjunto de instrucciones definidas como etiquetas en los archivos encabezados del código fuente del driver en cuestión. Esta función logra controlar una variedad de mecanismos del sistema operativo. Esta característica se debe a que la misma se implementa como una llamada de sistema que multiplexa diferentes comandos hacia la función del espacio de Kernel correspondiente. Una llamada a IOCTL tiene tres argumentos:

- un archivo descriptor (o socket)
- un identificador de comando
- un argumento

El multiplexado se basa en el archivo descriptor y en el identificador de comando.

Esta herramienta presenta la desventaja fundamental de que ha quedado obsoleta, es decir que no se desarrolla más, y solo se mantiene en Linux, por compatibilidad.

3.2. Netlink

El Kernel de Linux presenta una característica muy útil para agregar funcionalidad a bajo nivel. Se trata de la posibilidad de manejar módulos, es decir código que puede ser cargado o eliminado bajo demanda, en el sistema operativo, sin necesidad de recompilar el Kernel. Esto permite extender los servicios que brinda el mismo, sin siquiera tener que reiniciar el sistema. Un ejemplo de módulo, son los controladores de dispositivos, que permiten al Kernel acceder a los dispositivos físicos conectados al sistema.

Además de lo recién mencionado, se tiene la ventaja de que el Kernel se vuelve más pequeño y portable.

Es aquí donde se vuelve muy importante el rol de Netlink [11], una interfaz que se utiliza para transferir información entre los módulos del kernel y los procesos del espacio de usuarios. Provee un canal de comunicación bidireccional y consiste, básicamente, en una interfaz basada en sockets para los procesos y una *Application Programming Interface* (API) en el kernel para los módulos.

En cuanto a los sockets, se deben definir algunos parámetros que los identifican,

Familia Netlink Es el subgrupo al que se quiere referir.

Dominio Para este tipo de sockets, el dominio se define como AF_NETLINK.

Tipo de socket Netlink es un servicio orientado a datagrama. Para este parámetro se puede usar indistintamente SOCK_DGRAM ó SOCK_RAW.

En cuanto al primer parámetro, las principales familias Netlink son,

NETLINK_ROUTE Comunicación de lectura, escritura con la tabla de ruteo Ipv4. Permite modificar rutas, direcciones IP, parámetros del enlace, configuración de vecinos, configuración de colas y clases de servicio.

NETLINK_FIREWALL Recibe los paquetes enviados por el firewall Ipv4.

NETLINK_ARPD Manejo de la tabla ARP.

NETLINK_ROUTE6 Comunicación de lectura/escritura con la tabla de ruteo Ipv6.

Netlink es un servicio orientado a datagrama, y funciona en un modo muy similar a un socket UDP, con la salvedad de que en este caso el paquete no sale del nodo local, como si sucede en los sockets para tal fin, cuyo dominio es AF_INET. Los mensajes son enviados a través de sockets abiertos, mediante las funciones destinadas a tal fin, y el tipo de comunicación es sin conexión. Si bien no hay garantía de que el paquete será recibido correctamente, se implementan algunos mecanismos de confiabilidad en el encabezado del paquete, los cuales no serán tratados en la presente tesis.

Siempre que se envíe o se reciba información a través de un socket de este tipo, se debe completar una estructura de dirección del socket. En ella se indica al kernel el destino del paquete, al tiempo que para el espacio de usuario, se informa el origen de un paquete.

Por otro lado, cada mensaje de Netlink, tendrá un encabezado, que debe ser construido. A continuación se muestra dicho encabezado.

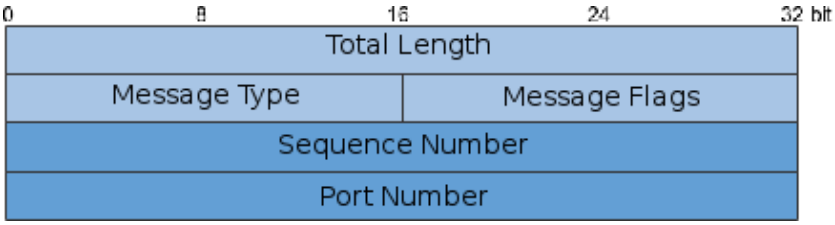


Figura 3.1: Encabezado de los paquetes Netlink

Total Length (32bits) El tamaño total del mensaje, incluyendo el encabezado.

Message Type (16 bits) Este campo especifica el tipo de payload que transporta el mensaje.

Message Flags (16 bits) Estas bandera se utilizan para modificar el comportamiento del campo type.

Sequence Number (32 bits) El número de secuencia es opcional, y puede ser utilizado como referencia a un mensaje anterior. Por ejemplo, un error de mensaje puede referir al requerimiento anterior para indicar que el mismo causó el error.

Port Number (32 bits) Especifica el modulo de kernel destino que recibirá el mensaje. Si este campo no se especifica, el mensaje será enviado al primer socket del lado del kernel que tenga la misma familia de protocolo.

En relación al segundo campo, referido al tipo de mensaje, Netlink distingue entre pedidos, notificaciones, y respuestas. Para los primeros, la bandera NLM_F_REQUEST está encendida, y se definen para pedir una acción por parte del receptor. Típicamente un pedido (request) se envía desde un proceso de espacio de usuario hacia el kernel. Aunque no es obligatorio, es recomendable que lleven un número de secuencia, que debe incrementarse con cada pedido enviado.

Dependiendo de la naturaleza del pedido, el receptor puede responder con otro mensaje Netlink. El número de secuencia de la respuesta debe ser el mismo.

Las notificaciones son de naturaleza informal, y no se espera ninguna respuesta, por ello el numero de secuencia lleva, típicamente, el valor de cero.

Los tipos de mensajes definidos son,

NLMSG_NOOP No tiene ninguna operación, el mensaje debe ser descartado.

NLMSG_ERROR Mensaje de error ó ACK.

NLMSG_DONE Fin de una secuencia multiparte.

NLMSG_OVERRUN Error.

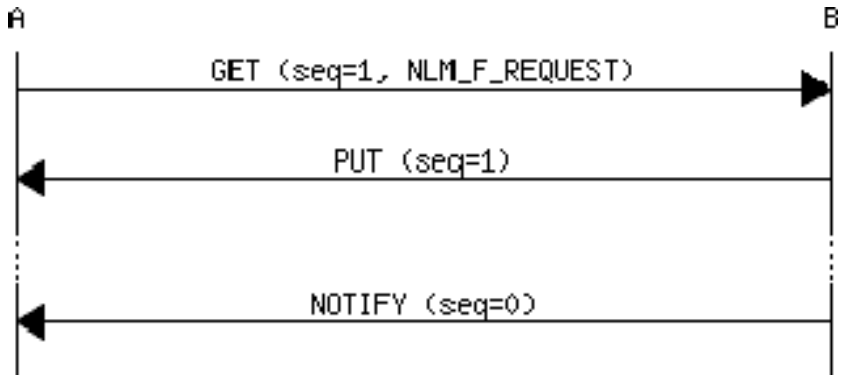


Figura 3.2: Intercambio de mensajes Netlink

Cada protocolo Netlink puede definir sus propios mensajes.

Aunque en teoría un mensaje puede ser tan grande como 4GB, los buffers de los sockets Netlinks no suelen ser tan grandes para alojar dicho tamaño. Por esta razón es común limitar los mensajes al tamaño de una página y usar el mecanismo multiparte para separar grande bloques de información, en varios mensaje. Para este tipo de mensajes se utiliza la bandera NLM_F_MULTI, y el receptor espera seguir recibiendo información hasta leer un mensaje de tipo NLMSG_DONE. A menudo se envían listas de objetos, donde cada mensaje es uno de ellos.

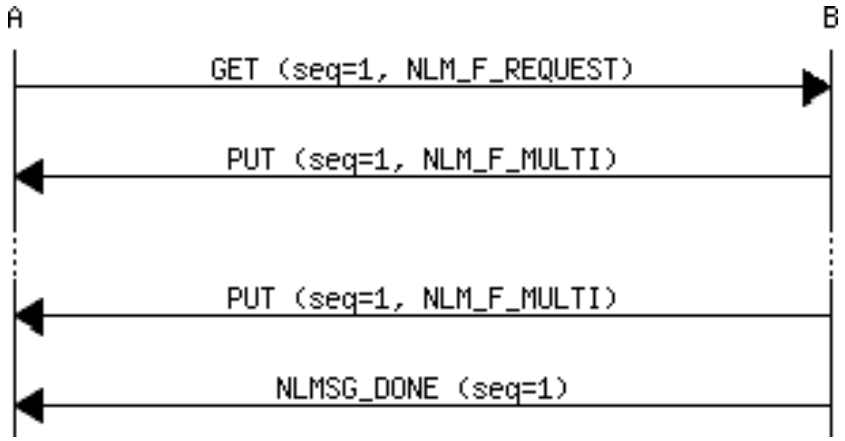


Figura 3.3: Intercambio de mensajes Netlink multiparte

Los pedidos pueden generar mensajes de error como respuesta. Estos deben usar el tipo de mensaje estándar NLMSG_ERROR. El payload consiste de un código de error y el encabezado del mensaje original que lo ocasiono.

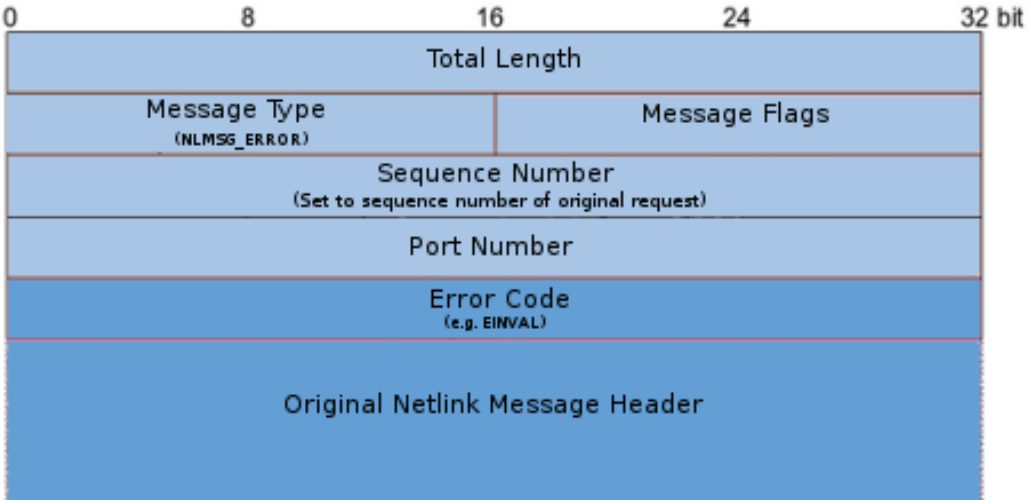


Figura 3.4: Paquete de error en Netlink

El número de secuencia del mensaje de error debe ser el mismo que aquel del mensaje que lo originó.

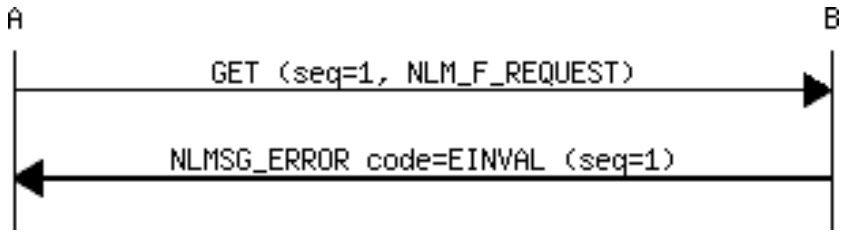


Figura 3.5: Mensaje de error en Netlink

Seteando la bandera NLM_F_ACK, es posible pedir que el receptor envíe una confirmación (ACK) por cada pedido procesado.

Las banderas estándar que define el protocolo son las siguientes,

- NLM_F_REQUEST 1 El mensaje es un requerimiento de información.
- NLM_F_MULTI 2 El mensaje es un bloque de información dentro de una lista que encadena más de uno. El receptor espera seguir recibiendo bloques hasta leer el tipo de mensaje NLMSG_DONE.

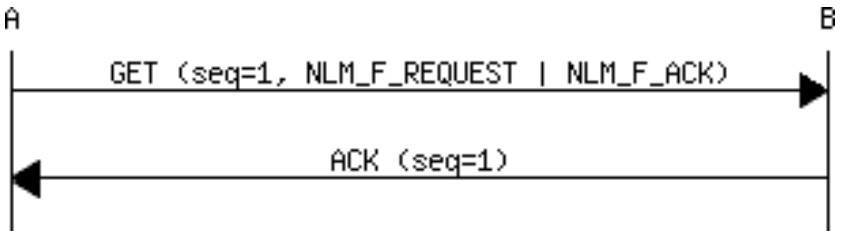


Figura 3.6: Mensaje de acks en Netlink

- NLM_F_ACK 4 El mensaje solicita una confirmación de recepción.
- NLM_F_ECHO 8 Es similar a NLM_F_ACK. Se puede utilizar en conjunto con NLM_F_REQUEST para que el receptor envíe una notificación al recibir un pedido.

El primer valor es el nombre de la etiqueta incluida en la biblioteca que implementa Netlink en el SO Linux. La segunda columna responde al valor numérico efectivo de dicha etiqueta. Puede verse que cada una de estas bandera se representa como un bit dentro de la palabra de dos bytes. De esta forma se puede configurar más de una bandera, simplemente aplicando una máscara "OR" entre la bandera a ser agregada y la palabra de banderas en el mensaje.

También se definen bandera adicionales que solo aplican para requests.

- NLM_F_ROOT 0x100 Devuelve la tabla completa, en lugar de solo una entrada.
- NLM_F_MATCH 0x200 Devuelve todos los elementos que concuerdan con un criterio dado en el mensaje.
- NLM_F_ATOMIC 0x400 Esta bandera es obsoleta. Se utiliza para pedir una operación atómica.
- NLM_F_DUMP (NLM_F_ROOT|NLM_F_MATCH) Devuelve una lista de todos los objetos. Es la combinación de las banderas NLM_F_ROOT y NLM_F_MATCH.

También existe una serie de banderas asociadas a los pedidos de crear o modificar objetos.

- NLM_F_REPLACE 0x100 Reemplaza el objeto, si es que existe.

- NLM_F_EXCL 0x200 No actualizar el objeto si es que ya existe.
- NLM_F_CREATE 0x400 Crear el objeto si es que no existe aún.
- NLM_F_APPEND 0x800 Agregar un objeto al final de la lista.

El número de secuencia constituye un parámetro muy importante de los paquetes Netlink. Este ayuda a relacionar pedidos con respuestas. A diferencia de protocolos como TCP, no es obligatorio el uso de los números de secuencia. Finalmente se dirá que se utilizan por sockets. Es decir que para dos sockets distintos, se puede tener mensajes con los mismos números de secuencia, sin perjuicio de su funcionalidad.

Cada socket puede suscribirse a cualquier número de grupos multicast del protocolo Netlink. De esa forma, el socket recibirá los mensajes que lleguen a cualquiera de ellos.

3.2.1. Sockets Netlink

Para poder usar Netlink, se requiere un socket de este tipo. Cada socket define un contexto totalmente independiente para enviar y recibir mensajes. Una aplicación puede usar diferentes sockets para el mismo protocolo Netlink. Por ejemplo, se puede tener un socket para enviar pedidos y recibir las respuestas, y otro socket suscripto a un grupo multicast para recibir notificaciones.

A continuación se presentan los atributos de los sockets.

Local Port El número de puerto local identifica unívocamente el socket y es usado para direccionarlo. Cuando el socket es definido, este número se define automáticamente. Consiste del ID del proceso (22 bits) y un número aleatorio (10bits), lo que permite un total de 1024 sockets por proceso. Es posible reescribir el número de puerto local, pero se debe estar seguro de que el mismo es único y ninguna otra aplicación lo está utilizando.

Peer Port Un puerto vecino se puede asignar a un socket, de modo tal que todos los mensajes unicast serán enviados por ese socket con la dirección de puerto vecino. Si no se especifica uno, el mensaje es enviado al kernel que intentará asociar el socket a uno del kernel, perteneciente a la misma familia de protocolo Netlink.

File Descriptor Netlink utiliza la interfaz de sockets BSD, por lo tanto un descriptor de archivo está detrás de cada socket. Si el socket se utiliza solo para recibir notificaciones, es conveniente configurarlo en modo no bloqueante y pollearlo periódicamente en busca de nuevas notificaciones.

Buffer size El buffer se utiliza para encolar mensajes entre emisor y receptor. Este parámetro especifica el máximo tamaño que se podrá enviar a un Netlink socket. Indirectamente definirá el máximo tamaño de mensaje.

3.2.2. Paquete Netlink

La mayoría de los protocolos Netlink, definen una estricta política de alineación, para todos los límites. El valor de alineación está definido por la etiqueta NLMSG_ALIGNTO y vale 4 bytes. Por lo tanto, todos los encabezados de mensajes Netlink, comienzos de payload, encabezados específicos de protocolo y la sección de atributos deben ubicarse en un desfasaje múltiplo de NLMSG_ALIGNTO.

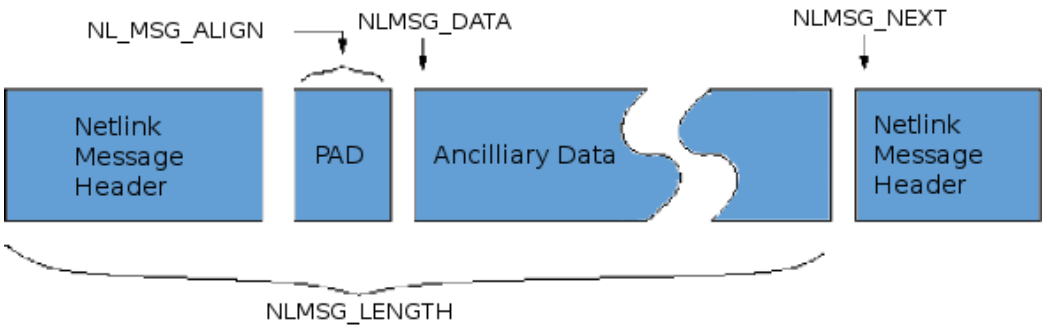


Figura 3.7: Estructura de un paquete Netlink

Con el fin de cumplir con el requerimiento recién explicitado, se agrega el campo de padding en la estructura del paquete.

La biblioteca ofrece dos formas diferentes de procesar los mensajes Netlink. Para aquellas aplicaciones que pretendan hacer el procesamiento manual, se tiene una interfaz de bajo nivel. Este método se explica a continuación.

Típicamente se recibe, del socket Netlink, un flujo de mensajes. Se cuenta con el buffer y su longitud. El mismo puede contener cualquier número de mensajes Netlink.

El primer encabezado de mensaje se ubica al comienzo del flujo. Los siguientes encabezados se acceden llamando a la función `nlmsg_next()`, en el header anterior. Dicha función automáticamente sustrae el tamaño del último mensaje, del resto de los bytes.

En verdad, no hay una indicación en el mensaje anterior acerca de si sigue otro mensaje o no. Se debe asumir de que siguen más mensajes hasta que todos los bytes del flujo hayan sido procesados.

Para simplificar esto, se implementa la función `nlmsg_ok()`, la cual devuelve verdadero si otro mensaje puede ser ubicado en el número restante de bytes en el flujo. La función `nlmsg_valid_hdr()` es similar. Esta última chequea si un mensaje específico contiene un mínimo número de bytes en su payload.

El payload del paquete, o carga útil, se ubica luego del encabezado del mismo, y se garantiza que comienza en un número de bytes múltiplo de NLMSG_ALIGNTO. Para asegurar esto, se agrega padding al final del encabezado si es necesario. La función `nlmsg_data()`, calculará el desfasaje, y devolverá un puntero al inicio del payload.

El tamaño del payload es devuelto por `nlmsg_datalen()`.

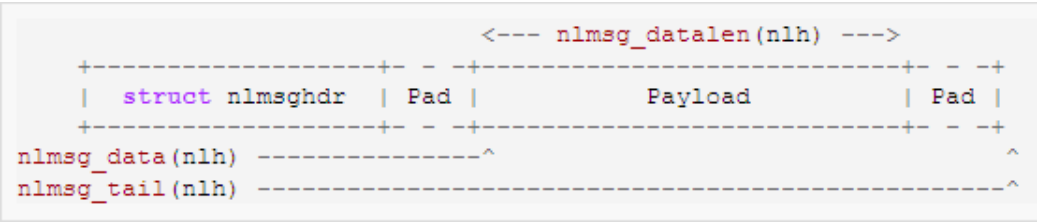


Figura 3.8: Payload de un paquete Netlink

La mayor parte de los protocolos de Netlink usan atributos. Esto da flexibilidad para extender el protocolo posteriormente. Nuevos atributos pueden definirse en cualquier momento, al tiempo que los viejos pueden ser reemplazados sin violar ninguna compatibilidad binaria del protocolo.

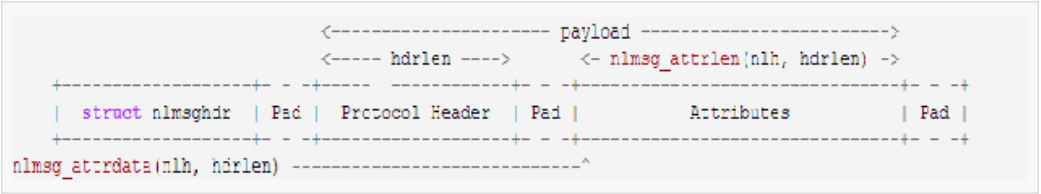


Figura 3.9: Atributos dentro de un mensaje Netlink

La función `nlmsg_attrdata()` devuelve un puntero al principio de la sección de atributos. El tamaño de esta sección es devuelto por la función `nlmsg_attrlen()`.

La función `nlmsg_parse()` valida un mensaje completo en un paso. Si hdrlen > 0, la misma llamará `nlmsg_valid_hdr()` para chequear si el encabezado de protocolo entra en el mensaje. Si hay más payload que procesar, asumirá que se trata de un atributo y lo procesará adecuadamente.

Los atributos presentan otra característica que brinda flexibilidad a la herramienta Netlink; la posibilidad de anidarlos. Tomando como ejemplo el caso en que se introduzca una nueva generación de dispositivos de red, que requiera un juego completamente nuevo de configuraciones, se tiene la posibilidad de definir un nuevo tipo de atributo que los contemple, al tiempo que es posible definir una sub-estructura de atributos que represente la configuración requerida por el mismo.

A continuación se muestra el formato general de paquete que se debe seguir para alocar atributos.

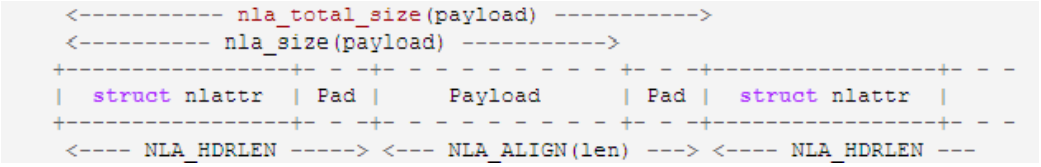


Figura 3.10: Formato de un paquete de atributos Netlink

Cada atributo debe comenzar en un múltiplo de NLA_ALIGNTO, que para esta implementación vale (4 bytes). Si se requiere saber si un atributo necesita incluir bytes de padding, es posible utilizar la función `nla_padlen()`, la cual devuelve la cantidad de los mismos que deberán ser utilizados. A continuación se muestra la estructura que deberán llevar.

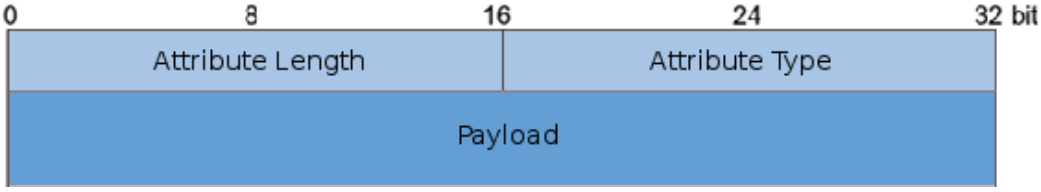


Figura 3.11: Estructura nlattr

Si bien existen funciones automáticas dentro de la familia `nlmsg_parse()`, también existe una interfaz para procesar los atributos en forma manual. Como se dijo antes, se tiene una secuencia o flujo de atributos. La función `nlmsg_attrdata()`, devuelve un puntero al encabezado del primero de ellos. Cualquier atributo subsiguiente es accesado a través de la función `nla_next()`, a la que se le debe pasar como parámetro el encabezado previo.

La semántica de estas funciones es la misma que para `nlmsg_next()`, con lo cual `nla_next()` también sustraerá el tamaño del encabezado anterior del numero restante de bytes en el flujo de atributos.

Al igual que los mensajes, los atributos no contienen un indicador de si otro atributo está encolado en el flujo. La única indicación es el número de bytes restantes en el flujo. La función `nla_ok()` existe para determinar si un atributo puede ser alocado en el número restante de bytes. Este calculo se basa, lógicamente, en el tamaño del mismo.

Una vez que los atributos individuales han sido separados, es posible acceder al encabezado y el payload de un atributo.

```
                            <- nla_len(hdr) ->
    +-----------------+- - -+- - - - - - - - - +- - -+
    |  struct nlattr  | Pad |     Payload       | Pad |
    +-----------------+- - -+- - - - - - - - - +- - -+
nla_data(hdr) ---------------^
```

Figura 3.12: Funciones de acceso al atributo

Las funciones `nla_len()` y `nla_type()` pueden ser usados para acceder al encabezado del atributo. La primer función devuelve la longitud del payload sin incluir bytes de padding. La segundo devuelve el tipo de atributo. La función `nla_data()` devolverá un puntero al payload del atributo.

Los atributos pueden ser anidados para lograr estructuras complejas en forma de árbol. Se usa normalmente para delegar la responsabilidad a un subsistema. También se usan frecuentemente para transmitir una lista de objetos. Cuando se usan estructuras de este tipo, aquellos anidados se incluyen como payload de un atributo contenedor.

Cuando se usan las funciones antes vistas de validación, solo los atributos en el primer nivel son validados. Ninguna de las funciones validará aquellos anidados. Para lograrlo se debe llamar explícitamente a `nla_validate()` o usar `nla_parse_nested()` para cada nivel anidado.

Como se pudo apreciar a lo largo de todo este apartado, Netlink es una herramienta muy flexible y potente, al tiempo que simple para la programación. Por estas razones, se utilizará esta herramienta.

3.3. Netfilter

Netfilter [1] está compuesto por una serie de ganchos ubicados a lo largo de la estructura de protocolos de red de un sistema Linux. En cada gancho es posible registrar un módulo de kernel, que realice algún manejo de paquetes en diferentes instancias del procesamiento del mismo por parte de los protocolos de red.

Un ejemplo de uso de esta herramienta de manejo de paquetes es Iptables, un *firewall* para Linux muy utilizado al momento de la redacción de esta Tesis. Iptables usa Netfilter como motor de captura de paquetes.

Aquí surge la opción de utilizar Iptables para realizar el manejo de paquetes. Sin embargo, esta alternativa debe ser invocada mediante el manejo de *scripting*, lo cual vuelve la solución poco flexible y por lo tanto se descarta en esta evaluación de herramientas.

[1] http://www.netfilter.org/

Netfilter inserta cinco ganchos asociados al protocolo IP (tanto versión 4 como 6) en Linux,

PREROUTING Todos los paquetes que llegan al nodo pasan por este gancho, que se encuentra antes de la decisión de ruteo y después de los chequeos de sanidad del encabezado IP.

LOCAL INPUT Todos los paquetes que tienen como destino la computadora local llegan a este gancho. Este es el último para aquellos paquetes dirigidos al nodo.

FORWARD Por aquí pasan los paquetes que son ruteados por este nodo como un próximo salto, pero que tienen un destino diferente.

LOCAL OUTPUT Este es el primer gancho en el camino de salida de aquellos paquetes que se originan en el nodo local.

POST ROUTING Este gancho se ubica luego de la decisión de ruteo. Absolutamente todos los paquetes salientes del nodo pasan por este gancho.

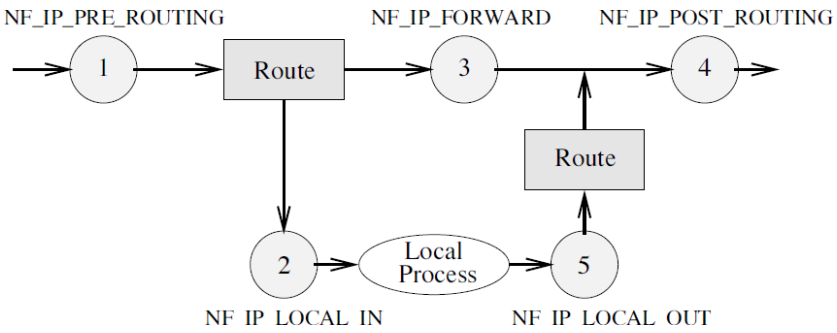


Figura 3.13: Diagrama de ganchos en la estructura Netfilter

Con este modelo en mente, es posible catalogar el tráfico en tres clases.

- Tráfico que será ruteado por el nodo, pero con una dirección de destino distinta a la propia. El camino seguido será, PREROUTING FORWARDING POSTROUTING
- Tráfico destinado al nodo local. El camino será PREROUTING INPUT
- Tráfico saliente del nodo local. El camino será OUTPUT POSTROUTING

Es posible registrar funciones de *callback* en cada uno de los ganchos. Es decir, funciones que se ejecutan automáticamente cuando un paquete atraviesa dicho gancho. De ese modo es posible realizar algún tipo de manejo de paquetes. Para cada registro de función se establece una prioridad, ya que es posible vincular varias de ellas a un mismo gancho.

Las funciones de callback pueden devolver diferentes valores, que se corresponden con diferentes acciones a tomar.

ACCEPT Permite que el paquete siga atravesando la estructura de protocolos de red.

DROP Descarta el paquete.

QUEUE Envía el paquete al Espacio de Usuario, de forma tal que un programa corriendo allí se encarga de manejarlo.

STOLEN Detiene el paquete hasta que se cumple una condición. Es decir que el paquete temporalmente interrumpe su recorrido. Está acción se usa normalmente para colectar paquetes fragmentados.

REPEAT Fuerza el paquete a ingresar nuevamente en el gancho.

3.4. Capa de enlace

En este apartado se estudia el funcionamiento de la capa de enlace sobre la que correrá ANTop.

El protocolo aquí utilizado es IEEE 802.11 [12]. En la sección 5.2.1 del citado documento, se define el concepto de *Independent Basic Service Set LAN* (IBSS LAN), para representar una red en la cual las estaciones se conectan directamente sin necesidad de ninguna entidad de control. También se indica que a este tipo de topología se le suele dar el nombre de *ad hoc network*.

Para que este servicio funcione correctamente, las estaciones (aquí serían los nodos de la red ANTop) deben trabajar en el mismo canal y compartir el mismo *Service Set Identifier* (SSID) para la red *ad hoc* a la que se conectan, de lo contrario no tendrán visibilidad a nivel de capa de enlace. Esto es una limitación del modo de funcionamiento *ad hoc* recién explicado. Cada nodo que se conecte a la red ANTop deberá usar el mismo SSID, por ejemplo

"antop". De esta forma, cuando un paquete contenga una dirección MAC de *broadcast* como destino, todos los nodos que estén dentro del alcance del que envió el paquete lo recibirán.

En el caso del estándar IEEE 802.11g [13], se define una velocidad de conexión máxima de 54 Mbps, para redes estructuradas. Sin embargo, para las redes ad hoc, el estándar no requiere alcanzar dicha tasa de transferencia, sino que será obligatorio alcanzar los 11 Mbps que define la norma b. Quedando, finalmente, la tasa máxima de transferencia en decisión del fabricante que proporcione el controlador del dispositivo, en este caso para un sistema operativo Linux.

3.5. Consideraciones finales

En la primer parte del capítulo se discutió acerca de donde colocar el código fuente que ejecute las características de red que aporta ANTop a una red ad-hoc inalámbrica de computadoras. Se vio que si bien colocar código en el Kernel del sistema operativo puede brindar un alto grado de eficiencia al eliminar todo el overhead en la comunicación de los procesos, se pierde portabilidad y se genera un riesgo considerable de que el sistema se vuelva inestable.

Por estas razones se determinó que la mejor opción sería correr un demonio en el Espacio de Usuario, el cual se comunique con todas las funciones de red residentes en el Kernel.

En este punto se debió optar por alguna herramienta para lograr dicha comunicación entre el Espacio de Usuario y el de Kernel. En primer medida se analizó *IOCTL*, pero se descarto debido a que es obsoleta y poco flexible.

Habiendo descartado IOCTL, se estudió Netlink, la herramienta que actualmente se encuentra en desarrollo y que es utilizada por la comunidad de programadores de red de Linux para cubrir las funciones requeridas. Se hizo un análisis en profundidad de la herramienta y se decidió adoptarla para las funciones de escritura y lectura de la tabla de ruteo y configuración de la interfaz de red.

Luego se estudió el modo para enviar los paquetes al Espacio de Usuario cuando los mismos atraviesan la pila de protocolos de red implementada en

el Kernel, y para enviarlos de nuevo al Kernel luego de su procesamiento por el demonio que corre ANTop.

La primer opción analizada fue la de utilizar *Iptables* a través de *scripting*, pero se descartó por ser una solución poco flexible. Finalmente, la herramienta adoptada para cumplir con esta función fue Netfilter.

Finalmente se definió el esquema que se utilizará para lograr la conexión de nodos a nivel de capa de enlace, al tiempo que se discutió acerca de las posibles tasas de transferencia máximas teóricas.

Capítulo 4

ANTop sobre IPv6

En este capítulo se estudiará como utilizar las herramientas elegidas previamente para interactuar con el sistema operativo, de modo de poder embeber en este último todas las funcionalidades de red que ANTop propone en su definición.

En la sección 4.1 se estudia como generar paquetes de control ANTop desde el demonio que corre en el Espacio de Usuario, y que modificaciones se debe hacer en los paquetes para que el tráfico pueda ser interpretado como tráfico a ser tratado por ANTop.

A continuación, en en el apartado 4.2 se estudian las posibilidades para embeber las direcciones relativas de ANTop (definidas como cadenas de bits sin formato), en direcciones IPv6. También se tienen en cuenta las implicaciones de cada tipo de dirección en el funcionamiento de la tabla de ruteo, así como la máscara de red que se debe adoptar al asignar direcciones IPv6 a la interfaz inalámbrica.

Más adelante en esta sección se estudia el proceso de conexión de nodos. Se ven los paquetes de control ANTop que se intercambian allí, así como el mecanismo mediante el cual se ceden direcciones. También se discute la relación de vecinos entre nodos cercanos. Por último se analiza que sucede cuando un nodo abandona la red.

A continuación se presenta una sección 4.3 que discute la función de ruteo. Para ello se analiza la comunicación con la tabla de ruteo, y sobre el final de la sección se discute el ruteo de los paquetes de control *Rendez-Vous*.

En la siguiente sección 4.5 se discute la importancia de contar con una identificación de los nodos de una red ad-hoc inalámbrica, independiente de la posición de los mismos en la topología de la red. Luego, se analiza la implementación del servicio *Rendez-Vous* que permite resolver direcciones universales a direcciones IPv6. Además se estudia como utilizar la estructura DNS ya instalada en el sistema operativo para embeber *Rendez-Vous* de un modo transparente.

La última sección 4.4 analiza un problema que surge cuando ANTop rutea tráfico TCP sobre la red ANTop.

4.1. Manejo de paquetes ANTop

Los nodos que participen de la red ANTop deben ser capaces de enviar y recibir los paquetes de control que el protocolo define. Para lograrlo, se utiliza un socket UDP, el cual se asocia a un puerto determinado que se usará como origen y destino para la capa de transporte. Además de crearse el socket, esta implementación de ANTop sobre IPv6 adjunta una función de callback para la lectura y procesamiento de los paquetes que lleguen al nodo, dirigidos al número de puerto antes mencionado.

Para generar los paquetes de control, se crea una estructura que contiene los campos definidos por el protocolo [1] (sección 2.2.4) y la misma se pasa como argumento a una función que genera los encabezados correspondientes, tanto de Ethernet, como de IPv6 y UDP, y coloca estos datos en el payload de la capa de transporte.

La definición de ANTop contempla mecanismos de retransmisión en caso de pérdida de paquetes de control. No es el objetivo del presente trabajo redefinir dicha especificación, por esta razón se utiliza UDP como protocolo de transporte.

Sin embargo, hubiera sido posible emplear sockets de tipo *RAW*, los cuales permiten modificar los campos de los encabezados IPv6 y UDP. Si bien esto brinda flexibilidad a la implementación, como se verá luego, estas modificaciones se realizan mediante Netfilter. Por otro lado, al permitir que los encabezados se generen automáticamente por el SO, se ahorra trabajo y se reducen los posibles puntos de error por mala manipulación de los campos. Un ejemplo de este caso es el cálculo del Checksum.

Los paquetes que son generados en el nodo local, son marcados con una etiqueta que indica que es tráfico que debe ser tratado por el demonio de espacio de usuario que ejecuta ANTop. Con el fin de ubicar dicha etiqueta, y otras que surgen del funcionamiento del protocolo, así como también una dirección IPv6 necesaria para ruteo (ver sección 4.3), se aloja un byte de banderas en el encabezado de extensión que la presente implementación de ANTop agrega. Más precisamente se usa un encabezado de tipo *Destination Options* E.

Tanto para el tráfico generado por las aplicaciones de las capas superiores, como el tráfico de control del protocolo, generado este último por el demonio de ANTop, los paquetes se interceptan, en el gancho LOCAL_OUT, y allí se marcan con la bandera de tráfico ANTop.

Cuando un nodo recibe un paquete, el mismo es enviado al Espacio de Usuario, mediante la función que registra Netfilter en el gancho LOCAL_IN. Allí se chequea si cuenta con la bandera que identifica al tráfico a ser tratado por ANTop. Si esto no es así, el paquete se devuelve al Espacio de Kernel sin más procesamiento, para que continúe su viaje.

4.2. Direccionamiento en ANTop sobre IPv6

Tal lo visto en la introducción a ANTop 2.3, cada nodo en la red tiene una dirección relativa dependiente de la topología, que representará las coordenadas de ese nodo en un hipercubo de dimensión n, y que se usará para rutear el tráfico entre dos nodos, mediante el criterio de mínima distancia al destino. Para los fines de la presente Tesis, las direcciones relativas adquieren un formato IPv6.

Para poder hacer uso de todas las herramientas y funcionalidades que presentan los sistemas operativos Linux, sin tener que reescribir código ya maduro y depurado, es necesario que las direcciones relativas cumplan con uno de los formatos unicast definidos por IPv6 [14],

Direcciones Globales Unicast

Las direcciones Unicast Globales tienen el siguiente formato,

Address type	Binary prefix	IPv6 notation
Unspecified	00...0 (128 bits)	::/128
Loopback	00...1 (128 bits)	::1/128
Multicast	11111111	FF00::/8
Link-Local unicast	1111111010	FE80::/10
Global Unicast	(everything else)	

Figura 4.1: Segmentación de direcciones IPv6

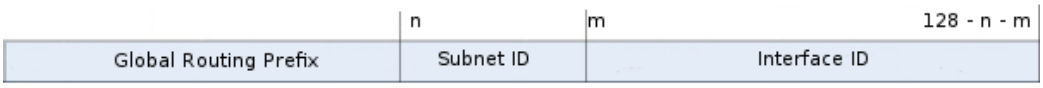


Figura 4.2: Formato de dirección Global Unicast IPv6

Aquí el campo *global routing prefix* es un valor asignado a un sitio, es decir un grupo de subredes. El campo *subnet ID* es un identificador de una red de área local, perteneciente al sitio. Finalmente, *Interface ID* define el direccionamiento para cada una de las interfaces de red que pertenecen a la red.

Direcciones Link-Local

Este tipo de dirección local debe cumplir con el siguiente formato,

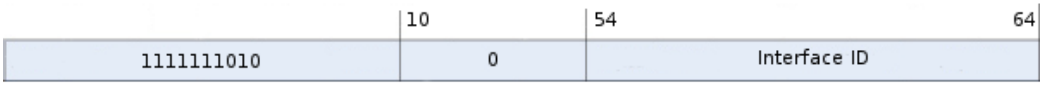


Figura 4.3: Formato de dirección Local IPv6

4.2.1. Mapeo de direcciones relativas en direcciones IPv6

En el presente trabajo se considera la red ANTop como aislada, sin salida a la red pública. El tráfico que intercambian los nodos que participan, no atraviesa ningún router. Para este tipo de red, el RFC 4291 [14] define las direcciones link-local. Los routers corriendo IPv6, no reenvían el tráfico con este tipo de direcciones.

Cuando un nodo quiere conectarse a la red, debe enviar un paquete PAR en modo broadcast, de tal forma que los nodos que están a su alcance y ya participan de la red ANTop, puedan ofrecerle direcciones IPv6 como respuesta a este pedido. Para que esta secuencia se complete exitosamente, se debe tener una dirección IPv6 que no coincida ni con una ya asignada en la red ANTop, ni con la de otro nodo que también intenta conectarse. Este último caso podría salvarse utilizando una configuración Link-Local, para la cual Linux adopta un mecanismo que busca asignar automáticamente una dirección IPv6, única en la red [15]. Dicho mecanismo solo es válido para interfaces que soportan comunicación multicast.

Con el objetivo de que una interfaz pueda obtener automáticamente una dirección IPv6, sin necesidad de configuración manual por parte del usuario, o de un servidor DHCPv6 corriendo en la red, el mecanismo antes mencionado, coloca la dirección física de dicha interfaz en la porción de la dirección IPv6 que hace referencia al identificador de la misma, según la figura 4.3. Por otro lado, se utilizará el prefijo *fe80::/64* previsto para este tipo de direcciones.

Sin embargo, aunque la dirección física de la interfaz es única, esto no asegura que otro nodo en la red local no esté utilizando esta misma dirección. Para resolver esto, el mecanismo en discusión contempla la validación de direcciones duplicadas, el cual consiste en enviar un mensaje *Nighbor Solicitation* (el mismo utilizado para obtener la dirección física de un nodo) consultando por la dirección IPv6 que se pretende utilizar. Si al cabo de un tiempo, no se reciben respuestas, significa que la dirección puede ser utilizada. En caso contrario, se detiene este procedimiento automático y la configuración debe hacerse manualmente.

En verdad, el procedimiento explicado va más allá de las direcciones de ámbito local. También se utiliza para direcciones globales. En este caso, también se determinará automáticamente el prefijo del sitio en el cual se está conectado físicamente. En este caso, se enviará un mensaje *Router Solicitation* al router que actúa como gateway del sitio, y el mismo contestará con la información requerida, en un mensaje *Router Advertisement*. Estos dos tipos de mensaje también se definen en el RFC 4861 [16].

Es así como se obtiene automáticamente el prefijo de la red en la que el nuevo nodo está conectado. Este dato, en conjunto con la dirección física determinarán la dirección IPv6 que se asignará a la interfaz de red.

Como se dijo con anterioridad, en la presente Tesis se considera la red ANTop como aislada y por ende no hay restricciones acerca del tipo de dirección a utilizar. Sin embargo, la discusión recién presentada es importante para el trabajo a futuro de la consolidación de ANTop como protocolo de redes ad-hoc.

Si bien utilizando direcciones de tipo link local se tiene un mecanismo definido para que no se dupliquen las direcciones de aquellos nodos que aún no se unieron a la red ANTop, aún puede suceder que la dirección de dichos nodos coincida con una presente en la red ANTop. De todas formas, como se verá más adelante, en la sección 4.5, no fue posible utilizar está configuración debido a un problema relacionado a la manera en que el SO maneja los pedidos DNS, los cuales tienen gran relevancia en el funcionamiento del servicio *Rendez-Vous* que incorpora ANTop. Este último permite resolver direcciones universales a relativas. Superar esta limitación forma parte del trabajo a futuro.

Al no poder utilizar direcciones de tipo link local debido al problema de resolución de nombres, se debió optar por las de tipo *Global Unicast*. Como se puede ver en la figura correspondiente 4.2, si bien no se define una cantidad específica de bits para cada campo, uno de ellos se utiliza para identificar al sitio. En el caso del presente trabajo, se tiene una red ad-hoc en la cual el sitio es único. Por esta razón, se puede pensar en un formato de dirección IPv6 con rango global, sin la necesidad de destinar m cantidad de bits a la identificación de la subred. En este modelo, será el campo *Interface ID* el que alberga la dirección relativa del nodo ANTop. Este es finalmente el esquema de mapeo que se emplea para llevar las direcciones relativas a direcciones IPv6.

Por último se debe tener en cuenta que al trabajar con direcciones IPv6, cuando se intente mandar un paquete a todos los nodos de la red, se deberá usar la dirección multicast IPv6, *FF02::1* como destino. Este será el caso de los paquetes de control ANTop PAR, PAN, y HB.

En lo que respecta a la capa de enlace, este paquete tendrá la dirección broadcast MAC *FF:FF:FF:FF:FF:FF* como destino.

4.2.2. Conexión de nodos

Para desarrollar el mecanismo de conexión de los nodos a la red ANTop, se toma un ejemplo con los siguientes parámetros.

- *n=64*
- *m=0*
- *Global Routing Prefix = 2001:0000:0000:0000*

En la sección A.1 se puede consultar el algoritmo de conexión de nodos, que se explica a continuación.

El primer nodo que se conecta a la red, no recibirá respuesta a cambio del envío del paquete PAR. Por esta razón el mismo se asignará la primera dirección del hipercubo. En verdad, ANTop prevé una primer dirección relativa cuyos bits tienen un valor de cero en su totalidad. Como se dijo en la sección 4.2.1, la dirección relativa estará representada por el campo *Interface ID*, con lo cual se espera tener un valor nulo para el mismo. Esto representa un problema, ya que el sistema operativo no permite instalar rutas que tengan como próximo salto la dirección de red propia, la cual concordaría con la del primer nodo conectado a la red. Esto se ejemplificará más tarde en el capítulo de simulaciones, cuando se vea que al tomar una dirección ANTop, se genera una entrada en la tabla cuyo destino es *2001::*, en el caso de los parámetros dados más arriba.

Para resolver este inconveniente, se toma una primer dirección de red para la cual el bit menos significativo del campo de dirección relativa ANTop tiene un valor de uno. Es decir que para los parámetros de este ejemplo, la primer dirección del hipercubo será *2001::1*.

El segundo nodo que quiera conectarse a la red, enviará un paquete PAR, en modo multicast, es decir a la dirección *FF02::1*. Normalmente, la tabla de ruteo del sistema operativo, incluye una entrada hacia dicha dirección de broadcast, indicando que el paquete debe salir por la interfaz inalámbrica. Cuando el primer nodo participante de la red, recibe dicho pedido, debe responder con un paquete PAN, cuyo destino será la dirección del segundo nodo, es decir que no será enviado en modo broadcast.

Con el fin de que tanto el PAP, como el resto de los paquetes de control ANTop con un destino único puedan ser ruteados por la interfaz inalámbrica, la máscara de la interfaz de red debe ser configurada con el mismo valor que n, es decir la longitud del campo *Global Routing Prefix*. De lo contrario, el sistema operativo no instalará la ruta necesaria para que los paquetes destinados a otros nodos ANTop puedan salir por la interfaz inalámbrica.

Figura 4.4: Primer nodo conectado a la red ANTop

Luego de enviar el PAR, el nodo 2 espera un determinado tiempo para colectar las ofertas de dirección relativa y vencido dicho intervalo, elige la opción con mayor máscara. Es decir, el espacio de direcciones más grande. En este caso, solo recibirá una respuesta que será

Dirección 2001:0000:0000:0000:8000:0000:0000:0001

Máscara 1

En este caso, el valor de la máscara ANTop, se aplica solo a los 64 bits que forman parte de la identificación de interfaz.

Una vez que el segundo nodo acepta esta dirección, el primero también incrementa su máscara a un valor de uno.

Como puede verse, los bits se modifican empezando por el más significativo, del MSB de la identificación de interfaz. Dicho byte en la dirección del segundo nodo, presenta el siguiente valor

- 10000000 = 128

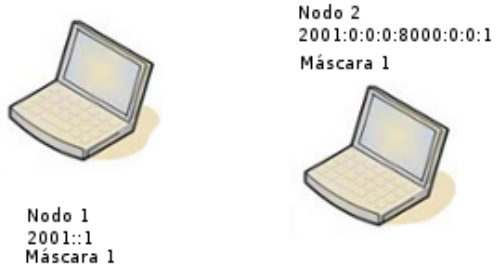


Figura 4.5: Segundo nodo conectado a la red ANTop

Aquí cabe aclarar que la máscara ANTop es un valor guardado en una variable del Espacio de Usuario, y es independiente del valor de la máscara de red que tiene la interfaz.

Cuando un tercer nodo se conecta a la red, si el mismo está en el alcance de los dos primeros, recibirá dos opciones de dirección relativa.

Dirección 2001:0000:0000:0000:c000:0000:0000:0001

Máscara 2

y,

Dirección 2001:0000:0000:0000:4000:0000:0000:0001

Máscara 2

La dirección que ofrece el nodo uno surge de asignar un valor de uno al bit más significativo del espacio de hosts ANTop. Es decir,

El valor del MSB del espacio de hosts de la dirección relativa del nodo 1 es 00000000, con máscara 1, con lo cual ofrece una dirección que solo difiere en un bit, de modo que esté byte vale 01000000, con máscara 2.

El valor del MSB del espacio de hosts de la dirección relativa del nodo 2 es 10000000, con máscara 1, con lo cual ofrece una dirección relativa para la cual cambia este byte. El mismo adquiere un valor de 11000000, con máscara 2.

En este caso, las dos opciones tienen la misma máscara ANTop, con lo cual el nodo 3 elegirá alguna de ellas indistintamente.

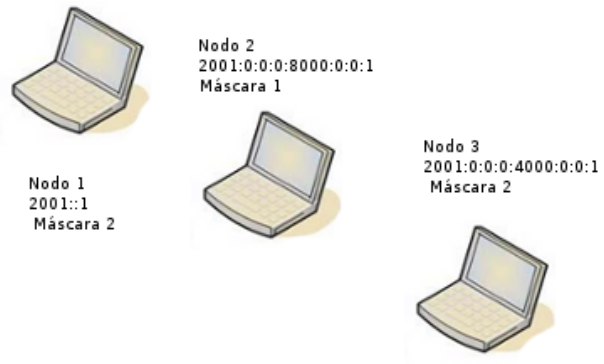


Figura 4.6: Tercer nodo conectado a la red ANTop

Como se dijo anteriormente ANTop presenta mecanismos para lidiar con la pérdida de paquetes de control. En lo que se refiere a la asignación de direcciones, el mecanismo es el siguiente.

El nodo que se quiere conectar a la red envía un paquete PAR en modo broadcast. Aquí se dispara un timer, y durante la vida del mismo se colectan respuestas, en forma de paquetes PAP (Primary Address Proposal). Estos paquetes UDP, tiene en su payload una estructura cuyos campos son, entre otros, la dirección propuesta, la máscara, el tipo de mensaje ANTop. Cada una de estas estructuras recibidas se guarda en el Espacio de Usuario en un vector de respuestas. Una vez vencido el timer, se chequea la longitud del vector. Si se recibieron respuestas, se elige la de mayor máscara, como ya se dijo, pero si la longitud del vector es nula, entonces puede haberse perdido el PAR, o sus respuestas, es decir los PAP. En este caso se envía nuevamente un PAR y se espera el mismo intervalo de tiempo por respuestas. Este proceso se repite un número configurable de veces, y si no se recibió respuesta entonces se considera que es el primer nodo de la red y se asigna el mismo una dirección de hipercubo.

En el caso de que se reciban respuestas, se envía un paquete PAN (Primary Address Notification), informando la dirección elegida. Este paquete se envía en modo broadcast para que todos los nodos que propusieron una dirección sepan que una fue elegida. El nodo que ofreció la dirección elegida cederá ese espacio de direcciones, y enviará un paquete de tipo PANC (Primary Address Notification Confirmation)

Si el paquete PAN que envía el nodo que quiere conectarse a la red, se perdiese, entonces el nodo que ofreció la dirección elegida no enviará el PANC, y por lo tanto el proceso de conexión del nuevo nodo queda incompleto. Frente a este escenario, el nodo entrante vuelve a iniciar el ciclo de envío de paquetes PAR.

Una vez que el proceso de conexión de un nodo ha finalizado, el mismo tiene una dirección relativa en el hipercubo y será vecino de todos aquellos nodos que cumplan con dos condiciones,

- Estar dentro del alcance de la placa inalámbrica.
- Diferencia de un bit en las direcciones relativas.

Para el esquema de ruteo que propone ANTop, las relaciones entre vecinos tienen gran importancia. En el caso de que no se tenga una ruta previamente instalada, el demonio que corre ANTop en el Espacio de Usuario, enviará el paquete por el vecino que esté más cerca al destino. La distancia, como se vio anteriormente, se define como la cantidad de bits que difieren entre las dos direcciones. Es decir que la segunda condición para que sean vecinos se reformula como distancia unitaria.

Al momento de rutear un paquete, cada nodo debe tener conocimiento de cuáles de sus vecinos están activos en la red, de lo contrario, se perderán paquetes por ser reenviados a nodos que ya no forman parte de la red. Para lograr esto, cada nodo envía periódicamente un paquete de control ANTop con tipo *Heart Beat* en modo broadcast. Esto último se logra con un timer al cual se le asocia una función de callback que una vez que el mismo vence, vuelve a dispararlo con el mismo tiempo de vida.

Cuando un nodo recibe un *Heart Beat*, chequea en su tabla de vecinos alojada en el Espacio de Usuario si ya tiene registrado el origen como un vecino. En caso negativo, agrega una entrada al final de la tabla (generada por una lista de registros), en caso positivo se marca mediante una bandera que un nuevo *Heart Beat* ha sido recibido para ese nodo.

Periódicamente, se recorre la tabla de vecinos y se chequean cuales vecinos no han enviado *Heart Beats*. En esos casos, se da un valor de uno a una variable que representa la cantidad de veces que no se ha recibido este aviso de actividad. Se repite este mecanismo, y se incrementa la variable si

corresponde. Cuando llega un *Heart Beat* de ese vecino, la variable toma un valor nulo nuevamente. Cuando la misma llega a un determinado valor, configurable, se considera que el nodo ya no está en la red.

4.2.3. Desconexión de nodos

Cuando un nodo se desconecte, sus vecinos dejarán de recibir sus *Heart Beats*. Al cabo de un determinado tiempo, ellos considerarán que el mismo abandonó la red, y lo eliminarán de su tabla de vecinos. Quien sea el padre del nodo, recuperará el espacio de direcciones cedido cuando el mismo se conecto, de modo tal que puede ser utilizado nuevamente. Cuando un nuevo nodo intenta conectarse dicho espacio es el primero en ofrecerse. Una vez que todos los recuperados fueron cedidos, se continua con la cesión del resto de las direcciones.

Como se verá en la sección 4.5, este nodo podrá almacenar información de direcciones relativas y universales de otros nodos. Al desconectarse, esta información no se perderá ya que la misma se encontrará duplicada en otros servidores de nombre.

4.3. Ruteo en ANTop sobre IPv6

En esta sección se estudia la utilización Netfilter para capturar los paquetes que entran y salen de un nodo, con el fin de enviarlos al Espacio de Usuario. Luego se analiza el modo de rutear dichos paquetes. Para eso se estudia el uso de Netlink para interactuar con la tabla de ruteo, finalmente se presenta el mecanismo de exploración de vecinos en búsqueda de una ruta, y se introduce el concepto de *Visited Bitmap*.

El algoritmo general de ruteo, se puede consultar en 3

Cuando un paquete llega al nodo, el mismo es enviado al Espacio de Usuario a través de la función que registra Netfilter en el gancho PRE_ROUTING. En el caso de que el destino sea el nodo local, entonces no se ejecuta ninguna acción y se permite que el mismo continúe su viaje a lo largo de la pila de protocolos de red. Si, por ejemplo, en el caso recién mencionado, el paquete es tráfico de control ANTop, entonces el mismo seguirá su curso normal, y será recibido por la función de lectura asociada al socket UDP que escucha el puerto ANTop definido previamente. De igual forma sucederá con cualquier otro tipo de tráfico destinado a un proceso del Espacio de Usuario.

Si por el contrario, el paquete tiene como destino una dirección relativa distinta a la propia, entonces sucede que este nodo se trata de un eslabón en la cadena de ruteo entre origen y destino. En este caso de deberá chequear en la tabla de ruteo del SO si se tiene una entrada con el destino apropiado. De no ser así, se deberá buscar una. Esto también es válido cuando el tráfico se genera localmente y debe ser enviado a un destino particular.

Para poder rutear el paquete, es necesario consultar la tabla de ruteo. Es aquí donde entra en juego la estructura Netlink. En primer instancia se crea un socket cuyo dominio 3.2 será AF_NETLINK, la familia será SOCK_DGRAM ó SOCK_RAW indistintamente, mientras que el protocolo será NETLINK_ROUTE. Luego se debe crear una estructura que representa el encabezado del paquete Netlink que se enviará al Espacio de Kernel. En este encabezado se indica que se estará haciendo un pedido de rutas. Cada una de ellas será devuelta como un atributo, de la respuesta Netlink, razón por la cual también se debe especificar que se desean obtener todas las entradas de la tabla, ya que de lo contrario estaremos recibiendo solo una. Por otro lado se aloja el payload, dentro del cual se incluye información específica del protocolo. En este caso se especifica que estaremos pidiendo rutas IPv6.

Una vez completado el paquete Netlink se envía a través del socket creado anteriormente, y luego se escucha el mismo para recibir la respuesta. Para el caso de la tabla de ruteo, se recibirá un paquete de Netlink por cada entrada. En el payload del mismo habrá un atributo por cada campo de la entrada, como ser el destino, el próximo salto o la máscara. Otro campo posible será el número de tabla de ruteo, siendo las más relevantes,

255 LOCAL Aquí se agregan automáticamente las direcciones de las interfaces de red del nodo.

254 PRINCIPAL Aquí es donde se agregarán las rutas generadas por el demonio de ANTop.

Para volcar la información de la tabla de ruteo al Espacio de Usuario, se define una lista de estructuras, donde cada una de ellas representa una ruta. Por cada mensaje Netlink que se recibe, se completa una estructura de ruta, cuyos campos copian los valores recibidos en los atributos.

Al finalizar la lectura de la tabla, el socket se cierra.

En este punto ya se tiene una copia de la tabla de ruteo en el Espacio de Usuario. A continuación se recorre la misma con el fin de determinar si existe una ruta para llegar al destino del paquete. Si este no fuera el caso, se dispara el mecanismo de búsqueda de ruta descripto en el algoritmo 2.

A grandes rasgos, este mecanismo de búsqueda, se trata de una exploración de los vecinos. Básicamente se envía el paquete hacia el que tiene menor distancia al destino y, aunque no se tiene seguridad de que exista una ruta efectiva a través de ese vecino, la misma es instalada con el primer intento de exploración. Para esto se emplea un mecanismo similar al de la consulta. En primer medida, se arma la estructura que representará el encabezado del paquete Netlink. En ella se especifica que se estará *creando* una ruta. Luego, en el payload se agrega un encabezado de protocolo en el que se indica que las direcciones serán IPv6 y que la tabla será la principal, entre otros datos. Finalmente se agrega un atributo por cada uno de los ítem que conformarán la ruta.

Una vez armado el paquete Netlink, el mismo se envía a través de un nuevo socket creado para este fin. Esta vez no es necesario seguir escuchándolo, por lo cual se cierra luego del envío.

Luego de agregada la ruta a la tabla, se configura un timer que invoca al borrado de la misma una vez que este expira. De esta forma se asegura la búsqueda de una nueva ruta luego de un tiempo, y por ende no rutear paquetes por caminos que ya no son válidas.

Una de las situaciones usuales que surgirá con el esquema de ruteo de ANTop es que al recibir un paquete que deba ser ruteado hacia un cierto destino, no se tengan más vecinos disponibles que el último salto por el que pasó. En esa situación el mismo será devuelto para que el nodo que lo envió continúe la búsqueda a través de otros vecinos. Con el fin de llevar un registro de los vecinos que el paquete ya visitó, se utiliza un *visited bitmap* el cual se compone de un vector de banderas por cada ruta instalada en la tabla. La cantidad de banderas es igual para todos los vectores. Cada una de ellas hace referencia a un vecino disponible, y se implementa como un bit que tendrá el valor de uno si ese vecino ya fue visitado al intentar rutear un paquete al destino o cero en el caso contrario.

La figura 4.7 muestra la forma del *Visited Bitmap*. La misma se muestra solamente a modo de ejemplo, y es tomada de la Tesis de Alejandro Marcu [1],

específicamente de la página 24 en el capítulo 2. En verdad, para la presente implementación se utilizarán direcciones IPv6, en lugar de cadenas de bits sin formato, como muestra la figura.

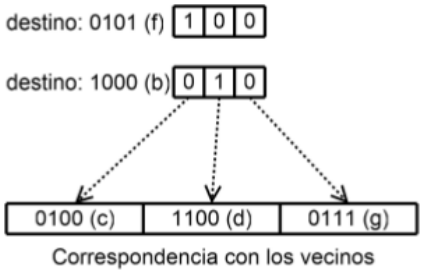


Figura 4.7: *Visited Bitmap*

De esta forma, cada ruta que se instala en la tabla del Kernel, se mapea a un vector de *Visited Bitmap* en el Espacio de Usuario.

Volviendo al caso en que el paquete es devuelto al último salto, se debe tener en cuenta que en verdad, no hay información en el paquete acerca de cual fue el último salto. Si bien en el encabezado de capa 2 se cuenta con la dirección física, no se tiene la dirección IPv6. Para incluir esta información en el paquete, se utilizan encabezados de extensión E, en particular *Opciones de Destino*.

Ruteo de paquetes *Rendez-Vous*

En el caso de este tipo de paquetes, el servidor *Rendez-Vous* destino se determina mediante una función de *Hash*. Por esta razón es probable que esa dirección no haya sido asignada a ningún nodo aún. Será el dueño del espacio de direcciones de dicho destino el que deba procesar el paquete. Para ello, cuando el mismo alcanza el gancho PRE_ROUTING, se chequea si la bandera que identifica al tráfico *Rendez-Vous* se encuentra activa. En ese caso, se chequea si el destino se encuentra en el espacio de direcciones que administra el nodo. De ser así, el paquete es procesado a pesar de no tener como destino la dirección local, y se le devuelve a la estructura de Netfilter la indicación de que lo descarte, ya que el mismo no debe seguir su camino a través de la pila de protocolos de red, de lo contrario sería ruteado incorrectamente por el sistema operativo.

Para el ruteo de paquetes de *Rendez-Vous*, al calcular la distancia se contempla la máscara del nodo. Por ejemplo, si el destino es 110111 y un nodo tiene dirección 010000/2, entonces la distancia utilizada en el ruteo de datos es 4, ya que hay 4 bits distintos. En cambio, en el ruteo de *Rendez-Vous*, para calcular la distancia se toman solamente los dos primeros bits de la dirección (dado por la máscara), y por lo tanto la distancia es 1.

4.4. TCP sobre ANTop

Bajo el esquema de implementación de ANTop sobre IPv6, se debe hace una consideración para que el funcionamiento de aquellas aplicaciones que utilizan TCP [17](Transmission Control Protocol) como protocolo de transporte.

Cuando el nodo cliente desea establecer una sesión TCP para intercambiar datos, envía un primer paquete al nodo servidor, con la bandera *SYN* activada. Además, en el campo de opciones, se envía el parámetro *MSS* (Maximum Segment Size). Este valor representa el máximo tamaño de segmento de datos que se enviará al lado opuesto.

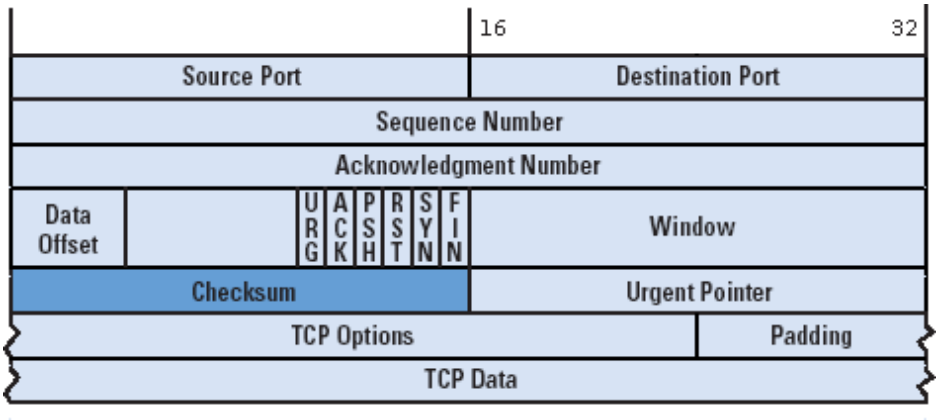


Figura 4.8: Encabezado TCP

Con el fin de calcular este valor, es necesario tener en cuenta un parámetro de configuración de la interfaz inalámbrica. Este es el *MTU* (Maximum Trasfer Unit). El MSS será como máximo el MTU, menos el tamaño de los headers IPv6, y TCP. Normalmente, utilizando Ethernet en la capa de enlace, el valor del MTU será de 1500 bytes, mientras que el tamaño de los headers asciende a 60 bytes, 40 bytes por IPv6 y 20 por TCP. De este modo

un valor típico de MSS es 1440 bytes. Sin embargo, aquí se debe recordar que ANTop utiliza un encabezado de extensión, según lo visto en 4.3. Dicho encabezado consume 32 bytes, resultando en un MSS de 1408 bytes.

Cuando el cliente envíe un paquete TCP con la bandera SYN activa, para iniciar una sesión, el mismo será interceptado por el gancho LOCAL_OUT de la estructura Netfilter, y será enviado al Espacio de Usuario. Allí se comprobará el valor de la bandera SYN, y luego se modificará el valor del MSS.

4.5. Resolución de nombres, servicio *Rendez-Vous*

Una de la características más distintivas de las redes ad-hoc es que los nodos que en ellas participan gozan de gran movilidad. Es por esta razón que la dirección dependiente de la ubicación en la red, cambiará frecuentemente. Esto último se vuelve incomodo cuando se intenta establecer una comunicación con el nodo, ya que se debe hacer un seguimiento de esta dirección. Para independizar la identificación del nodo, de su dirección relativa, se utiliza la dirección universal. La misma será una cadena de caracteres, por ejemplo *nodo_antop_matias*.

Según el esquema de resolución de nombres presentado en [1] (sección 2.1.2), cada nodo que ingresa a la red, deberá registrarse en su servidor *Rendez-Vous*. La información de resolución de nombres, se encuentra dispersa a lo largo de toda la red, es decir que no hay una entidad centralizada que la administre.

Cada nodo utilizará una función de *Hash* aplicada a su dirección universal (la cual es asignada por el usuario del nodo), y como resultado obtendrá una serie de direcciones que serán los servidores *Rendez-Vous* en los cuales deberá registrarse. Con este esquema se obtiene redundancia de la información en caso de que un nodo se desconecte en forma imprevista, ya que la información de resolución de nombres en el contenida, se encuentra replicada en un número de servidores *Rendez-Vous*, que dependerá de la dimensión de la red.

Para concertar el registro de la dirección en el servidor *Rendez-Vous*, el nodo entrante envía un paquete de control ANTop, en la misma forma que un PAR, solo que en lugar de ser un multicast a todos los nodos, se envía a un grupo de nodos en particular. Este registro se envía periódicamente, con

lo cual la información se mantiene actualizada. Para estos registros también se asocia un timer que luego de vencido borra la entrada.

El algoritmo 6 describe como un servidor *Rendez-Vous* procesa los pedidos de registro.

Hasta aquí se ha visto el mecanismo por el cual los nodos se registran en los servidores *Rendez-Vous*. Ahora se verá como cada nodo hace uso de la información de nombres alojada en dichos servidores para comunicarse con aplicaciones en otros nodos.

Cuando el usuario de una PC que participa en una red ANTop, desea entablar una conversación a nivel de aplicación de otro nodo, debe hacer referencia o bien a su IP, o bien a su nombre de host. Aquí se debe recordar que el funcionamiento de ANTop sobre IPv6, será transparente a las capas superiores, es decir que se utiliza la estructura de comunicación, basada en una pila de protocolos de comunicación, ya existente. Para utilizar cualquier aplicación, el usuario tiene la posibilidad de utilizar la dirección IPv6 del nodo con el que quiere comunicarse, la cual contendrá la dirección relativa, embebida en el modo visto en la sección 4.2.1. Además contará con otra opción de referencia al nodo del otro extremo, que será la dirección universal. Esta opción será la predilecta, ya que es mucho más significativa para el usuario a la hora de hacer referencia a otro nodo.

Los algoritmos 6, 4 y 5 describen el mecanismo mediante el cual los nodos resuelven direcciones universales a relativas, y a continuación se encara una explicación detallada de dicho mecanismo.

Con el objetivo de presentar el modo en el que ANTop resuelve las direcciones universales a relativas (ruteo indirecto), se presenta un ejemplo, en el que se supone que el usuario local desea conocer el tiempo de respuesta a otro nodo, para el cual conoce que su dirección universal es *nodo_ antop_ matias*. Este usuario, típicamente, abrirá una terminal de comandos y ejecutará una línea del estilo *ping nodo_ antop_ matias*. Este comando disparará una consulta DNS en la siguiente forma,

Típicamente, al utilizar la herramienta Ping, se indica la interfaz mediante la cual se desea rutear el tráfico. En este caso, Linux detecta que dicha interfaz está configurada con un esquema de dirección IPv6, emitiendo por lo tanto una consulta DNS para obtener una dirección con ese mismo formato. Esta

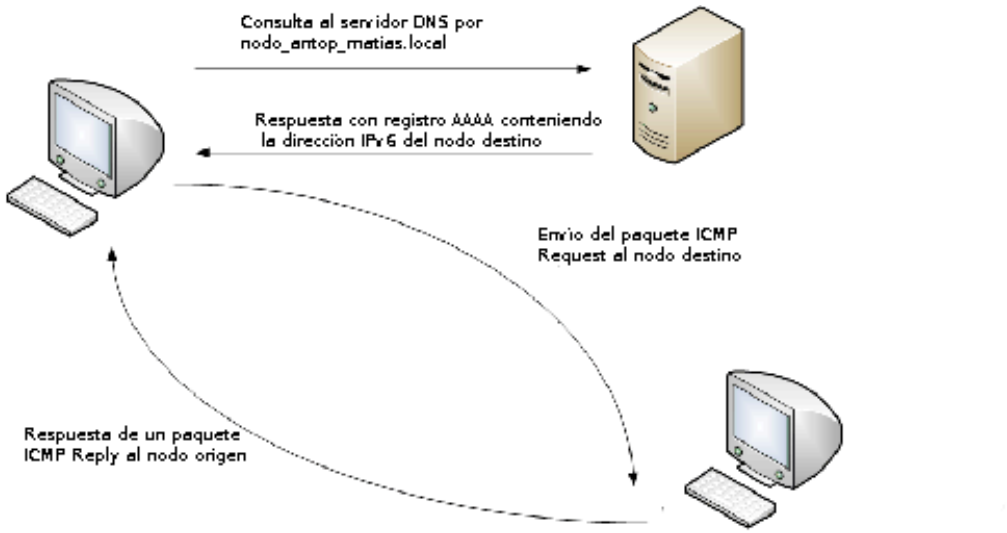


Figura 4.9: Consulta tradicional DNS IPv6 a un servidor del dominio local

consulta es en verdad un pedido de registro $AAAA$, que asocia una IP a un nombre de host, y se denomina de tal forma haciendo referencia a la longitud de dichas direcciones, las cuales constan de 128 bits, mientras que las de IPv4 constan de 32 bits, protocolo para el cual el tipo de consultas se denomina A.

Para resolver las direcciones universales, ANTop se vale del esquema DNS recién presentado, pero con algunas modificaciones. Sin embargo, para la capa de aplicación, se envía una consulta por un registro $AAAA$ y se recibe una respuesta con dicho requerimiento. Es decir que el mecanismo de ruteo indirecto es transparente para el sistema operativo, como ya se menciono anteriormente.

Aquí se debe recordar que el mecanismo de resolución de nombres de ANTop es descentralizado, y por lo tanto no existe una estructura centralizada en la que un nodo en particular, se encargue de albergar la información de nombres. Por esta razón la figura de un servidor DNS deja de existir, al tiempo que su función se distribuye entre todos los nodos de la red ANTop.

La consulta DNS da origen a un paquete que en su búsqueda por el servidor de nombres, alcanzará el gancho LOCAL_OUT de la estructura Netfilter. Este paquete será enviado al Espacio de Usuario, más precisamente al demonio que corre ANTop. Este último comprobará que es tráfico UDP, mirando

el campo *Next Header* del encabezado de opciones IPv6. Luego chequeará el puerto destino, verificando que se trata de tráfico DNS. Finalmente, comprobará que se trata de un pedido de resolución de nombre, mirando el segundo campo del paquete DNS, el de banderas. Más específicamente la primera de izquierda a derecha en la figura 4.10, llamada *QR*. Esta bandera de un bit, tendrá un valor de cero en caso de que el mensaje se trate de un pedido, y de uno en caso de que sea una respuesta.

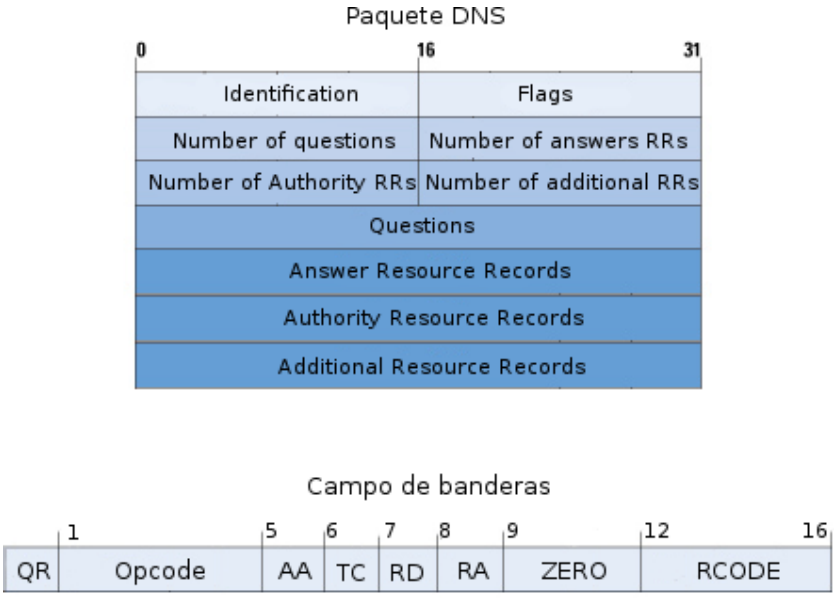


Figura 4.10: Estructura general de un paquete DNS

Habiendo corroborado que es un pedido DNS, el siguiente paso es chequear la tabla cache de direcciones, en la que se guardan las últimas consultas resueltas. Si allí no se encuentra una entrada con la dirección requerida, entonces se envía un paquete de control ANTop, de tipo Address Lookup, que contiene la dirección universal del nodo para el que se quiere obtener la relativa. Por otro lado, se guardará el paquete con el pedido DNS, el cual será necesario a la hora de armar una respuesta.

El destinatario del paquete será el servidor *Rendez-Vous* encargado de resolver esa dirección. Sin embargo, como se vio recientemente cada par formado por la dirección universal y la relativa, está registrado en un número de nodos que dependerá de la dimensión del hipercubo. El pedido se enviará, concretamente, al servidor más cercano, es decir el que tenga una dirección relativa con menor distancia a la dirección del nodo local.

Este paquete es de tipo *Rendez-Vous*, con lo cual el nodo que lo rutea deberá chequear si el destino se encuentra dentro de su espacio de direcciones.

El primer paso al procesar el pedido, será buscar en la tabla *Rendez-Vous* la dirección universal contenida en el paquete. Esta tabla contiene los pares dirección universal - relativa. Independientemente del resultado de la búsqueda, enviará un paquete de control *Rendez-Vous* de tipo Address Solve. En el caso de que la consulta sea resuelta, se debe activar la bandera *MASK_SOLVED* del campo correspondiente en la estructura de control ANTop que viaja en el paquete UDP.

Cuando el pedido Address Lookup alcanza el gancho *LOCAL_OUT* en su camino fuera del nodo local, se chequea si el mismo tiene un destino dentro del espacio de direcciones que el propio nodo administra. Esta comprobación se hace por ser un paquete *Rendez-Vous*. En caso positivo, es el nodo local el servidor *Rendez-Vous*, y por ende se realizará el proceso de búsqueda arriba mencionado, localmente.

Este esquema también podrá darse en el caso del registro de nombre. Será el nodo local quien guarde su propia información de direcciones en la tabla *Rendez-Vous*. Por ejemplo, el primer nodo de la red ANTop, será dueño de todo el espacio de direcciones del hipercubo, con lo cual deberá registrarse a si mismo. Si bien no habrá otros participantes con los que interactuar, será posible ejecutar un *Ping* a la dirección IPv6 de la interfaz inalámbrica invocando la dirección universal.

Volviendo al proceso de resolución de nombres, cuando el nodo que emitió la consulta recibe el paquete Address Solve, si el mismo tiene la bandera de dirección resuelta activa, entonces se está en condiciones de informar a la aplicación que originó el pedido, la dirección IPv6 con la cual deberá comunicarse. Para lograrlo, se usará el servicio DNS montado nativamente en el sistema operativo. Para ello se debe tener en cuenta que desde su punto de vista, habiendo realizado un pedido de resolución de nombre, ahora se está esperando una respuesta por parte del servidor. Como ya se dijo antes, no hay un servidor DNS, por ser este un esquema descentralizado de resolución de nombres. Será el demonio que corre ANTop en la capa de aplicación quien se haga cargo de dar la respuesta en nombre de dicho servidor. Para ello, se utiliza el paquete DNS guardado al emitir la consulta *Rendez-Vous*, creándose la respuesta en base al mismo.

Dicha respuesta será, también, un paquete DNS, que se enviará a través de un socket UDP creado para tal fin. Se tomará como dirección destino, la local, de modo tal que es el propio nodo quien se hace cargo de la misma. Este paquete autogenerado, viajará por la pila de protocolos de red hasta ser procesado por el servicio de DNS corriendo en el sistema operativo. Este último comunicará a la aplicación que disparó el pedido, la dirección IPv6 que deberá utilizar. Cerrando así el proceso de resolución de nombres, cuyo motor es el servicio *Rendez-Vous*, enmascarado bajo la estructura del servicio DNS corriendo en el sistema operativo. Por esta razón ANTop introduce un mecanismo transparente para brindar esta funcionalidad.

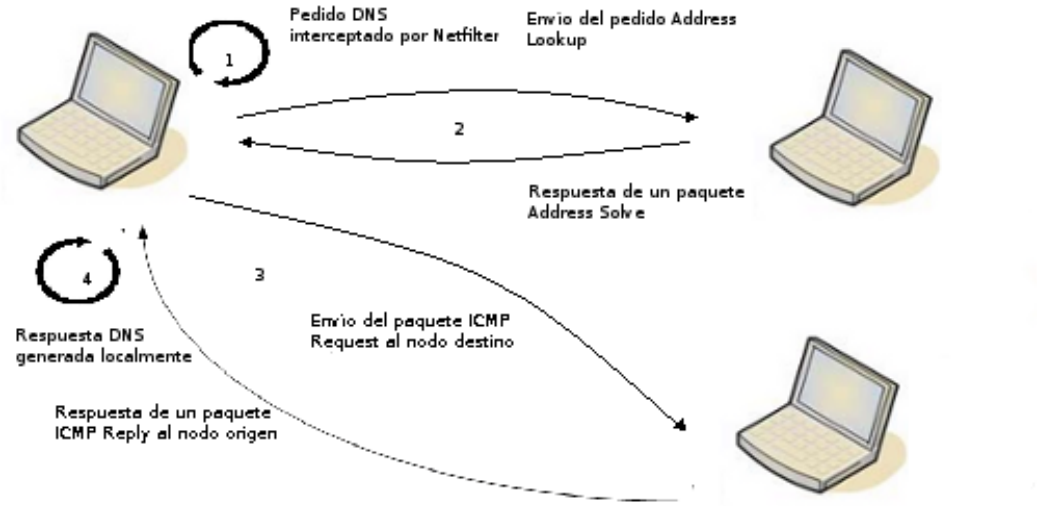


Figura 4.11: Resolución de nombres mediante *Rendez-Vous*

El esquema presentado hasta aquí hace uso de la estructura del servicio DNS montada en el sistema operativo Linux, sin embargo, para que el funcionamiento de este diseño sea el esperado, se debe tener en cuenta una consideración acerca de la configuración de la dirección IPv6 de la interfaz de red.

Según lo mencionado en la sección 4.2.1, se debió utilizar direcciones de tipo *Global Unicast*, ya que para las de tipo *Link Local*, en lugar de generarse pedidos DNS por requerimiento de las aplicaciones, el sistema operativo Linux genera paquetes de tipo mDNS *Multicast DNS*[1]. Estos paquetes se generan en respuesta a un servicio de resolución de nombres implementado

[1] http://www.multicastdns.org/

en Linux, para aquellas redes que no disponen de un servidor DNS. Este servicio se vale de consultas en modo multicast en lugar de las unicast, dirigidas a un servidor, que utiliza DNS. En este caso, serán los propios nodos que participan de la red, quienes guarden los registros de nombres.

Durante el desarrollo de la presente Tesis no fue posible generar las respuestas mDNS localmente al igual que se lo hace con aquellas de tipo DNS y por esa razón no se pudo avanzar en la implementación del servicio *Rendez-Vous* sobre la estructura mDNS montada en Linux. El análisis de factibilidad y posteriormente de funcionamiento de esta solución no es contemplado dentro de esta Tesis y se considera parte del trabajo a futuro.

Finalmente, al configurar la interfaz de red inalámbrica con una dirección de tipo *Global Unicast*, las consultas generadas son de tipo DNS, y el esquema adoptado para la resolución de direcciones universales hacia relativas es el desarrollado anteriormente en esta sección.

Capítulo 5

Pruebas de funcionamiento de ANTop

En este capítulo se encaran pruebas de funcionamiento de la implementación de ANTop sobre IPv6, según el protocolo de prueba definido en la sección D. En dicho apartado se definen más pruebas que las que aquí se realizan, quedando como trabajo a futuro completar las mismas.

Este capítulo se divide en tres secciones. La primera presenta la estructura de los paquetes ANTop utilizados en esta implementación y las diferencias con los paquetes originales de la definición del protocolo. Las restantes dos secciones presentan las pruebas, clasificadas como de bajo y alto nivel.

5.1. Paquetes de control ANTop

Antes de comenzar con las pruebas, se verá el formato de los paquetes utilizados y su diferencia con aquellos definidos durante el diseño de la plataforma de simulación QUENAS [1] iniciado en la Tesis de grado de Alejandro Marcu [1] y continuado en la Tesis de grado de Gaston Tejia [2].

En la figura 5.1 se puede ver la definición original de los paquetes de control ANTop.

packet type El tipo del paquete en 5 bits, permitiendo hasta 32 tipos distintos.

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

Figura 5.1: Diseño original de un paquete de control genérico ANTop

flags 3 flags de uso general, que cada tipo de paquete puede asignar para lo que necesite.

total length Longitud total del paquete, incluyendo encabezados opcionales

MAC Address Dirección física del nodo que envía el paquete. Se utiliza para identificar unívocamente a los nodos cuando aún no poseen una dirección de hipercubo

Primary Address Length Longitud en bits de la dirección primaria (dimensión del hipercubo)

Primary Address Dirección primaria, almacenada en n bits, siendo n el mínimo múltiplo de 8 mayor que Primary Address Length

Optional Headers Cada paquete puede transportar encabezados opcionales.

En la implementación de ANTop aquí encarada se sigue el mismo esquema, es decir que se define un paquete de control genérico, cuyos campos tomarán distintos valores en función de la función requerida.

En la figura 5.2 se puede ver el paquete de control usado en esta Tesis, mientras que en la figura 5.3 se muestra la forma del encabezado opcional *Additional Address*.

A continuación se detalla el valor de los cada uno de los campos de un paquete de control genérico para cumplir con las diferentes instancias del proceso de conexión de nodos.

*** Paquete PAR *(Primary Address Request)***

packet type Este campo toma el valor de 1.

source address La dirección IPv6 asignada a la interfaz inalámbrica, antes de correr el demonio ANTop.

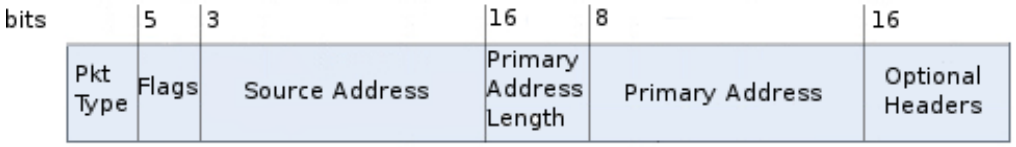


Figura 5.2: Paquete de control utilizado en esta Tesis

Figura 5.3: Encabezado opcional *Additional Address*

*** Paquete PAP *(Primary Address Proposal)***

packet type Este campo toma el valor de 2.

primary address La dirección IPv6 del nodo, asignada para participar en la red ANTop

primary address length La dimensión del hipercubo

optional headers Se incluye un encabezado opcional de tipo *Additional Address*.

El encabezado opcional contiene los siguientes valores,

mask El valor de la máscara ANTop si se tiene una dirección relativa que ofrecer, ó cero en otro caso.

address La dirección IPv6 que contiene embebida la dirección relativa ofrecida al nodo que quiere conectarse.

*** Paquete PAN *(Primary Address Notification)***

packet type El tipo de paquete vale 3 en este caso.

primary address La dirección IPv6 elegida para participar en la red ANTop.

source address La dirección IPv6 de la interfaz inalámbrica antes de conectarse a la red ANTop.

optional headers Se incluye un encabezado opcional de tipo *Additional Address*.

El encabezado Additional Address tiene los siguientes valores en sus campos,

address La dirección de aquel nodo que ofreció la dirección elegida.

*** Paquete PANC *(Primary Address Notification Confirmation)***

packet type Este campo toma el valor de 4.

Para el encabezado opcional se tiene solamente el siguiente valor,

address La dirección de aquel nodo que ofreció la dirección elegida, que debería concordar con la del nodo que envía el paquete PANC.

Para el proceso de cesión de direcciones secundarias, se utiliza un paquete SAP y un SAN.

*** Paquete SAP *(Secondary Address Proposal)***

packet type Este campo lleva el valor de 7.

source address La dirección IPv6 del nodo que ofrece la dirección secundaria

primary address length La dimensión del hipercubo

optional headers Se envía un encabezado Additional Address

El encabezado Additional Address tiene los siguientes valores en sus campos,

address La dirección IPv6 con la dirección relativa secundaria embebida, tal como fue visto en la sección 4.2.1

mask La máscara ANTop de la dirección secundaria.

*** Paquete SAN *(Secondary Address Notification)***

packet type Este campo lleva el valor de 8.

source address La dirección IPv6 del nodo que confirma la dirección secundaria

optional headers Se envía un encabezado Additional Address

Para el encabezado opcional se tiene solamente el siguiente valor,

address La dirección IPv6 elegida.

*** Paquete HB *(Heart Beat)***

packet type Este campo lleva el valor de 6.

primary address La dirección IPv6 del nodo que envía el HB.

optional headers Se envía un encabezado Additional Address

Para el encabezado opcional se tiene solamente el siguiente valor,

mask La máscara ANTop del nodo que envía el HB.

A partir de aquí se mostrarán los paquetes de control *Rendez-Vous* utilizados. En la figura 5.4 puede verse el paquete original definido en la tesis de Alejandro Marcu [1] (sección 2.4.3).

Figura 5.4: Paquete de control *Rendez-Vous* original

En el campo de datos se incluyen mayormente direcciones universales y relativas. Esto depende del tipo de paquete *Rendez-Vous* que se esté representando.

En la figura 5.5 se muestra el paquete RV genérico utilizado en esta implementación. Luego se detallan los valores de estos campos para cada uno de los tipos de paquetes.

PAQUETE *REGISTER*

packet type Este campo lleva el valor de 9.

primary address La dirección IPv6 del nodo que se registra.

universal address La dirección universal del nodo que se registra.

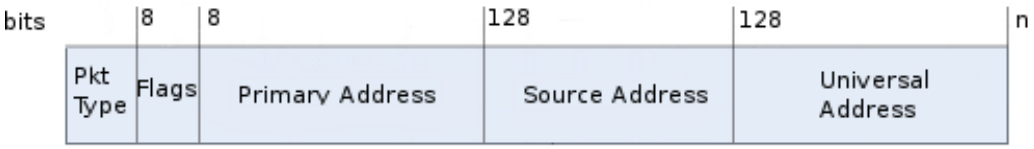


Figura 5.5: Paquete de control *Rendez-Vous*

PAQUETE *ADDRESS LOOKUP*

packet type Este campo lleva el valor de 10.

source address La dirección IPv6 del nodo que pide la resolución.

universal address Dirección universal a resolver.

PAQUETE *ADDRESS SOLVE*

packet type Este campo lleva el valor de 11.

flags Este campo lleva el valor de 1 si la dirección fue resuelta, cero en otro caso.

primary address La dirección IPv6 que se corresponde con la dirección universal de la búsqueda.

universal address Dirección universal a resolver.

5.2. Pruebas de Bajo Nivel

Este primer apartado agrupa pruebas relacionadas con funciones de bajo nivel del protocolo, como ser la conexión de nodos y el ruteo de paquetes. Dichas funciones son transparentes al usuario.

5.2.1. Conexión de nodos

En esta sección se reproduce la conexión de los primeros tres nodos a una red ANTop.

El primer nodo que se conecta a la red tiene la siguiente configuración para su interfaz de red.

```
matias@notebook:~$ ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:21:00:b4:55:31
          inet6 addr: 2001::8/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:18 errors:0 dropped:0 overruns:0 frame:37414
          TX packets:104 errors:35 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3136 (3.1 KB)  TX bytes:11880 (11.8 KB)
          Interrupt:18
```

Figura 5.6: Configuración de la placa inalámbrica

Inicialmente, el nodo presenta una dirección de red que debe ser configurada por el usuario. A dicha dirección inicial se destinarán los paquetes PAP provenientes de los nodos que ya están conectados a la red ANTop. En este caso la dirección elegida al configurar la conexión inalámbrica ad-hoc es *2001::8*, con máscara 64.

La tabla de ruteo tendrá las siguientes entradas.

Destination	Gateway	Netmask	Interface
169.254.0.0	0.0.0.0	255.255.0.0	eth1
2001::	::	64	eth1
fe80::	::	64	eth0
::	::	0	eth1
::	::	0	lo
::1	::	128	lo
2001::8	::	128	lo
fe80::223:5aff:fe2a:b5f2	::	128	lo
ff00::	::	8	eth1
::	::	0	lo

Figura 5.7: Tabla de ruteo

En este punto se ejecuta el demonio ANTop en el Espacio de Usuario. Según lo discutido en 4.2.2, se asigna la primer dirección de la red, aquella bajo la cual se encuentra todo el espacio de direcciones del hipercubo.

```
matias@notebook:~/Desktop/resumido$ ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:21:00:b4:55:31
          inet6 addr: 2001::1/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:18 errors:0 dropped:0 overruns:0 frame:52171
          TX packets:161 errors:35 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3136 (3.1 KB)  TX bytes:20527 (20.5 KB)
          Interrupt:18
```

Figura 5.8: Configuración de la interfaz, al correr ANTop

Es posible ver que la máscara de red de la interfaz vale 64. No se debe confundir la misma con la máscara ANTop, la cual vale 0 y está definida en una variable del Espacio de Usuario.

En la imagen 5.9 se presenta la tabla de ruteo luego de obtener la nueva dirección IPv6.

Ahora, se conecta un segundo nodo a la red. El mismo envía un paquete PAR en modo broadcast, y recibe un PAP del primer nodo.

La primer dirección de la imagen 5.10 es aquella con la que se envía el PAR. Como respuesta se recibe un paquete de control ANTop, cuyo tipo tiene el

Destination	Gateway	Netmask	Interface
169.254.0.0	0.0.0.0	255.255.0.0	eth1
2001::	::	64	eth1
fe80::	::	64	eth0
fe80::	::	64	eth1
::	::	0	lo
::1	::	128	lo
2001::	::	128	lo
2001::1	::	128	lo
fe80::	::	128	lo
fe80::223:5aff:fe2a:b5f2	::	128	lo
ff00::	::	8	eth1
::	::	0	lo

Figura 5.9: Tabla de ruteo, al correr ANTop

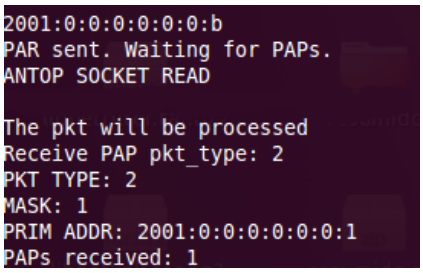


Figura 5.10: Mensaje impreso en la terminal con información del PAP recibido

valor 2, definido de tal forma para representar un PAP. Este último llega de la dirección *2001::1*, con una máscara ANTop ofrecida de 1.

Por ser este el único PAP recibido, se acepta la propuesta. Obteniendo, así, la dirección relativa con la que el nodo se conectará al hipercubo.

Según la imagen, la dirección obtenida es *2001::8000:0:0:1*. Para una máscara de red de 64, se dispone de 64 bits para representar a cada nodo, es decir 8 bytes. En la notación utilizada, cada byte es representado por dos dígitos hexadecimales. En este caso el primer dígito de la dirección relativa tiene el valor de 8, lo cual implica que solo el bit más significativo toma el valor de 1. Esto es lo esperado según planteado en el mecanismo de asignación de direcciones ANTop.

La tabla de ruteo, ahora contiene las siguientes entradas,

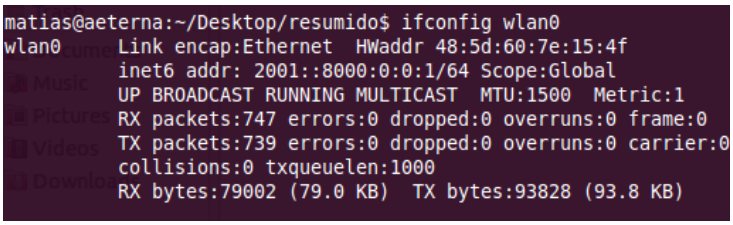


Figura 5.11: Configuración de la placa inalámbrica para el nodo 2

Destination	Gateway	Netmask	Interface
2001::6:0:0:0	2001::1	128	wlan0
2001::4006:0:0:0	2001::1	128	wlan0
2001::	::	64	wlan0
fe80::	::	64	wlan0
::	::	0	lo
::1	::	128	lo
2001::	::	128	lo
2001::8000:0:0:1	::	128	lo
ff00::	::	8	wlan0
::	::	0	lo

Figura 5.12: Tabla de ruteo para el nodo 2

Más adelante se verá como surgen las dos primeras entradas.

Ahora se conecta un tercer nodo a la red. Se repite el mecanismo de pedido de dirección, pero en este caso, se obtendrán dos respuestas, ya que tanto el nodo 1 como el 2 administran un espacio de direcciones de hipercubo.

Como se puede ver en la imagen 5.13, si recibe tanto un PAP del nodo 1, cuya dirección es *2001::1*, como del nodo 2, cuya dirección es *2001:0:0:0:8000:0:0:1*. Como es de esperarse, ambas direcciones ofrecidas tienen máscara 2.

De las ofertas recibidas se escoge aquella con mayor espacio de direcciones, es decir menor máscara ANTop. En este caso ambos ofertas tiene la misma máscara, con lo cual se elige una de ellas indistintamente. En verdad se elegirá la opción que llegué primero.

Una vez que se hubo elegido, se envía un paquete PAN (Primary Address Notification), en modo broadcast indicando la elección. Aquel nodo cuya propuesta es elegida, debe responder con un PANC (Primary Address

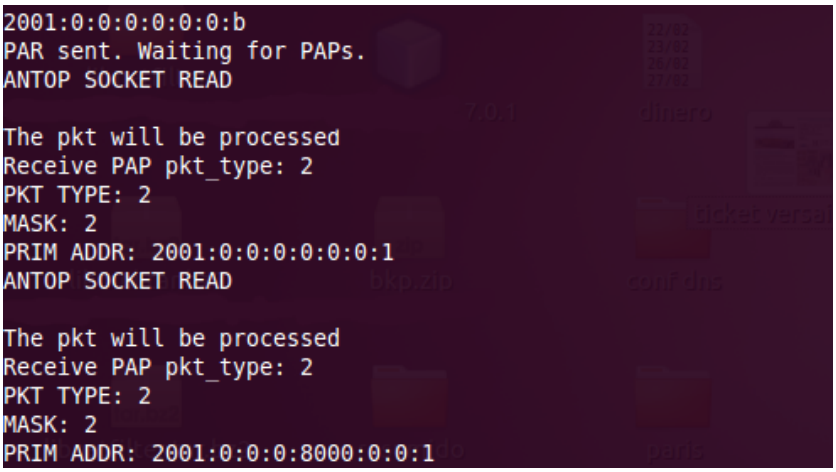


Figura 5.13: Ofertas de dirección para el nodo3

Notification Confirmation). Una vez recibido dicho paquete, se acepta efectivamente el espacio de direcciones y se configura la interfaz inalámbrica con la nueva dirección IPv6.

En la figura 5.14, se muestra el envío de un paquete PAN con la dirección original de la interfaz inalámbrica, configurada por el usuario. En este caso es *2001::b*. También se puede distinguir la recepción del paquete PANC, el cual tiene un valor de 4 para el campo *Type*. En este caso se eligió la dirección *2001:0:0:0:4000:0:0:1*, ofrecida por el nodo 1, con dirección *2001::1*.

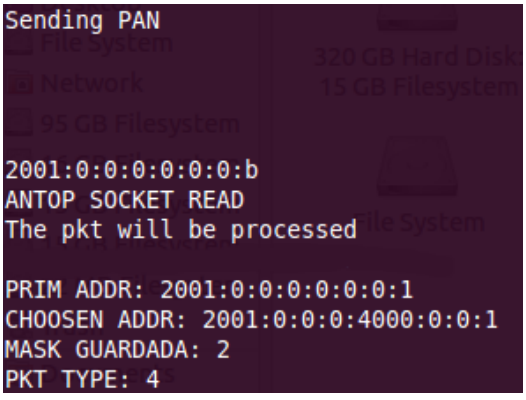


Figura 5.14: PANC recibido para confirmar la cesión del espacio de direcciones

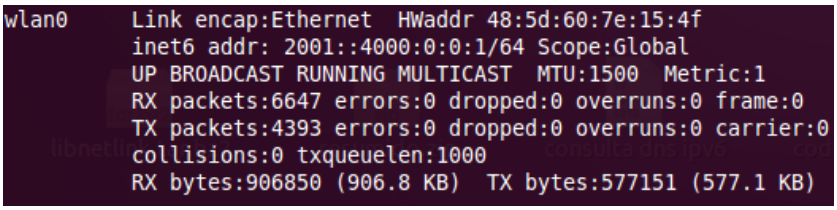


Figura 5.15: Nueva configuración de la interfaz inalámbrica, para el nodo 3

En la figura 5.16, se presenta el contenido de la tabla de ruteo del nodo 3, luego de conectarse a la red ANTop.

Destination	Gateway	Netmask	Interface
169.254.0.0	0.0.0.0	255.255.0.0	wlan0
2001::	::	64	wlan0
fe80::	::	64	wlan0
::	::	0	lo
::1	::	128	lo
2001::	::	128	lo
2001::4000:0:0:1	::	128	lo
ff00::	::	8	wlan0
::	::	0	lo

Figura 5.16: Tabla de ruteo para el nodo 3

Si en este punto se mira la configuración de la placa inalámbrica del nodo 2, se distingue una segunda dirección IPv6. *2001:0:0:0:6000:0:0:1*.

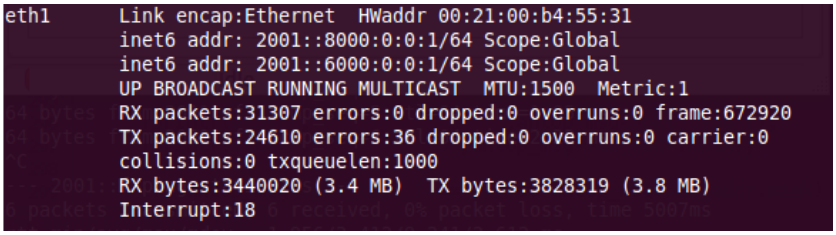


Figura 5.17: Configuración en el nodo 2, luego de la conexión del nodo 3

Se agrega una dirección secundaria en el nodo 2 para lograr tener conectividad a nivel del protocolo ANTop, con el nodo 3. En la imagen 5.18, se puede ver las direcciones asignadas a los tres nodos hasta el momento. Allí

se aprecia que las direcciones relativas primarias de los nodos 2 y 3 difieren en dos bits. Es decir que la distancia entre sus direcciones vale 2.

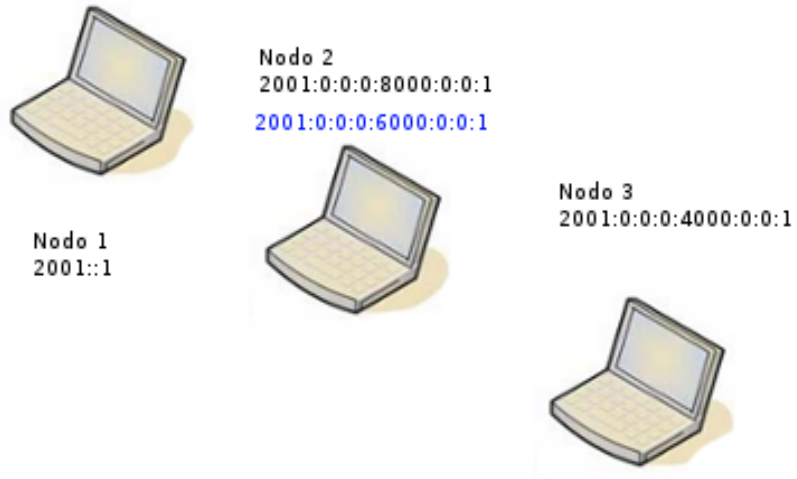


Figura 5.18: Direcciones de los nodos conectados a la red ANTop

Cuando el nodo 2 envía un Heart Beat, tanto el nodo 1 como el 3 lo reciben. Sin embargo, el nodo 3 se da cuenta que la distancia entre las direcciones es mayor a uno, y que por lo tanto no podrán ser vecinos. Por esta razón el mismo le envía un SAP (Secondary Address Proposal), ofreciendo la dirección *2001:0:0:0:6000:0:0:1*, la cual es aceptada. De aquí en más el nodo 2 enviará Heart Beats tanto por la dirección primaria como por la secundaria.

La figura 5.19, muestra la tabla de vecinos para el nodo 3. En ella se ve la relación de vecinos con el nodo 2, mediante la dirección secundaria cedida, la cual tiene máscara 3. El parámetro *Neighbor Count* refiere a la cantidad de períodos consecutivos, durante los cuales no se ha recibido un Heart Beat. En este caso, el valor de -1 indica que no se ha perdido ninguno.

En las figuras 5.20 y 5.21 se muestran las tablas de vecinos de los nodos 2 y 1, respectivamente.

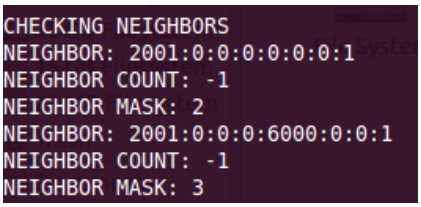


Figura 5.19: Tabla de vecinos para el nodo 3

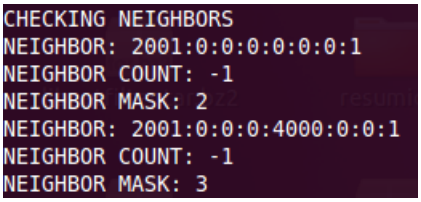


Figura 5.20: Tabla de vecinos para el nodo 2

En este punto se desconecta el nodo 3, cerrando el proceso del demonio ANTop que se ejecuta en el Espacio de Usuario. Luego de tres intervalos de espera por Heart Beats, sin recibirlos tanto el nodo 2 como el 1 borran al 3 de su tabla de vecinos, cuya dirección IPv6 era *2001:0:0:0:4000:0:0:1*. Esta última fue cedida por el nodo 1, el primero conectado a la red ANTop. Al ver que la dirección que abandona la red corresponde a un sucesor, se dispone a recuperar dicho espacio de direcciones. Esto puede verse en la figura 5.22.

Luego de unos instantes, se ejecuta nuevamente el demonio ANTop en el nodo 3. Según la figura 5.23, la dirección IPv6 que se utiliza como origen del paquete PAR es *2001:0:0:0:4000:0:0:1*, es decir la que utilizaba el nodo al estar conectado a la red ANTop. Se repite el mecanismo de obtención de dirección primaria, y se obtienen dos ofrecimientos de conexión. La primera de ellas es realizada por el nodo 1, quien vuelve a ofrecer la dirección que se acaba de recuperar, es decir *2001:0:0:0:4000:0:0:1*. El segundo es realizado por el nodo 2, cuya dirección primaria es *2001::0:0:0:8000:0:0:1*. De estas dos opciones, el nodo 3 elige la segunda, ya que la misma cuenta con la menor máscara. La diferencia se debe a que el espacio de direcciones de *2001:0:0:0:4000:0:0:1* disminuyó su tamaño al otorgar una dirección secundaria al nodo 2, aquella con IPv6 *2001:0:0:0:6000:0:0:1*. El espacio asociado a *2001:0:0:0:8000:0:0:1*, por otro lado se mantuvo intacto.

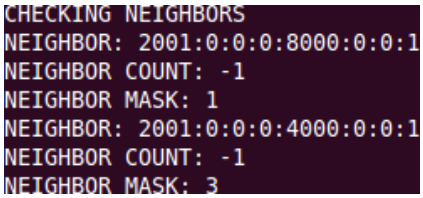


Figura 5.21: Tabla de vecinos para el nodo 1

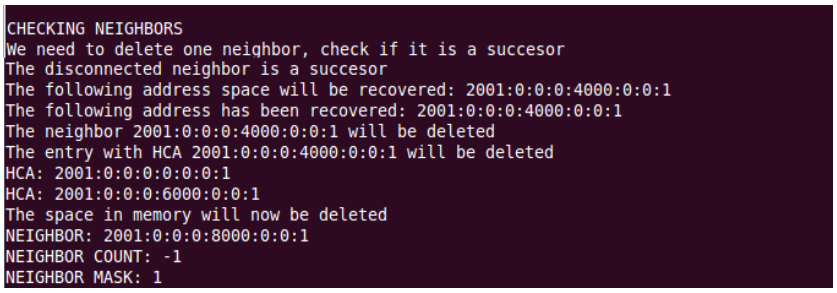


Figura 5.22: El nodo 1 borra al 3 de su tabla de vecinos

5.2.2. Ruteo de datos

Como se explicó anteriormente, el ruteo de paquetes depende fundamentalmente de la relación de vecinos que cada nodo entabla con sus pares ubicados a una distancia próxima en sentido topológico. Para convalidar dicha relación, son muy importantes los paquetes de tipo HB.

De aquí en más se desarrollan las pruebas de ruteo de paquetes de datos.

Para la siguiente prueba, se tiene una topología de tres nodos, en la cual si bien cada uno de ellos está dentro del alcance de la señal inalámbrica de los otros, no se asignarán direcciones secundarias. De esta forma, no todos podrán comunicarse entre sí, mediante el ruteo de ANTop .El esquema de direcciones resultante es el presentado en la figura 5.24.

En esta red, los nodos 2 y 3 son sucesores del 1. Ambos recibieron su espacio de direcciones, y por ende su dirección del primer nodo conectado a la red. Además, obviamente 1 será vecino de 2 y de 3, ya que los nodos padre e hijo, por defecto lo son. Por otro lado, 2 y 3 no serán vecinos, ya que sus direcciones difieren en dos bits.

La figura 5.25, presenta la tabla de vecinos para el nodo 1, mientras que la figura 5.26, hace lo propio con los nodos 2 y 3.

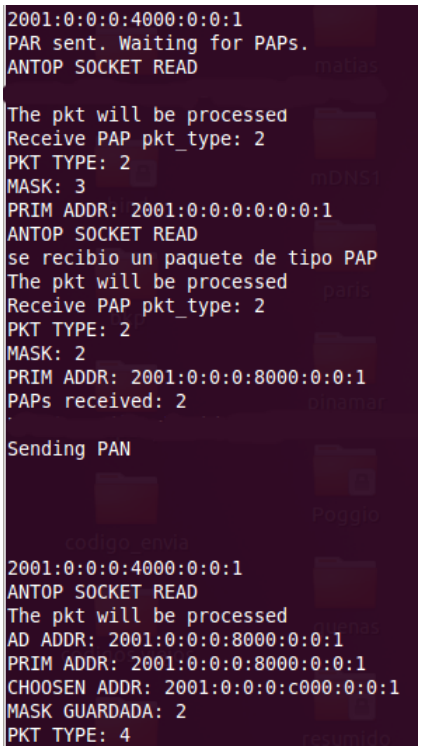


Figura 5.23: Reconexión del nodo 3

El nodo 1 ya ha cedido dos espacios de direcciones y por ende su máscara ANTop tiene un valor de 2, mientras que los nodos 2 y 3 no han cedido espacios de direcciones, conservando su máscara en el valor original de 1.

En este esquema, los nodos 2 y 3 no pueden enviarse tráfico directamente entre si, sino que deben hacerlo a través del nodo 1. En esta prueba se enviará un Ping desde el nodo 2 hacia el 3. Para ello ejecutamos el siguiente comando en el nodo 2.

```
ping6 -I wlan0 2001:0:0:0:4000:0:0:1
```

Dicho comando dispara un paquete ICMPv6 Echo Request con destino *2001::4000:0:0:1*, y origen *2001::8000:0:0:1* que es interceptado por Netfilter cuando el mismo alcanza el gancho LOCAL_OUT. Allí es enviado al Espacio de Usuario, y recibido por el demonio que corre ANTop, el cual chequea que no hay una entrada en la tabla de ruteo que tenga como destino la dirección IPv6 del nodo 3. Esto dispara el mecanismo de obtención de rutas.

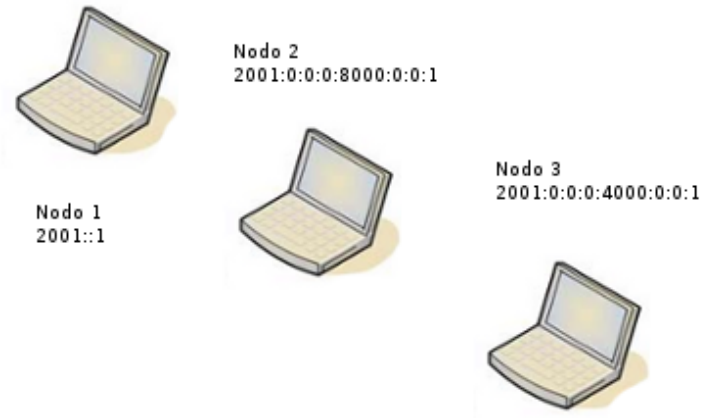


Figura 5.24: Topología de prueba de ruteo

```
CHECKING NEIGHBORS
NEIGHBOR: 2001:0:0:0:8000:0:0:1
NEIGHBOR COUNT: -1
NEIGHBOR MASK: 1
NEIGHBOR: 2001:0:0:0:4000:0:0:1
NEIGHBOR COUNT: -1
NEIGHBOR MASK: 1
```

Figura 5.25: Tabla de vecinos para el nodo 1

Según la solución que propone ANTop, el paquete se debe enviar a aquel vecino cuya distancia al destino sea menor. En este caso solo se tiene un vecino, el nodo 1, y allí debe enviarse el paquete. Para lograrlo, se instala en la tabla de ruteo una nueva entrada, con los siguientes parámetros.

Destino 2001::4000:0:0:1

Próximo Salto 2001::1

Al consultar la tabla de ruteo, luego de ejecutar el Ping, se obtiene el resultado mostrado en la figura 5.27. Allí puede apreciarse, enmarcada, la ruta correspondiente. Tanto la entrada con destino *2001::5:0:0:0*, como aquella con un valor de *2001::4005:0:0:0*, para el mismo parámetro, son agregadas por el servicio *Rendez-Vous*. Dichas entradas se explicarán en la sección 5.2.3.

En este punto, el sistema operativo del nodo 2, al encontrar una entrada en la tabla de ruteo cuyo destino es el del paquete, chequea la tabla de vecinos IPv6 [16]. En caso de que no halle una entrada para la dirección IPv6 del

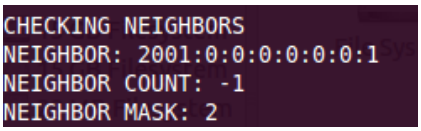


Figura 5.26: Tabla de vecinos para el nodo 2 y 3

Destination	Gateway	Netmask	Interface
169.254.0.0	0.0.0.0	255.255.0.0	eth1
2001::5:0:0:0	2001::1	128	eth1
2001::4000:0:0:1	2001::1	128	eth1
2001::4005:0:0:0	2001::1	128	eth1
2001::	::	64	eth1
fe80::	::	64	eth1
::	::	0	lo
::1	::	128	lo
2001::	::	128	lo
2001::8000:0:0:1	::	128	lo
ff00::	::	8	eth1
::	::	0	lo

Figura 5.27: Tabla de ruteo del nodo 2 luego de ejecutar el Ping

nodo 1, debe enviar un paquete Neighbor Solicitation en modo broadcast, preguntando por la dirección física del nodo 1. Este último, al recibir el pedido, responde con un paquete Neighbor Advertisement, cuya dirección de destino es la del nodo 2.

Luego de ejecutar el Ping en el nodo 2, se ejecuta el comando *ip neigh*, y se observa la entrada de la figura 5.28, entre otras.

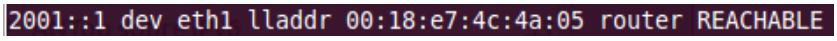


Figura 5.28: Entrada con la dirección física del nodo 1

Una vez obtenida la dirección física del próximo salto, el nodo 2 envía el paquete ICMPv6 Echo Request hacia el nodo 1. Allí es interceptado nuevamente por Netfilter, pero esta vez en el gancho PRE_ROUTING, para luego ser enviado al Espacio de Usuario. Allí, se detecta que el destino del paquete es un vecino, el nodo 3, con lo cual no se busca una ruta, sino que se deja que el paquete sea ruteado a través de la ruta default, aquella que en este caso tiene los siguientes parámetros,

Destino 2001::

Máscara 64

Interfaz eth1

En la tabla de vecinos del nodo 1 se hallan dos entradas con las direcciones físicas de los nodos 2 y 3. Esto puede verse en la figura 5.29.

```
2001::4000:0:0:1 dev wlan0 lladdr 48:5d:60:7e:15:4f router REACHABLE
2001::8000:0:0:1 dev wlan0 lladdr 00:21:00:b4:55:31 router REACHABLE
```

Figura 5.29: Entradas con las direcciones físicas de los nodos 3 y 2, respectivamente

Es así como finalmente el paquete ICMPv6 Echo Request llega a su destino, el nodo 3. En las figuras 5.30, 5.31, 5.32 y 5.33 se puede apreciar una captura de paquetes en la que se muestra la ruta que siguen tanto los paquetes de Request como los de Reply. Las primeras dos figuras muestran el pedido ICMPv6 que se rutea desde el nodo 2 al nodo 1 y luego desde allí al nodo 3. Las últimas dos figuras de la serie, muestran la ruta realizada por la respuesta ICMPv6, la cual se rutea desde el nodo 3 hacia el 1, para finalmente dirigirse desde allí al nodo 2.

En las capturas es posible ver encuadradas las direcciones físicas de los saltos, las cuales concuerdan con aquellas contenidas en las tablas de vecinos mostradas anteriormente. Por otro lado, es posible ver el encabezado de opciones que se agrega en el paquete para contener la dirección IPv6 del último salto por el que transitó el paquete. Esta última información se utiliza en caso de que el algoritmo de búsqueda de ruta de ANTop, dé como resultado la devolución del paquete por donde llego al nodo local.

Por último se puede observar para el campo de *Flow Label*, un valor de 4. Si bien el sistema operativo da un valor de cero por defecto para este campo, el demonio que corre ANTop, representa aquí las banderas, dando un valor de uno en este caso para aquella que determina que el tráfico debe ser tratado procesado por ANTop.

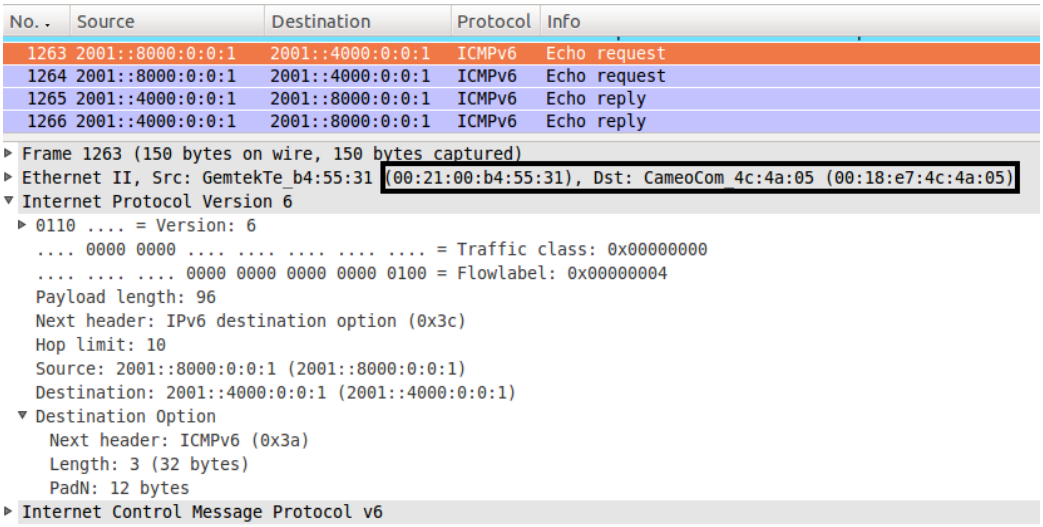


Figura 5.30: Primer salto para el Echo Request

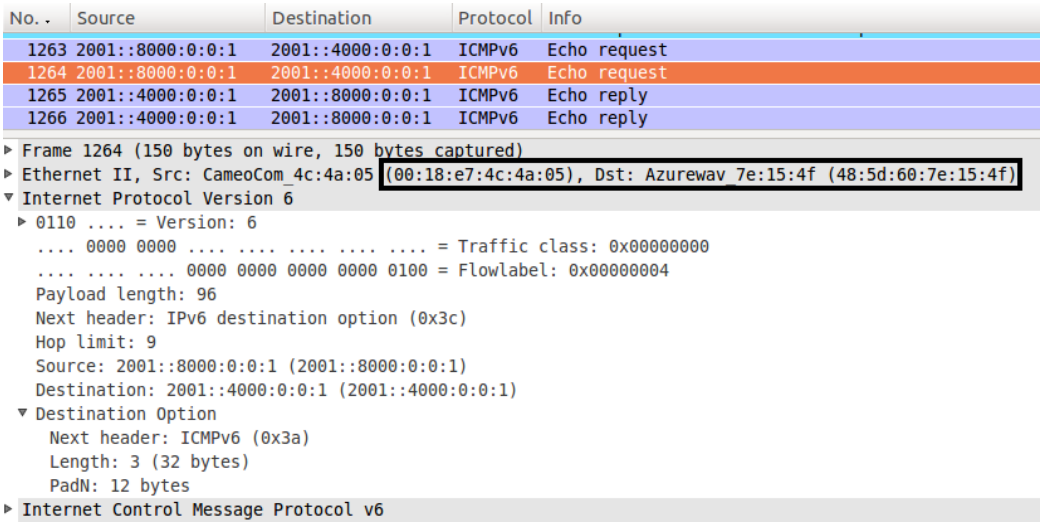


Figura 5.31: Segundo salto para el Echo Request

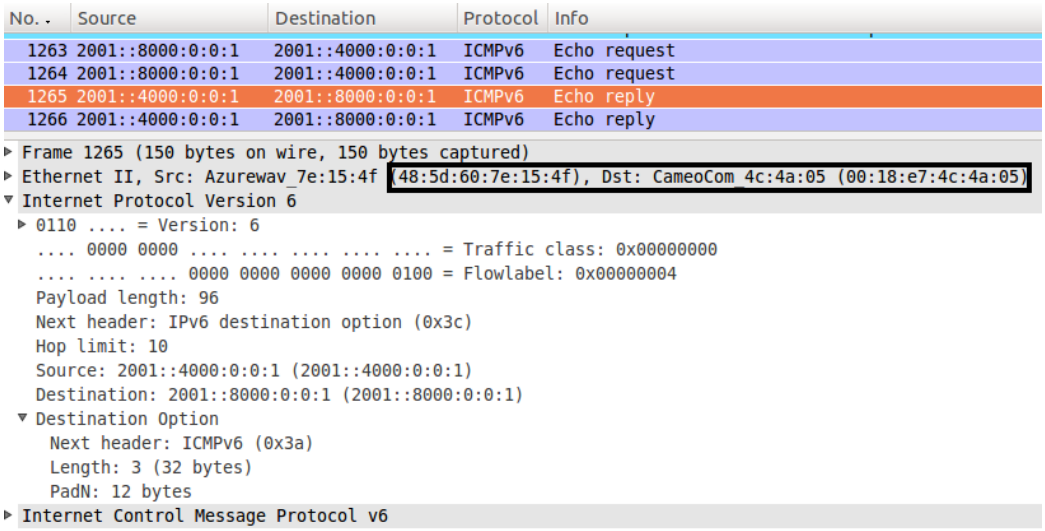


Figura 5.32: Primer salto para el Echo Reply

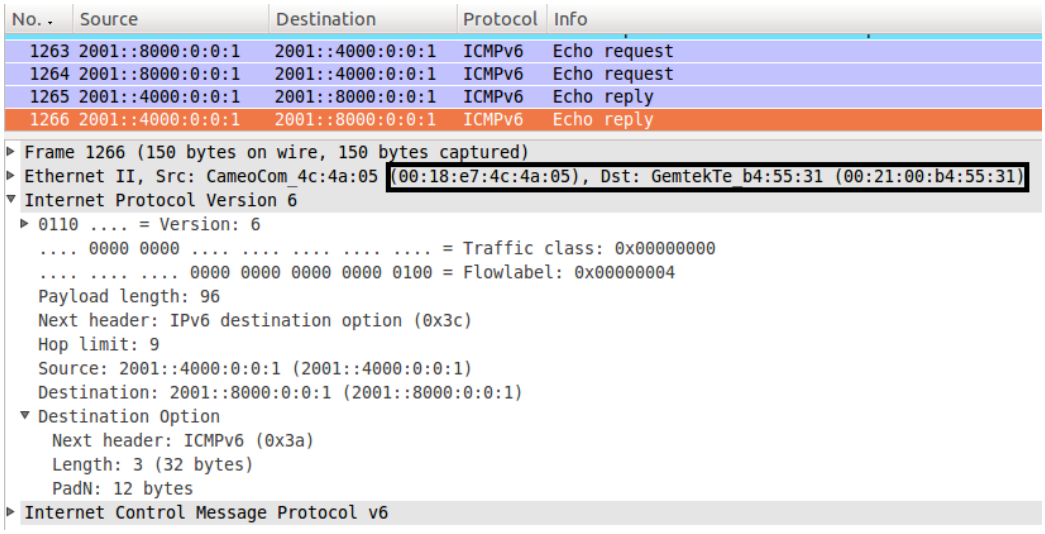


Figura 5.33: Segundo salto para el Echo Reply

5.2.3. Ruteo de tráfico de control Rendez-Vous

Al presentar la tabla de ruteo del nodo 2, en la figura 5.27, se dijo que las dos primeras entradas eran instaladas por el servicio *Rendez-Vous*. Para explicar esto, se debe recordar que los nodos que participan de la red ANTop,

se registran en servidores de nombres (servidores *Rendez-Vous*), distribuidos a lo largo de todo el hipercubo. Es decir que no se tiene una estructura centralizada. Para que el peso de este servicio se distribuya uniformemente en toda la red, se utiliza una función de Hash que toma como entrada la dirección universal del registro que se quiera localizar.

Para el caso del nodo 2, luego de aplicar la función de Hash a la dirección universal, se obtienen tres direcciones IPv6 de servidores *Rendez-Vous* a los cuales se debe enviar periódicamente un paquete de control ANTop con la dirección universal y la relativa. Esto puede verse en la figura 5.34

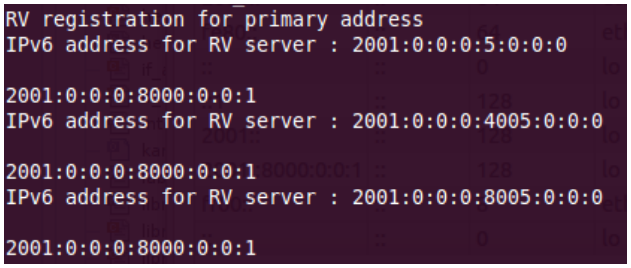


Figura 5.34: Registro del nodo 2 en los servidores *Rendez-Vous*

Al igual que el resto de los paquetes de control ANTop, los de registro a servidores *Rendez-Vous* son paquetes UDP, los cuales en su camino hacia el destino, alcanzarán el gancho LOCAL_OUT de la estructura Netfilter. Allí se buscará una ruta para los mismos, en caso de que aún no la haya.

Uno de los primeros pasos en esta búsqueda de rutas será comprobar si el destino de los paquetes se encuentra dentro del espacio de direcciones del nodo actual. Esto es muy importante ya que, como se dijo con anterioridad, las direcciones de los servidores *Rendez-Vous* pueden no existir.

En el caso del tercer servidor de la figura 5.34, el mismo tiene una dirección *2001:0:0:0:8005:0:0:0:*, la cual se encuentra en el espacio de direcciones del nodo actual. Por esa razón, el paquete en verdad no sale del nodo actual, sino que es procesado localmente. Esto se logra enviando desde el Espacio de Usuario hacia el Espacio de Kernel un veredicto de descarte a la interfaz Netfilter.

La copia del paquete recibida en el Espacio de Usuario se utiliza para procesar el pedido de registro de dirección universal en el nodo local. En este caso no se debe agregar ninguna entrada a la tabla de ruteo. Es por esta razón que mirando nuevamente la figura 5.34, solo se observan entradas para el primer y segundo servidor *Rendez-Vous*, pero no para el tercero.

Para el caso de los servidores con dirección IPv6 *2001:0:0:0:5:0:0:0* y *2001:0:0:0:4005:0:0:0*, ninguno de ellos se encuentra dentro del espacio de direcciones del nodo 2, razón por la cual se debe buscar una ruta. En ambos casos, los paquetes son enviados al único vecino disponible, es decir el nodo 1, con dirección IPv6 *2001::1*.

Cuando el pedido de registro destinado a *2001:0:0:0:5:0:0:0*, llega al nodo 1, el mismo se procesa localmente, ya que se encuentra dentro del espacio de direcciones que se administra. El pedido destinado a *2001:0:0:0:4005:0:0:0* en cambio, no se encuentra dentro del espacio de direcciones que el nodo 1 administra, pero si del espacio de direcciones del nodo 3, el cual tiene dirección IPv6 *2001:0:0:0:4000:0:0:1*. Por esta razón, el paquete en enviado finalmente al nodo 3, el cual se encarga de guarda la información de las direcciones del nodo 2. En la imagen

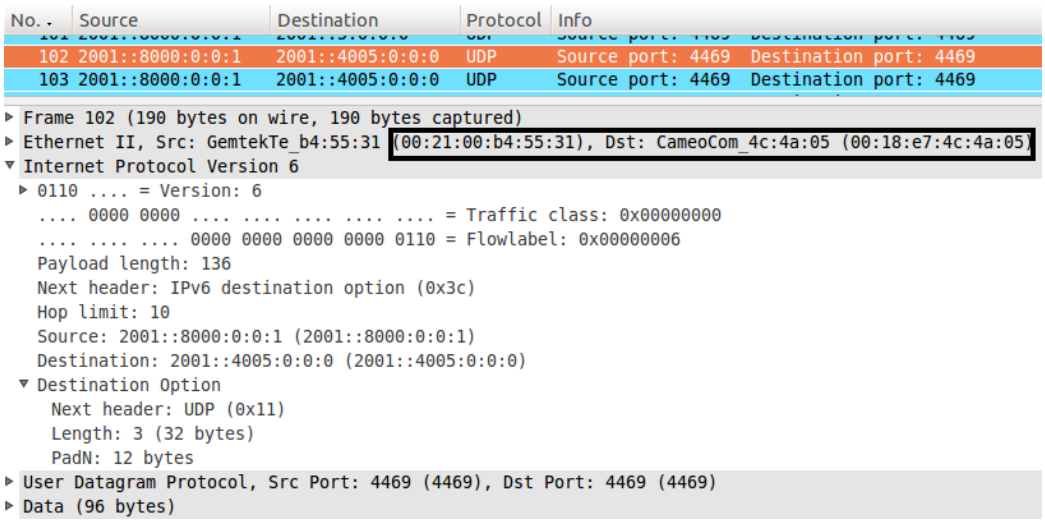


Figura 5.35: Ruteo del paquete *Rendez-Vous* desde el nodo 2 hacia el 1

Para este caso en particular, los tres nodos guardarán la información de direcciones universal y relativa para el nodo 2.

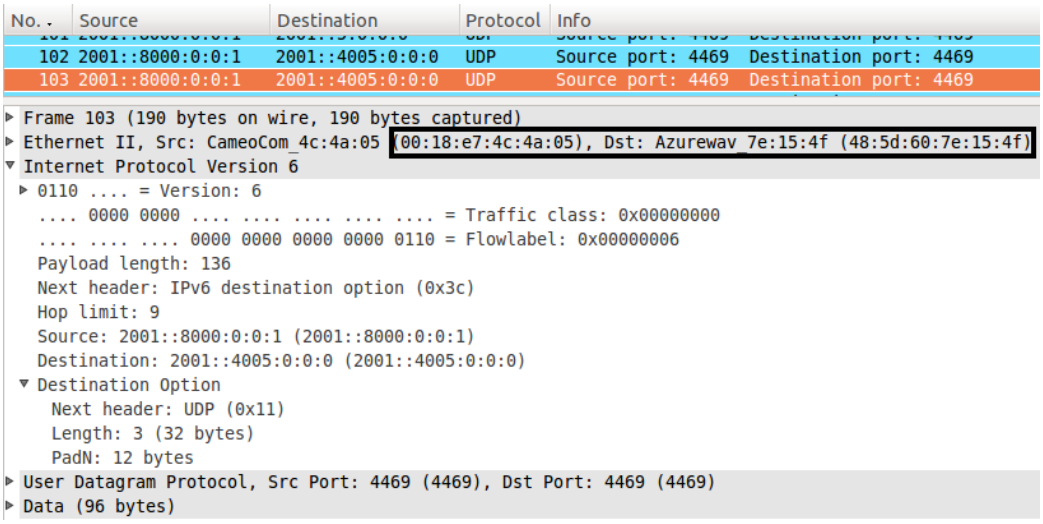


Figura 5.36: Ruteo del paquete *Rendez-Vous* desde el nodo 1 hacia el 3

5.3. Pruebas de Alto Nivel

Esta segunda sección se ocupa de las pruebas relacionadas con los servicios al usuario. Aquí se incluyen las pruebas de resolución de nombres y la de transferencia de datos.

5.3.1. Resolución de nombres

De aquí en más se presentan efectivamente las pruebas realizadas para chequear el funcionamiento del servicio de resolución de nombres.

Para probar este servicio en funcionamiento sobre una red ANTop de computadoras se contempla la topología de la figura 5.37.

En el nodo 1 se ejecuta,

```
ping6 -I eth1 nodo2
```

Siendo eth1 el nombre de la interfaz inalámbrica del nodo 1.

Como se explico en el apartado 4.3, la ejecución del Ping utilizando el nombre del host destino dispara una consulta DNS, de la cual se muestra una captura en la figura 5.38.

Figura 5.37: Topología para prueba del servicio de resolución de nombres

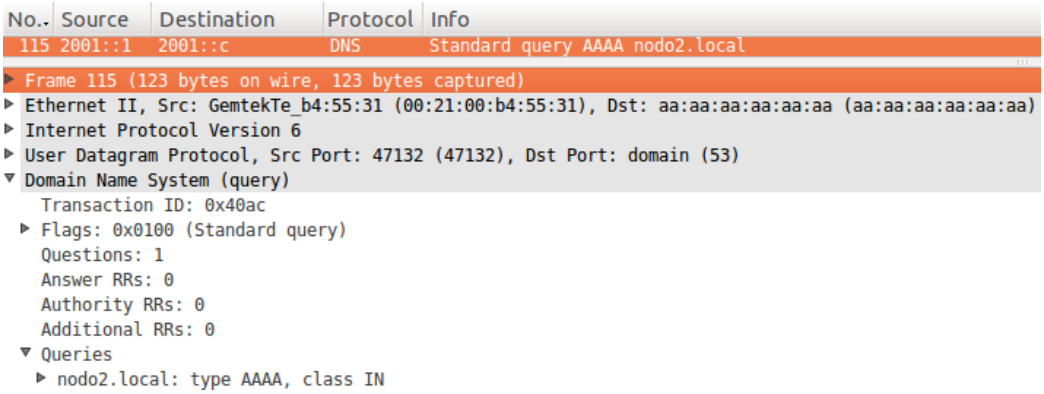


Figura 5.38: Envío de un pedido DNS

En dicha imagen se puede observar que la dirección física de destino, es decir del próximo salto es *aa:aa:aa:aa:aa:aa*. Para entender esto, es necesario recordar que en las redes ANTop no se tiene verdaderamente un servidor DNS a quien consultar por un nombre de host, sino que se utiliza la interfaz de dicho servicio, montada en el sistema operativo, para establecer un canal de comunicación de pedidos y respuestas de resolución de nombres entre la capa de aplicación y ANTop.

Teniendo esta última consideración en mente, se aclara que verdaderamente es el demonio que corre ANTop en el Espacio de Usuario quien agregar una entrada estática en la tabla de vecinos, para que el paquete que contiene el pedido DNS, pueda ser armado con una dirección MAC correspondiente a la dirección IPv6 del servidor DNS, la cual también será ficticia. En la figura 5.39, es posible ver dicha entrada forzada en la tabla de vecinos del nodo 1.

```
2001::c dev eth1 lladdr aa:aa:aa:aa:aa:aa PERMANENT
```

Figura 5.39: Entrada ficticia correspondiente al servidor DNS en la tabla de vecinos del nodo 1

Con este arreglo, el sistema operativo tiene todos los datos necesarios para conformar el paquete de pedido DNS. Dicho paquete, en su camino hacia el destino, ficticio en este caso, alcanzará el gancho LOCAL_OUT de la estructura Netfilter. Allí se envía al Espacio de Usuario y será el demonio de ANTop quien detecte que alguna aplicación requiere la resolución del nombre de un host (dirección universal ANTop) en una dirección IPv6 (que contiene la dirección relativa del nodo en el hipercubo).

La presencia de este pedido DNS dispara a su vez un pedido de resolución ANTop, para el cual se consultará al servidor *Rendez-Vous* más cercano, entre aquellos que tengan a su cargo mantener la información de direcciones para el nombre *nodo2*. En este caso en particular, luego de aplicar la función de Hash se obtiene como resultado la dirección IPv6 *2001:0:0:0:1:0:0:0*. El demonio ANTop, entonces, genera un paquete con un pedido de resolución de nombre *Rendez-Vous* utilizando dicha dirección como destino.

En este caso, el destino del paquete se encuentra dentro del espacio de direcciones del nodo 1, con lo cual se procesa localmente. Luego de dicho procesamiento, en el cual se obtiene la dirección IPv6 del nodo 2, se envía un paquete de respuesta DNS para el cual la dirección destino será la del propio nodo 1 *2001::1*, y la de origen será la del inexistente servidor DNS, es decir *2001::c*. Este paquete generado localmente en verdad no sale del nodo y es procesado por la aplicación que en este caso fue quien disparo el Ping y estaba aguardando una respuesta a su consulta de nombre de host. En la figura 5.40, se muestran los resultados de dicho Ping.

```
matias@notebook:~/Desktop/resumido$ ping6 -I eth1 nodo2
PING nodo2(2001::8000:0:0:1) from 2001::1 eth1: 56 data bytes
64 bytes from 2001::8000:0:0:1: icmp_seq=1 ttl=10 time=2.25 ms
64 bytes from 2001::8000:0:0:1: icmp_seq=2 ttl=10 time=2.22 ms
64 bytes from 2001::8000:0:0:1: icmp_seq=3 ttl=10 time=2.06 ms
64 bytes from 2001::8000:0:0:1: icmp_seq=4 ttl=10 time=2.00 ms
64 bytes from 2001::8000:0:0:1: icmp_seq=5 ttl=10 time=2.17 ms
^C^C64 bytes from 2001::8000:0:0:1: icmp_seq=6 ttl=10 time=1.97 ms

--- nodo2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 25038ms
rtt min/avg/max/mdev = 1.973/2.117/2.258/0.110 ms
```

Figura 5.40: Resultado del Ping dirigido al nodo 2

5.3.2. Transferencia de datos

Sea la topología la indicada en la figura 5.37. En el nodo 2 se abre una sesión ssh contra el nodo 1, con el siguiente comando,

```
ssh -6 nodo1
```

En dicho comando, el parámetro *6* refiere a que se desea abrir una conexión ssh sobre el protocolo IPv6, al tiempo que *nodo1* es, por supuesto, el nombre del host con el que queremos comunicarnos. Este valor concuerda con la dirección universal del mismo.

En el nodo 2 se obtiene el resultado que se muestra en la figura 5.41,

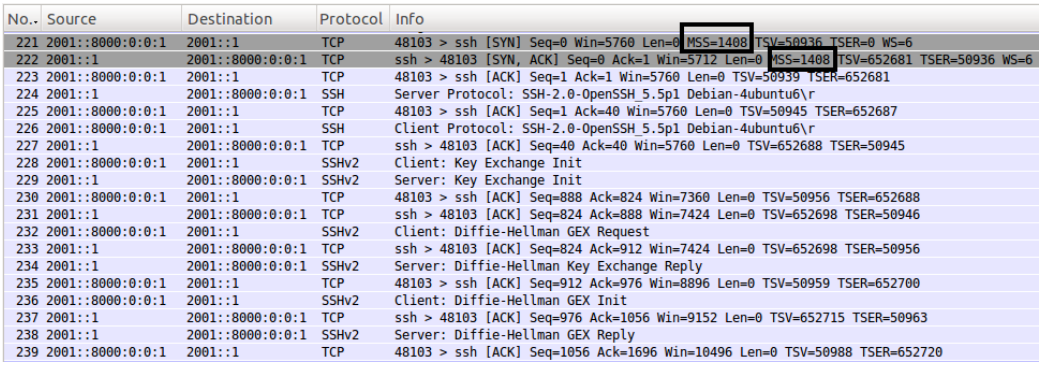
```
matias@aeterna:~/Desktop/resumido$ ssh -6 nodo1
The authenticity of host 'nodo1 (2001::1)' can't be established.
RSA key fingerprint is 0f:76:3f:fc:98:c9:a5:dd:73:2f:aa:ef:ee:3b:da:9a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'nodo1' (RSA) to the list of known hosts.
Warning: the RSA host key for 'nodo1' differs from the key for the IP address '2001::1'
Offending key for IP in /home/matias/.ssh/known_hosts:2
Are you sure you want to continue connecting (yes/no)? yes
matias@nodo1's password:
Linux notebook 2.6.35-22-generic #33-Ubuntu SMP Sun Sep 19 20:34:50 UTC 2010 i686 GNU/Linux
Ubuntu 10.10

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

404 packages can be updated.
227 updates are security updates.

Last login: Thu Apr 19 11:10:38 2012 from 2001::8000:0:0:1
matias@notebook:~$ lspci
00:00.0 Host bridge: Advanced Micro Devices [AMD] RS780 Host Bridge
00:01.0 PCI bridge: Hewlett-Packard Company Device 9602
00:04.0 PCI bridge: Advanced Micro Devices [AMD] RS780 PCI to PCI bridge (PCIE port 0)
00:05.0 PCI bridge: Advanced Micro Devices [AMD] RS780 PCI to PCI bridge (PCIE port 1)
00:06.0 PCI bridge: Advanced Micro Devices [AMD] RS780 PCI to PCI bridge (PCIE port 2)
00:07.0 PCI bridge: Advanced Micro Devices [AMD] RS780 PCI to PCI bridge (PCIE port 3)
00:11.0 SATA controller: ATI Technologies Inc SB700/SB800 SATA Controller [AHCI mode]
```

Figura 5.41: Resultado del comando ssh

Como puede verse en dicha imagen, la dirección universal es resuelta según el mecanismo explicado en la sección 5.3.1, en la dirección IPv6 *2001::1*. Una vez ingresada la clave de autenticación para el nodo 1, se abre una sesión remota de linea de comandos en el host de nombre *notebook*. Aquí se debe tener en cuenta que la dirección universal del nodo no será necesariamente igual al nombre del host.

En la parte inferior de la figura es posible observar el resultado de ejecutar el comando lspci, el cual devuelve un listado de los dispositivos conectados al equipo. Aquí se muestra solo un fragmento de la lista.

La figura 5.42 muestra una captura de tráfico que muestra los primeros paquetes de la sesión TCP. Allí puede verse el establecimiento de la conexión. Recuadrado en negro se puede ver el parámetro *MSS* que intercambian los nodos. Es posible apreciar que según lo explicado en la sección 4.4, su valor es de 1408 bytes, correspondientes al tradicional 1440 bytes para TCP sobre IPv6, menos los 32 bytes correspondientes a la longitud del encabezado de extensión de tipo *Destination Options*.



No..	Source	Destination	Protocol	Info
221	2001::8000:0:0:1	2001::1	TCP	48103 > ssh [SYN] Seq=0 Win=5760 Len=0 MSS=1408 TSV=50936 TSER=0 WS=6
222	2001::1	2001::8000:0:0:1	TCP	ssh > 48103 [SYN, ACK] Seq=0 Ack=1 Win=5712 Len=0 MSS=1408 TSV=652681 TSER=50936 WS=6
223	2001::8000:0:0:1	2001::1	TCP	48103 > ssh [ACK] Seq=1 Ack=1 Win=5760 Len=0 TSV=50939 TSER=652681
224	2001::1	2001::8000:0:0:1	SSH	Server Protocol: SSH-2.0-OpenSSH_5.5p1 Debian-4ubuntu6\r
225	2001::8000:0:0:1	2001::1	TCP	48103 > ssh [ACK] Seq=1 Ack=40 Win=5760 Len=0 TSV=50945 TSER=652687
226	2001::8000:0:0:1	2001::1	SSH	Client Protocol: SSH-2.0-OpenSSH_5.5p1 Debian-4ubuntu6\r
227	2001::1	2001::8000:0:0:1	TCP	ssh > 48103 [ACK] Seq=40 Ack=40 Win=5760 Len=0 TSV=652688 TSER=50945
228	2001::8000:0:0:1	2001::1	SSHv2	Client: Key Exchange Init
229	2001::1	2001::8000:0:0:1	SSHv2	Server: Key Exchange Init
230	2001::8000:0:0:1	2001::1	TCP	48103 > ssh [ACK] Seq=888 Ack=824 Win=7360 Len=0 TSV=50956 TSER=652688
231	2001::1	2001::8000:0:0:1	TCP	ssh > 48103 [ACK] Seq=824 Ack=888 Win=7424 Len=0 TSV=652698 TSER=50946
232	2001::8000:0:0:1	2001::1	SSHv2	Client: Diffie-Hellman GEX Request
233	2001::1	2001::8000:0:0:1	TCP	ssh > 48103 [ACK] Seq=824 Ack=912 Win=7424 Len=0 TSV=652698 TSER=50956
234	2001::1	2001::8000:0:0:1	SSHv2	Server: Diffie-Hellman Key Exchange Reply
235	2001::8000:0:0:1	2001::1	TCP	48103 > ssh [ACK] Seq=912 Ack=976 Win=8896 Len=0 TSV=50959 TSER=652700
236	2001::8000:0:0:1	2001::1	SSHv2	Client: Diffie-Hellman GEX Init
237	2001::1	2001::8000:0:0:1	TCP	ssh > 48103 [ACK] Seq=976 Ack=1056 Win=9152 Len=0 TSV=652715 TSER=50963
238	2001::1	2001::8000:0:0:1	SSHv2	Server: Diffie-Hellman GEX Reply
239	2001::8000:0:0:1	2001::1	TCP	48103 > ssh [ACK] Seq=1056 Ack=1696 Win=10496 Len=0 TSV=50988 TSER=652720

Figura 5.42: Captura del tráfico ssh

A continuación se muestran los resultados para una prueba de transferencia segura de archivos entre dos nodos. Para ello, se utiliza la herramienta *scp*, la cual se basa en *ssh*. Para este ejemplo se transfirió un archivo de 43 MB ubicado en el nodo 2, al nodo 1. En la figura 5.43 se muestran los resultados de invocar dicha herramienta en el nodo 1.

```
matias@notebook:~/Desktop/resumido$ scp -6 matias@nodo2:/home/matias/file.sh /home/matias/file.sh
The authenticity of host 'nodo2 (2001::8000:0:0:1)' can't be established.
RSA key fingerprint is 0a:ee:de:b2:8f:0a:a6:05:e3:eb:38:22:85:07:21:62.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'nodo2,2001::8000:0:0:1' (RSA) to the list of known hosts.
matias@nodo2's password:
file.sh                                                   100%   43MB   1.1MB/s   00:41
```

Figura 5.43: Transferencia segura de un archivo

La transferencia del archivo de 43 MB demoró 41 segundos, logrando una velocidad de transferencia de aproximadamente 8.4 Mbps. Cabe destacar que

los nodos se encontraban próximos entre sí. En la imagen se muestra la última transferencia medida, y no el promedio.

En otra prueba, para la cual los nodos se encontraban a una distancia de 10 metros, separadas por dos paredes con puertas abiertas, se transfirió el mismo archivo. El tiempo registrado esta vez fue de 50 segundos. Logrando una tasa de transferencia de 6.8 Mbps

En una última prueba se llevaron los nodos a pisos contiguos en una casa, separados por un techo y luego dos paredes. En esta situación el camino entre ambos nodos estaba mucho más obstaculizado que en el caso anterior. Aquí el tiempo de transferencia ascendió a 2:48 minutos, logrando una transferencia de aproximadamente 2 Mbps.

Estos últimos dos resultados se muestran en la figura 5.44

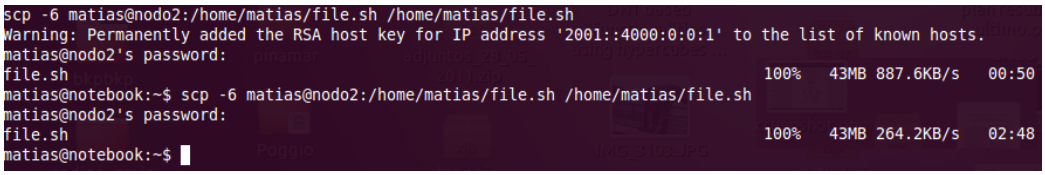


Figura 5.44: Transferencia segura de un archivo alejando los nodos

Para realizar todas estas pruebas, se utilizó el canal 1 y la norma 802.11b, ya que los controladores no soportaron trabajar con la norma g, según lo discutido en el capítulo 3.4. De esta forma, la tasa de transferencia máxima teórica es de 11 Mbps.

5.4. Resumen de las pruebas

En lo que refiere a la conexión de nodos se lograron realizar todas las pruebas definidas en D.1.

Para la característica de ruteo, se llevaron a cabo las dos primeras pruebas que analizan los aspectos básicos de esta función. Restando por realizar las pruebas de funcionamiento relacionadas con el orden de exploración de los vecinos en la búsqueda por rutear el paquete, instalación de rutas inversas, y detección de bucles de ruteo.

Para la última sección de pruebas relacionadas con el servicio de resolución de nombres, resta por encarar las pruebas de resolución en un nodo remoto, y mantenimiento del servicio en caso de desconexión de un servidor *Rendez-Vous*.

Capítulo 6

Conclusiones

Se logró embeber las funciones de red de la versión reactiva de ANTop sobre IPv6, en redes reales inalámbricas de computadoras.

El esquema básico del software obtenido consta de un demonio que corre en el Espacio de Usuarios, el cual está en constante comunicación con el Espacio de Kernel, leyendo y escribiendo datos, y un modulo de Kernel que se carga al momento de la ejecución del demonio, cuya tarea es la de enviar los paquetes de tráfico al Espacio de Usuarios donde serán procesados. Como resultado de este procesamiento se disparan las diversas funcionalidades de red que presenta ANTop.

Con la presente versión del software se armó un paquete Debian [18] para instalarlo a través de las herramientas estándares de este tipo de distribuciones Linux.

El lenguaje utilizado para programar el demonio fue *C*, ya que las interfaces del sistema operativo están escritas con dicho lenguaje.

Se logró establecer un canal de comunicación con el Kernel del sistema operativo, mediante la utilización de sockets Netlink. De esta forma fue posible leer y escribir las entradas de la tabla de ruteo, así como las direcciones IPv6 de la interfaz inalámbrica y las máscaras.

Se logró mapear las direcciones relativas de ANTop en direcciones IPv6. Para ello se utilizó direcciones de tipo *Global Unicast*.

Se logró intercambiar con otros nodos mensajes ANTop generados por el demonio corriendo en el Espacio de Usuarios. Para esto se utilizaron sockets UDP.

Fue posible utilizar la herramienta Netfilter, la cual dispone una seria de ganchos a lo largo de la pila de protocolos de red. En estos ganchos se puedo registrar una función de manejo del evento generado cuando un paquete pasa por alguno de ellos. En esta implementación, dicha función envía los paquetes al Espacio de Usuarios. Gracias a esto se pudo leer y modificar los diversos campos de un paquete de datos, como por ejemplo las direcciones IPv6 origen y destino.

Se logró insertar información esencial para el funcionamiento de ANTop en los paquetes ruteados por cada nodo. Se pudo implementar encabezados de extensión, más específicamente, el tipo *Destination Options* para registrar en el paquete la dirección del último por el que fue ruteado.

Dicha serie de modificaciones a los paquetes, en algunas ocasiones generó la necesidad de recalcular el Checksum tanto de UDP como TCP, pudiendo lograrlo exitosamente.

Fue posible montar el servicio de resolución de nombres que plantea ANTop: Rendez-Vous. Para esto se utilizó la estructura DNS ya incluida en el sistema operativo, logrando hacer esta implementación transparente a los fines de las aplicaciones y del usuario mismo. Aquí también fue necesario agregar una entrada a la tabla ARP, lo cual se logró exitosamente desde el demonio que corre en el Espacio de Usuarios.

Se logró realizar trasferencias seguras de archivos a través de *Secure Copy* (SCP). Los resultados obtenidos fueron los siguientes,

- 8.4 Mbps para nodos próximos
- 6.8 Mbps para una distancia de 10 metros, con una pared entre los nodos
- 2 Mbps para nodos en pisos contiguos de un edificio, separados por el techo y luego dos paredes

6.1. Trabajo a futuro

- *Pruebas de funcionamiento*: Completar las pruebas de funcionamiento definidas en el anexo D. De este modo se podrá hacer una evaluación general de los resultados obtenidos con el diseño creado en la presente Tesis, y además buscar solución para problemas que puedan surgir de dichas pruebas.

- *Uso de direcciones Link Local*: Es deseable utilizar este tipo de direcciones, ya que el uso de las mismas es más natural para una red privada. Para lograrlo, se debe resolver el uso del servicio mDNS en lugar de DNS al realizar un pedido de resolución de dirección universal. El primero es el que el sistema operativo implementa al configurar la interfaz inalámbrica con una dirección IPv6 Link Local.

- *Conexión de la red ANTop con otras redes*: En el presente trabajo se consideró que la red ANTop está aislada. A futuro es deseable que los usuarios de la misma puedan acceder a Internet, para ello se debe resolver el punto anterior e implementar algún sistema de NAT.

- *Chequear unicidad de direcciones universales*: Se deberá implementar un mecanismo por el cual los servidores Rendez-Vous corroboren que los registros de direcciones universales recibidos sean únicos. En caso de que esto no sea así deberán dar aviso al nodo que intenta registrarse y en este último se deberá pedir al usuario que elija una nueva dirección.

Apéndice A

Algoritmos

En esta sección se presentan los algoritmos definidos en el trabajo [1], para la implementación de ANTop en su versión reactiva.

A.1. Algoritmo de Conexión de Nodos

En este apartado se ve el algoritmo principal 1 que especifica el mecanismo de conexión de los nodos a la red ANTop.

A.2. Algoritmos de Ruteo

Aquí se presentan los algoritmos diseñados en la especificación de ANTop para responder a las funciones de red requeridas.

El algoritmo 2 describe el mecanismo de exploración de vecinos para la búsqueda de rutas. El algoritmo 3 presenta el mecanismo general de ruteo de paquetes.

A.3. Algoritmos Rendez-Vous

Aquí se describen los algoritmos que definen las funciones de resolución de nombres.

Los primeros dos algoritmos 4 y 5 se ejecutan en el cliente Rendez-Vous, y describen el proceso mediante el cual un nodo, realiza una consulta de resolución de nombre.

Algorithm 1: Establecimiento de dirección primaria

```
1.1  begin
1.2      i:=N_PAR;
1.3      repeat
1.4          Enviar paquete PAR;
1.5          Espera un tiempo T_PAP, almacenando los paquetes PAP
             recibidos;
1.6          i:=i − 1;
1.7          if se recibió al menos un paquete PAP con dirección disponible
             then
1.8              ir a 1.11;
1.9          end
1.10     until i==0 ;
1.11     if se recibió al menos un paquete PAP con dirección disponible
         then
1.12         Elegir entre las direcciones propuestas una con la menor
             máscara;
1.13         Enviar un paquete PAN en modo broadcast notificando la
             dirección elegida;
1.14         Esperar a que llegue un paquete PANC o que transcurra un
             tiempo T_PANC;
1.15         if se recibió un paquete PANC then
1.16             La dirección primaria elegida ya puede ser utilizada. fin del
                 algoritmo;
1.17         end
1.18         ir a 1.2;
1.19     else
1.20         if se recibió un paquete PAR indicando que no hay direcciones
             disponibles then
1.21             No se puede conectar. Fin del algoritmo;
1.22         end
1.23         Utilizar la dirección compuesta por todos ceros;
1.24     end
1.25 end
```

Algorithm 2: sendToNextNeighbor(M. w)

```
2.1  begin
2.2      if no existe el VisitedBitmap de vecinos para v-> z then
2.3          crear el VisitedBitmap, y marcar el vecino w como visitado;
2.4          iniciar Bitmap Timer (entrada) /* eliminará solo el
                 Visited Bitmap de la entrada                    */
2.5      end
2.6      if están todos los bits marcados then
2.7          marcar en la entrada de la tabla de ruteo que no hay ruta para
             z;
2.8          buscar en la tabla de ruteo v-> x : y, n;
2.9          devolver el paquete a y;
2.10         fin del algoritmo;
2.11     end
2.12     if hay NBP + 1 bits marcados then
2.13         NextHop = primer bit /* padre                      */
2.14     else
2.15         if el primer bit está marcado then
2.16             NextHop = siguiente bit no marcado;
2.17         end
2.18     end
2.19     if el primer bit no está marcado then
2.20         NextHop = bit no marcado correspondiente al vecino con
             mínima distancia al destino en el VisitedBitmap;
2.21     end
2.22     iniciar Route Timer (entrada) /* eliminará toda la entrada
             en la tabla de ruteo                            */
2.23     marcar NextHop en el VisitedBitmap y enviar el paquete por ese
         nodo.;
2.24 end
```

Algorithm 3: route(M,w)

```
3.1  begin
3.2      v recibe un mensaje M(x, z) del vecino w;
3.3      if M está marcado como devuelto then
3.4          SendToNextNeighbor(M, w);
3.5          fin del algoritmo;
3.6      end
3.7      if se encuentra en la tabla de pares a (x, z) con distancia al origen
         menor que M then
3.8          devolver el paquete por donde vino;
3.9          fin del algoritmo;
3.10     end
3.11     buscar en la tabla de ruteo v− > x : w, n (entrada inversa) y
         agregarla si no se encuentra;
3.12     buscar en la tabla de ruteo v− > z : y, m;
3.13     if se encontraron entradas then
3.14         enviar el paquete por la entrada con mínima distancia.;
3.15     else
3.16         sendToNextNeighbor(M.w);
3.17     end
3.18     agregar en la tabla de pares a (x,z) si no estaba anteriormente;
3.19 end
```

Algorithm 4: send(U,M)

```
4.1  begin
4.2      buscar el par (U, V) en la caché;
4.3      if se encontró (U, V) then
4.4          enviar M a la dirección V;
4.5      else
4.6          E = H(U);
4.7          enviar un paquete AddressLookup a E pasándole la dirección
             U;
4.8          agregar (U, M) en el vector WaitQueue
4.9      end
4.10 end
```

Algorithm 5: receive(M)

```
5.1  begin
5.2      if M es un paquete de tipo AddressSolve then
5.3          if el flag solved de M vale verdadero then
5.4              agregar en la caché (U, V);
5.5              for cada par (Um, M) en WaitQueue do
5.6                  if Um == U then
5.7                      enviar M a la dirección V;
5.8                  end
5.9              end
5.10         else
5.11             notificar mediante un mensaje interno que U no se puede
                 resolver;
5.12         end
5.13     end
5.14 end
```

En el algoritmo 6 se ejecuta en el servidor Rendez-Vous y describe el procesamiento de los paquetes que recibe. Aquí se debe aclarar que se eliminaron las lineas correspondientes al intercambio de mensajes de tipo

- Deregister
- Lookup Table
- Lookup Table Received
- ReadyForDisc

Ya que estos mensajes no se utilizan en la presente implementación de ANTop y su presencia en el algoritmo sumaba una complejidad innecesaria.

Estos mensajes tiene la finalidad de controlar el proceso de desconexión de un nodo. En particular, el segundo y el tercero hacen referencia a la transferencia de los registros de la tabla de resolución de nombres. Sin embargo, como se vio en la sección 4.5, la información de los registros es redundante ya que la misma es enviada a diversos servidores dispersos por toda la red ANTop. Esta información es enviada periódicamente por cada uno de los nodos que participa de la red y goza de una dirección relativa asociada a una dirección universal.

En general, el proceso de desconexión encarado en esta implementación es automático y no requiere de intercambio de paquetes de control.

Algorithm 6: receive(M)

```
6.1  begin
6.2      if M es un paquete de tipo Register then
6.3          agregar (U, V) en la tabla de resolución;
6.4      else
6.5          if M es un paquete de tipo AddressLookup then
6.6              buscar U en la tabla de resolución;
6.7              if se encontró U en la tabla de resolución then
6.8                  responder con un paquete AddressSolve con la dirección
                     de red correspondiente;
6.9              else
6.10                 responder con un paquete AddressSolve con
                     Solved = false;
6.11             end
6.12         end
6.13     end
6.14 end
```

Apéndice B

Filtro de Bloom

Genéricamente se define un filtro de Bloom como un vector de m bits que codifica la pertenencia de un juego

$$A = (a_1, a_2, ..., a_n)$$

de n elementos. La idea es generar un vector v de m bits, inicialmente todos nulos, y luego elegir k funciones de Hash independientes,

$$h_1, h_2, ..., h_k$$

cada una con rango

$$\{1, ..., m\}$$

Para cada elemento a de A, los k bits en las posiciones

$$h_1(a), h_2(a), ..., h_k(a)$$

llevan un valor de 1.

Para verificar si un elemento b pertenece al conjunto A, es suficiente verificar si todos los k bits en las posiciones

$$h_1(b), h_2(b), ..., h_k(b)$$

en v tienen un valor de 1. Si alguno de ellos tiene valor nulo, entonces se tiene certeza de que b no pertenece a A. Si todos tienen valor 1, entonces es probable que b sea un elemento de A. Sin embargo existe una probabilidad de que b no sea miembro de A. Esto se llama un *falso positivo*.

La probabilidad de tener falsos positivos, decrece a medida que el tamaño del filtro m crece.

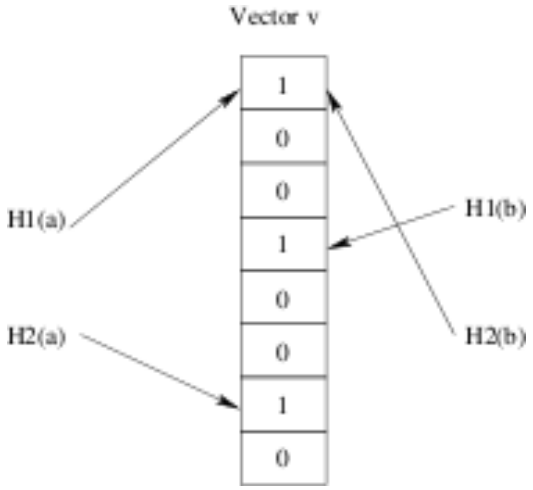


Figura B.1: Filtro de Bloom

En el modelo propuesto, el conjunto A está formado por las direcciones IP de los nodos intermedios en el camino entre origen y destino. Por otro lado, n es igual a la longitud del camino.

El overhead, ahora, estará dado por el tamaño del filtro de Bloom, ya que es lo que se envía en el paquete IP, en lugar del listado con las direcciones intermedias. De esta forma, cuanto mayor sea el overhead, menor será la probabilidad de tener falsos positivos, y por ende de que las rutas sean incorrectas.

Con la mejora de performance presentada por HB-DSR, se vuelve factible montar una red DSR, sobre IPv6. Para ello también se hace uso de los encabezados de extensión, los cuales permiten incorporar información adicional en el encabezado IPv6. En particular se utiliza el tipo de encabezado *Hop-by-Hop Options*, que permite agregar opciones en el paquete que deben ser procesados por cada nodo intermedio.

Esta funcionalidad de IPv6 se explica con mayor profundidad en la sección E.

Apéndice C

Configuración de la PC para usar ANTop

Para usar las funcionalidades que ANTop presenta, es necesario realizar algunas configuraciones sobre el sistema operativo.

C.1. Instalación de bibliotecas

Netlink es una poderosa herramienta para interactuar con la configuración de los funcionalidades de red del sistema operativo. Para volver más fácil el manejo de su interfaz de desarrollo, a fines de 2010 se desarrolla *libmnl*, acrónimo de Library Minimalistic Netlink. Esta es una biblioteca de Espacio de Usuario orientada a desarrolladores de Netlink, y que simplifica muchas tareas de formateo, validación y construcción de encabezados. Estas tareas de otro modo pueden volverse muy repetitivas, y fácilmente causar errores.

La biblioteca libmnl es parte del proyecto *Netfilter.org*[1].

La presente implementación hace uso de la biblioteca libmnl, con lo cual debe bajarse el código fuente de la página del proyecto Netfilter. Una vez descomprimidos los archivos, se deben ejecutar los siguientes comandos, en dicha carpeta.

```
./configure
make
sudo make install
```

[1]http://www.netfilter.org

Por otro lado, se debe instalar la biblioteca de Espacio de Usuario *libnetfilter_queue*, la cual presenta una API para manejar los paquetes que han sido encolados por el filtro del Espacio de Kernel. Esta biblioteca reemplaza el viejo mecanismo *ip_queue/libipq*.

Para que *libnetfilter_queue* funcione, se debe instalar previamente *libnfnetlink*. Esta última es una biblioteca de bajo nivel para comunicación entre los espacios de usuario y kernel, en cuanto a la estructura Netfilter. Se desarrolla no como una API pública para desarrolladores de aplicaciones, sino de uso interno para otras herramientas del proyecto Netfilter.

Ambas bibliotecas deben instalarse de igual forma que libmnl.

C.2. Configuración de la conexión inalámbrica

Para configurar la conexión inalámbrica se debe abrir el administrador de conexiones de red y crear una nueva. El modo debe llevar el valor de *Ad-hoc*.

Figura C.1: Configuración de la conexión inalámbrica

Se debe setear una dirección inicial de tipo Global Unicast. Según lo discutido en 4.2.2, la máscara de red deberá valer los 128 bits de la dirección IPv6, menos la cantidad de bits que representarán a la dirección relativa ANTop.

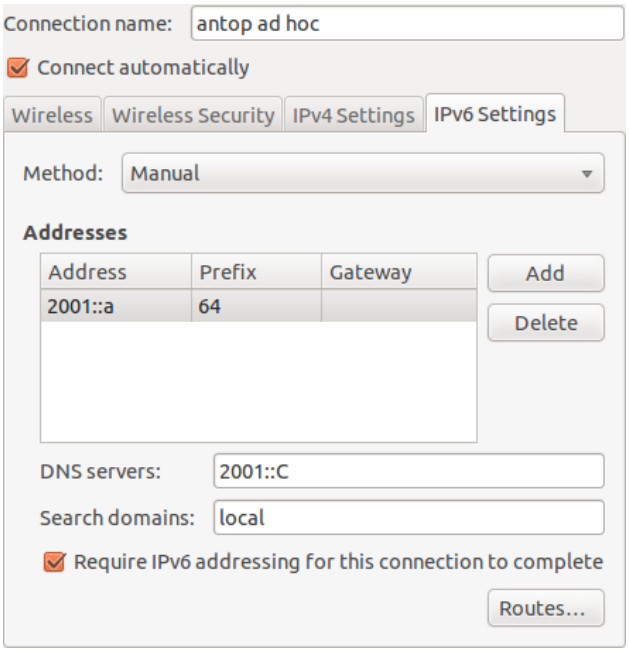


Figura C.2: Configuración de la conexión inalámbrica

C.3. Ejecución del demonio ANTop

Para ejecutar el demonio ANTop en el Espacio de Usuario, se debe especificar la interfaz de red, la dirección universal de nodo y la dimensión del hipercubo.

Por ejemplo, la siguiente es una ejecución válida

```
sudo ./antopd -i wlan0 -u nodo_matias -l 64
```

El programa debe ejecutarse con privilegios de súper usuario.

Cuando se finaliza el proceso que corre ANTop, es necesario remover el módulo de Kernel para manejo de paquetes, insertado al momento de la ejecución del proceso. Para ello se debe usar el siguiente comando,

```
sudo rmmod kantop
```

Apéndice D

Protocolo de prueba

Aquí se especifican las secuencias de pasos a seguir, con el fin de probar que el funcionamiento de la presente implementación de ANTop se corresponda con el diseño del mismo, encarado este último en los trabajos de Alejandro Marcu [1] y Gaston Tejia [2].

Dicho protocolo busca básicamente corroborar el funcionamiento de ANTop, según la definición de los algoritmos del anexo A.

Cada una de las secciones de este capítulo responde a un aspecto básico de los servicios de red que el protocolo brinda. En el primer apartado se trata la conexión de nodos. El segundo apartado busca corroborar el desempeño del algoritmo de ruteo de tráfico, mientras que el último capítulo se aboca a la resolución de direcciones universales en relativas, realizada por el servicio *Rendez-Vous*.

D.1. Conexión de nodos

Este apartado encara la verificación del comportamiento de la presente implementación de ANTop, en cuanto a la conexión y desconexión de nodos a la red. La metodología utilizada se basa en el seguimiento de la evolución de la topología de una red ANTop, iniciando desde su etapa más temprana, cuando aún no se ha conectado ningún nodo.

- El primer nodo que quiere participar en la red, envía los paquetes *PAR*, pero al no tener respuesta, se asigna a si mismo el espacio de direcciones completo, adoptando una máscara de valor nulo.

- Luego se une un segundo nodo. Este recibe un espacio de direcciones del nodo 1, con máscara 1. Este último deberá incrementar el valor de su máscara. Ahora cada uno de los nodos, administra mitad del espacio de direcciones del Hipercubo.

- El siguiente paso es conectar un tercer nodo a la red ANTop. Este último deberá estar dentro del alcance de la placa inalámbrica de los dos primeros, de modo tal que reciba ofertas de dirección de ambos. Dichas ofertas tendrán igual máscara con un valor de 1, con lo cual el tercer nodo elige la oferta que recibe primero.

- Esta experiencia se debe repetir algunas veces, y si se observa que la oferta de uno de los dos nodos es elegida con mayor preferencia, se debe medir el tiempo de respuesta entre el nodo 3 y los dos primeros, mediante un Ping. Para el nodo que se elige con mayor preferencia se debe observar un tiempo de respuesta inferior, que justifique la tendencia.

- Como siguiente paso, se deberá procurar que cada uno de los tres nodos, ahora conectados a la red ANTop, se mantenga dentro del alcance de los otros dos. En primer instancia, no se podrá establecer relaciones de vecinos entre todos los nodos, ya que algún par de direcciones primarias diferirán en más de un bit. Sin embargo, cuando esta situación sea detectada por los nodos, mediante el intercambio de *Heart Beats*, alguno de ellos se deberá procurar una dirección secundaria, de modo tal que cada uno de los nodos sea vecino de los otros dos.

- Habiendo verificado la cesión de direcciones secundarias, se procede a desconectar de la red al segundo nodo. Luego de algunos instantes utilizados para la reconvergencia de la red, el nodo 1 deberá recuperar el espacio de direcciones del nodo desconectado por ser el padre del mismo.

- Ahora, se vuelve a conectar el segundo nodo, estando dentro del alcance de ambas conexiones inalámbricas. El mismo deberá recibir dos ofertas de espacio de dirección. El nodo 1 ofrecerá el espacio recuperado, el cual tendrá una máscara menor que la oferta del nodo 3. Por esta razón, deberá elegir dicha opción y continuar participando de la red ANTop normalmente, como lo estaba haciendo antes de la desconexión.

D.2. Ruteo de tráfico

La primer prueba de este protocolo tiene el objetivo de corroborar el funcionamiento del mecanismo de forwardeo en cada nodo, es decir que se pone a prueba la capacidad de cada nodo para reenviar paquetes que no lo tienen como destinos a aquel vecino que tenga la menor distancia con la dirección a la cual está dirigida el paquete.

La segunda prueba que se diseña tiene el objetivo de probar que cada nodo procese aquellos paquetes *Rendez-Vous* cuyo destino está dentro del espacio de direcciones local.

La tercer prueba del protocolo se monta en una topología algo más compleja que las anteriores y tiene por objetivo corroborar el funcionamiento general de gran parte de las funciones de ruteo definidas en ANTop.

En la cuarta y última prueba se verifica el funcionamiento del mecanismo por el cual se detectan y rompen lazos de ruteo.

A partir de aquí se encara la primera de las pruebas con una topología como la que se muestra en la figura 5.24.

- Se cuenta con los primeros tres nodos conectados a la red ANTop, con la particularidad de que el nodo 3 está dentro del alcance del nodo 1, pero no del 2. De este modo serán vecinos el 1 con el 2 y el 3, pero estos dos últimos no lo serán entre sí. Este esquema fuerza a dichos nodos a comunicarse a través del nodo 1. Este último deberá cumplir con la función de reenviar paquetes que no le son destinados. La prueba efectiva consiste en enviar un Ping entre los nodos 2 y 3 y que el mismo sea enrutado correctamente por el nodo 1, tanto de ida como de vuelta.

Para la segunda de las pruebas se tiene una topología simple con dos nodos. Como aquella vista en la figura 5.37.

- La prueba consiste básicamente en buscar una dirección universal para el nodo 2, de modo tal que uno de los servidores RV resultantes luego de aplicar la función de *Hash* al mismo, caiga dentro del espacio de direcciones del nodo 1, sin ser precisamente la del nodo 1. Para facilitar esta tarea se debe considerar que cuanto mayor sea la dimensión del hipercubo definida por el primer nodo conectado a la red ANTop, mayor será la cantidad de servidores RV a los que se envíe el registro de

dirección universal y por ende mayor la probabilidad de que la condición de esta prueba se cumpla.

Para llevar adelante la tercer prueba se requiere una topología algo más compleja que en los casos anteriores. La misma se presenta en la figura D.1.

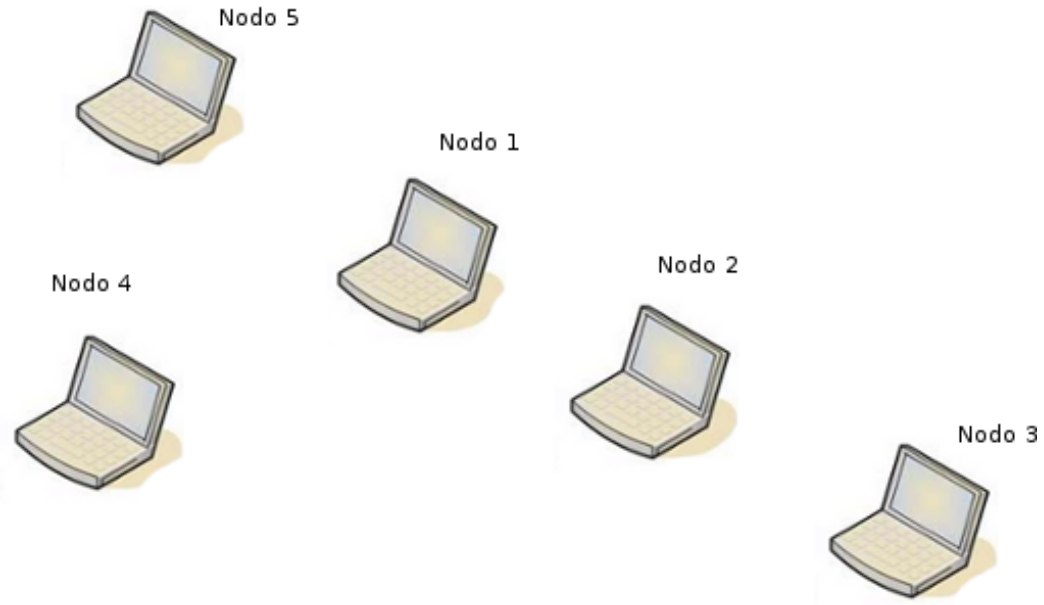


Figura D.1: Topología para la tercer prueba de este protocolo

- Aquí el nodo 1 será vecino del 2, del 4 y del 5. El 3 por otro lado, solo tendrá como vecino al 2. La prueba consiste básicamente en enviar un Ping desde el nodo 1 hacia el 3. En dicho caso, el primero deberá buscar en sus vecinos cual es el que tiene menor distancia al destino, es decir al nodo 3. En caso de que esta condición la cumplan o bien el nodo 5 o el 4, se deberá corroborar con un capturados de tráfico, que el paquete ICMPv6 sea dirigido hacia alguno de ellos dos. Luego el mismo deberá ser devuelto, ya que ninguno de los dos tiene otro vecino a quien reenviarlo. Una vez devuelto el paquete, el nodo 1 chequeará su tabla de vecinos visitados para este destino, y decidirá enviarlo a través del nodo 2.

  Por otro lado, si el nodo 2 es que tiene originalmente menor distancia al nodo 3, entonces se debe corroborar mediante un capturador de tráfico que el paquete ICMPv6 Echo Request sea ruteado directamente a través del nodo 2.

En este punto, el nodo 2 deberá enviar el paquete hacia el 3, ya que el 1 es marcado inmediatamente como un vecino ya visitado. Una vez que el paquete llegue al nodo 3, se debe corroborar en la tabla de ruteo que se haya instalado una entrada inversa para llegar al nodo 1 a través del 2.

Como respuesta al paquete ICMPv6 Echo Request, el Echo Reply deberá ser ruteado sin inconvenientes hacia el nodo 1.

Si el comportamiento de la red al rutear el paquete concuerda con lo dicho aquí, entonces se habrán probado las siguientes funcionalidades

- Elección de próximo salto según criterio de distancia mínima al destino
- Devolución de un paquete al último salto, en caso de no tener una ruta disponible.
- Instalación de rutas inversas.
- Marcado del último salto como vecino ya visitado en el *Visited Bitmap* de dicha ruta.

La última prueba se aboca a corroborar el funcionamiento de la detección de bucles. Para ello se considera una topología como la de la figura D.2.

- En este caso, se instalará una entrada en la tabla de ruteo del nodo 1, con una dirección destino inexistente en la red, pero cuyo próximo salto será el nodo 2. En este último también se instala una entrada con el mismo destino, pero en este caso el próximo salto será el nodo 3. Para los nodos 3 y cuatro se repite el mismo procedimiento.

  Cuando el nodo 1 envía un Ping a la dirección inexistente utilizada como destino de las rutas instaladas, el paquete generado circulará en la red tal según el sentido indicado por las flechas de la figura D.2.

  El nodo 3 es vecino del 2 y del 4, pero no del 1. Por esta razón, al recibir el paquete ICMPv6 que deberá reenviar al nodo 4, instalará una entrada inversa en su tabla de ruteo. Dicha ruta tendrá como destino el nodo 1 y como próximo salto el nodo 2.

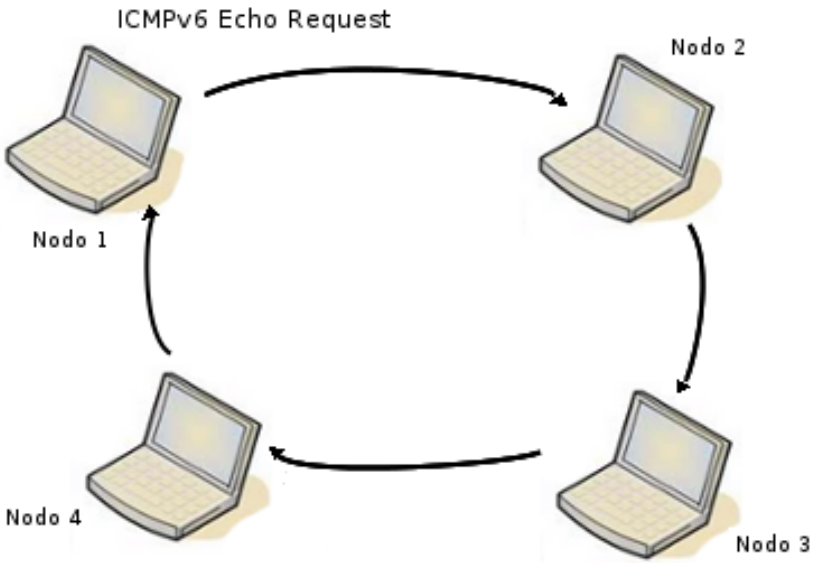


Figura D.2: Topología para prueba de detección de loops

El paquete continuará viajando en la topología siguiendo el sentido de circulación indicado por las flechas, decrementando el valor de su campo *Hop Limit* en el encabezado IPv6. Al llegar al nodo 3, el mismo utiliza dicho campo para calcular los saltos que ha recorrido este paquete y luego corrobora que tiene una entrada inversa instalada con una distancia al origen menor. De esta forma detecta que el mismo está en un bucle de ruteo. Por esta razón dicho nodo marcará el paquete como devuelto y lo enviará al nodo 2, es decir al último salto.

El nodo 2, al ver que el paquete tiene la bandera de "devuelto" encendida, inicia el mecanismo de exploración de vecinos, cuyo primer paso será marcar como visitado el vecino 3. Como resultado de dicho mecanismo, el paquete será enviado al nodo 1, que es el único vecino restante por visitar. El nodo 1 por su parte detecta que el mismo originó el paquete con lo cual lo devuelve al 2. Este último al ya haber visitado todos los vecinos, marca en su tabla que no hay ruta hacia ese destino y descarta el paquete. Con esto último se previene que esta situación vuelva a suceder en lo inmediato.

Si el comportamiento de la red es el descripto aquí se habrá corroborado exitosamente el mecanismo de detección de bucles y además la detección por parte de la red ANTop de que en verdad no existe una ruta hacia el destino

requerido.

D.3. Servicio Rendez-Vous

En este apartado se presentan tres pruebas para verificar el funcionamiento del servicio de resolución de nombres *Rendez-Vous*. La primera evalúa el criterio de selección del servidor de nombres a menor distancia, particularmente que la resolución del nombre se realice localmente. La segunda fuerza la resolución de una dirección universal mediante el intercambio de paquetes *Address Lookup* y *Address Solve*. Por último se define una prueba para corroborar la redundancia de los registros de nombres, en caso de caída de un servidor *Rendez-Vous*.

La primer prueba utiliza una topología como la vista en la figura 5.37, con dos nodos conectados a la red ANTop. El primero de la red, y su sucesor.

- La dimensión del hipercubo y la dirección universal del nodo 2 se deben elegir de tal forma que uno de los registros del mismo caiga en el nodo 1. En ese caso, cuando el nodo 1 deba seleccionar el servidor que se encuentre a menor distancia, para resolver la dirección universal del nodo 2, se seleccionará a si mismo.

  Bajo estas consideraciones se envía un Ping del nodo 1 al 2, esperando que el comportamiento sea el recién presentado.

- Para la segunda prueba se utiliza la misma topología que en el caso anterior, pero aquí se debe procurar que la dirección universal que se desea resolver no se encuentre registrada en el nodo local.

  Bajo este esquema, se deberá observar el intercambio de paquetes *Address Lookup* y *Address Solve* entre ambos nodos. Esto puede hacerse con un analizador de tráfico.

Para la última prueba se utiliza una topología como aquella de la figura D.3.

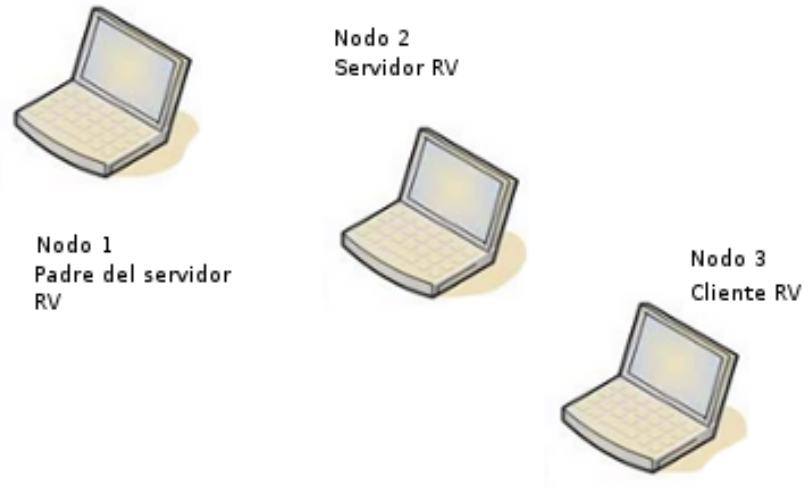


Figura D.3: Topología para prueba del servicio Rendez-Vous

- En esta prueba, el nodo 2 resuelve una consulta generada por el 3. Posteriormente, el nodo 2 se desconecta de la red. Ahora, al realizar la misma consulta de dirección universal por parte del nodo 3, deberá ser el 1 quien se haga cargo de la resolución de la misma, tras haber recuperado el espacio de direcciones cedido previamente cuando 2 se conectó.

  Con un analizador de tráfico se deberá observar que ahora los paquetes de *Address Lookup* y *Address Solve* se intercambian entre los nodos 1 y 3.

Apéndice E

Encabezados de extensión en IPv6

El protocolo IPv6 [19] presenta la oportunidad de colocar información adicional referente a la capa de red, la cual puede ser ubicada entre el encabezado IPv6 y el encabezado de la capa superior en un paquete. Hay un número reducido de encabezados para información adicional, y cada uno de ellos se define por un valor de *Next Header*. Como se puede ver en la figura, un paquete IPv6 puede transportar cero, uno o más encabezados de extensión.

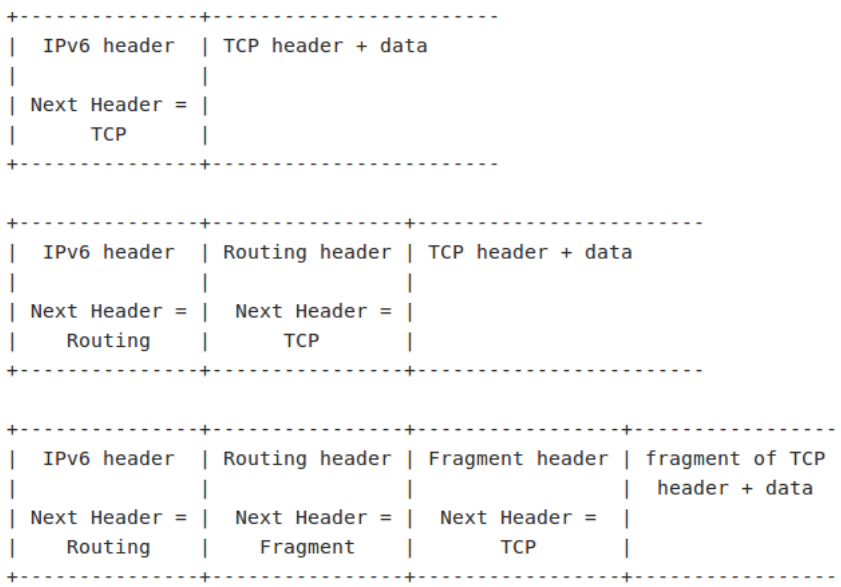


Figura E.1: Encabezados de extensión en IPv6

Cada encabezado de extensión tiene un tamaño múltiplo de 8 octetos, de forma de obtener dicha alineación para los encabezados subsiguientes. Los campos con n octetos son situados con un desplazamiento de de n octetos

desde el inicio del encabezado, con

$$n = 1, 2, 4, 8$$

Una implementación completa de IPv6 debe incluir los siguientes encabezados de extensión,

- Hop-by-Hop Options
- Routing (Type 0)
- Fragment
- Destination Options
- Authentication
- Encapsulating Security Payload

El único encabezado que es examinado por nodos intermedios, es aquel correspondiente a opciones *Hop-by-Hop*. Adicionalmente, es procesado por el origen y el destino. Cuando este encabezado está presente, debe ubicarse inmediatamente después del encabezado IPv6. Su presencia se indica con un valor de cero en el campo *Next Header* IPv6.

Cada uno de estos encabezados puede estar presente una vez o ninguna, a excepción del tipo *Destination Options*, que puede estar incluido hasta dos veces en el mismo paquete, una antes de un encabezado de tipo *Routing* y una antes del encabezado de capa superior.

Los tipos de encabezados *Hop-by-Hop* y *Destination Options* llevan un número variable de opciones formateadas como *type-length-value* (TLV), de la siguiente forma,

Figura E.2: Opciones de encabezados de extensión en IPv6

Option Type Identificador de 8 bits del tipo de opción

Opt Data Len Entero sin signo de 8 bits. Longitud del campo de datos de esta opción, expresada en octetos.

Option Data Campo de longitud variable. Datos específicos del tipo de opciòn.

Para el campo *Option Type*, los valores se codifican de modo tal que los dos bits más significativos especifican la acción a realizar en caso de que el nodo que recibe el paquete no reconoce el tipo de opción.

- 00 - Saltear esta opción y continuar procesando el paquete
- 01 - Descartar el paquete
- 10 - Descartar el paquete y mandar un ICMP Code 2.
- 11 - Descartar el paquete y mandar un ICMP, si el destino es unicast

La opción *Hop-by-Hop* se utiliza para llevar información adicional que debe ser examinada por cada nodo a lo largo del camino de ruteo. Si bien en el RFC 2460, solo se definen opciones de Pad, de este tipo, es posible definir nuevas.

La opción *Destination Option* se utiliza para enviar información adicional que solo debe ser examinada por el destino.

A continuación se muestra la forma de este tipo de encabezado.

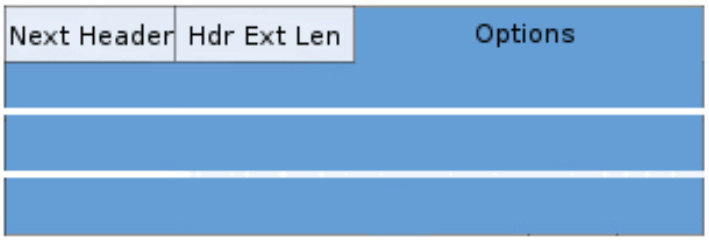


Figura E.3: Encabezado de Opciones de Destino

Next Header Campo de 8 bits. Identifica el tipo de encabezado que sigue al de Opciones de Destino.

Header Extension Length Campo de 8 bits. Tamaño del encabezado de opciones de destino, expresado en unidades de 8 octetos, sin incluir los primeros 8 octetos.

Options Campo de longitud variable. Su tamaño deberá ser tal que el tamaño completo del encabezado de opciones de destino, es un entero múltiplo de 8 octetos.

Es en el campo de opciones, donde se incluirá la dirección IPv6 del nodo que rutea el paquete, de modo tal que quien lo recibe, puede consultar este dato. Aquí se debe respetar el formato que se muestra en la figura E.2.

Bibliografía

[1] A. Marcu, *Desarrollo y simulacion de un protocolo para redes ad-hoc*. Universidad de Buenos Aires, Facultad de Ingeniería, July 2007. http://cnet.fi.uba.ar/alejandro.marcu/Tesis_Alejandro_Marcu.pdf.

[2] G. Tejia, *Estudio y análisis de mecanismos orientados al robustecimiento de ANTop, utilizando ruteo proactivo*. Universidad de Buenos Aires, Facultad de Ingeniería, Nov. 2009. http://cnet.fi.uba.ar/gaston.tejia/Tesis_Gaston_Tejia.pdf.

[3] J. I. Alvarez-Hamelin, A. C. Viana, and M. D. de Amorin, "DHT-based functionalities using hypercubes," in *Ad-hoc Networking*, vol. 212/2006, pp. 157–176, IFIP International Federation for Information Processing, Springer Boston, ISSN 1571-5736 (Print) 1861-2288 (Online).

[4] E. Perkins, C. Belding-Royer and S. Das, "RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing," July 2003.

[5] Y. Johnson, D. Hu and D. Maltz, "RFC 4728: The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," July 2007.

[6] D. B. Johnson, "Routing in ad hoc networks of mobile hosts," in *In Proc. of IEEE Workshop on Mobile Computing Systems and Applications*, pp. 158–163, 1994.

[7] D. B. Johnson and D. A. Maltz, *Mobile Computing*. Kluwer Academic Publishers, 1996.

[8] M. Barbeau and E. Kranakis, *Principles of AD HOC Networking*. John Wiley & Sons, 2007.

[9] C. Castelluccia, "Hash-based dynamic source routing (hb-dsr)," tech. rep., Institut National de Recherche en Informatique et en Automatique, 2003.

[10] B. H. Burton, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, 1970.

[11] J. Salim, H. Khosravi, A. Kleen, and A. Kuznetsov, "RFC 3549: Linux Netlink as an IP Services Protocol," July 2003.

[12] "IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1 –2793, 29 2012.

[13] "Iso/iec standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements- part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 4: Further higher data rate extension in the 2.4 ghz band," *ISO/IEC 8802-11:2005/Amd.4:2006(E) IEEE Std 802.11g-2003 (Amendment to IEEE Std 802.11-1999)*, pp. c1 –68, 2006.

[14] S. Deering and R. Heiden, "RFC 4291: IP Version 6 Addressing Architecture," Feb. 2006.

[15] T. Narten, S. Thomson, and T. Jinmei, "RFC 4862: IPv6 Stateless Address Autoconfiguration," Sept. 2007.

[16] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)," Sept. 2007.

[17] R. W. Stevens, *TCP/IP Illustrated, Vol 1: The Protocols.* Addison-Wesley, 1994.

[18] "ANTop (Adjacent Networks Topologies) ad-hoc routing." El enlace estará disponible en breve.

[19] S. Deering and R. Heiden, "RFC 2460: Internet protocol, Version 6 (IPv6) Specification," Dec. 1998.