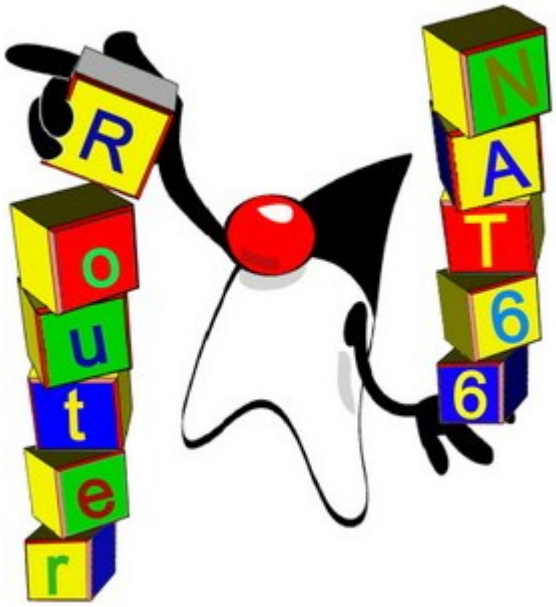

Proyecto de Grado 2008

Anexo II

IP4JVM – Router & NAT66

Autores:

Leandro Scasso

Marcos Techera

Tutor:

Ariel Sabiguero

Tribunal:

Andrés Aguirre

Eduardo Grampín

Carlos Martínez

Tabla de contenidos

1.Introducción

Durante este capitulo se presentar el problema afrontado en esta etapa del proyecto **IP4JVM**, comenzando con una descripción del estado inicial del mismo para luego pasar a enumerar cuales son los objetivos que se pretenden alcanzar al finalizar esta etapa. Por último, y no menos importante, se describe la organización del resto del documento.

1.1.Presentación del Problema

En la actualidad no existen demasiadas implementaciones de **IPv6** en compración con la cantidad de dispositivos IP que existen hoy en día y los pocos que cuentan con la aprovación de IPv6 Ready[IPv6R] (no llegan a 300 dispositivos). IPv6 Ready tiene como objetivo la definición de los tests de conformidad e interoperabilidad para IPv6 y es quien da la certificación **IPv6 Ready Logo** en caso de pasar estos tests de forma satisfactoria.

Por otro lado se encuentra el proyecto IP4JVM, el cual ya cuenta con el aporte de dos trabajos academicos que analizaron y realizaron desarrollos sobre una máquina virtual Java para que esta tuviera un soporte propio para el manejo de operaciones de redes en IPv6 y no uno dependiente del sistema operativo. A partir de lo realizado en etapas anteriores de este proyecto, es que se decidió continuar extendiendo el mismo realizando una implementación que permita configurar un nodo para realizar las funcionalidades básicas de un router. A estás funcionalidades básicas se decidió agregar también la realización de **NAT** / **PAT** para **IPv6**.

1.2.Situación inicial del proyecto IP4JVM

Al comenzar esta etapa, el proyecto **IP4JVM** permitía la configuración de nodos como hosts. Pero no existía ninguna funcionalidad de router. Por otra parte, siempre se había considerado en el diseño la posible implementación router en algún momento.

1.3.Resultados Esperados

Al culminar esta etapa del proyecto se pretende contar con una versión del proyecto **IP4JVM** ampliada que permita realizar la configuración de un nodo para poder realizar las funciones de un host y de un router.

Otros resultados deseados son:

•Implementación de envío de ***ICMPv6 Redirect***

•Estado del arte de **IP4JVM** de funcionalidades que permitan realizar funciones de router

•Estudio de alternativas para la implementación de **NAT** para **IPv6**

•Implementación de una solución de **NAT** para **IPv6**

•Realizar tests sobre el proyecto de forma que este tenga un nivel aceptable de funcionamiento

1.4.Organización de este documento

El presente documento se encuentra organizado en capítulos, cada uno de los cuales presenta un aspecto funcional del desarrollo del proyecto.

Capítulo 1, Introducción: en la introducción se define los términos generales del problema, los objetivos que se desean alcanzar y se describe la organización del documento.

Capítulo 2, Estado del arte: en este capítulo presenta lo que es un router y como funciona el ICMPv6 redirect[RFC4861]. También es definido que es NAT y es presentado el NAT66[NAT66], un borrador perteneciente a la IETF [IETF]. Por último se presenta cual es el estado actual del proyecto para afrontar las modificaciones necesarias para soportar Routing y NAT.

Capítulo 3, Análisis de la solución: aquí se presenta la solución implementada para que el proyecto tuviera soporte para Routing y NAT. También se hace mención a los errores, correcciones y nuevas impelemetaciones que se debieron realizarse sobre lo que ya se encontraba implementado en el proyecto para que este pudiera poseer las funcionalidades de Routing y NAT.

Capítulo 4, Cronogramas y organización del trabajo: en este capítulo se presentan la distribución de las horas de trabajo en las distintas tareas llevadas a cabo durante esta etapa del proyecto.

Capítulo 5, Resultados y Conclusiones: aquí se presentan las conclusiones arribadas luego de culminar la etapa correspondiente a la extensión de funcionalidades para permitir realizar las funciones de Routing y NAT, las limitaciones de la solución implementada y cuales son las posibles mejoras a realizarle.

2.Estado del Arte

Durante la fase de investigación, se estudio el estado en que se encontraba el proyecto IP4JVM y sus limitantes. Para está parte se hizo un énfasis especial en el estudio de todo el proceso llevado a cabo para seleccionar la interfaz y la dirección IP por donde se enviará el paquete, debido a la importancia de este proceso en el diseño de los requerimientos a agregar.

Se estudio también los distintos **RFCs** que definen a un router **IPv6**, RFCs 4861 [RFC4861] y 4862 [RFC4862]. Así como también distintas posibles implementaciones de NAT para IPv6, entre ellas un borrador de **NAT66** presentado en el **IETF** "**IPv6-to-IPv6 Network Address Translation**" [NAT66], y la implementación de NAT para IPv4, con las modificaciones necesarias para que funcione para IPv6.

Si bien también existen borradores de NAT64 o NAT46 no fueron tomados en cuenta debido a que el proyecto es un stack que sólo soporta IPv6.

2.1.Router

Un router, como se define en la terminología del RFC 4861, es un nodo que realiza el **forwarding** (o encaminamiento) de paquetes no explícitamente dirigidos a él. Mientras que un host es definido como cualquier nodo que no es un router.

Los routers permiten la conexión de dos o más interfaces lógicas a nivel de capa de red.

Dentro de las funcionalidades de un router la que se tuvo mayor interés en estudiar e implementar fue ***ICMPv6 Redirect***. El cual será explicado en el siguiente punto. Fue seleccionado debido a que es considerado importante en IPv6, tanto el envío por parte de los routers, como la recepción por parte de los hosts, para obtener el IPv6 Ready Logo [RE HACIPV6 GO].

Si bien en IPv6 el router puede implementar otras funcionalidades como el envío de **Router Advertisements** para anunciar el prefijo de red, mensajes de configuración de **DHCPv6**, etc; estos son opcionales.

2.1.1.ICMPv6 Redirect

A diferencia de **ICMPv4** el redirect en **ICMPv6** es un mensaje de información y no un mensaje de error.

El formato y cuando se deben utilizar se encuentra definido en el **RFC 4861**, **Neighbor Discovery** en **IPv6** [RFC4861]. Los mensajes de **ICMPv6** redirect son utilizados por los routers para notificar a los hosts que existe una mejor ruta para un destino cuando un paquete es enviado. Pueden ser redirigir a un router que se encuentra en un mejor primer paso o que el destino es un vecino del host.

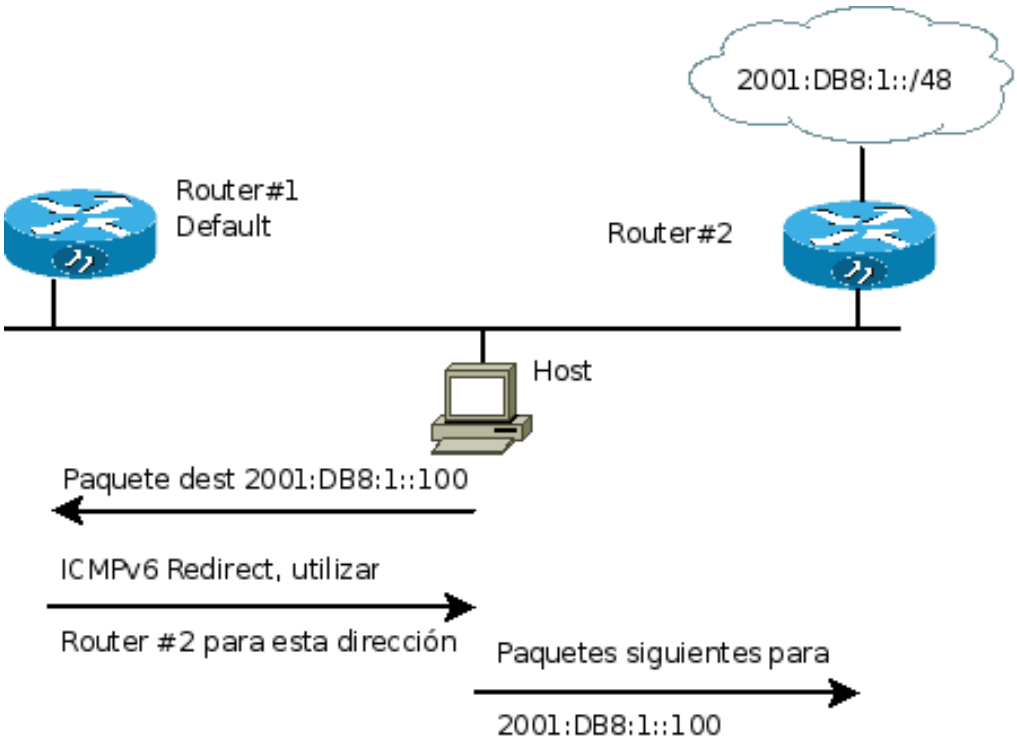

Ilustración 1: Secuencia de envío ICMPv6 Redirect

En el ejemplo de la ilustración anterior, el host desea enviarle un paquete al destino **2001:DB8:1::100**, dado que no tiene ninguna ruta hacia el destino selecciona la ruta por defecto que es hacía el router 1. El router 1 al recibir el mensaje detecta que el host y el router 2 se encuentran en el mismo enlace. Entonces envía el mensaje de redirect al host informando que el router 2 es una mejor opción. Cuando el host recibe el redirect, actualiza su **Destination Cache**, y luego los paquetes siguientes para **2001:DB8:1::100** se envían directamente a través del router 2.

Formato de mensaje de redirect

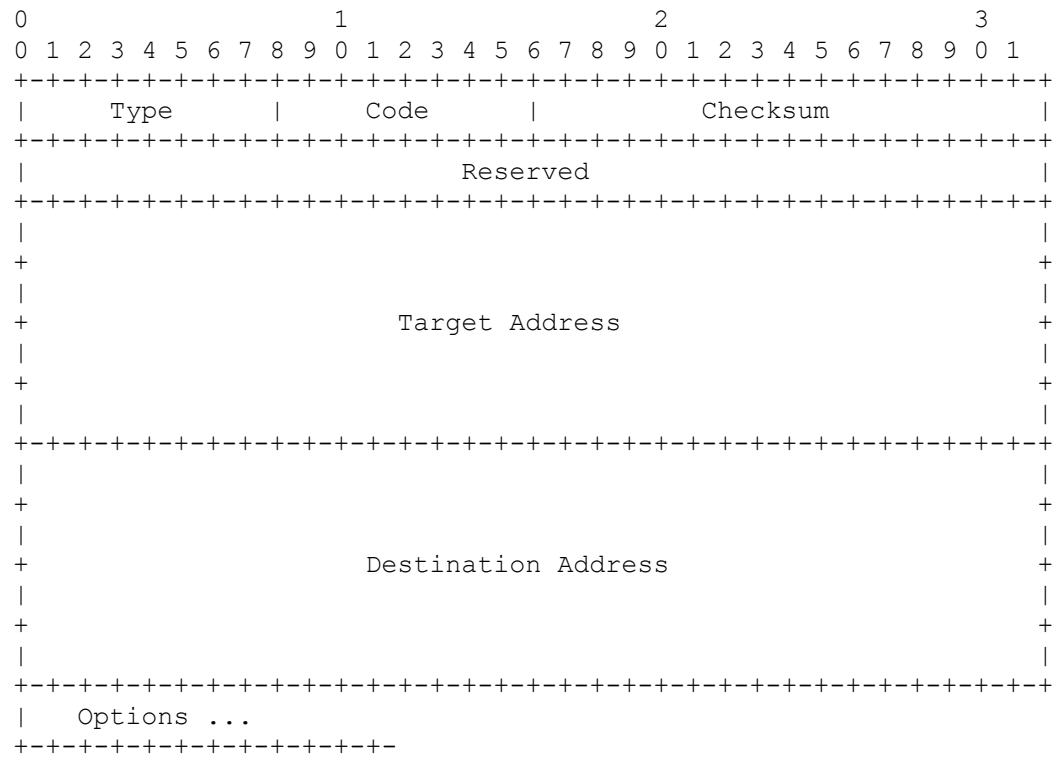

Ilustración 2: Formato de mensaje redirect

Campos IP:

- **Source Address:** Debe ser la dirección link-local asignada a la interfaz desde donde el mensaje es enviado
- **Destination Address:** El origen del paquete que generó el redirect
- **Hop Limit:** 255, los mensajes de redirect no se forwardean

Campos ICMP:

- **Type:** 137
- **Code:** 0
- **Checksum:** El checksum de ICMPv6
- **Reserved:** No se utiliza, debe ser inicializado con 0 por quien envía, e ignorado por el receptor
- **Target Address:** La dirección IPv6 del mejor primer paso para alcanzar el destino del mensaje

•**Destination Address:** La dirección IPv6 del destino que es redirigido hacia la Target Address

Opciones posibles:

•**Target link layer address:** La dirección de capa de enlace para la Target Address. Debería ser incluida si se conoce.

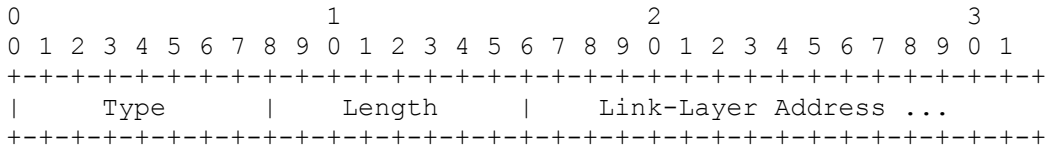

Ilustración 3: Formato del mansaje Taget Link Layer Address

•**Redirected Header:** El paquete IP que genero el envío del mensaje de Redirect, en caso de ser mayor que el mínimo MTU especificado se debe enviar el máximo posible.

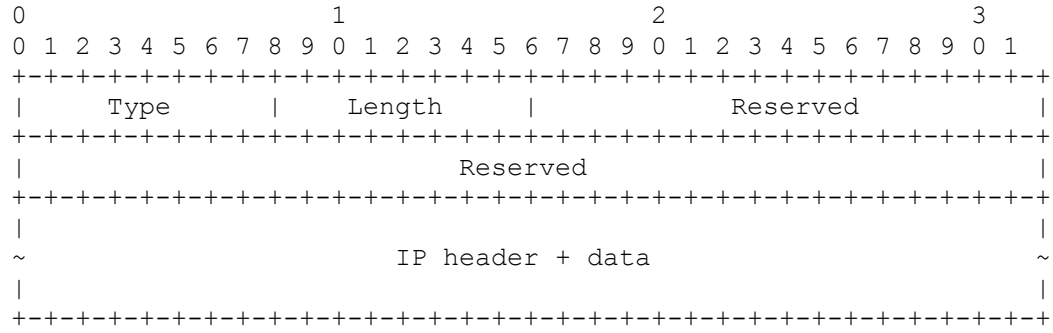

Ilustración 4: Formato del mensaje Redirected Header

2.2.IPv6 NAT

2.2.1.IPv4 NAT

NAT (Network Address Translation), como se explica en el **RFC 2663** [RFC2663] para **IPv4**, es un mecanismo de los routers, mediante el cual traducen direcciones **IPv4**. Cuando un paquete llega, es reescrito por el dispositivo **NATv4**, para poder realizar el forwarding a un host que no es la IP destino. El dispositivo **NATv4** mantiene la información de la traducción, para que luego cuando el nodo envíe la respuesta el dispositivo **NATv4** pueda traducir nuevamente la dirección pero en el sentido contrario.

NAPT (Network Address Translation Port), explicado en el **RFC 3022**[RFC3022], extiende la traducción un paso más, traduciendo los identificadores de capa de transporte (los puertos de TCP y UDP). Esto permite que los identificadores de transporte de varios hosts con direcciones privadas, ser

multiplexados a identificadores de una única dirección externa. Esto implica que NAPT permite a varios hosts compartir una única dirección.

2.2.2.Borrador NAT66

Los **Internet Drafts** o borradores son una serie de documentos, en los que aún se esta trabajando, publicados por la **Internet Engineering Task Force (IETF)** [IETF]. Estos documentos tienen una duración máxima de seis meses, luego del cúal deben ser actualizados a una nueva versión o serán eliminados, o pueden llegar a transformarse en un RFC.

Introducción

NAT66 (IPv6-to-IPv6 Network Address Translation) [NAT66] es un borrador presentado en la **IETF** el 27 de octubre de 2008 en su primer versión, y el 3 de noviembre de 2008 su segunda versión, que será discutido en un **BOF** (Birds of a Feather), en el 74 encuentro de la **IETF**, a desarrollarse en San Francisco, Estados Unidos, entre le 22 y el 27 de marzo de 2009.

El borrador describe una solución, sin estado, agnóstico de capa de transporte, que implementa traducción de direcciones **IPv6 a IPv6**, que provee los beneficios de independencia de direcciones asociado con **IPv4 a IPv4 NAT (NAT44)**, y a la vez minimizando, pero no eliminando completamente los problemas de **NAT44**.

Algunos de los beneficios de **NAT44**, como el de la necesidad de compartir una dirección global routeable para conservar el espacio de direcciones, no aplica para IPv6, como está definido actualmente. Pero provee otros como "Seguridad Simple", autonomía de direcciones, y ocultamiento de topología también provistos por **NAT44**.

Si bien estos beneficios son discutibles, no es la idea de este documento, ni del proyecto entrar en discusiones sobre si **NAT** provee o no estos beneficios. **NAT** es una herramienta que puede ser utilizada, es mejor tenerla y optar por no utilizarla que no disponer de ella.

Motivación

Como parte del borrador se aclara varias veces que no es intención de los autores alentar a la implementación o uso del **NAT66**, sino que intentar minimizar los impactos negativos en caso que algunos implementadores decidan desarrollar un mecanismo de **IPv6 NAT**, y que algunos administradores de red decidan utilizarlo.

Los autores recomiendan antes de implementar o realizar el deploy de **NAT66**, leer el **RFC 4864**, Local Network Protection for IPv6 [RFC4864].

Cuando se comenzó a utilizar **NAT44**, no existían estándares describiendo como se debía comportar los dispositivos NATs. Esto determinó que exista una gran diversidad de funcionalidades e implementaciones de NAT, que complicaron el diseño e implementación de protocolos de transporte y aplicación. A partir de esto, se realizó un trabajo retroactivo para clasificar las distintas funcionalidades de **NAT44**, para luego realizar recomendaciones para mejorar las implementaciones de **NAT44**, y para diseñar mecanismos para poder atravesar los dispositivos **NAT44**. Sin embargo aún se está lidiando con la complejidad de soportar diversos dispositivos **NAT44**.

Parte de la motivación de **NAT66** es evitar los altos costos de reenumerado. Si un cliente de un ISP decide cambiar a otro ISP, entonces deberá cambiar todas las direcciones IP de su red. Usando **NAT66** esto implicaría que sólo debería modificar el prefijo de acceso a Internet, y que toda la red interna no debería ser reconfigurada.

Si bien **NAT66** comparado con **NAT44** mejora en muchos aspectos, no elimina todos los problemas de arquitectura descriptos en Architectural Implications of NAT [RFC2993].

NAT66 modifica los encabezados IP en transito, por lo tanto no es compatible con mecanismos de seguridad que involucren encriptación end-to-end del cabezal IP.

También puede interferir en el uso de protocolos de capa de aplicación que transmiten direcciones IP en el paquete especifico de aplicación. Si bien esto es un problema de diseño en las aplicaciones de la capa de aplicación, es importante considerarlo pues es algo presente desde el FTP y recurrente en varias aplicaciones.

Descripción general

NAT66 podría ser implementado en un router IPv6 para mapear un prefijo de direcciones IPv6 a otro prefijo IPv6 a medida que cada paquete IPv6 transita a través del router.

En su forma más simple, un dispositivo **NAT66** se encuentra conectado a dos interfaces de red, una que es la perteneciente a la red interna, y la otra interfaz perteneciente a la red externa con conectividad a la red global de Internet.

Todos los host de la red interna utilizan direcciones de un único prefijo routeable localmente, y estás direcciones se traducirán hacia o desde direcciones de prefijos routeables globalmente a medida que los paquetes IP transitan el dispositivo **NAT66**.

La siguiente ilustración muestra un dispositivo **NAT66** conectado a dos redes. En este ejemplo, la dirección de red interna utiliza una **IPv6** Unique Local Addresses (**ULAs**), definidas en el **RFC 4193** [RFC4193], equivalentes a la direcciones privadas de **IPv4**, para representar la red interna, y la externa utiliza direcciones IPv6 routeables globalmente. Cabe destacar, que el **RFC 3849** [RFC3849] define que el prefijo **2001:DB8::/32** queda reservado para la utilización en documentación, por lo tanto es el utilizado para el ejemplo.

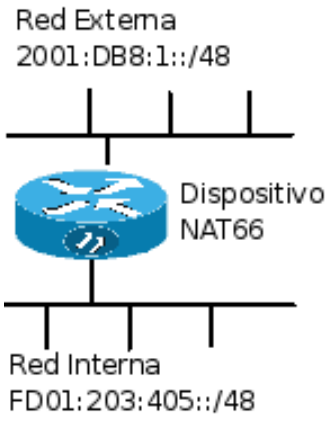

Ilustración 5: Ejemplo deploy NAT66

Cuando un dispositivo **NAT66** realiza un forwarding en dirección saliente, desde la red interna hacia la externa, **NAT66** sobreescribe la dirección **IPv6** de origen en el cabezal **IPv6**, con la dirección correspondiente para el prefijo externo. Cuando los paquetes son forwardeados en dirección entrante, desde la red externa hacia la red interna, es la dirección de destino **IPv6** la sobreescrita utilizando la dirección correspondiente al prefijo interno.

Utilizando los prefijos definidos en la ilustración anterior, cuando un paquete pasa por el dispositivo **NAT66** en dirección saliente, el prefijo de la dirección de origen (**FD01:203:405::/48**), se sobreescribe con el prefijo de la dirección externa (**2001:DB8:1::/48**). En un paquete entrante, el prefijo de destino (**2001:DB8:1::/48**) se sobreescribe con el prefijo de la red interna (**FD01:203:405::/48**).

Mecanismos de mapeo NAT66

Este borrador presenta dos soluciones, una puramente algorítmica (**Two-Way Algorithmic Address Mapping**), la cual es obligatoria de implementar en los dispositivos NAT66 al ser considerada "*MUST be implemented*" en el borrador; y otra que se realiza mediante configuración o manteniendo el estado de las

conexiones (**Topology Hiding Option**), la cual es opcional al ser considerada "*SHOULD be implemented*"

Ambas soluciones plantean un mapeo 1:1 (uno a uno), esto quiere decir que dada una dirección interna siempre es mapeada a la misma dirección externa.

En el mapeo **Two-Way Algorithmic** no se mantiene ningún estado en el dispositivo **NAT66**, ni por flujo, ni por nodo. Las direcciones de los paquetes entrantes y salientes son traducidos únicamente algoritmicamente utilizando sólo la información del cabezal IPv6. Debido a esta propiedad, el mapeo **Two-Way Algorithmic** puede soportar conexiones entrantes y salientes sin configuraciones previas o manteniendo el estado de los flujos.

El mapeo **Topology Hiding Option** intenta oscurecer la información de la red interna de la red externa, para que un nodo externo no pueda determinar la estructura de la red interna observando el tráfico fuera del dispositivo **NAT66**. Está característica es llamada "Topology Hiding" u ocultamiento de topología, y es uno de los beneficios que se consideran asociados a **NAT44**, intenta utilizar "Security Through Obscurity", que si bien es cuestionable que se logre, como ya se aclaro previamente, no es la intención de este documento, ni del proyecto discutir sobre estos temas. Como no es posible determinar toda la dirección interna solamente observando la dirección externa, el dispositivo **NAT66** debe mantener el estado para poder copiar la dirección interna en los paquetes entrantes. Esto implica además que no es posible establecer conexiones entrantes, a menos que se tomen previsiones como por ejemplo la configuración estática de los estados en el dispositivo **NAT66**, algo similar a lo que se debe realizar en NAT para IPv4.

Mapeo de checksum neutro

El mapeo de direcciones descripto en el borrador es de checksum neutro, esto quiere decir, que el resultado del mapeo es un pseudo cabezal **IPv6** con el mismo checksum que el original cuando el mismo es calculado mediante el estándar checksum de Internet [RFC1071]. Esto implica en que los protocolos de capa de transporte, como TCP y UDP, obtengan el mismo checksum del pseudo cabezal **IPv6** para los paquetes en la forma externa e interna, lo que determina que no sea necesario modificar los cabezales de capa de transporte.

Para poder lograr esto, el dispositivo **NAT66** calcula la suma del complemento a uno de 16 bits del prefijo interno y la suma del prefijo externo también en complemento a uno. La diferencia entre el prefijo original y el prefijo a mapear es calculada en complemento a uno de 16 bits, y la diferencia es sumada a 16 bits en otra área de la dirección IPv6, resultando un cabezal IPv6 que tendrá el mismo checksum del pseudo cabezal que el cabezal original.

Aunque el mismo mecanismo es usado en ambas implementaciones de **NAT66**, existen diferencias en que partes del cabezal IPv6 son mapeadas y donde se realiza el cambio.

Two-Way Algorithmic Address Mapping

El borrador especifica que es obligatorio implementarlo en todos los dispositivos **NAT66**. El mapeo consiste en modificar 16 bits de subred en los bits 49 a 64 de la dirección **IPv6**. La misma transformación es realizada tanto en conexiones entrantes como salientes, de está manera sólo es necesario conocer los prefijos de dirección interno y externo.

Para la configuración de red mostrada en la ilustración 5, se podría considerar el siguiente ejemplo.

Prefijo Interno: FD01:203:405::/48

Prefijo Externo: 2001:DB8:1::/48

Si un nodo con la dirección interna **FD01:203:405:1::1234** envía un paquete a través del dispositivo **NAT66**, el resultado de la dirección externa sería **2001:DB8:1:D550::1234**, como se muestra en la ilustración 6.

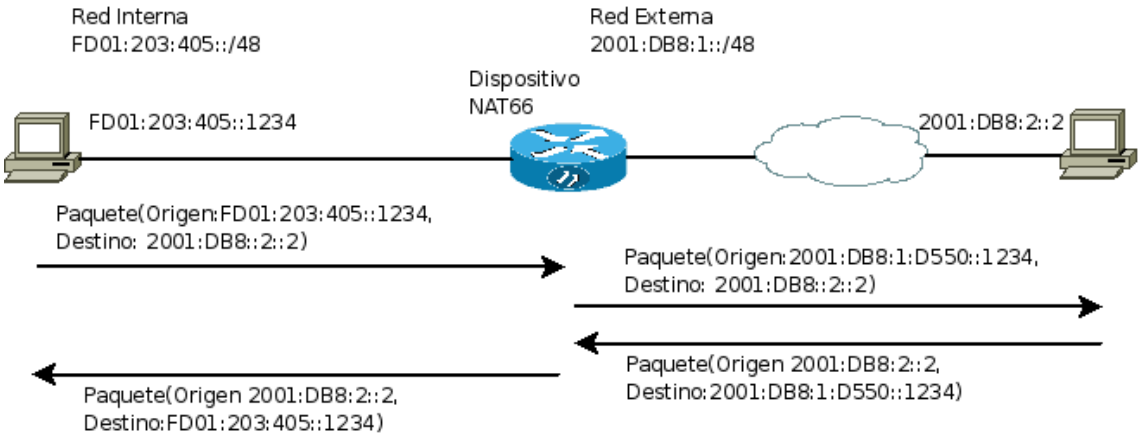

Ilustración 6: Two Way Algorithmic Protocol

Este mapeo implica que el identificador de la interfaz (**IID**) [RFC4291], no sufre modificaciones. Por lo tanto no interferirá con protocolos futuros que podrían llegar a utilizar IID únicos para identificar nodos.

La utilización de este mapeo se encuentra restringido a casos donde tanto el prefijo externo e interno son de largo 48 bits o menor.

Topology Hiding Option

El borrador especifica que los dispositivos **NAT66** deberían implementarlo. El mapeo es similar que el mapeo **Two-Way Algorithmic**, excepto que los bits de subred son transformados a 0 en la dirección saliente y restaurados a su valor original en los paquetes entrantes. El ajuste de checksum es realizado en los últimos 16 bits de la dirección IPv6, por lo tanto modificando el identificador de interfaz (**IID**). Como la interfaz podría no ser única, el bit "u" [RFC4291], es puesto en 1 en el **IID**, este cambio también debe ser tenido en cuenta en el ajuste del checksum.

Para la configuración de red mostrada en la ilustración anterior, se podría considerar el siguiente ejemplo.

Prefijo Interno: FD01:203:405::/48

Prefijo Externo: 2001:DB8:1::/48

Si un nodo con la dirección interna **FD01:203:405:1::1234** envía un paquete a través del dispositivo **NAT66**, el resultado de la dirección externa sería **2001:DB8:1:0:2::E782**, como se muestra en la ilustración 7.

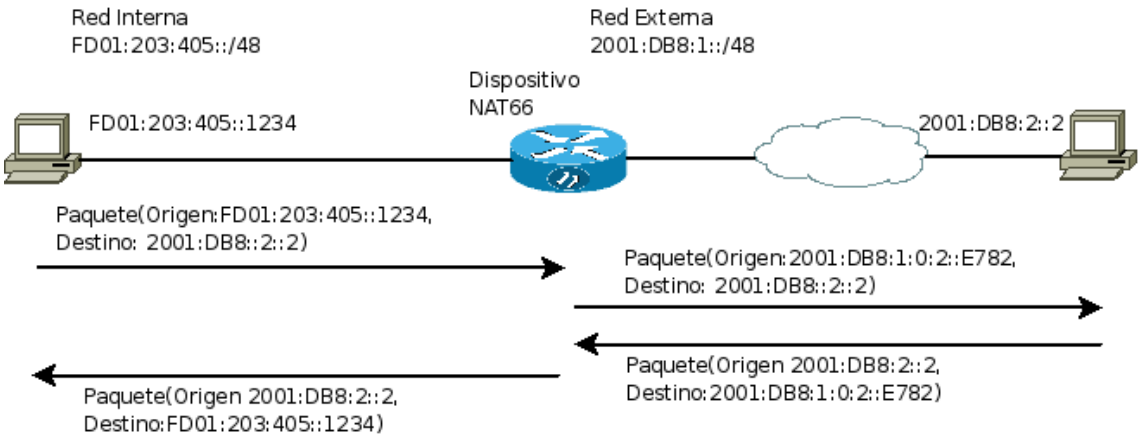

Ilustración 7: Topology Hiding Option Protocol

Cuando un paquete de respuesta es recibido, va a contener como dirección de destino **2001:DB8:1:0:2::E782**, pero debido a que los bits de subred fueron transformados a 0, no existe suficiente información con la dirección y los prefijos para realizar una transformación reversa algoritmicamente. Por lo tanto el

dispositivo **NAT66** deberá consultar su estado interno para realizar el mapeo inverso de la dirección.

El estado interno utilizado para este mapeo puede consistir en una mapeo por nodo dinámico, como la mayoría de los dispositivos **NAT44**, o puede ser mediante un mapeo estático de direcciones externas a direcciones internas. Si se utiliza el mapeo dinámico, las conexiones entrantes hacia nodos que no hallan comunicado con el exterior fallarán, debido a que no se encontrará en el estado.

Prefijos para las direcciones

Si bien en el borrador no es una necesidad, se recomienda la utilización de direcciones ULAs [RFC4193] (**Unique Local Address**) para los nodos de la red interna. Aunque los dispositivos **NAT66** deben soportar la configuración manual de cualquier, mientras sea válido, prefijo de direcciones unicast, como se describe en el **RFC 4291**.

NAT por otro nombre

Una de las consideraciones planteadas por los autores del borrador, fue el de cambiar el nombre, renombrar la propuesta con nombres como MAT, NAC, etc. De esta manera como principal ventaja tenían la de diferenciar **NAT66** de las implementaciones de **IPv4 NAT**, pero como desventaja que podría no ser tenida en cuenta por otros al no ser una solución **IPv6 NAT**.

2.3.Estado inicial del proyecto IP4JVM

Cuando se inicio el trabajo sobre el proyecto este posibilitaba el trabajo con distintos protocolos de red dentro de los cuales se encuentran una implementación parcial de los **IPv6 core protocols**, **TCP**, **UDP** e **ICMPv6**. Además en forma paralela coexistía un proyecto que también trabajaba sobre **IP4JVM** con el fin de poder crear túneles con el objetivo de soportar el protocolo de **IP Móvil** [IP4JVM].

A partir de estos protocolos es que el proyecto brinda funcionalidades tales como el envió/recepción de paquetes desde un origen o hacia un destino perteneciente al stack de direcciones. Esto quiere decir que brinda solamente funcionalidades para configurar hosts, pero ningún soporte para routers.

Se estudio en profundidad el algoritmo de selección de dirección IP de origen, y se resolvió que el mismo no era correcto, debido a que sólo se tenía en cuenta que el scope del origen sea igual al de destino, a partir de ahí seleccionaba cualquier IP que cumpla está condición.

Además se detecto que no se estaban generando los prefijos de las direcciones link-local, así como, si bien se permitía la configuración de direcciones estáticas, no se podía configurar el prefijo de las mismas, lo que no permitía seleccionar la dirección de origen correctamente.

2.3.1.Algoritmo de selección de dirección de origen existente

Las reglas utilizadas eran las definidas en el manual del desarrollador de FreeBSD [FHBD], pero no se encontraban implementadas en completitud.

1. Si la dirección de origen es especificada por el usuario, entonces se utiliza la misma.

2. Si existe una dirección asignada a la interfaz que tiene el mismo scope que el destino, entonces esa debe ser utilizada.

3. Si no existe ninguna dirección que satisfaga las condiciones anteriores, entonces elegir una dirección global asignada a una de las interfaces del nodo que envía.

4. Si no existe ninguna dirección que satisfaga las condiciones anteriores, y el destino tiene un scope de sitio, entonces elegir una dirección de scope de sitio de una de las interfaces del nodo que envía.

5. Si no existe ninguna dirección que satisfaga las condiciones anteriores, entonces se debe seleccionar la dirección asociada al destino de la tabla de ruteo.

Estas reglas son las utilizadas por defecto por la implementación de IPv6 del sistema operativo FreeBSD, pero difieren con las especificadas por el RFC 3484, Default Addresses Selection [RFC3484]. Debido a esto se definió reimplementar el algoritmo de selección de dirección IP de origen, a partir de las reglas definidas en el RFC 3484 [RFC3484].

Cabe destacar que la regla 4 no correspondería más, debido que el RFC 3879 [RFC3879] declara obsoletas las direcciones site local.

2.3.2.RFC 3484 Default Address Selection for Internet Protocol version 6 (IPv6)

El algoritmo de selección de dirección de origen está compuesto por 8 reglas que deben ser seguidas en orden, este algoritmo no debe ser utilizado para sobreescribir las direcciones seleccionadas por las aplicaciones o protocolos de capas superiores.

1. **Preferir la misma dirección**, si existe una dirección igual al destino, seleccionarla.

2. **Preferir scope apropiado**, seleccionar las direcciones que tengan el mismo scope que el destino.

3. **Evitar direcciones desaprobadas**, preferir las direcciones preferidas ante las desaprobadas.

4. **Preferir las home addresses**, cuando se utiliza IP móvil preferir las direcciones home addresses ante el resto.

5. **Preferir la interfaz de salida**, si una dirección se encuentra asignada a la interfaz por donde saldrá el paquete, mientras que la otra se encuentra asignada a otra interfaz, entonces elegir la primera.

6. **Preferir misma etiqueta**, preferir las direcciones que tengan la misma etiqueta ante las que no.

7. **Preferir direcciones públicas**, preferir las direcciones públicas sobre las temporales.

8. **Utilizar la que tenga el prefijo común más largo**, el largo de prefijo común se calcula comparando los bits de corrido iguales de izquierda a derecha que tienen en común dos direcciones IPv6, por tanto pueden ser de 0 a 128.

2.4.Resumen

En esta etapa se realizó un estudio del proyecto y su funcionamiento colocando el énfasis en el estudio del proceso llevado a cabo para seleccionar la interfaz y la dirección IP por donde se enviará el paquete, debido a la importancia de las misma en el diseño de los requerimientos a agregar. Al detectarse fallas en la implementación del proyecto se tuvieron que considerar soluciones alternativas para las mismas, por lo tanto se definió que se debía reimplementar el algoritmo de selección de dirección de origen con la solución provista por el **RFC 3484** [RFC3484].

Se realizó un estudio de lo que se debía implementar para permitir al proyecto **IP4JVM** realizar las funciones de router.

Y se estudio el **ICMPv6 Redirect** que era de una de las funcionalidades interesantes de explorar para luego implementar. Como nota a destacar, en **ICMPv4** el redirect es considerado un mensaje de error, pero mantiene sus diferencias con los otros errores de **ICMPv4**, debido a que no es un error de envío, sino una ineficiencia. Debido a esto fue catalogado como mensaje de información

para **ICMPv6**, pero también mantiene sus diferencias con el resto de los mensajes de información **ICMPv6** ya que a diferencia del resto es en reacción a un paquete común **IP**, y que además incluye una copia del paquete que la origino, como los mensajes de error **ICMPv6**.

Se realizó un estudio detallado de las distintas alternativas para la implementación de **NAT** para **IPv6**, siendo la elegida para abordar el borrador de **NAT66**, debido a que, si bien cualquier opción que se tomara sería una implementación totalmente innovadora, **NAT66** es un borrador, el cual se encuentra en estudio, que podría llegar a ser aprobado por el **IETF** [IETF], al cual se le podrían encontrar mejoras y/o errores al ser implementado, y de está manera poder contribuir a una mejor solución de **NAT** para **IPv6**.

3.Análisis de la solución

En este capítulo se presentan las soluciones a los objetivos planteados en capítulo 1, así como también los problemas presentados en el capítulo 2.

3.1.Corrección de errores

Antes de comenzar la implementación de las nuevas funcionalidades se procedió a la corrección de errores e implementación de funcionalidades básicas que no se encontraban implementados.

3.1.1.Algoritmo de selección de dirección de origen

Durante el estudio del arte se detectó que el algoritmo de selección de la dirección de origen no era correcto, debido a esto se busco alternativas para solucionarlo, encontrándose el **RFC 3484** [RFC3484]. De las reglas definidas por este RFC se implementado todas menos la regla 5, debido a que el concepto de etiquetas no es manejado en el proyecto.

3.1.2.Prefijos de direcciones

Se agregaron los prefijos de las direcciones link-local generadas automáticamente en la lista de prefijos de **Neighbor Discovery**.

Además se permitió la configuración de prefijos para las direcciones estáticas leídas del archivo de configuración **ip4jvm.config**.

3.1.3.JNI

Al ejecutar el proyecto con varias interfaces físicas distintas conectas al stack se detectaron problemas de concurrencia en el código implementado en **C**, que realiza la comunicación entre la biblioteca **Pcap** [Pcap] y el proyecto. Luego de detectado el error fue rápidamente subsanado.

La modificación se efectuó en el archivo ip4jvm_javafwrk_NetManagerRead.c, la versión anterior fue comiteada en la revisión 317 en el SVN, mientras que la actual se comiteó en la revisión 673.

Durante el testing, cuando se realizaban transferencias de archivos muy grandes, principalmente en el testing de router, luego de enviar cierta cantidad de

paquetes la aplicación dejaba de enviar y comenzaba a mostrar un error al no poder abrir la comunicación con la biblioteca Pcap. Esto se debía que al enviar un paquete se abría un manejador para enviarlo pero este luego no se cerraba.

```
handle = pcap_open_live(dev, 65536, 1, 1000, pcap_errbuf);

if (handle == NULL) {

        fprintf(stderr, "Couldn't open device %s: %s\n", dev, pcap_errbuf);

        return;

}

c = pcap_sendpacket(handle, packet, packet_size) ;

if (c != 0) {

        fprintf(stderr,"\nError sending the packet: \n",
pcap_geterr(handle));

        return 0;

}

pcap_close(handle);

free(packet);
```

Ilustración 8: Modificación Manager Write

En la ilustración 8 se muestra un extracto del código del archivo ip4jvm_javafwrk_NetManagerWrite.c que realiza la comunicación con la biblioteca Pcap para el envío de paquetes, en negrita se muestra la inserción que fue necesaria realizar para cerrar el manejador, el cual se abria en la primer línea del extracto de código, de esta manera se subsano el error.

La modificación se efectuó en el archivo ip4jvm_javafwrk_NetManagerWrite.c, la versión anterior fue comiteada en la revisión 157 en el SVN, mientras que la actual se comiteó en la revisión 726.

3.2.Router

Luego de solucionados los errores, se comenzó el diseño e implementación de las funcionalidades de router.

3.2.1.Forwarding

El modelo conceptual del nodo, que se encontraba implementado, es el descripto en el punto 5 del RFC 4861 [RFC4861]. Está compuesto por:

- **Neighbor Cache**: que mantiene un cache de vecinos con los cuales existió tráfico recientemente.
- **Destination Cache**: que mantiene un cache de destinos con los cuales existió tráfico recientemente, además es el que se actualiza al recibir un redirect de un router.
- **Prefix List**: que mantiene la lista de prefijos recibidos de los Router Advertisement.
- **Default Router List**: que mantiene la lista de routers y la tablas de ruteo.

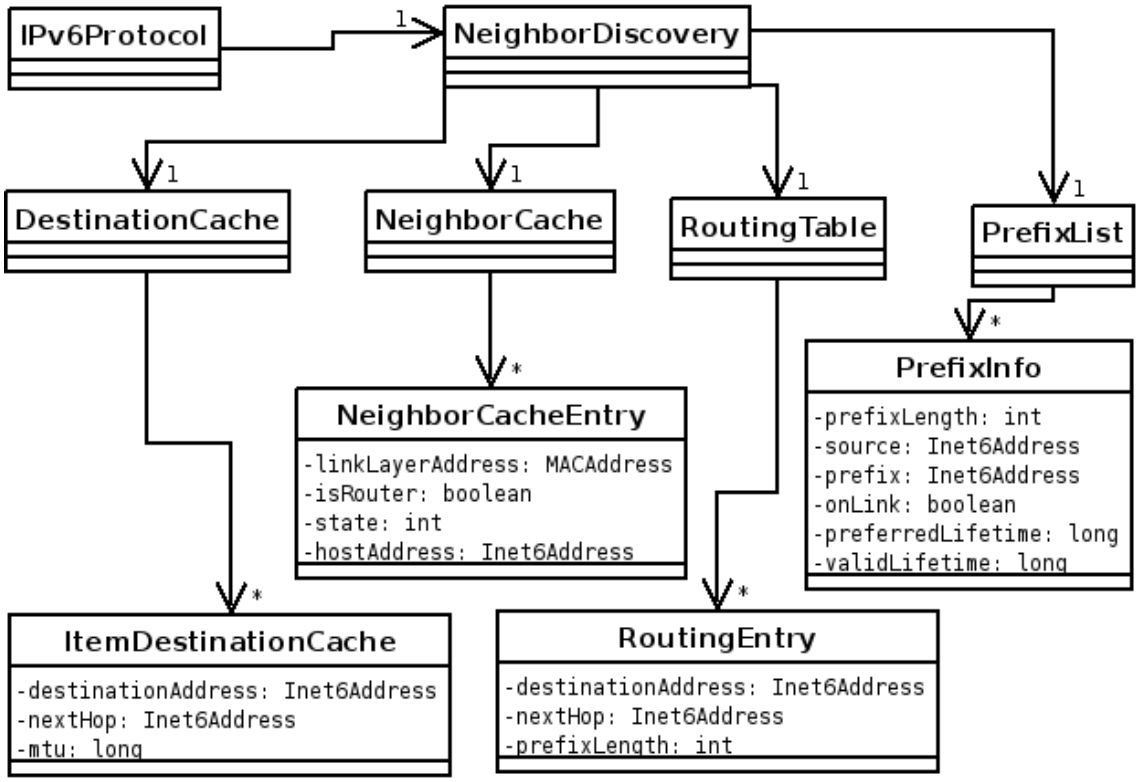

Ilustración 9: Vista parcial de diagrama de clases IPv6Protocol

La ilustración 9 muestra una vista parcial de diagrama de clases para la implementación de el modelo conceptual del protocolo **IPv6** que se encontraba implementado.

Como su nombre lo indica **IPv6Protocol** representa la clase encargada del manejo de paquetes **IPv6** en capa de red, **NeighborDiscovery** es la encargada de

el **Neighbor Discovery** y así sucesivamente el resto de las clases con el modelo conceptual explicado anteriormente.

La ilustración 10 muestra el algoritmo implementado para enviar un paquete. Este algoritmo se encuentra implementado en la clase **IPv6Protocol**, con la colaboración de **NeighborDiscovery** para realizar las consultas al **Destination Cache** y al **Neighbor Cache**, y a la lista de prefijos y de routers.

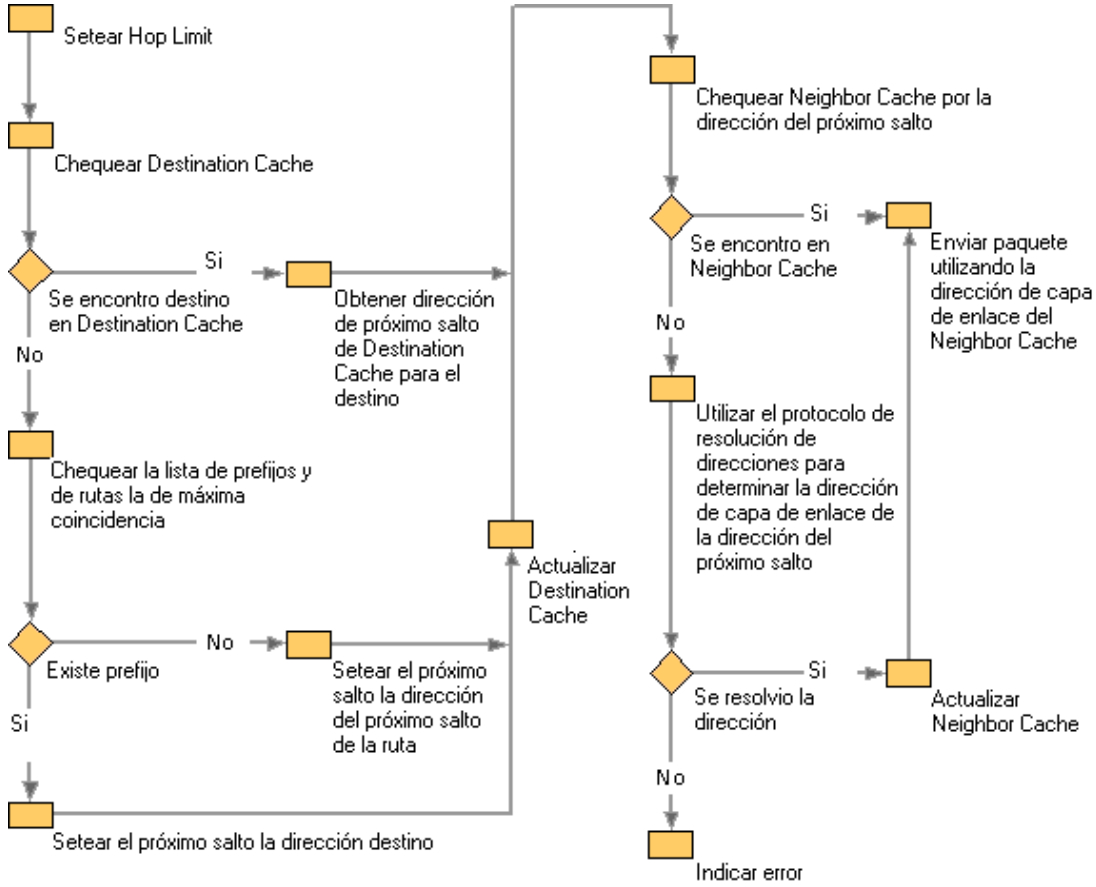

Ilustración 10: Envío de un paquete

Al recibir **IPv6Protocol**, un paquete de la capa superior, o de un túnel para ser enviado, lo primero que realiza es setear el Hop Limit. Luego de esto chequea que el destino se encuentre en el cache, si se encuentra se obtiene la dirección del próximo salto y se realiza la resolución de direcciones del próximo salto, en caso contrario se chequea la lista de prefijos y rutas para lograr llegar a destino seleccionando y se setea la dirección del próximo paso; y se actualiza el **Destination Cache**, para que de está manera los próximos paquetes a enviar no deban realizar la selección de prefijos y rutas.

A partir de ahí comienza el algoritmo de resolución de direcciones, el mismo consiste en hallar la dirección de la interfaz en capa de enlace del próximo salto, primero se chequea que no exista una entrada en el **Neighbor Cache**, si no existe utiliza el protocolo de resolución de direcciones de capa de enlace para hallarlo, si se resuelve, se actualiza el **Neighbor Cache** y se envía, si no se pudo resolver entonces se indica que existió un error.

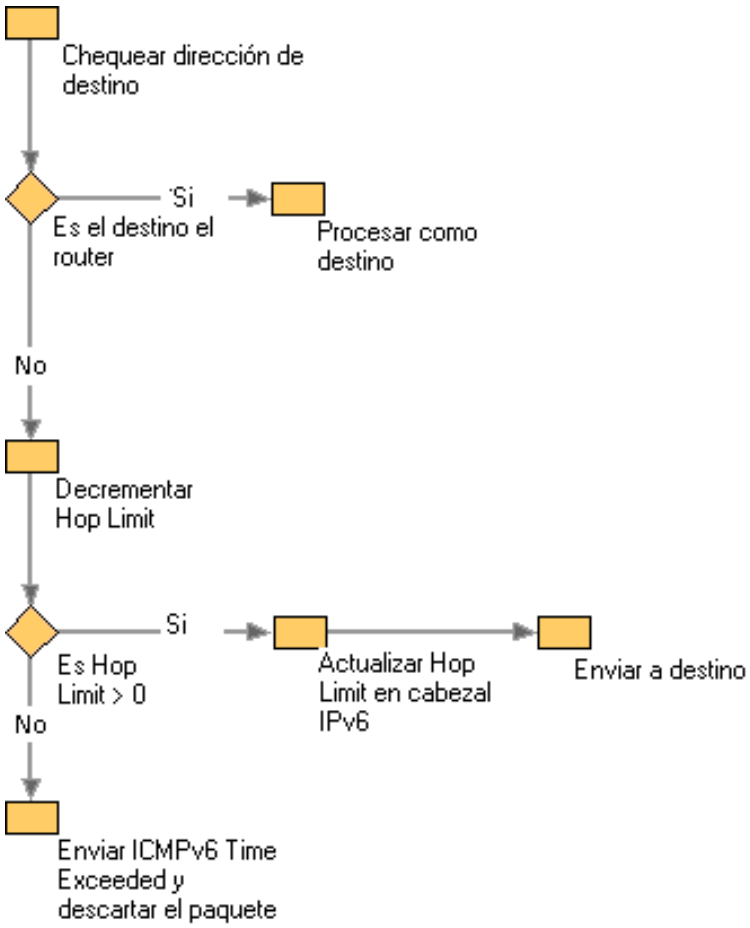

Ilustración 11: Forwarding de un paquete

Para implementar el forwarding se extendió el algoritmo de recepción de paquetes desde una capa inferior (o desde un túnel) en IPv6Protocol. Ahora al recibir un paquete, si el proyecto se encuentra configurado para realizar tareas de router, chequea que si pertenece a algunas de sus direcciones de cualquier interfaz (cuando está configurado como host sólo chequea contra la interfaz donde recibe el paquete), si el destino es el propio router entonces se procesa como de costumbre. En cambio, si no es así, se decrementa en uno el Hop Limit del paquete y luego se realizan los pasos anteriormente descriptos para enviar un paquete, con la

diferencia que en caso que no se pueda resolver la dirección de capa de enlace del próximo salto se debe enviar un mensaje de error al origen ***ICMPv6 Destination Unreachable***, para indicarle que el destino es inalcanzable. Si el Hop Limit no fuera mayor a 0 entonces se envía un mensaje de error ***ICMPv6 Time Exceeded*** al origen del paquete y se descarta el mismo.

3.2.2.ICMPv6 Redirect

Como se explicó anteriormente son utilizados por los routers para notificar a los hosts que existe una mejor ruta para un destino cuando un paquete es enviado.

Lo primero que se analizó fue cuando correspondía de parte de un router el envío de un ***ICMPv6 Redirect***, para esto el router tiene que poder determinar que el nodo origen y el nodo objetivo del primer paso se encuentran conectados en el mismo enlace, esto es posible de determinar y se cumple cuando el paquete se recibe desde una interfaz y luego de determinado el próximo salto se determina que el paquete debe salir por la misma interfaz Además el router debe determinar si el host origen es vecino del router, esto se determina chequeando la entrada del host si se encuentra en el enlace.

Por lo que se debió seguir realizando modificaciones en la clase IPv6Protocol, en la recepción y envió a destino de paquetes para poder brindar soporte para el ***ICMPv6 Redirect***.

3.2.3.Multicast

Debido a que se implementaron funciones de router hubo que implementar funciones de multicast propias de un router. Se agregó la dirección de multicast para todos los routers FF02::2, en caso que las funciones de router se encuentren activadas. Así como también se implementó la respuesta a los pings a la dirección FF02::2, que se encuentra definida en el RFC 2373, IP Version 6 Addressing Architecture [RFC2373].

3.3.NAT66

Como se explicó en el capitulo anterior, luego del estudio del arte de las distintas alternativas para implementar una solución **IPv6 a IPv6 NAT**, optamos por la del borrador de **NAT66** [NAT66].

Se decidió que se implementarían ambas soluciones, la obligatoria Two Way Algorithmic y la opcional Topology Hiding Option.

Para esto se extendió la clase **IPv6Protocol** con la clase que implementa NAT66, a la que llamamos **IPv6NatProtocol**. De esta manera, para utilizar NAT66, se debe sustituir en el stack de protocolos instalados **IPv6Protocol** por **IPv6NatProtocol**, habiendo seteado previamente los prefijos internos y externos parar realizar el NAT66. Al recibir un paquete está clase, en el método ***read***, determina si el paquete tiene como destino una dirección perteneciente al sistema, en caso de ser así es enviado al método ***read*** de **IPv6Protocol**, para ser procesado con normalidad como paquete a destino. En caso contrario se realiza el chequeo para ver si corresponde hacer NAT66. Existe cuatro casos posibles los cuales deben ser considerados en orden.

1. Si las direcciones destino y origen corresponden al del prefijo interno, esto implica que son paquetes dentro de la red interna, entonces lo que corresponde hacer es el forwarding del paquete.

2. Si sólo la dirección origen corresponde al del prefijo interno, entonces implica que se debe realizar un NAT66 saliente, o sea cambiar la dirección de origen del paquete. Está dirección se determina mediante la función ***getIPNatAddress*** que recibe como una dirección, el prefijo a utilizar y si el paquete es entrante o saliente, en este caso corresponde a la dirección origen, a el prefijo externo, y a que es saliente.

3. Si la dirección destino corresponde al prefijo externo, entonces se debe realizar un NAT66 entrante, en este caso lo que se debe modificar es el destino del paquete, que se determina también mediante la función ***getIPNatAddress***, pero recibe como parámetros la dirección destino, el prefijo interno, y que es entrante.

4. En caso que no corresponda a ninguno de los anteriores esto implica que el paquete debe ser forwardeado sin modificar las direcciones si corresponde.

Debido a que el mapeo se realiza distinto en las dos implementaciones NAT66, y para permitir otros mapeos nuevos posibles, la clase **IPv6NatProtocol** y la función ***getIPNatAddress*** *es* abstracta. De esta manera la implementación del método ***getIPNatAddress*** para la solución para el protocolo Two-Way Algorithmic se encuentra en la clase **IPv6NatTwoWayAlgorithmicProtocol**, que extiende **IPv6NatProtocol**, y para el protocolo Topology Hiding Option se encuentra en la clase **IPv6NatTopologyHidingProtocol** que también extiende **IPv6NatProtocol**.

El método ***getIPNatAddress*** en **IPv6NatTwoWayAlgorithmicProtocol** no tiene en cuenta si el paquete es entrante o saliente debido que a partir de la dirección y el prefijo recibido las cuentas a realizar, para que el **Checksum** sea neutro, son las mismas.

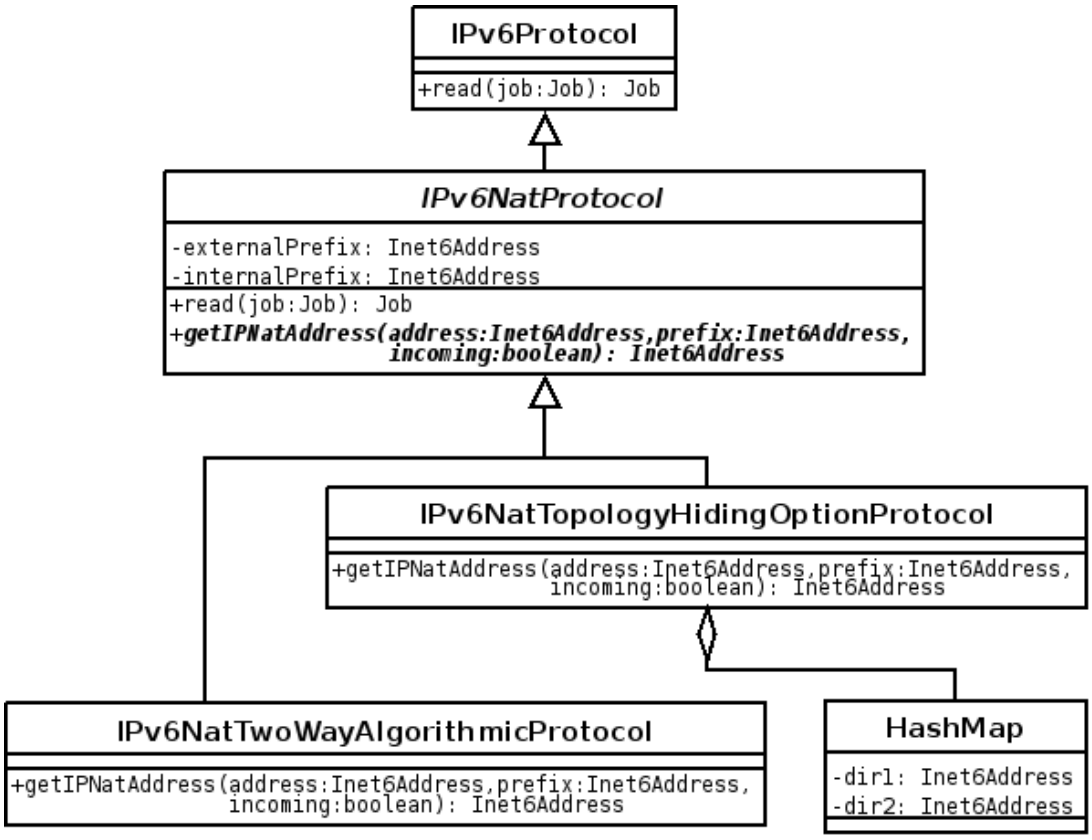

Ilustración 12: Vista parcial de diagrama de clases IPv6Protocol e IPv6NatProtocol

Por su parte en **IPv6NatTopologyHidingProtocol**, si el paquete es saliente debe realizar las cuentas para determinar la dirección de origen, pero cuando el paquete es entrante se debe revisar la tabla de mapeos para chequear si existe una entrada en la misma, si existe se retorna esa dirección, en caso contrario el paquete es desechado. Existe dos formas de cargar la tabla de mapeos:

- Mediante el tráfico, cada vez que un paquete sale se actualiza la tabla donde se coloca la dirección origen de la red interna, y el resultado del algoritmo, que es el origen para la red externa, y destino al recibir un paquete.

- Mediante archivo de configuración, existe un archivo de configuración **NATMapping.config**, que permite realizar un mapeo de forma estática entre las direcciones externas y las internas.

El formato del archivo es el siguiente:

```
#NAT Mapping File for Topology Hiding Option

#The lines starting with '#' are for comments

#IP1 <---> IP2

#Must be Checksum-Neutral Mapping



FD01:203:405:1::1234      2001:DB8:1:0:0:2::E782
```

Ilustración 13: Formato archivo de configuración NATMapping.config

Donde el # indica que la línea es un comentario, un par de direcciones mapeadas deben estar expresadas en una misma linea con uno o más espacios en blanco separando ambas direcciones, además las mismas deben cumplir la condición de ser neutro al checksum.

3.4.Resumen

En una primera instancia se solucionaron los errores que se habían detectado durante el estudio del estado del arte, como la resolución de la dirección de origen, y los prefijos de direcciones. Además al momento de implementar y testear se detectaron errores en el código C encargado de realizar la comunicación entre la librería Pcap y el proyecto. Este error se daba al conectar varias interfaces físicas al stack.

Luego se amplió la clase **IPv6Protocol** para brindar soporte para habilitar el soporte de router en el proyecto, implementandose además el **ICMPv6 Redirect**.

A partir de la implementación de las funciones de router permitió realizar de forma más fácil la implementación del borrador de **NAT66** [NAT66].

4.Cronograma

En este capítulo se presentan la distribución de las horas de trabajos en las distintas tareas.

El siguiente diagrama de Gantt presenta las tareas llevadas a cabo para lograr extender IP4JVM para brindar **soporte de routers** y realizar las tareas de **IPv6 a IPv6 NAT** acompañadas con los tiempos insumidos para cada una de ellas.

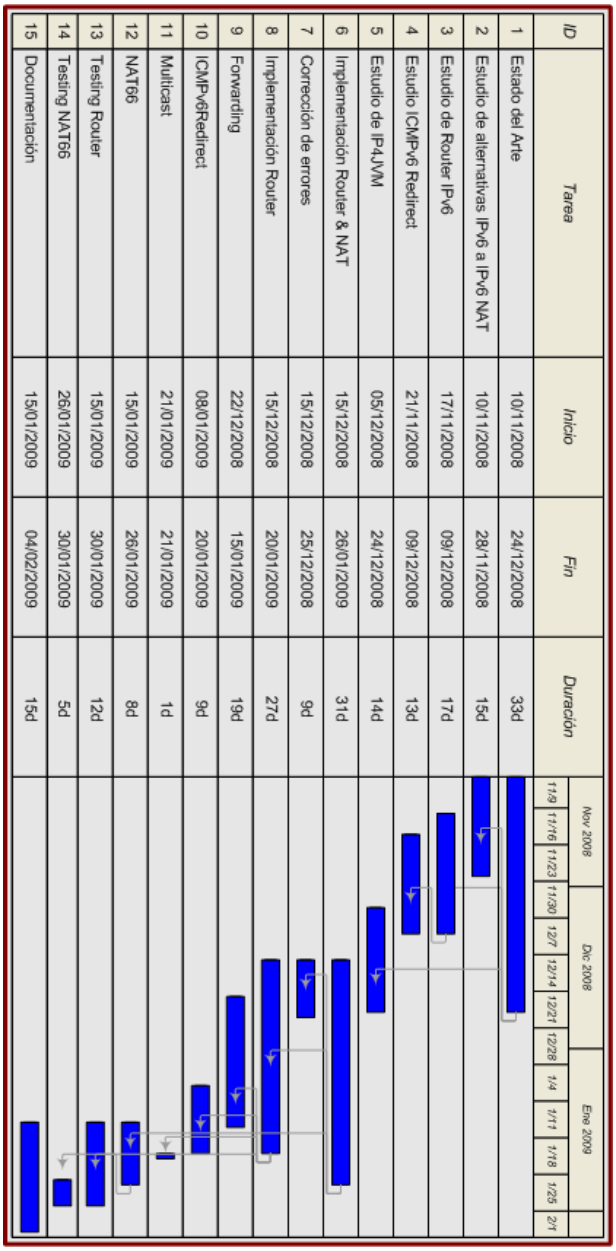

Ilustración 14: Gantt Tiempo Insumido

El primer aspecto que debe mencionarse es que la implementación se dividió en tres grandes tareas distintas, la primera el estudio del estado del arte, la segunda la implementación de Router y NAT66, y por último la documentación. Si bien las tareas se realizaron de forma concurrente son los tres grandes grupos que se encuentran representados en el diagrama.

A continuación se presenta la tabla de tiempos estimados y tiempos insumidos para la realización de cada tarea.

Tarea	Tiempo Estimado	Tiempo Insumido
Estudio del estado del arte	4 semanas	**5 semanas**
Estudio de alternativas IPv6 a IPv6 NAT	2 semanas	2 semanas
Estudio de router IPv6	2 semanas	2 semanas
Estudio ICMPv6Redirect	2 semanas	2 semanas
Estudio de IP4JVM	2 semanas	**2,5 semanas**
Implementación Router & NAT	5 semanas	**5,5 semanas**
Corrección de errores	1,5 semanas	1,5 semanas
Implementación Router	4 semanas	**4,5 semanas**
Forwarding	2,5 semanas	2,5 semanas
ICMPv6Redirect	1,5 semana	1,5 semanas
NAT66	2 semanas	**1 semana**
Testing Router	1 semana	**2 semanas**
Testing NAT66	1 semana	1 semana
Documentación	2 semanas	**2,5 semanas**
Total	11 semanas	**13 semanas**

Ilustración 15: Tabla de tiempos estimados e insumidos

Como se puede apreciar en la tabla de tiempos estimados e insumidos, los tres grandes grupos de tareas tuvieron una desviación de lo inicialmente planificado, si bien la diferencia en todos los casos como máximo es de una semana. Al desglosar la tarea de Estudio del Arte en las subtareas, se aprecia que la única variación fue en el estudio de **IP4JVM** el cual insumió media semana más, esto junto a alguna desviación más en el inicio de las subtareas determinó la semana de diferencia.

En la tarea de implementación, la desviación fue de media semana, y desglosando en subtareas, se puede apreciar que las mayores desviaciones se encontraron en la implementación de **NAT66** y el testing de router. En uno de los casos, **NAT66**, el tiempo insumido fue menor, una semana menos, debido a que la implementación fue más sencilla de lo esperado. Esto puede deberse a que en el momento de realizar las implementaciones de router se realizaron algunas consideraciones para facilitar la posterior implementación de **NAT66**. Por el contrario el testing del router insumió el doble del tiempo estimado, esto se debió principalmente en que la creación del ambiente de testing para router fue más complejo de lo que se pensó inicialmente, por más información ver anexo **Anexo VI – Configuración de un Ambiente de Testing en IP4JVM**.

Por último la desviación en la generación de la documentación fue de media semana.

5.Resultados y conclusiones

La etapa inicia con el planteo de incorporar al proyecto IP4JVM la capacidad de proveer **funcionalidades de router**, así como también realizar un estudio e implementación de soluciones **IPv6 a IPv6** de **NAT.**

Se implementó las funcionalidades mínimas de un router IPv6, la única funcionalidad extra que se implementó fue el **ICMPv6Redirect**.

Por el lado de **IPv6 a IPv6 NAT**, primero se consideraron las distintas alternativas posibles, resolviendo luego la implementación del borrador NAT66 presentado ante el IETF.

En los siguientes puntos se presentan los logros, limitaciones y mejoras de los componentes diseñado e implementado, así como también pros y contras del borrador **NAT66** implementado.

5.1.Logros

Se lograron cumplir todos los objetivos planteados al inicio de esta etapa del proyecto, luego de realizar el estudio del estado del arte, considerar las distintas soluciones, y testear las mismas.

5.2.Limitaciones

Dentro de las principales limitaciones se encuentra la no posibilidad de envío de **Router Advertisement**, lo cual implica que el proyecto haciendo las veces de router no puede avisar a los host de la red que el mismo es un router.

Otra de las limitaciones del router es que no se brinda la posibilidad de habilitar distintas interfaces para realizar el **forwarding** de paquetes. Al habilitar las funcionalidades de router se habilitan en todas las interfaces.

5.3.Trabajo futuro y posibles mejoras

Dentro de las posbles mejoras se encuentra exteneder la solución actual para brindar soporte a las limitaciones descriptas en el punto anterior, envio de **Router Advertisement**, y configuración de interfaces para realizar **forwarding**.

Otra posible extensión es la de implementar NAT4466, es decir NAT como se encuentra implementado para IPv4 pero para IPv6, con traducción de direcciones y puertos.

5.3.1.NAT66

Durante el transcurso de esta etapa del proyecto, se recibió por medio de la lista de NAT [LSTNAT] el 8 de febrero, información sobre modificaciones en NAT66 en el **Topology Hiding Option Protocol**. Debido a que el nuevo borrador no fue publicado antes de la culminación del proyecto, a que aún no se encuentra claro como será el nuevo algoritmo y a que la implementación de este ya se encontraba finalizada, se decidió no implementar la nueva solución. Como parte de la solución se previo este tipo de situaciones, por lo que la dificultad de agregar el protocolo nuevo o las modificaciones pertinentes dependerá solamente del grado de dificultad del nuevo algoritmo.

Al implementarse un borrador se detectaron problemas que aún no habían sido considerados por los autores, ni por otras personas.

Tal fue el caso de los problemas que surgen cuando un nodo se encuentra dentro del mismo prefijo externo que el dispositivo **NAT66**. En estas situaciones al querer comunicarse con un nodo que se encuentra en la red interna, el nodo que se encuentra en la red externa no envía directamente el paquete, sino que debido a que se encuentra en el mismo prefijo que el nodo destino del paquete, envía un **Neighbor Discover**, para resolver la dirección de capa de enlace de quien considera que es su vecino. Debido a que el borrador no especifica que el dispositivo **NAT66** deba responder a estos **Neighbor Discovery**, ni deba realizar una traducción de los mismos, ni deba realizar las tareas de un proxy de **Neighbor Discovery**, la resolución de direcciones no se completa, lo que implica que el paquete no se logra enviar.

Para solucionar este tipo de caso existen dos soluciones. La primera implica modificar las rutas del nodo que envía, poniendo como next hop del mismo el dispositivo **NAT66**, de esta manera la resolución de direcciones se realiza contra el dispositivo **NAT66** y el paquete se envía a el, y luego es traducido para ser enviado al nodo destino de la red interna. La segunda opción, y que no se encuentra implementada, y puede ser considerada para trabajo futuro es que el dispositivo NAT66 implemente el **proxy** de **Neighbor Advertisements** descripto en el **RFC 4861** [RFC4861], sección 7.2.8 con algunas variantes de traducción de direcciones. De esta manera el dispositivo indica que se encuentra en condiciones de aceptar paquetes para ese destino.

5.4.Conclusiones

Luego de finalizado el estudio del estado del arte e implementados los cambios consideramos por la extensiones realizadas al proyecto en esta etapa, que la arquitectura del proyecto es extensible, ya que no implica grandes desafíos la introducción de nuevos cambios como los aplicados en esta etapa.

5.4.1.NAT66 vs NAT44

Como principal ventaja encontrada en **NAT66** frente a **NAT44** se encuentra el no mapeo de puertos, no afectando de esta manera el checksum de los cabezales TCP y UDP. Esto permite la aplicación de otros protocolos en la capa de transporte, siempre y cuando el checksum se calcule mediante el estándar **Internet Checksum**, **RFC 1071** [RFC1071], pero manteniendo los problemas en los protocolos que utilizan su dirección IP como referencia y son enviados y modifcadas las direcciones IP en ruta. Ejemplos de estos protocolos son **FTP** y **SIP**.

Otra de las mejoras es el mapeo uno a uno, descripto para el **Two Way Algorithmic Protocol**, lo que permite que se acepten conexiones entrantes, reduciendo, por ejemplo los problemas generados en las aplicaciones **peer to peer** cuando se está detrás de un dispositivo **NAT44**, muy comunes en la transferencia de archivos y de difusión multimedia en tiempo real.

6.Índice de ilustraciones