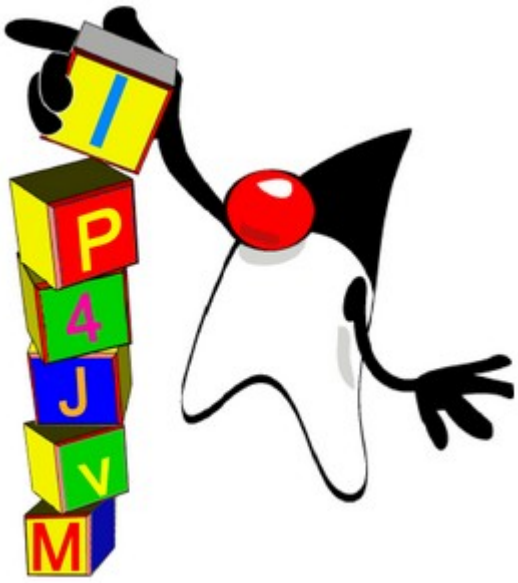

Proyecto de Grado 2008

Anexo VI

IP4JVM - Testing en IP4JVM

Autores:

Leandro Scasso

Marcos Techera

Tutor:

Ariel Sabiguero

Tribunal:

Andrés Aguirre

Eduardo Grampín

Carlos Martínez

Tabla de contenidos

1.Introducción

En este documento se explican todos los test realizados sobre el proyecto IP4JVM al comienzo del proyecto, antes de comenzar a implementar y luego en las distintas fases para cada una de ellas.

1.1.Objetivos

Comprobar el correcto funcionamiento del proyecto durante las distintas etapas del desarrollo. Además se desea obtener metricas para poder comparar la implementación IPv6 con otras implementaciones existentes.

1.2.Organización de este documento

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

Capítulo 1, Introducción: se definen los términos generales del problema a tratar en el presente documento, los objetivos que se desean alcanzar y se describe la organización del documento.

Capitulo 2, Test básicos: se presentan los test realizados para comprobar la correcta implementación por parte de IP4JVM de la autoconfiguración "Stateless Address Autoconfiguration".

Capítulo 3, Test de conformidad básicos de TAHI: se presentan los tests realizados para comprobar la correcta instalación del TAHI en el ambiente de testing y conocer el estado general del proyecto al comenzar el desarrollo.

Capítulo 4, DHCPv6: se presentan los test realizados para probar la funcionalidad del cliente DHCPv6 implementado.

Capítulo 5, Test Router & NAT66: se presentan los test realizados para testear el correcto funcionamiento de la implementación de las funcionalidades básicas de Router, de redirect y de NAT66.

2.Test Básicos

En este punto se presentan los test realizados para comprobar la correcta implementación por parte de IP4JVM de la autoconfiguración "Stateless Address Autoconfiguration".

Para poder llevar a cabo estos test se utilizo la herramienta radvd[RVD] para el envío de Router Advertisement el cual siempre se encontraba corriendo en un host distinto al host en el cual corría el proyecto IP4JVM, la tarjeta de red por la cual este demonio envía los mensajes es la vmnet1.

La configuración del proyecto IP4JVM para todos los test que se corrieron bajo los objetivos expresados más arriba fue la misma y estaba dada por el archivo de configuración ip4jvm.config que contenía las siguientes líneas:

jth0 1 eth0 00:0C:29:E4:EA:88

jth1 1 eth0 00:0C:29:E4:EA:89

De lo anterior se desprende que la topología sobre la cual se corrieron los test es la siguiente:

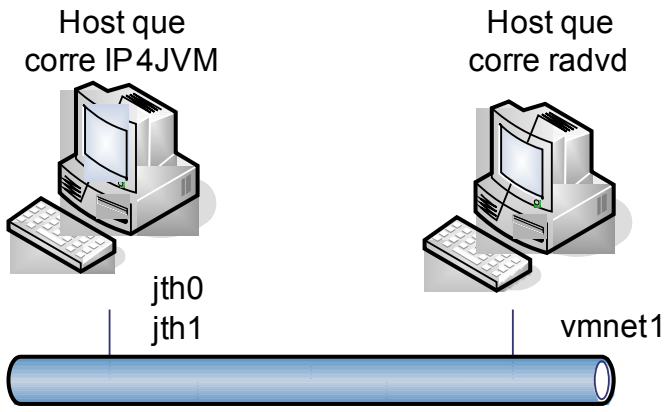

Ilustración 1: Topología inicial

2.1.Test 1

El propósito de este test es el de comprobar el correcto funcionamiento del método de autoconfiguración "Stateless address autoconfiguration". Para esto se configuro el radvd con un prefijo en donde la bandera autonomous tenía el valor en 1.

La configuración del radvd fue la siguiente:

```
interface vmnet1 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        prefix 2003::/64 {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
                AdvPreferredLifetime 50;
                AdvValidLifetime 100;
        };
};
```

Ilustración 2: Configuración radvd

Luego de correr el test el stack agrego los prefijos y además configuro las interfaces a partir de los mismos y la topología que se obtuvo fue la siguiente:

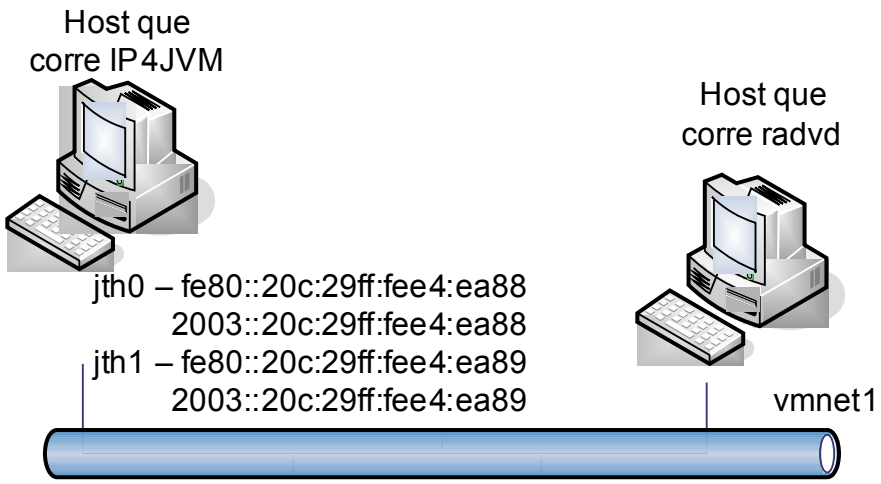

Ilustración 3: Topología final test 1

2.2. Test 2

El propósito de este test es el de comprobar los tiempos de validez de la interfaces configuradas mediante el método de configuración "Stateless address autoconfiguration". Para esto al igual que en el test 1 se configuro el radvd con un prefijo en donde la bandera autonomous se encuentra en 1, pero luego de que el radvd estuviera corriendo durante un tiempo se paro al mismo. Otra variante que se probo fue la de cortar de forma abrupta la comunicación entre el host que corre el radvd y el host que corre IP4JVM.

La configuración del radvd fue la siguiente:

```
interface vmnet1 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        prefix 2003::/64 {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
                AdvPreferredLifetime 50;
                AdvValidLifetime 100;
        };
};
```

Ilustración 4: Configuración radvd

El resultado obtenido fue el esperado, mientras el radvd estuvo en funcionamiento las interfaces configuradas fueron renovando sus tiempos de vida y estuvieron disponibles. Al detener el radvd este envía un Router Advertisement indicando que el tiempo de vida del router es cero. Las interfaces continúan configuradas mientras el tiempo de vida valido que se encuentra configurado para las mismas no venza. Luego de esto las interfaces dejan de estar disponibles.

El resultado obtenido al cortar de forma abrupta la comunicación entre ambos host fue la misma, la diferencia se encuentra en que en este caso el radvd

no tiene la posibilidad de enviar un Router Advertisement indicando que tiempo de vida del router es cero.

La topología obtenida durante y al final del test es la siguiente:

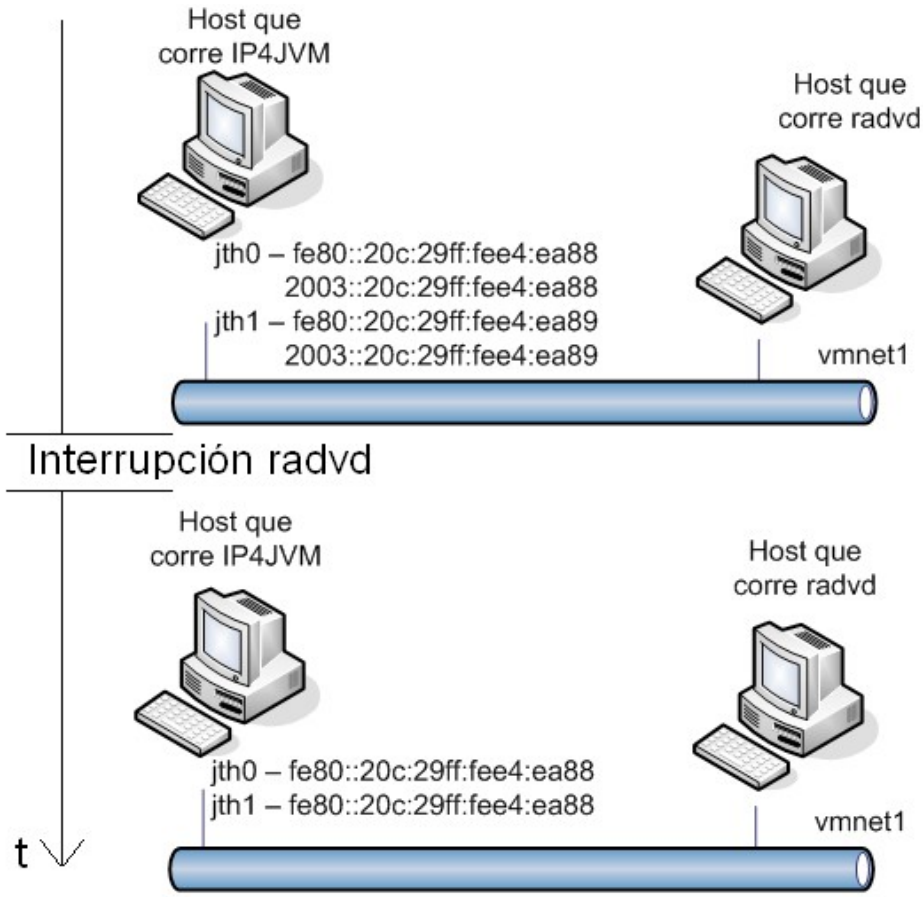

Ilustración 5: Topología final test 2

2.3. Test 3

El propósito de este test es el de comprobar que únicamente se encuentra implementado el método de autoconfiguración "Stateless address autoconfiguration". Para esto se configuro el radvd para que enviara Router Advertisements en donde el prefijo tenía la bandera autonomous en 0. Y las banderas M y o se encontraban con el valor 1.

La configuración del radvd fue la siguiente:

```
interface vmnet1 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        AdvManagedFlag on;
        AdvOtherConfigFlag on;
        prefix 2003::/64 {
                AdvOnLink on;
                AdvAutonomous off;
                AdvRouterAddr on;
                AdvPreferredLifetime 50;
                AdvValidLifetime 100;
        };
};
```

Ilustración 6: Configuración radvd

Como se esperaba el stack agrego los prefijos enviados en la Router Advertisement pero no configuro una nueva dirección para ninguna de las jth. Con esto nos aseguramos que cuando se pone el prefijo autonomous en 0 no se usa "Stateless address autoconfiguration". Además se realizaron capturas de tráfico de red mediante la herramienta Wireshark en donde se pudo comprobar que la aplicación IP4JVM dejo de enviar los Router Solicitations.

Además mediante la misma herramienta se pudo comprobar que no se enviaron mensajes UDP por el puerto 547, lo cual confirma que la la configuración por DHCP no está implementada.

La topología resultante de este test fue la siguiente:

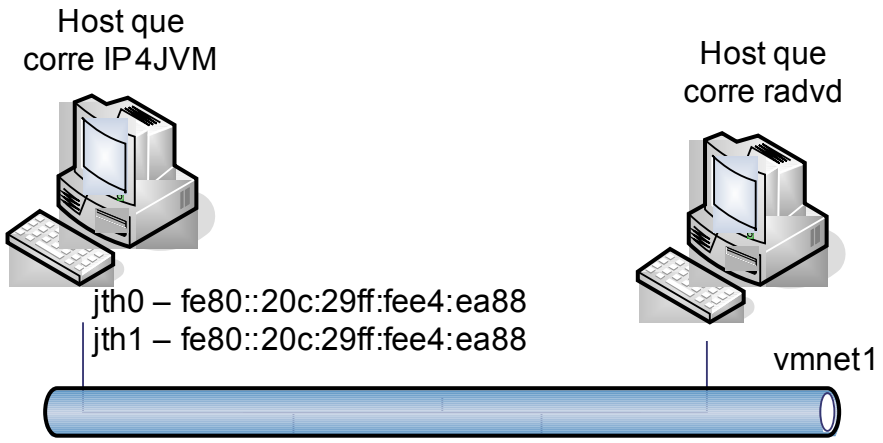

Ilustración 7: Topología final de Test 3

2.4. Test 4

En el marco del evento "Hacia IPv6"[HIPV6] que realizó el UY6TF[UY6TF], el IPv6 Task Force Uruguayo, los días 7 y 8 de julio de 2008 se tuvo la oportunidad de probar el proyecto contra un router IPv6 real y no contra un router simulado mediante software que es lo que se realizó en los tres test anteriores.

Gracias a esto se pudo apreciar que se había omitido un posible test a realizar con el simulador, ya que el proyecto no podía configurarse de forma correcta contra el router. El problema que se presentaba era que la header option de ICMPv6 MTU no se encontraba bien implementado ya que faltaba el campo reserved dentro de la implementación.

Luego de solucionado el problema se reprodujo con el radvd la configuración que tenía el router del evento y se volvió a probar. En esta ocasión el proyecto si se puedo configurar correctamente.

La configuración del radvd que emulaba la configuración del router real fue la siguiente:

```
interface vmnet1 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        AdvLinkMTU 1000;
        prefix 2003::/64 {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
                AdvPreferredLifetime 50;
                AdvValidLifetime 100;
        };
};
```

Ilustración 8: Configuración radvd

La topología que se obtuvo es la misma a la que se llego luego de correr el test 1.

2.5.Conclusiones

Los test confirmaron que el proyecto de momento solo maneja el mecanismo Stateless address autoconfiguration y que aun no está disponible la configuración mediante DHCP.

3.Test de conformidad básicos de TAHI

El TAHI Project[TAHI] tiene como objetivo desarrollar tecnologías para poder verificar IPv6, contribuir en el mejoramiento de calidad de los stacks IPv6, brindar test de interoperabilidad entre las distintas implementaciones, reducir los esfuerzos de los implementadores para realizar el testing y clarificar las ambigüedades de los RFCs. v6eval es la herramienta que utiliza el proyecto TAHI para generar los tests. Por más información ver **Anexo V – Configuración de un Ambiente de Testing en IP4JVM**.

Los tests que presentados en esta sección son los que se encuentran disponibles con la instalación de la herramienta v6eval de TAHI. Estos test se corrieron buscando dos objetivos:

1. Poder comprobar que la correcta instalación del proyecto TAHI en el ambiente de testing
2. Conocer el estado general del proyecto.

Los resultados generados por el proyecto TAHI se pueden encontrar en el DVD que se adjunta en la documentación dentro de la carpeta tests.

En todos los test:

NUT node under test

TN tester node

Por más información ver **Anexo VII – Glosario**.

Los test que se corrieron y sus funcionalidades son los que se explican en los siguientes puntos de este capítulo.

3.1.Link Local Ping Test

El objetivo de este test es el de enviar un paquete ICMP del tipo echo request desde el TN al NUT y esperar como respuesta un paquete ICMP echo reply enviado desde el NUT. Además en caso de que el NUT envíe un Neighbor Solicitation el TN responderá con un Neighbor Advertise.

Este test es superado de forma exitosa por el proyecto IP4JVM

3.2. Fragmentation Test

Al igual que en el caso anterior este test pretende realizar un ping sobre el NUT, pero esta vez el paquete se envía de forma fragmentada.

Este test no es superado por el proyecto IP4JVM.

3.3. Global Address allocation test

En este caso el TN envía dos mensajes, primero hace el envío de un Router Advertisement indicándole al NUT que debe de configurar un dirección global. Después de esto le envía un paquete ICMP echo request a la dirección que el NUT debió configurar a partir del Router Advertisement.

El NUT debe de enviar dos mensajes para poder aprobar el test, un Neighbor Solicitation, el cual será respondido con el correspondiente Neighbor Advertise por parte del TN, y un ICMP echo Reply.

Este test es superado de forma exitosa por el proyecto IP4JVM.

3.4. Time Exceeded

Este test verifica que el NUT envía un paquete valido del tipo ICMPv6 Time Exceeded en respuesta de un paquete que no puede ser re-armado.

Este test no es superado por el proyecto IP4JVM.

3.5. UnknownNext Header Type

Este test verifica que el NUT envié un paquete ICMPv6 de tipo Parameter Problem en respuesta a un paquete que incluye un valor desconocido para el campo next header.

El test no sólo es superado por enviar un paquete del tipo antes mencionado sino que además la dirección de origen de dicho paquete debe de ser la dirección global del NUT a donde fue enviado el paquete inicial y la dirección destino debe de ser la dirección desde don del TN envió el paquete inicial. Otro control realizado corresponde al lugar en donde se encuentra el error, el paquete ICMPv6 debe indicar que el error se encuentra en la posición 40.

3.6.Port Unreach

Este test verifica que el NUT envié un paquete ICMPv6 del tipo Destination Unreachable en respuesta a un paquete que es enviado por el TN con la destination address del NUT pero a un puerto que este no está utilizando.

Este test es superado por el proyecto IP4JVM

3.7.MLD

En este test se verifica el funcionamiento básico del Multicast Listen Discovery, pero el mismo aun no está implementado en el proyecto IP4JVM, con lo cual este test falla.

3.8.HBH-DST

Este test verifica que el NUT envía un mensaje ICMv6 de tipo Parameter Problem luego de recibir un ICMPv6 de tipo echo request con un Destination Option Header.

Este test tampoco es superado por el proyecto IP4JVM

3.9.MTU Option

En este caso se verifica que el NUT pueda recibir un Router Advertisement con la opción MTU. Para esto el TN envía un Router Advertisement con la mencionada opción y además envía un ICMPv6 echo request. Estos mensajes deben de ser contestados por el NUT con un ICMPv6 echo reply y un Neighbor Solicitation.

Este test no es superado por el el proyecto IP4JVM.

3.10.Conclusiones

Si bien los test cumplieron con sus dos objetivos, probar la configuración de TAHI y ver el estado actual del proyecto, los resultados obtenidos en los mismos no fueron los deseables.

Algunos de los errores no requieren de mucho esfuerzo para su corrección pero debido a que el proyecto anterior aun se encontraba en su etapa de culminación no fueron llevadas a cabo dichas correcciones si no que fueron enviadas a los integrantes del mencionado proyecto.

4.DHCPv6

Para testear las funcionalidades del cliente DHCPv6 se utilizaron dos tipos de prueba, en una primera instancia se utilizó el Dibbler] como servidor DHPCv6 de manera de poder configurar al cliente y realizar las pruebas básicas, luego se utilizo los tests del proyecto TAHI[TAHI] para clientes DHCPv6, de manera de realizar pruebas más estandarizadas.

4.1.Test Dibbler

El radvd se utilizó para indicar al proyecto IP4JVM mediante los **Router Advertisement** si correspondia hacer stateful o stateless DHCPv6. La configuración en esto tests básicos vario de acuerdo a si se debía hacer stateful o stateless DHCPv6.

```
interface vmnet1 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        AdvLinkMTU 1000;
        prefix 2003::/64 {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
                AdvPreferredLifetime 50;
                AdvValidLifetime 100;
                AdvManagedFlag on;
                AdvOtherConfigFlag on;
        };
};
```

Ilustración 9: Configuración radvd

En estos tests se modificaron las banderas **AdvManagedFlag** y **AdvOtherConfigFlag** con los valores **on** y **off** de manera de indicar que parámetros solicitar al servidor DHCPv6.

A continuación se presenta un ejemplo de configuración utilizado para el servidor de DHCPv6 Dibbler.

```
iface "vmnet1" {
        t1 1800-2000
        t2 2700-3000
        prefered-lifetime 3600
        valid-lifetime 7200
        class {
                pool 2001:DB8:/64
        }
        option dns-server 2001:DB8::FF,2001:DB8::FE
        option domain example.com
}
```

Ilustración 10: Configuración Dibbler Server

En este ejemplo se indica que el tiempo T1 es entre 1800 y 2000 segundos, T2 entre 2700 y 3000 segundos, el preferredLifetime es de 3600 segundos y el tiempo válido validLifetime 7200. Además que el pool de direcciones a utilizar es 2001::DB8/64. Indica además que los servidores de DNS se encuentran en 2001:DB8::FF y 2001:DB8::FE, y el dominio es example.com. Todos estás configuraciones solo aplican a la interfaz vmnet1.

4.2.Test de conformidad DHCPv6

Estos test corresponden al conjunto de pruebas elaboradas por el proyecto TAHI con la intención de corroborar el cumplimiento de los RFCs

- RFC3315 - Dynamic Host Configuration Protocol for IPv6(DHCPv6)
- RFC3646 - DNS Configuration options for Dynamic Host Configuration Protocol for IPv6(DHCPv6)

- RFC3736 - Stateless Dynamic Host Configuration Protocol(DHCP) Service for IPv6

De todos los test conformados para poder probar las especificaciones de los mencionados tests, sólo se corrieron los test correspondientes al RFC3315, ya que para poder realizar el resto de los test se debería de tener una implementación del protocolo de DNS, el cual excede el alcance del del proyecto.

El objetivo a cumplir con estos test es el de obtener una evaluación de la solución encontrada para el problema planteado al inicio del proyecto. Teniendo la ventaja de que el proyecto TAHI publica los resultados de los test de distintas aplicaciones también permite comprar los resultados obtenidos por la aplicación realizada en comparación con otras implementaciones disponibles, como ser el Dibbler herramienta de la cual hacemos uso en su versión de servidor DHCPv6.

Los test realizados para el cumplimiento del RFC 3315 son noventa y cinco y los mismos se explican a continuación agrupando los mismos en grupos de funcionalidades a probar. Los grupos son los mismos que utiliza TAHI.

4.2.1.Basic Message Exchange

Consta de 6 partes donde se chequea que la respuesta dada a un **Request**, **Confirm**, **Renew**, **Rebind**, **Release** y a un **Decline Message** sea correcto. La topología en todos los casos es la siguiente:

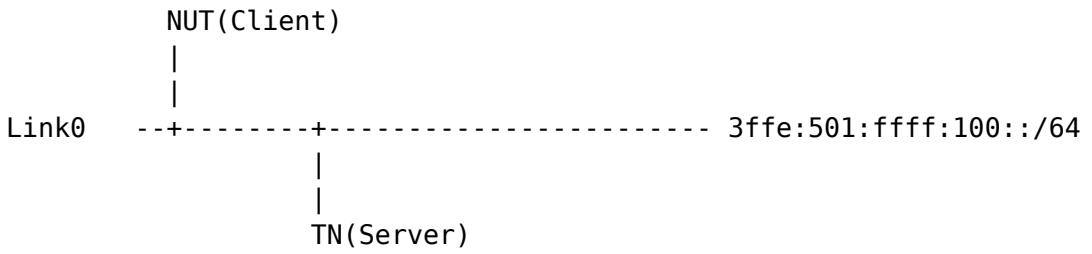

Ilustración 11: Topología Basic Message Exchange

Cada uno de los procedimientos de testing difiere dependiendo del mensaje que espera testear, a modo de ejemplo se muestra el testeo del Rebind y los mensajes que se esperan durante el transcurso del test.

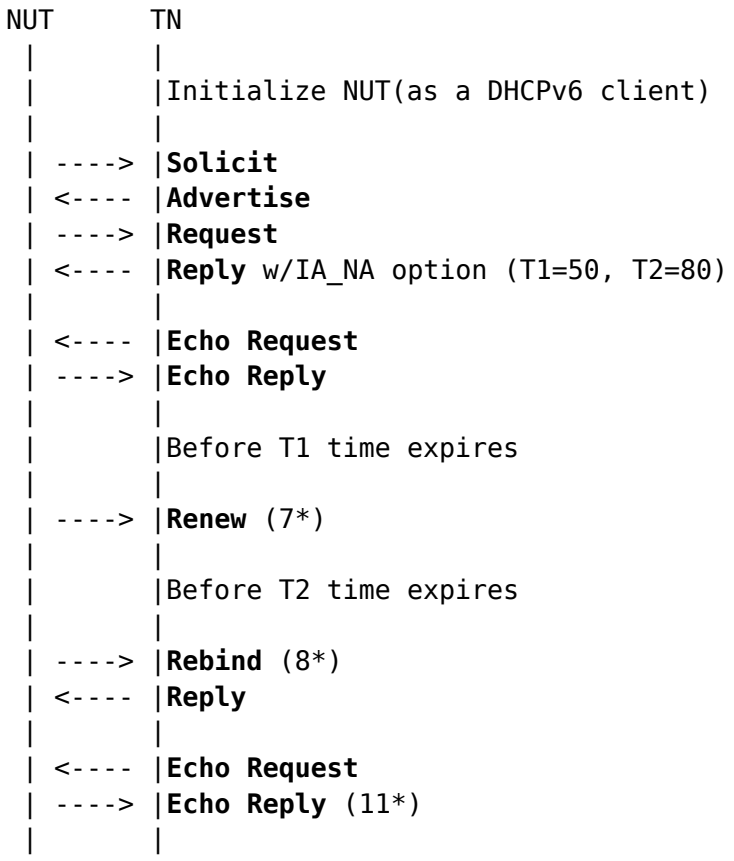

Ilustración 12: Test Rebind

Todos los tests de este grupo son pasados en forma satisfactoria.

4.2.2. Router Advertisement Processing

Este grupo consta de dos tests. Los mismos cambian la bandera **M** de los **Router Advertisement**, de manera que el host deba procesar las mismas y decidir si utilizar DHCPv6 o no. El primero de los tests es pasado de forma correcta, pero el segundo no. En las versiones anteriores de este test era pasado de forma correcta pero en esta última (DHCPv6_Self_Test_P2_1_0_17) no se pasa.

4.2.3. Stateful and Stateless Address-configuration

Este grupo consta apenas de un test, y se envía un **Router Advertisement** donde se avisa que se debe configurar mediante **Stateful Address Configuration**, pero además envía un prefijo de dirección de manera que también se configure otra dirección mediante **Stateless Address Configuration**.

Este test es pasado de forma satisfactoria por el proyecto IP4JVM.

4.2.4.Implementation of DHCP constants

En este grupo de tests, que consta de tres tests, se chequea que se utilice las constantes determinadas en el RFC 3315, como es el puerto UDP del servidor destino de los paquetes, así como también que se utilice la dirección **FF02::1:2** para solicitar todos los servidores DHCPv6 y todos los Relay Agents DHCPv6.

Este grupo de tests es pasado de forma satisfactoria.

4.2.5.Client Message Format

Consta de un test donde se chequea que el formato del **Solicit Message** sea el correcto.

Este test es pasado en formado satisfactoria.

4.2.6.Client Identifier Format

También consta de un test donde se comprueba que el formato del **Client Identifier Option** sea el correcto.

También es pasado de forma satisfactoria.

4.2.7.Client DHCP Unique Identifier Contents

Es un grupo de tests que consta de 6 tests. Los primeros dos chequean el formato y la consistencia del **DUID**, en este caso **DUID-LLT**, el DUID formado a partir de la dirección de la capa de enlace. Estos test son pasados de forma satisfactoria.

El resto de los tests del grupo chequean el formato y la consistencia del **DUID**, en sus formas **DUID-EN** y **DUID-LL**, los cuales no aplican en nuestro caso.

4.2.8.IA_NA Option Format

Consta de un test donde se chequea que el formato de la opción **Identity Association for Non-temporary Addresses Option** sea el correcto.

Este test es pasado en forma correcta.

4.2.9.IA Address Option Format

También consta de un test solo, en este test se chequea la consistencia del **IAID**, la opción **IA_NA** incluye la opción **IAID** y el mismo debe ser consistente luego de que el cliente DHCPv6 reinicie.

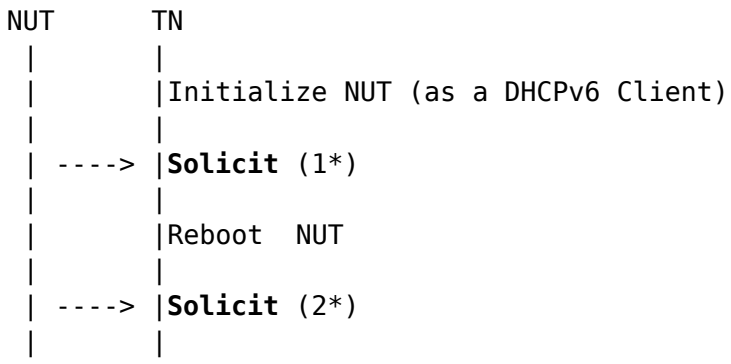

Ilustración 13: Test IA Address Option Format

Para pasar este test se espera que el **IAID** en (1*) sea el mismo que en (2*).

Este test también es pasado de forma satisfactoria.

4.2.10.IA Elapsed Time Option Format

Este grupo consta de 3 tests, los cuales chequean las opciones enviadas en la opción de **Elapsed Time**, que indica el tiempo transcurrido desde el primer mensaje del intercambio. A continuación se presenta uno de ellos:

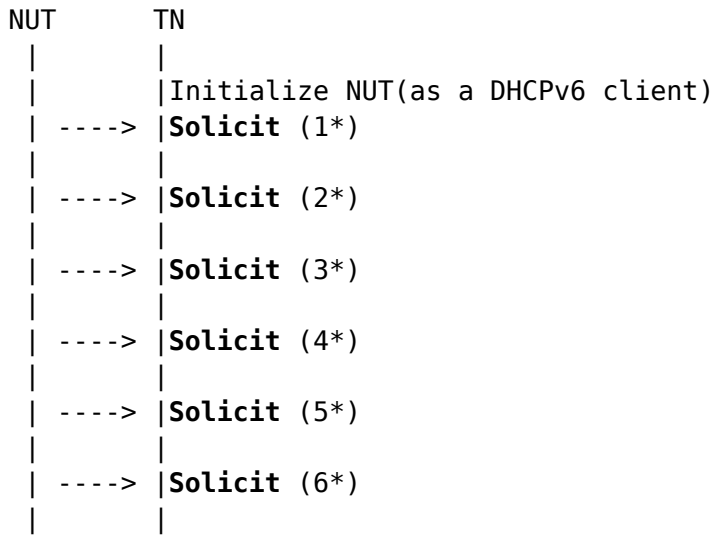

Ilustración 14: Test Elapsed Time Option Format

En este caso se chequea que el tiempo transcurrido sea correcto en todos los casos.

De estos tests el segundo falla debido a variaciones en los tiempos, si bien las variaciones son mínimas las misma existen.

4.2.11.Transmission of Solicit Message

Este grupo de tests consta de 4 tests, en estos se prueba que el **Solicit Message** sea enviado de forma correcta, se chequea que la retransmisión sea correcta, así como también los tiempos de la misma.

De estos test, en forma aleatoria, el último del grupo falla, esto se debe a pequeñas diferencias de tiempos que existen entre los mensajes.

4.2.12.Message Exchange Termination for Solicit Message

Este grupo consta de 2 tests donde se chequea la duración de la transacción entre un **Solicit Message** y un **Advertise Message**.

En el primer test el cliente DHCPv6 debe esperar un tiempo IRT antes de enviar un **Request Message** luego de haber enviado el primer **Solicit Message**, para poder recoger los **Advertise Message**, a continuación se muestra un diagrama del test:

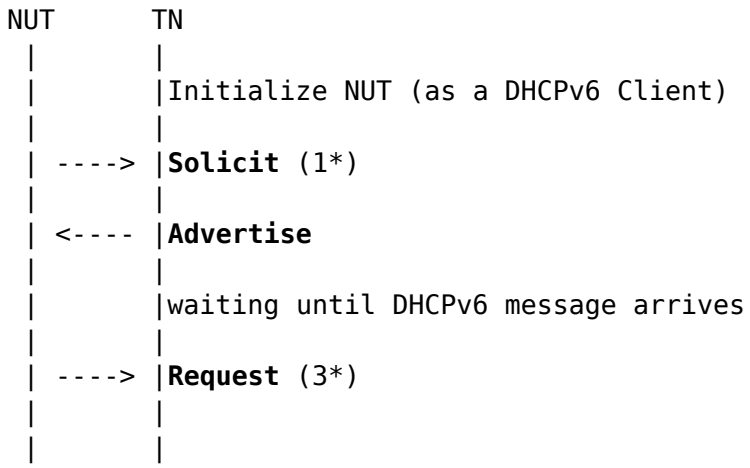

Ilustración 15: Test Message Exchange Termination for Solicit Message

En (1*) se guarda el tiempo del **Solicit Message**, en (3*) se comprueba que el intervalo entre el **Request Message** y el **Solicit Message** sea menor que **IRT + RAND*IRT** y mayor que **IRT**.

Aleatoriamente este test falla debido a que los tiempos no se encuentran dentro de los intervalos aceptados.

En el segundo test de este grupo el primer **Solicit Message** no recibe ninguna respuesta, luego de retransmitirse el mismo, recibe un **Advertise Message** que debe ser procesado inmediatamente y respondido por un **Request Message**. A continuación se muestra un diagrama del test:

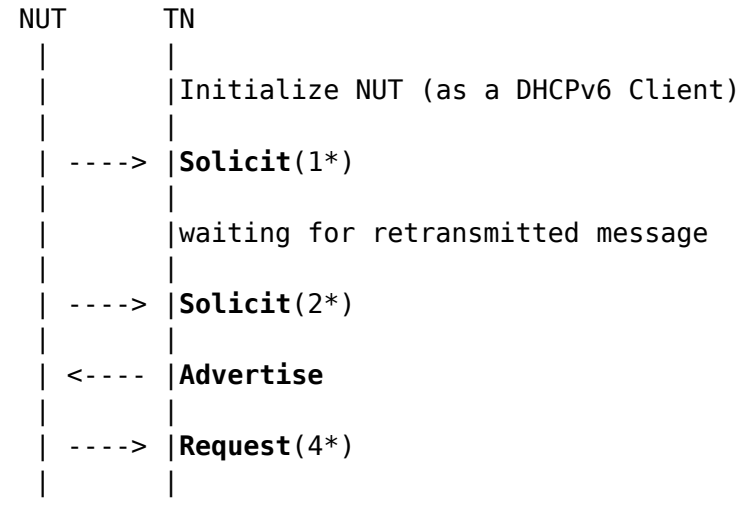

Ilustración 16: Test Message Exchange Termination for Solicit Message II

Este test es pasado en forma satisfactoria.

4.2.13.Transmission of Request Message

Este grupo de tests consta de 4 tests, en estos se prueba que el **Request Message** sea enviado de forma correcta, se chequea que la retransmisión sea correcta, así como también los tiempos de la misma.

Dos de estos tests no son pasados debido a diferencias de tiempos en los intervalos estipulados.

4.2.14.Transmission of Renew Message

Este grupo de tests consta de 4 tests, en estos se prueba que el **Renew Message** sea enviado de forma correcta, se chequea que la retransmisión sea correcta, así como también los tiempos de la misma.

Estos tests son pasados de forma satisfactoria.

4.2.15.Transmission of Rebind Message

Este grupo de tests consta de 4 tests, en estos se prueba que el **Rebind Message** sea enviado de forma correcta, se chequea que la retransmisión sea correcta, así como también los tiempos de la misma.

Los dos últimos tests de este grupo de no son pasados debido a diferencias en los intervalos de tiempo.

4.2.16.Transmission of Release Message

Este grupo de tests consta de 4 tests, en estos se prueba que el **Release Message** sea enviado de forma correcta, se chequea que la retransmisión sea correcta, así como también los tiempos de la misma.

Este grupo de tests es pasado de forma satisfactoria.

4.2.17.Reception of Decline Message

Este grupo de tests consta de 4 tests, en estos se prueba que el **Decline Message** sea enviado de forma correcta, se chequea que la retransmisión sea correcta, así como también los tiempos de la misma.

El tercero de estos tests no es pasado de forma satisfactoria debido a diferencias en los intervalos de tiempo.

4.2.18.Reception of Advertise Message

Este grupo consta de un solo test donde se chequea que el cliente descarte los **Advertise Message** con **status code** de **NoAddrsAvail**, que indica que no hay direcciones disponibles.

Este test es pasado en forma satisfactoria.

4.2.19.Client initiated Exchange – Reception of Reply Messages

Este grupo de tests consta de 5 tests donde se chequea que las opciones recibidas en la recepción de un **Reply Message** sea correctamente procesada.

Este grupo de tests es pasado en forma satisfactoria.

4.2.20.Client initiated Exchange – Reception of Reply Messages cont'd

Este grupo de tests es una continuación del grupo anterior donde se prueban distintas opciones en la recepción de **Reply Message**, consta de 10 tests, donde se prueban muchas más opciones del Reply.

De estos tests el cuarto de ellos falla, el mismo corresponde a un **Reply Message** en respuesta a un **Confirm Message** con la opción **NotOnLink** en el **Option Status Code**, el mismo no es procesado de forma correcta por el proyecto IP4JVM. Cabe destacar que en versiones anteriores del programa de test, este test era pasado de forma satisfactoria.

El resto de los tests es pasado de forma satisfactoria.

4.2.21.Reception of Invalid Advertise Message

Este grupo consta de 4 tests donde se envían opciones inválidas, o faltan opciones, en los **Advertise Message**.

Este grupo de tests es pasado de forma satisfactoria.

4.2.22.Reception of Invalid Reply Message

Al igual que el grupo anterior consta de 4 tests y se envían opciones inválidas, o faltan opciones, pero en este caso es en los **Reply Messages**.

Estos 4 tests son pasados de forma satisfactoria.

4.2.23.Client Message Validation

Este es el último grupo de tests, consta de 9 tests, donde se cheque que el cliente descarte los mensajes recibidos que no corresponden a un cliente. Por ejemplo uno de ellos se le envía un **Solicit Message** a un cliente DHCPv6 y se espera la respuesta del mismo, el cliente ante esta situación lo que debe hacer es ignorar el mismo.

Estos tests son pasados satisfactoriamente.

4.2.24.Resumen

De los 95 tests que conforman el DHCPv6 Self Test para clientes DHCPv6 del proyecto TAHI para el RFC 3315, el proyecto IP4JVM pasa un total de 80 de los mismos, falla en 10 y no aplican 5, estos 5 no aplican hasta el momento a ningún cliente de DHCPv6. La mayor parte de los tests que se fallan se deben a diferencias de tiempos, y las mismas se han ido limando desde que se comenzó a testear mediante el proyecto TAHI por ambas partes, gran parte de las modificaciones en las distintas versiones del tests de DHCP han sido por problemas de tiempo, pero aún existen diferencias.

En comparación a aplicaciones que se encuentran en producción como son el Dibbler] y el Wide-DHCP [WIDE] en sus versiones cliente, se está en mejor situación en los tests del proyecto TAHI. Dibbler en última versión del tester, la misma que el proyecto IP4JVM, pasa 71 tests y falla en 19. Mientras que Wide DHCP en la versión anterior del tester, pasa 69 y falla en 21. Al igual que para el proyecto IP4JVM las principales diferencias se encuentran en los tiempos esperados y los tiempos recibidos.

5.Test Router & NAT66

En este capítulo se presentan los tests realizados para testear el correcto funcionamiento de la implementación de las funcionalidades básicas de Router, de redirect y de NAT66. Para esto se realizaron dos tipos distintos de tests, el primero un test básico para probar las funcionalidades de routing e interoperabilidad con otros sistemas operativo mediante aplicaciones como Ping6 y SSH, y el segundo, realizando los test del proyecto TAHI, para comprobar que se estuvieran pasando los test para ICMPv6 Redirect.

5.1.Test básicos

En estos test se probaron distintas topologías de red con distintos sistemas operativos y distintas aplicaciones para comprobar la correcta implementación de las funcionalidades de routing.

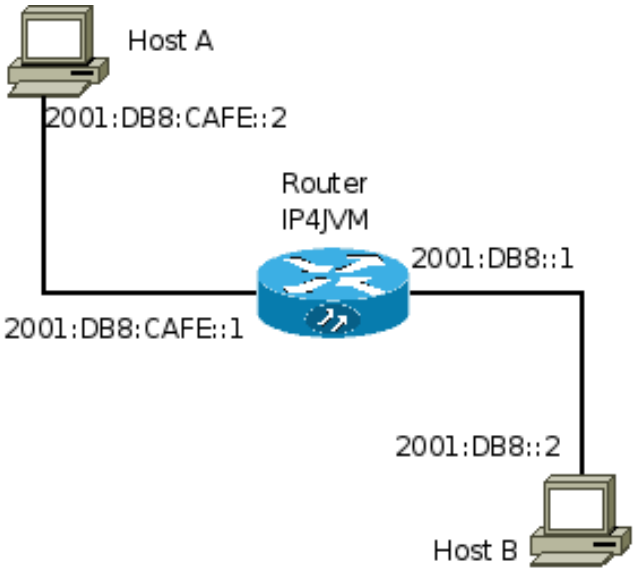

Ilustración 17: Testing Router

En la ilustración 17 , se presenta un formato general de las pruebas realizadas, donde el router IP4JVM presenta dos interfaces de red, una con dirección IPv6 2001:DB8:CAFE::1 conectada con el Host A con dirección IPv6 2001:DB8:CAFE::2, y la otra con dirección IPv6 2001::DB8::1 en un enlace distinto que el anterior conectada al Host B con dirección 2001:DB8::2.

En un principio para los Host A y B se utilizaron los sistemas operativos Debian GNU/Linux Lenny 5.0 y FreeBSD 6.3 respectivamente.

Los tests inicialmente consistieron en pinguear desde los Host A y B las distintas direcciones IPv6. Por ejemplo desde el Host A pinguear la dirección 2001:DB8::2 perteneciente al Host B, para chequear que se este realizando correctamente el forwarding de los paquetes, el **ICMPv6 Request** del pedido del ping y el **ICMPv6 Reply** de la respuesta. También realizar pings a la interfaz del Router no conectada en el enlace, por ejemplo realizar un ping desde el Host A a la dirección **2001:DB8::1**.

Otra de las pruebas realizadas fue conectarse desde el Host A al Host B a través del router mediante SSH y aplicaciones web, de esta manera se testeo que el forwarding sea correcto en distintas aplicaciones.

Asimismo se utilizaron otros sistemas operativos en los hosts, como Windows XP, Windows Vista y Ubuntu GNU/Linux con los mismos resultados.

Todos estos tests se pasaron de forma satisfactoria.

5.2. Test ICMPv6 Redirect

Para testear la funcionalidades de ICMPv6 Redirect se utilizó los test provistos por el proyecto TAHI, así como también se realizaron test básicos con la configuración de la ilustración 18.

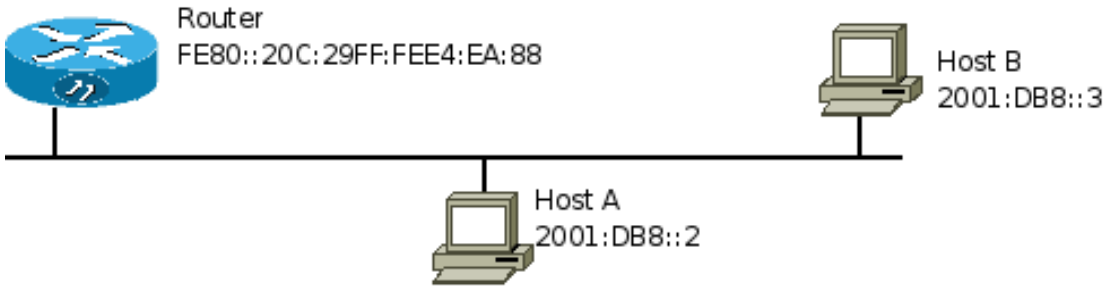

Ilustración 18: Testing ICMPv6 Redirect

5.2.1. Test básicos

El test consistió en configurar el proyecto IP4JVM como router y en el **Host A** poner como default gateway el Router y ninguna otra ruta. De esta manera al intentar comunicarse el **Host A** con el **Host B** se envíe a través del Router, al recibir el paquete el router envía un **ICMPv6 Redirect** al **Host A** para informarle que el destino se encuentra en el mismo enlace, entonces a partir de este momento la comunicación del **Host A** y el **Host B** es directa.

5.2.2.TAHI

El proyecto TAHI contiene para chequear la funcionalidad del **ICMPv6 Redirect** cuatro test.

En todos los test:

RUT indica Router Under Test, en este caso el proyecto IP4JVM

TNX tester node X, donde X representa un número

TRX tester router X, donde X representa un número

Por más información ver **Anexo VII – Glosario**.

El primero de ellos **v6LC_2_3_16_A - Redirect – Transmit**, es similar al básico realizado en el punto anterior. Con la siguiente topología:

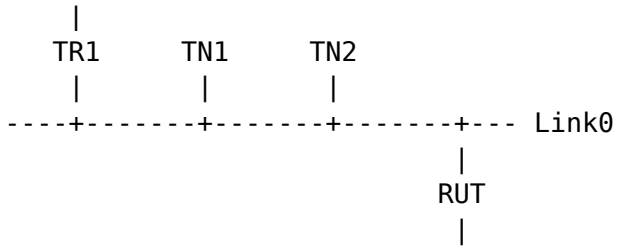

Ilustración 19: Topología Redirect - Transmit

Donde se envía un ping del **TN1** al **TN2** y se espera el redirect del **RUT**, como se muestra en el siguiente diagrama:

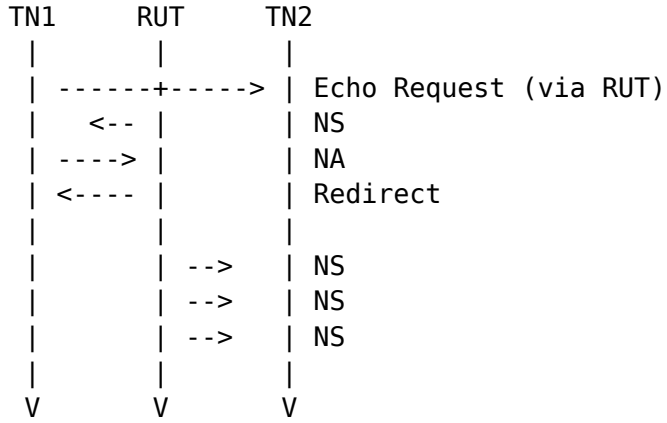

Ilustración 20: Redirect - Transmit

El segundo test **v6LC_2_3_16_B - Redirect – Transmit**, tiene una topología más compleja:

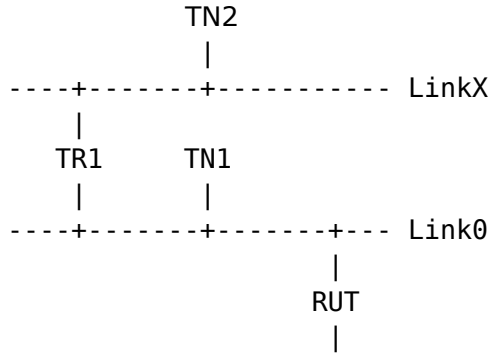

Ilustración 21: Topología Redirect - Transmit II

En este caso el **TN1** envía un ping al **TN2** que es enviado a través del **RUT**, por lo que se espera que el **RUT** envíe un **ICMPv6 Redirect** a través del **TR1** como se muestra en el siguiente diagrama:

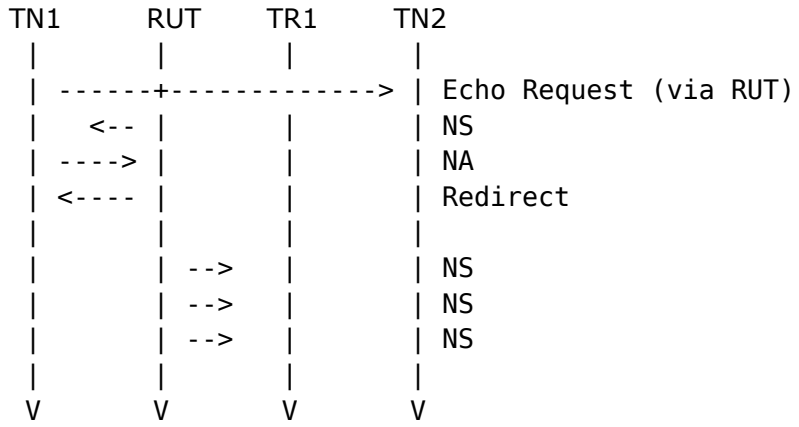

Ilustración 22: Redirect - Transmit II

El tercer caso **v6LC_2_3_16_C - Redirect – Transmit** utiliza la misma topología que el anterior, pero en este caso el **TN1** envía también un ping al **TN2** pero el origen del mismo no es una dirección del nodo **TN1**, ni de ninguna interfaz conectada al **Link0**, de esta manera lo que se espera es que el router no envíe un **ICMPv6 Redirect**, como muestra el siguiente diagrama:

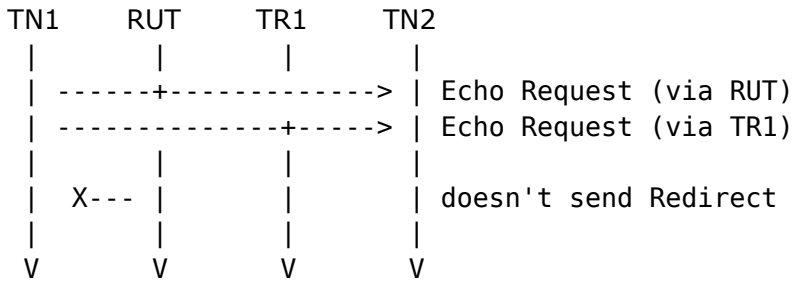

Ilustración 23: Redirect - Transmit III

En el último caso de prueba para Redirect, **v6LC_2_3_16_D - Redirect – Transmit**, también utiliza la misma topología que los dos anteriores. En este caso lo que se intenta probar es que cuando el destino es una dirección de multicast tampoco se debe enviar un mensaje de **ICMPv6 Redirect**, como se presenta en el siguiente diagrama:

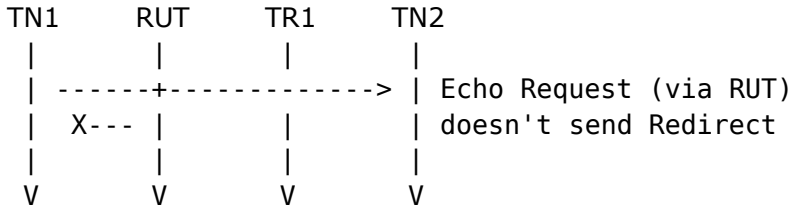

Ilustración 24: Redirect - Transmit IV

El proyecto IP4JVM pasa estos cuatro test, por más información ver datos de prueba de los test en la carpeta tests en el DVD que se adjunta a la documentación.

5.3.Test NAT66

Debido a que NAT66 es un borrador no se dispone aún casos de prueba como TAHI, por lo que los únicos test realizados para probar las funcionalidades de NAT66 fueron similares a los primeros casos de prueba de Router, de aplicaciones como ping, SSH y aplicaciones web. También se utilizaron pruebas con distintos prefijos de red, tanto de la red interna, como de la externa. En la figura VII se presenta una de las topologías testeadas.

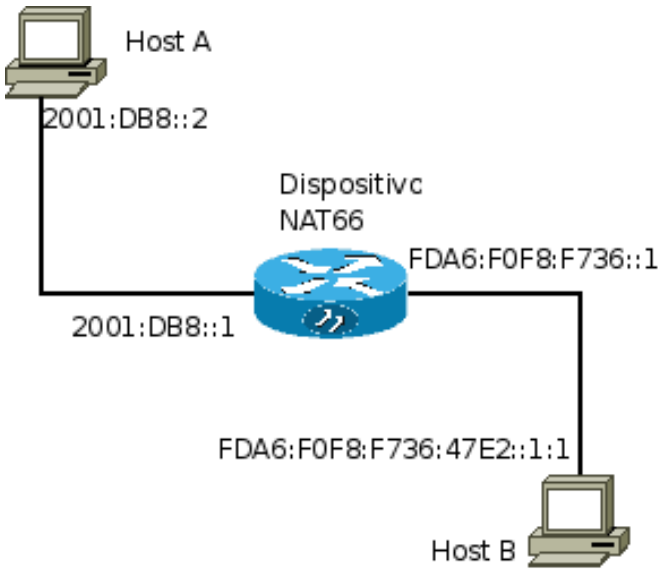

Ilustración 25: Testing NAT66

En este caso **2001:DB8::/48** es el prefijo externo del **dispositivo NAT66**, mientras que el prefijo interno es **FDA6:F0F8:F736::/48**. La dirección **FDA6:F0F8:F736:47E2::1:1** es mapeada al prefijo externo como **2001:DB8::1:1** de esta manera desde el **Host A** para pinguear al **Host B** debe hacerlo a la dirección **2001:DB8::1:1**. De la mismo modo que todo paquete que salga del **Host B** al pasar por el **dispositivo NAT66** cambia su origen a **2001:DB8::1:1**.

Cabe destacar que el prefijo de la red interna **FDA6:F0F8:F736::/48**, es una dirección ULA (Unique Local Address) [RFC4193] registrada en el sitio de SixXS[SIXXS] para la utilización en este proyecto. Si bien el registro de prefijos de las direcciones ULAs no es obligatorio, este sitio puede llegar a ser una buena herramienta en el futuro donde todo aquel que utilice un prefijo ULA lo registre de manera de evitar colisiones. Si bien las direcciones son no routeables globalmente es posible que existan colisiones, al igual que existen con las direcciones privadas de IPv4, por ejemplo al conectar VPNs de distintas empresas. Actualmente existen 261 prefijos registrados en el sitio de SixXS incluyendo el del proyecto IP4JVM.

5.4.Benchmarking

Si bien es claro la superioridad de las soluciones nativas del sistema operativo frente a soluciones montadas sobre el mismo en máquinas virtuales, se decidió realizar pruebas para ver que tan grandes eran las diferencias en tiempos entre el forwarding nativo del sistema operativo, y el forwarding realizado por el proyecto IP4JVM.

La red, en ambos casos, fue configurada con dos hosts y un router, todos corriendo en la misma máquina mediante virtualización. Uno de los hosts se utilizó en la máquina host de la virtualización, mientras que el router y el otro hosts fueron los guests de la misma.

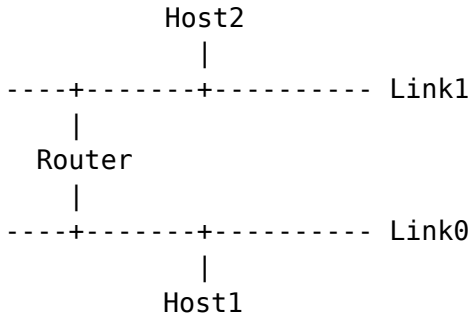

Ilustración 26: Topología forwarding

Se realizó pings entre el **Host 1** y el **Host 2**, de esta manera el **Router** debió realizar el forwarding del **ICMPv6 Request** y su respectiva respuesta **ICMPv6 Reply**.

A continuación se muestra las gráficas de tiempo de la realización de **25 pings** con distinto tamaño de paquete. En azul y mediante cuadrados se presenta los tiempos para el router del proyecto IP4JVM, mientras que en rojo y con diamantes se presenta los tiempos para un router GNU/Linux.

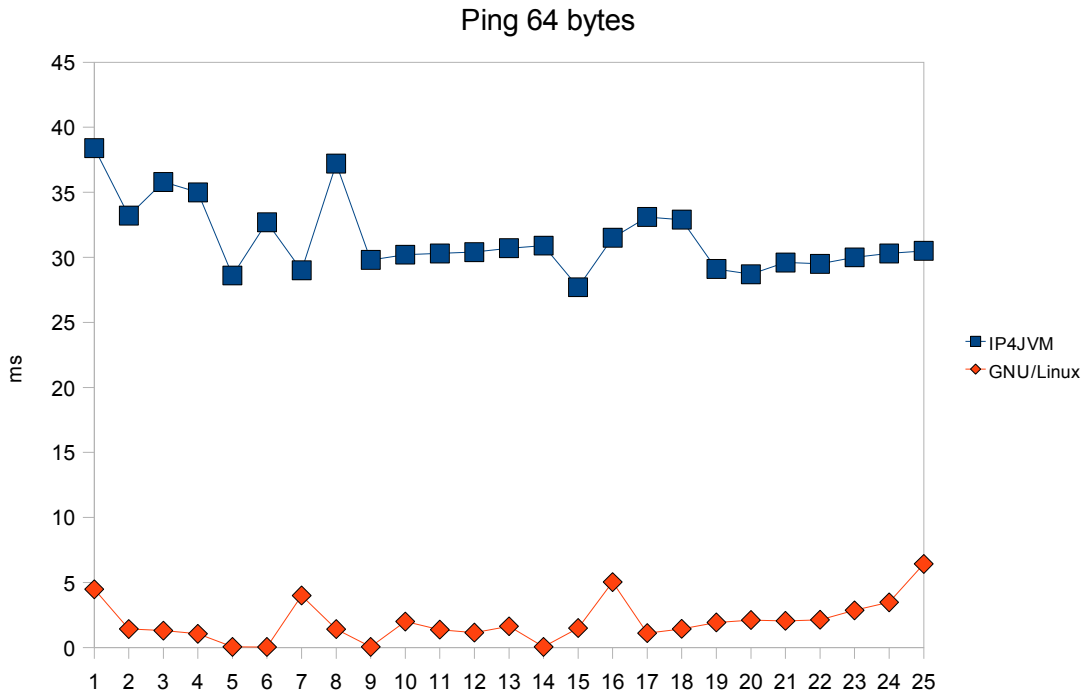

Ilustración 27: Gráfica Ping 64 bytes

La ilustración 27 presenta los tiempos insumidos para realizar un ping entre dos nodos a través del router. El tiempo promedio cuando se utilizó el proyecto IP4JVM fue de 31 ms, frente a los 2 ms cuando se utilizó GNU/Linux.

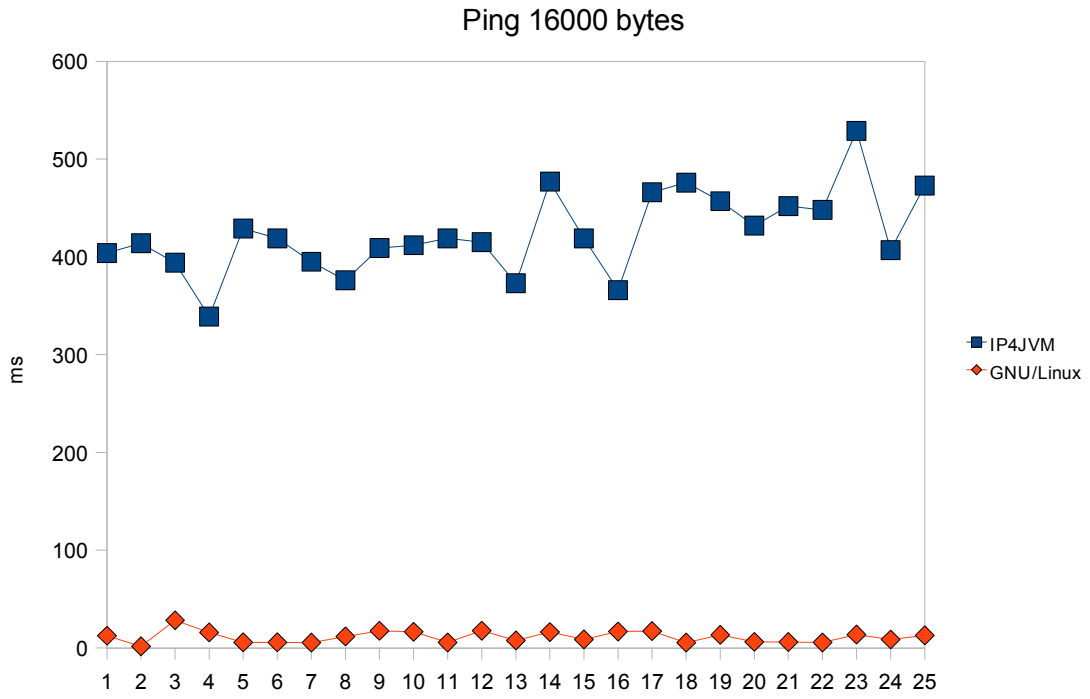

Ilustración 28: Gráfica Ping 16000 bytes

En la ilustración 28, se aprecian los tiempos insumidos para un ping de 16000 bytes, en este caso el tiempo promedio en el caso de utilizar el proyecto IP4JVM fue de 424 ms, frente a 12 ms del sistema operativo GNU/Linux.

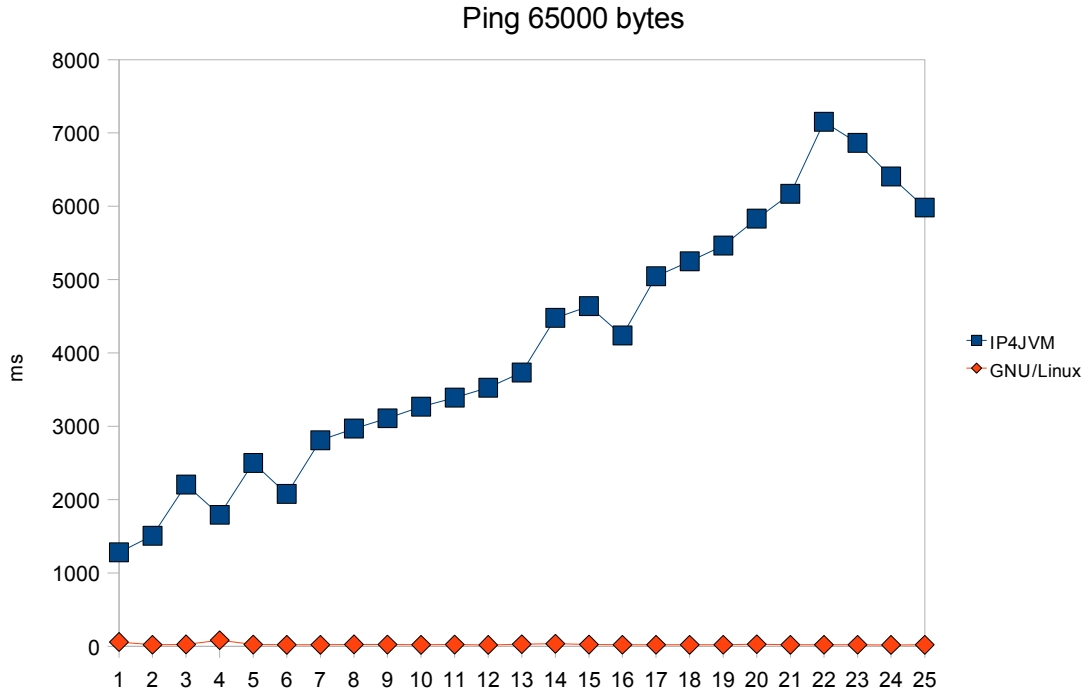

Ilustración 29: Gráfica Ping 65000 bytes

A continuación se realizó un scp, secure copy, de archivos de 1 MB, 5MBs, 10MBs y 20MBs, entre el **Host 1** y el **Host 2** con una topología igual a la presentada en la ilustración 26, comparando los tiempos de transferencia cuando se ruteo mediante el proyecto IP4JVM y cuando se ruteo con el sistema operativo GNU/Linux.

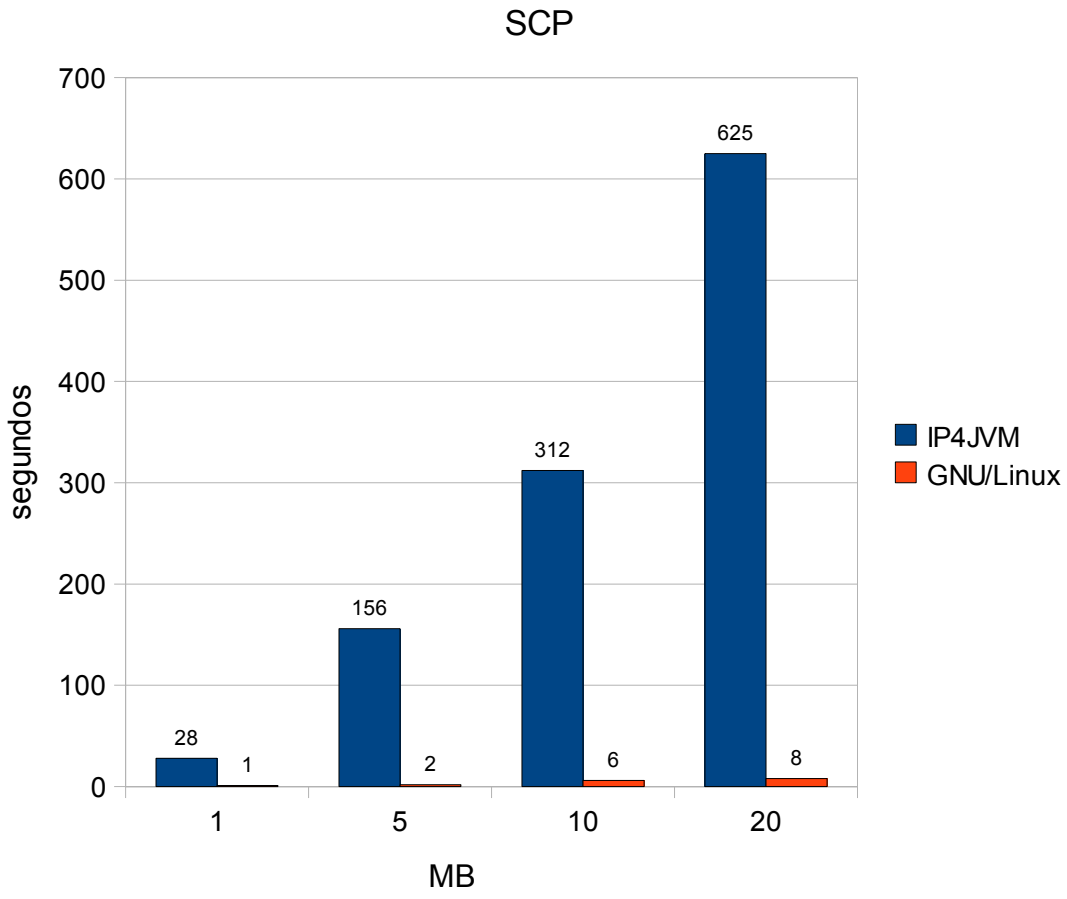

Ilustración 30: Gráfica SCP

La ilustración 30 presenta los segundos que demoró en realizarse la transferencia de los distintos tamaño de archivos con las distintas variantes de ruteo. En promedio cuando se utilizó el proyecto IP4JVM la tasa de transferencia fue de 36 KB/s mientras que cuando se utilizó el sistema operativo GNU/Linux fue de 1800 KB/s.

5.5.Conclusiones

Luego de concluidos los tests de Router y de NAT66 podemos afirmar que se llego al nivel esperado para esta etapa del proyecto. Si bien los tiempos ya se sabía que serían muy superiores los de forwarding nativo en un sistema operativo que los del proyecto IP4JVM, no se consideró que fuera de tanta magnitud la diferencia entre ambos, principalmente cuando se congestiona como en el caso del ping de 65000 bytes, y en el caso de la transferencia de archivos donde es en el orden de 50 veces más rápido la solución nativa.

6.Índice de ilustraciones