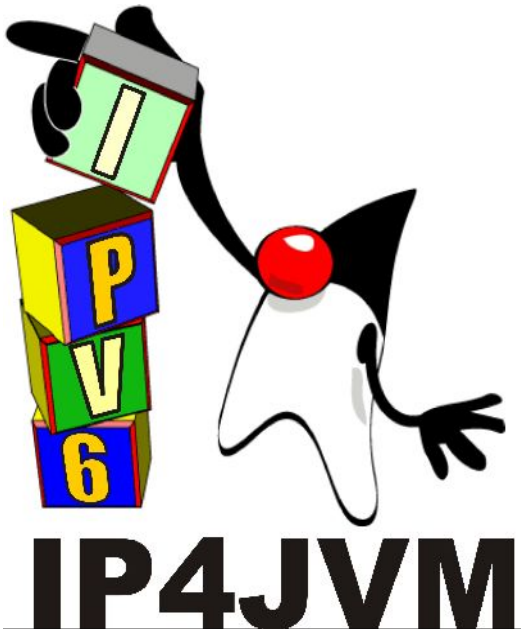


Proyecto de Grado 2010

CERTIFICACION DE IP4JVM

Anexo V – Interoperabilidad

Autor: Daniel Enrique Rosano Lorenzo

Tutores:
Ariel Sabiguero Yawelak
Leandro Scasso

Instituto de Computación

Facultad de Ingeniería

Universidad de la República

Octubre de 2011

Índice

1 Introducción

Los tests de interoperabilidad son la segunda parte para poder conseguir la certificación de IPv6ReadyLogo.

Esta etapa consiste en ejecutar un conjunto de tests, que a diferencia de los tests de conformidad, no tiene una herramienta automática implementada, por lo tanto todos los tests son manuales, o en el mejor de los casos, semiautomático.

A diferencia de IPv4, que comenzó con un pequeño y cerrado grupo de implementadores, la universalidad de IPv6 lleva a un gran número de implementaciones. La interoperabilidad siempre ha sido considerada como una característica fundamental. Para un producto es muy importante dar una buena señal probando el nivel de interoperabilidad con diferentes productos, y para evitar la confusión, se creó un conjunto de tests único.

2 Implementación de la máquina de testing

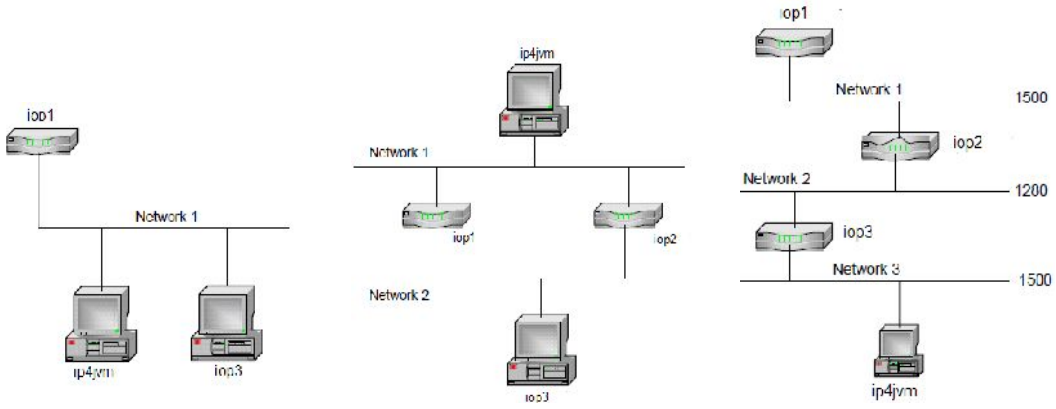
La topología utilizada en los tests de interoperabilidad consiste en 2 (o 3) nodos que actúan como Routers y 3 nodos que actúan como Hosts. Uno de esos 3 nodos Host es un nodo con ip4jvm instalado (nodo bajo test).

Ya que es muy complejo tener 5 nodos físicos, se crearon 5 nodos virtuales en VMWare en una máquina que llamaremos **iopmaster**, a saber:

iop1, iop2, iop3, iop4: nodos que pueden actuar como Routers y Hosts

ip4jvm: un nodo que tiene instalado ip4jvm

A su vez se crean un conjuntos de redes virtuales que interconectan nodos, que llamaremos **Network 1**, **Network 2** y **Network 3**



Un par de ejemplos de topologías que se puede armar con nuestros nodos virtuales: Izquierda (Test 1), centro (Test 29) y derecha (Test 36). Como se puede observar, los nodos pueden actuar tanto como Host o como Router (en el test 1 iop3 es Host y en el 36 es Router)

3 Herramienta de testing

Como es complicado configurar los nodos iopX cada vez que se realiza un test, se necesita alguna suerte de herramienta que configure los nodos de manera automática y ejecute los comandos necesarios para los tests.

Desarrollada por el tutor de este proyecto, Ariel Sabiguero, se tiene una herramienta que facilita bastante el trabajo. Esta herramienta se compone de un conjunto de scripts que implementan los pasos de los tests de interoperabilidad y además se encarga de la inicialización de los nodos y las redes virtuales.

Esta herramienta se encuentra en la máquina que aloja las máquinas virtuales (iopmaster).

Del lado del usuario que corre el test, únicamente debe manejar la máquina con ip4jvm y del resto se encarga la herramienta.

3.1 Uso de la herramienta

Estando en el directorio raíz de la máquina iopmaster, en primer lugar hay que crear todas las redes virtuales que se identifican como eth0.X (donde X es un número de 13 a 20). Para esto se utilizan estos dos comandos:

$ cd interop/ip4jvm
$ ./config_master config

Este último comando crea y habilita todas las redes virtuales.

Luego de esto, es necesario inicializar todas las máquinas virtuales, o sea configurar todos los nodos para que actúen, ya sea como nodos o como routers. Para esto, hay que cambiarse al directorio out y correr el comando execute 0.

$ cd out
$ ./execute 0

El comando anterior configura todas las variables de las máquinas virtuales iopX.
A partir de aquí podemos comenzar a correr los diferentes tests

$ ./execute X
(donde X es un número entre 1 y 40)

Cada test está compuesto de un conjunto de pasos (Step) la mayoría de los cuales son ejecutados automáticamente por la herramienta.

El usuario de la herramienta únicamente debe preocuparse de ejecutar los comandos que competen al nodo “ip4jvm” (léase el nodo que tiene instalado la herramienta ip4jvm)

Cuando la herramienta le pida al usuario que ejecute un cierto comando le da tiempo suficiente para hacerlo y luego presionando la tecla enter continúa la ejecución del test.

Las operaciones que se le “piden” a ip4jvm se dividen esencialmente en las siguientes:

- Habilitar la red
- Deshabilitar la red
- Hacer un ping a otro nodo (Do Ping)
- Cambiar la IP del nodo (remover la IP vieja y asignar una nueva)

4 Requerimientos del nodo

- Se necesita poder configurar una dirección global y un default router, al recibir un Router Advertisement.

- Se debe poder usar una aplicación del estilo ping6 e imprimir los resultados indicando la recepción de un ICMPv6 Echo Reply.

5 Tests de interoperabilidad

Los tests de interoperabilidad se dividen en 6 categorías particulares que analizan aspectos del intercambio de información entre nodos con capacidades IPv6. Algunas de estas 6 categorías, a su vez se subdividen en diferentes tests que prueban ciertos casos de prueba.

Interop 1.1 : ICMPv6 Echo Interoperability

Parte A : Direcciones link-local unicast (Host vs Host) -> Tests 1 y 4
Parte B : Direcciones unicast globales (Host vs Host) -> Tests 2 y 5
Parte C : Direcciones multicast (Host vs Host) -> Tests 3 y 6
Parte D : Direcciones link-local unicast (Host vs Router) -> Tests 7 y 10
Parte E : Direcciones unicast globales (Host vs Router) -> Tests 8 y 11
Parte F : Direcciones multicast (Host vs Router) -> Tests 9 y 12

Interop 1.2 : Address Autoconfiguration and DAD

Parte A : DAD – Tentative Address Unique (Host vs Host) -> Tests 13 y 15
Parte B : DAD – Tentative Address Duplicated (Host vs Host) -> Tests 14 y 16
Parte C : DAD – Tentative Address Unique (Host vs Router) -> Tests 17 y 19
Parte D : DAD – Tentative Address Duplicated (Host vs Router) -> Tests 18 y 20

Interop 1.3 : Procesamiento de RA's – Prefix Discovery

Parte A : Single Prefix Discovery (Host vs Router) -> Tests 21 y 24
Parte B : Multiple Prefix Discovery (Host vs Router) -> Tests 22 y 25
Parte C : Prefix lifetime expires (Host vs Router) -> Tests 23 y 26

Interop 1.4: Procesamiento de RA's – Router Lifetime

-> Tests 27 y 28

Interop 1.5 : Redirect

-> Tests 29 y 30

Interop 1.6: Path MTU Discovery y Fragmentación

Parte A : PMTU Discovery (Host vs Router) -> Tests 31 y 32
Parte C : Fragmentación / Rearmado (Host vs Host) -> Tests 33 y 34
Parte D : Fragmentación / Rearmado (Host vs Router) -> Tests 36 y 37

6 Estado inicial de las pruebas

En la primera corrida de los tests se obtuvieron los siguientes resultados:

1	1.1.A	OK	11	1.1.E	FALLA	21	1.3.A	OK	31	1.6.A	OK
2	1.1.B	OK	12	1.1.F	FALLA	22	1.3.B	OK	32	1.6.A	OK
3	1.1.C	FALLA	13	1.2.A	OK	23	1.3.C	OK	33	1.6.C	FALLA
4	1.1.A	OK	14	1.2.B	OK	24	1.3.A	OK	34	1.6.C	FALLA
5	1.1.B	OK	15	1.2.A	OK	25	1.3.B	OK	35	Setup	OK
6	1.1.C	FALLA	16	1.2.B	OK	26	1.3.C	OK	36	1.6.D	FALLA
7	1.1.D	FALLA	17	1.2.C	OK	27	1.4	OK	37	1.6.D	FALLA
8	1.1.E	FALLA	18	1.2.D	OK	28	1.4	OK			
9	1.1.F	FALLA	19	1.2.C	OK	29	1.5	OK			
10	1.1.D	FALLA	20	1.2.D	OK	30	1.5	OK			

Esencialmente, los errores se agrupaban en 3 categorías

Tests 3, 6, 7, 9, 10 y 12: Se enviaban 6 Echo request pero se recibían 3 Echo reply

Tests 8 y 11: Error *connect : Network is unreachable*

Tests 33, 34, 36 y 37: Faltaba implementar una funcionalidad que permitiera enviar pings de cierto tamaño a través de la interfaz web de ip4jvm

7 Soluciones

7.1 Error A: Tests 3, 6, 7, 9, 10, 12

Estos tests consisten en enviar 6 Echo Requests y recibir 6 Echo Replies.

El problema de dicho test, es que de 6 Echo Requests, solamente se reciben 3.

La explicación de este error viene de la creación de la función “Habilitar/deshabilitar”

Cuando se deshabilita una interfaz, al intentar rehabilitarla se deja el estado del nodo incorrecto, lo que lleva a este error

Para evitar este error se modificó el comportamiento de la función que “rehabilita” una interfaz. La solución simplemente consistió en colocar una llamada a la función Network.start() que inicializa las interfaces de red.

7.2 Error B: Tests 8, 11

Al intentar hacer ping hacia/desde ip4jvm se produce el siguiente error:

Connect: Network is unreachable

La solución a este error, aunque sencilla fue muy complicada de encontrar. Primero se intento atribuirla a errores en ip4jvm. Se debuggeo el código para ver si había algún problema para este test en particular, con resultados negativos.

Pero se descubrió que si se ejecuta este test únicamente, se corre correctamente.

Sin embargo, si se corren otros tests antes, no se corre correctamente.

Por lo tanto el error pasó a estar del lado de la máquina tester iopmaster.

Se intentó por varios caminos identificar el error hasta que se llegó a la conclusión de que el error radicaba en que no se daba correctamente de baja la interfaz iop1.

```
ifconfig eth0 down
ifconfig eth0 0.0.0.0
```

Resultaba que también se necesitaba dar de baja la dirección global creada en uno de los primeros pasos

Para esto se agregó en los scripts de los tests que utilizan direcciones globales dentro de ip4jvm/interop/out/iopmaster/, la siguiente línea de código:

```
ifconfig eth0 del direccion_global
```

7.3 Error C: Tests 33, 34, 36, 37

En este caso no se trata de un error, sino de una funcionalidad extra que se necesito implementar.

Era necesario poder enviar desde la interfaz web de ip4jvm pings de un cierto tamaño.

Ejemplo (Test 33):

Step 19
ip4jvm Send 5 ICMP Echo Requests (1452 bytes) to 3ffe:501:ffff:1:20c:29ff:feec:e2cc via interface 1

Para solucionar ese error se agregó un campo dentro de “doPing.jsp” para especificar el tamaño del ping, y se modificó la clase Ifaces.java, de modo de contar con esta funcionalidad.

8 Conclusiones

Si bien esta etapa debiera haber sido relativamente sencilla de resolver, tuvo sus complicaciones:

1) Intentar resolver los errores sin un ambiente de testing adecuado. Primero se intentó hallar las soluciones a los errores sin tener dos máquinas involucradas y probando en paralelo. Esto se prolongó aproximadamente por un par de meses sin resultados positivos ya que al momento de debuggear cambios había que subir todo al SVN para que del otro lado se probara la interoperabilidad.

2) Armar el ambiente de testing con dos PCs. Visto que no se pudo resolver los tests con una única máquina, se armó una PC completa con el ambiente de testing la cual se conectaba con la máquina bajo test (la que tenia IP4JVM). De esta forma era posible realizar cambios y probarlos en el momento.

3) Identificar la causa de los errores A y B e implementar el error C. Error C fue sencillo de implementar. Tal vez la peor parte de esta etapa fue lo complicado de identificar las causas de los errores A y B.

En el caso del error A se intentó buscar la causa a en un nivel más bajo (a nivel de procesos por ejemplo), se probaron varias cosas hasta llegar a la conclusión de que era un tema de estados y de objetos que no quedaban bien inicializados a la hora de rehabilitar.

Encontrar la solución del error B fue más difícil ya que el error que se obtenía no era para nada descriptivo y del lado de la máquina tester (no la máquina con IP4JVM). Se debuggeo código y se observó el tráfico intentando encontrar una explicación hasta que se llegó a la conclusión de que el error estaba en la máquina tester (iopmaster). Ahí comenzó otra etapa intentando identificar donde estaba el problema, si era un problema de Kernel de Linux o un problema de configuración de la red. Finalmente se llegó con mucha investigación a que el problema se encontraba al cerrar las conexiones de red creadas por los scripts que implementan los tests de interoperabilidad.