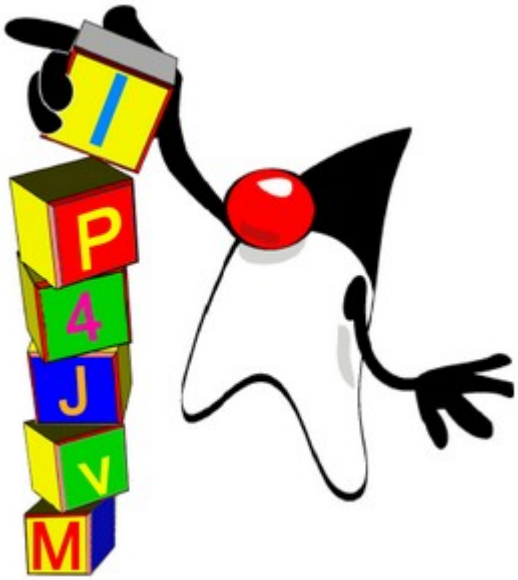

Proyecto de Grado 2008

Anexo V

Configuración de un Ambiente de Testing en IP4JVM

Autores:

Leandro Scasso

Marcos Techera

Tutor:

Ariel Sabiguero

Tribunal:

Andrés Aguirre

Eduardo Grampín

Carlos Martínez

Tabla de contenidos

1.INTRODUCCIÓN.....3

2.HERRAMIENTAS.....4

2.1.Wireshark.....4
2.2.Eclipse.....5
2.3.Apache Tomcat.....5
2.4.VMware Server.....6
2.5.OpenJDK.....6
2.6.Proyecto IP4JVM.....6
2.7.openSUSE.....7
2.8.FreeBSD.....7
2.9.TAHI.....7
2.9.1.TAHI v6eval.....7
2.9.2.TAHI IPv6 Core Protocols.....8
2.9.3.TAHI DHCPv6.....8
2.10.radvd.....8
2.11.dibbler-server.....9
2.12.Sistemas Operativos.....9

3.CONFIGURACIÓN ARCHIVOS IP4JVM.....10

3.1.ip4jvm.config.....10
3.2.dhcpv6.config.....12

4.ÍNDICE DE ILUSTRACIONES.....14

1.Introducción

En el presente documento se explica como instalar y dejar en funcionamiento las distintas herramientas que permiten realizar los tests sobre el sistema IP4JVM. Así como también adjunta configuraciones de ejemplo para las distintas herramientas

La siguiente tabla muestra los productos utilizados, así como la versión durante el desarrollo y testing del proyecto.

Producto	Versión
Wireshark	1.0.2
Eclipse	3.4
Subclipse	1.2.x
Apache Tomcat	6.0.18
OpenJDK	1.7
VMware Server	2.0.0
openSUSE	10.3
FreeBSD	6.3
TAHI v6eval	3.0.16
TAHI DHCPv6	1.0.16
TAHI IPv6 Core Protocols	4.0.4
RADVD	1.1
Dibbler-server	0.7.1

Ilustración 1: Productos utilizados

2.Herramientas

2.1.Wireshark

Wireshark[WARK] es un analizador de protocolos de red que permite capturar tráfico en tiempo real para ser analizado. De está manera es posible ver los paquetes que son enviados y recibidos por el proyecto IP4JVM y de esta manera corroborar que estén bien formados y que sean los esperados. Es posible instalarlo en diversos sistemas operativos. Para instalarlo se debe bajar el paquete desde la página de wireshark http://www.wireshark.org/download.html o mediante administradores de paquete provistos por el sistema operativo.

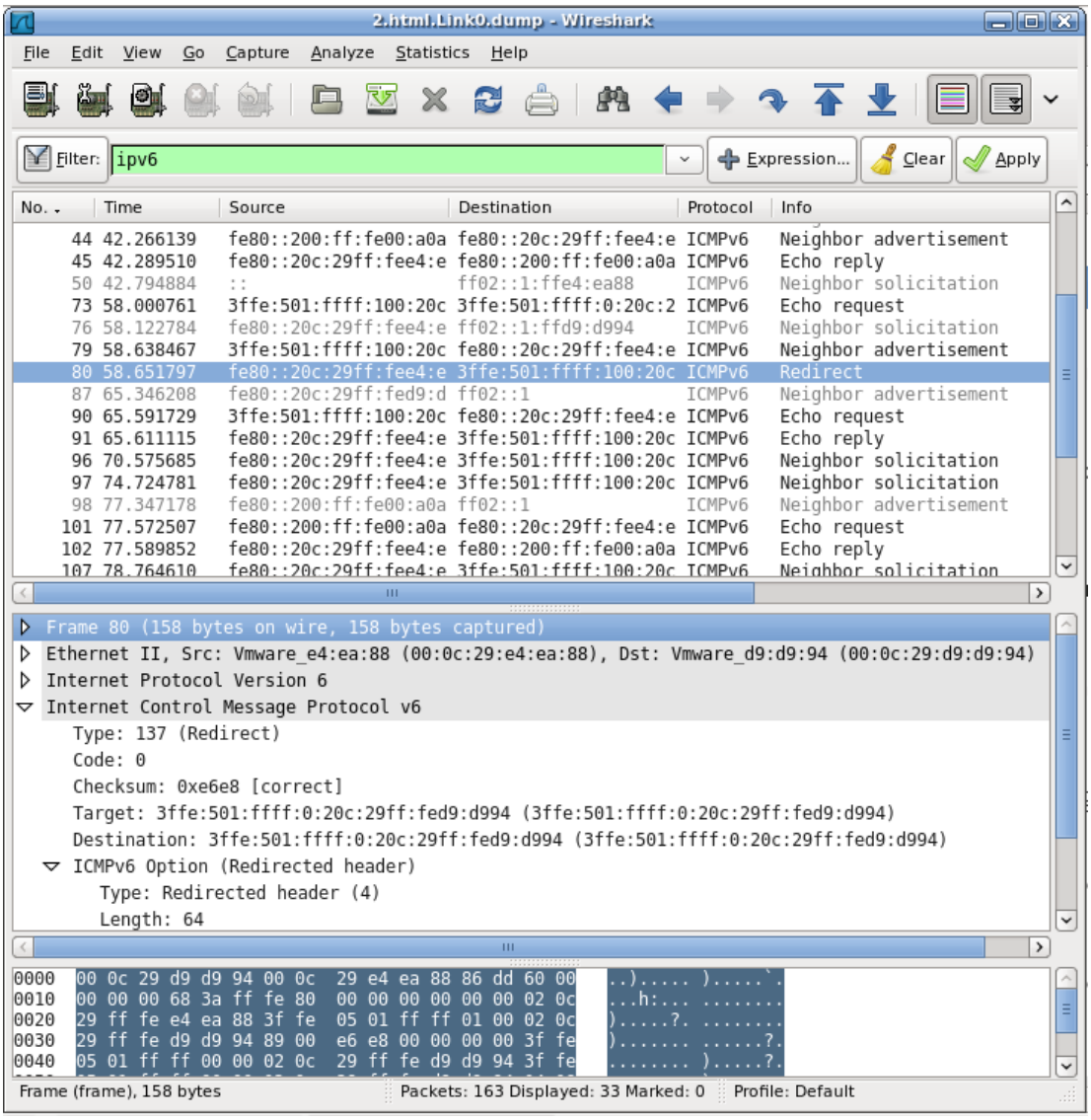

Ilustración 2: Captura de paquetes Wireshark

En la ilustración 2 se muestra una captura de paquetes realizada con el Wireshark. En la parte superior se muestran los paquetes capturados, en la parte central se muestra un detalle del paquete seleccionado, con el respectivo análisis por capas y tipo de paquete. Y en la parte inferior se muestra en sistema hexadecimal el paquete seleccionado.

2.2.Eclipse

Para realizar cambios e introducir modificaciones en el código del proyecto se utilizó el IDE Eclipse [EPSE]. La herramienta no requiere instalación, basta con descomprimir el archivo de que se puede bajar de la página http://www.eclipse.org/download/ en sus distintas versiones para los distintos sistemas operativos. Para el proyecto de configuración WEB, se utilizó el plugin para Apache Tomcat de Eclipse, el mismo puede ser instalado desde el eclipse mediante el gestor de plugins.

Además para poder utilizar el repositorio SVN y utilizar el sistema de control de versiones disponible desde el Eclipse, se instaló el plugin de Subclipse [SCPE], versión 1.2.x . Las instrucciones para lograr instalarlo en el IDE Eclipse se encuentran en la siguiente página http://subclipse.tigris.org/install.html.

También para facilitar el desarrollo de la aplicación Web, se utilizo el plugin de Eclipse para Apache Tomcat.

Se determinó utilizar este IDE debido a que se tenía mayor conocimiento del mismo, frente a otros como Netbeans o JDeveloper, etc. Además el grupo anterior venía desarrollando también en Eclipse.

2.3.Apache Tomcat

Apache Tomcat[TCAT] es un servidor web que brinda soporte para servlets y páginas JSPs escrito en Java, como se explicó en el **Anexo III – Aplicación Web**, se realizó una aplicación web para realizar las tareas de configuración del proyecto que corre sobre este servidor. La instalación y configuración del mismo forma parte del **Anexo III – Aplicación Web**.

2.4. VMware Server

VMware Server [VARE] es un software gratuito de virtualización, es por esto que nos permite la virtualización de varias máquina lógicas en una sola máquina física. De esta manera se puede realizar distintas topologías de red en una sola máquina física, lo cual es ideal para el testing.

Se prefirió ante otras alternativas, como es VirtualBox [VIRB] que dispone de una licencia open source, debido a que tiene un mejor rendimiento que el resto, la facilidad para configuraciones de red. Durante el desarrollo del proyecto se comenzó utilizando la versión VMware Server 1.0.5, pasando por las distintas versiones 1.0.x hasta la 1.0.8 y por último pasando a la versión 2.0.0.

2.5. OpenJDK

OpenJDK[OPJDK] es la versión libre de la plataforma de desarrollo Java de SUN[SUN]. Fue la máquina virtual Java utilizada a lo largo del proyecto debido a que era la que venía siendo utilizada en versiones anteriores del proyecto.

Los pasos a seguir para la instalación de la misma integrando la herramienta IP4JVM se encuentran en la documentación del grupo que realizó el proyecto previamente [IP4JVM] en el punto 10.2.4 Instalación en OpenJDK en la página 105.

2.6. Proyecto IP4JVM

El proyecto se encuentra alojado en el SVN del INRIA[INRIA] en la siguiente dirección https://gforge.inria.fr/projects/ip4jvm/ , para poder acceder al mismo es necesario contar con los permisos adecuados. Se comenzó a trabajar en la revisión 530, en un principio se trabajó de forma concurrente con otro grupo académico que se encontraba extendiendo el proyecto IP4JVM, hasta la revisión 671 el 17 de diciembre de 2008.

Para realizar el checkout del proyecto se recomienda la lectura del informe final del trabajo academico realizado por Roger Abelenda e Ignacio Corrales [IP4JVM] en la página 97, Proyecto IP4JVM.

El SVN se encuentra dividido en varias directorios:

- **doc** donde está la documentación del proyecto en todas sus etapas.
- **lib** donde se encuentran las **.dll** y los **.so** que realizan la conexión entre IP4JVM y el **Pcap** ya compiladas.

- **src** donde se encuentra el código fuente del proyecto. Este a su vez se divide en subcarpetas en donde se encuentra el código fuente C, el código del proyecto en general, los fuentes de DHCPv6, el proyecto Common y el proyecto para la configuración web
- **IP4JVM iter2** donde se encuentra alojado un tag realizado por el el grupo que trabajo antes que el nuestro.
- **vmachine** donde están las máquinas virtuales en la versión utilizada.
- **Apache-tomcat-6.0.18** donde se encuentra la versión modificada de Apache Tomcat para IP4JVM

2.7.openSUSE

openSuse [OPSU] es un sistema operativo Linux, fue seleccionado entre los distintos sistemas operativos existentes debido a que existía ya una documentación generada de como configurar este sistema operativo para poder utilizar IP4JVM sobre el. Se utilizó en su versión 10.3, que era la actual al momento de comenzar el proyecto.

2.8.FreeBSD

FreeBSD [FBSD] es un sistema operativo derivado de BSD[BSD], la versión UNIX desarrollada en la Universidad de California, Berkeley.

Se utilizó para las pruebas debido a que el proyecto TAHI debía instalarse en este sistema operativo

2.9.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 tests de interoperabilidad entre las distintas implementaciones, reducir los esfuerzos de los implementadores para realizar el testing y clarificar las ambigüedades de los RFCs.

2.9.1.*TAHI v6eval*

v6eval es la herramienta para generar los tests. Como requerimiento luego de instalar el FreeBSD, se debe instalar los paquetes lang/p5-Expect y security/ps-

Digest-MD5, los cuales se encuentran disponibles en los puertos del FreeBSD. Trae ejemplos básicos de tests para poder ver si quedo bien instalado.

2.9.2.*TAHI IPv6 Core Protocols*

Son los tests de los protocolos centrales de IPv6, el test lo realiza a partir de las especificaciones de los RFCs más importantes de IPv6, RFC 2460 [RFC2460], RFC 4861[RFC4861], RFC 4862[RFC4862], RFC 1981[RFC1981], y RFC 4443[RFC4443], tanto para hosts como para routers. Para instalarlo se debe solamente descomprimir la carpeta, realizar la configuración pertinente a lo que se quiere testear y luego **make ipv6ready_p2_host** si el nodo a testear es un host y **make ipv6ready_p2_router** si es un router.

2.9.3.*TAHI DHCPv6*

Contiene los tests del protocolo DHCPv6, especificados por los RFC 3315[RFC3315], RFC 3646[RFC3646] y RFC 3736[RFC3736], tanto para clientes, servers o relay agents DHCPv6. Al igual que los **Core Protocols** para instalarlo se debe descomprimir la carpeta, realizar la configuración pertinente a lo que se desea testear y luego ejecutar los tests mediante **make ipv6ready_p2_client**, para un cliente DHCPv6, **make ipv6ready_p2_server** para testear un servidor, y **make ipv6ready_p2_relay** si es un relay agent. En nuestro caso debido a que se implementó sólo el cliente y sin configuraciones de DNS, solamente se ejecutó el test para el cliente testeando la implementación del RFC

2.10.radvd

radvd [RVD] es un demonio de router advertisements para IPv6 en GNU/Linux. Escucha los router solicitations y envía los routers advertisements como se describe en el RFC 4861[RFC4861]. Mediante estos advertisements los hosts pueden automáticamente configurar sus direcciones y otros parámetros como ser la búsqueda de un servidor DHCPv6.

Para poder iniciar el demonio es necesario que el nodo tenga habilitado el forwarding de paquetes IPv6. Para esto se recomienda realizar **echo 1 > /proc/sys/net/ipv6/conf/all/forwarding** como root del sistema antes de iniciar el demonio.

2.11.dibbler-server

dibbler-server] es una implementación de un servidor DHCPv6. Soporta tanto la configuración stateful y stateless. Es posible correrlo en sistemas operativos GNU/Linux, así como también en Windows XP y 2003. Es una implementación de los RFC 3315[RFC3315], RFC 3319[RFC3319], RFC 3646[RFC3646], RFC 3736[RFC3736] y el RFC 3898[RFC3898].

2.12.Sistemas Operativos

Como sistemas operativos host del VMWare Server fueron utilizados los sistemas operativos Windows XP[WINXP], Windows Vista[WVST], Debian GNU/Linux Lenny 5.0 [DIAN], Ubuntu GNU/Linux 8.04 Hardy Heron [UBU]. Los sistemas operativos host, fueron utilizados como herramienta básica de testing (ping entre el sistema operativo host y el proyecto IP4JVM), y para testing de la herramienta de configuración. En todos los casos se utilizó el stack IPv6 del host para realizar los tests por lo que podemos resaltar la interoperabilidad del proyecto IP4JVM.

3.Configuración archivos IP4JVM

En el presente punto se explica parámetros y configuraciones de los archivos de configuración de IP4JVM.

3.1.ip4jvm.config

Es el archivo de configuración de los parámetros específicos de IP4JVM al booteo de la máquina virtual. En la ilustración 3 se presenta un ejemplo del mismo.

El archivo de configuración presenta tres zonas delimitadas por por una linea de guiones ("--------"). Los comentarios son precedidos por el símbolo de numeral ("#").

En la primera división se encuentra las configuraciones básicas del stack, esto es, si se desea que el stack soporte IP móvil se tiene que encontrar la siguiente linea "**mipv6Enabled 1**". Si queremos que el proyecto se comporte como un router entonces se debe encontrar la línea "**isRouter 1**". Por último si queremos que NAT66 este habilitado entonces se debe encontrar la línea "**natEnabled tipo prefijoExterno prefijoInterno"**, donde el tipo 1 representa el **TwoWayAlgoritmicProtocol** y el tipo 2 es **TopologyHidingOptionProtocol**. Como su nombre lo dicen **prefijoExterno** representa el prefijo de red externo del dispositivo NAT66 y **prefijoInterno** representa el prefijo de red interno del dispositivo NAT66.

En la segunda parte se especifican las interfaces de red. El formato general es **identificador tipo opcionesEspecificas**. El **identificador** es el nombre de la interface de red en el stack. Existen tres posibles valores para el **tipo**:

El tipo **1** que representa interfaz de red común. Como opciones especificas tiene el **nombre** (eth0 por ejemplo en un sistema operativo de GNU/Linux) de la interfaz del sistema operativo por donde leerá y saldrán los paquetes, la **MAC** de la interfaz a crear, y por último como parámetro opcional el **MTU** de la interfaz, en caso de no encontrarse por defecto se utiliza 1500.

El tipo **2** representa las direcciones de loopback y no tiene opciones especificas.

El tipo **3** representa los túneles para IP Móvil. Como opciones especificas tiene la interfaz de red mapeada al túnel, el punto de salida, el punto de entrada y el prefijo de punto de salida. Por más información de como configurar este tipo ver MIPv6 with IP4JVM and Irisa Ipv6[MIPV6IP].

```
#The format of the file must be respected, one line starting with '-' is a division
between kinds of options

# one starting with '#' is a line of comments

#General Settings (identifier, value)

#this option enables the use of MIPv6 support

#mipv6Enabled 1

#this option enables the use of IPv6 forwarding support

isRouter 1

#this option enables the use of NATv66 support (draft-mrw-behave-nat66-01.txt)

#option 1 TwoWayAlgoritmic

#option 2 Topology Hiding Option

#Address1=Outbund Prefix

#Address2=Inbound Prefix

#natEnabled 1 2001:DB8:: fda6:f0f8:f736::

------------------------------

#Interfaces (identifier, type (1-eth, 2-lo, 3- tun), specific options)

#Specific Options:

#interfaces eth: physical interface identifier, MAC, mtu (optional, Default:1500)

#interfaces lo: (no options)

#interfaces tun: mapped interface, exit point, entry point, exit point prefix length

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

jth1 1 eth1 00:0C:29:E4:EA:89

------------------------------

#IPS (identifier, ip, prefixLength (optional, Default:128 tentative (optional,
Default:true))

jth0 2001:DB8::1 32

jth1 2001:DB8:CAFE::1 32

------------------------------
```

Ilustración 3: Archivo de configuración ip4jvm.config

En la tercera y última parte se especifican las IPs que se configurarán al iniciar el stack. El formato general es **identificador IP largoPrefijo tentativa**. El **identificador** corresponde al nombre seleccionado para la interfaz en el stack. La **IP** es la dirección IPv6 a configurar, **largoPrefijo** es un parámetro opcional que corresponde al largo de prefijo de la dirección IPv6, por defecto es 128. Y por

último **tentativa** que define si la dirección es tentativa o no, esto quiere decir, si es necesario realizar el Duplicated Address Detection, por defecto el valor es true.

3.2.dhcpv6.config

Es el archivo general de configuración para un cliente DHCPv6 en el stack. Al igual que para el archivo de configuración ip4jvm.config el # indica que la linea es un comentario.

El archivo está compuesto por las opciones generales de DHCPv6 y opciones más especificas de cada interfaz. El formato general es **opción=valor**.

```
#Sat Sep 13 14:07:09 UYT 2008

DUID=000100011035a70d005056c00001

dhcpEnabled=FALSE

jth0_option_dns_server=TRUE

#jth0_IA=3 2001:DB8::3

#jth0_rapidCommit=TRUE

#jth0_T1=2000

#jth0_T2=2000

#jth0_preferredLifetime=1000

#jth0_validLifetime=1000
```

Ilustración 4: Archivo de configuración de dhcpv6.config

La primera opción corresponde al **DUID** el cual es utilizado para identificar el dispositivo cuando se intercambian mensajes. Se debe de expresar su valor en formato hexadecimal.

La siguiente opcion que se puede apreciar en el ejemplo es la **dhcpEnabled.** Esta opción determina si se debe de utilizar DHCPv6 para configurar el nodo o no. Esta caso de que la opción tome el valor FALSE aunque los Router Advertisements indiquen que se debe de realizar la configuración de las interfaces por medio de DHCPv6 esta configuración no se realizara por que el proyecto IP4JVM tiene apagado el soporte para DHCPv6. En caso de que la opción tome el valor TRUE entonces sólo se realizara la configuración DHCPv6 si los Router Advertisements así lo indican.

El resto de las opciones pueden ser generales, para todas las interfaces, o se puede especificar la interfaz a configurar. Por ejemplo si se quiere que sola interfaz

jth0 pida por un servidor de DNS, se debe configurar en el archivo de configuración **jth0_option_dns_server=TRUE**, pero si se quiere que todas soliciten un servidor de DNS se pone **option_dns_server=TRUE** y esto bastara para que todas las interfaces del stack soliciten por el servidor de DNS.

La opción **rapidCommit**, si se encuentra activado indica que se desea utilizar la opción de Rapid Commit que permite asignar direcciones en una transferencia de dos mensajes en lugar de cuatro.

La opción **T1**, permite al cliente enviar el tiempo T1 que prefiere al servidor DHCPv6. El tiempo T1 determina el tiempo a esperar luego de finalizado el intercambio de mensajes para enviar un Renew.

La opción **T2**, al igual que T1 es el tiempo que el cliente envía al servidor indicando su preferencia de tiempo luego de configurado para enviar un Rebind.

La opción **preferredLifetime** indica el tiempo que desea el cliente que la dirección solicitada siga siendo preferida. La opción **validLifetime** indica el tiempo que desea el cliente que la dirección solicitada sea válida.

En todos los casos los tiempos son los preferidos por el cliente DHCPv6 pero no necesariamente el servidor DHCPv6 enviará los tiempos solicitados por el cliente.

La opción **IA** y la opción **TA**, son de la forma **cantidad IPv6 IPv6 ... IPv6**, donde **cantidad** indica la cantidad de direcciones a solicitar al servidor DHCPv6 y a continuación las direcciones IPv6 solicitadas las cuales son opcionales.

4.Índice de ilustraciones

Ilustración 1: Productos utilizados.....3

Ilustración 2: Captura de paquetes Wireshark.....4

Ilustración 3: Archivo de configuración ip4jvm.config.....11

Ilustración 4: Archivo de configuración de dhcpv6.config.....12