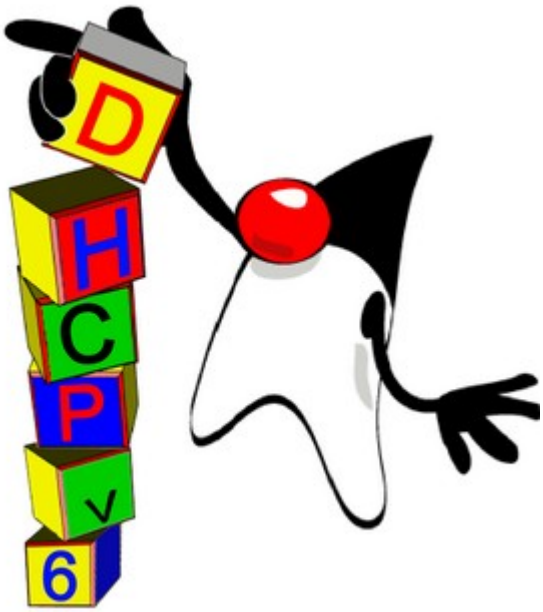

Proyecto de Grado 2008

Anexo I

DHCPv6 - IP4JVM

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. Aprovechando los avances realizados en el proyecto IP4JVM, es que se decidió continuar extendiendo el mismo realizando una implementación que permitiera realizar la configuración de un nodo mediante **DHCPv6**.

1.2.Situación inicial del proyecto IP4JVM

Al comenzar esta etapa, el proyecto IP4JVM manejaba los protocolos **UDP**, **TCP**, **ICMP** los cuales permiten el procesamiento de **Router Advertisements** y **Neighbor Solicitation**, así como el envío de **Neighbor Solicitation** los cuales a su vez permiten la autoconfiguración de direcciones IPv6 mediante el mecanismo de **Stateless Autoconfiguration** descripto en el RFC 4862[RFC 4862].

Debido a todo lo mencionado en el párrafo anterior es indispensable para el funcionamiento de **DHCPv6** es que se realizaron diversos tests para poder corroborar que las implementaciones de los mencionados protocolos así como el manejo de los distintos mensajes fuera el correcto. Por más información sobre los tests y las herramientas utilizadas para llevar a cabo los mismos ver los documentos **Anexo V – Configuración del Ambiente de Testing en IP4JVM** y **Anexo VI – Testing en IP4JVM**.

1.3.Resultados Esperados

Al culminar esta etapa del proyecto se pretende contar con una versión del mismo que permita realizar la configuración de un nodo mediante **DHCPv6**.

Otros resultados deseados son:

- Estado del arte del proceso de autoconfiguración de un nodo.
- Estudio de herramientas que posibiliten el test del proyecto.
- Obtener un diseño orientado a objetos que permita una sencilla incorporación de los protocolos que se configuran a partir de **DHCPv6**, como por ejemplo **DNS**.
- 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 definen 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 se detalla el estado del arte de la configuración DHCPv6 así como de las herramientas a utilizar durante el mismo. Por último se presenta el estado actual del proyecto IP4JVM.

Capítulo 3, Análisis de la solución: aquí se presenta en forma genérica la solución encontrada en función de sus componentes y explicando los principales puntos que componen su arquitectura y como estos se relacionan entre sí.

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

Capítulo 5, Resultados y Conclusiones: aquí es donde se presentan las conclusiones arribadas luego de culminar la etapa correspondiente a la configuración DHCPv6 de un nodo, las limitaciones de la solución realizada para obtener la mencionada configuración y las posibles mejoras a realizarle.

2.Estado del Arte

Durante la fase de investigación, se estudió el estado actual del proyecto IP4JVM para saber cuales eran sus virtudes y defectos en cuanto al manejo de los protocolos IPv6, TCP, UDP e ICMPv6. También se realizó un estudio de todo el proceso llevado a cabo para la autoconfiguración de un nodo mediante el uso de **DHCPv6** así como de las herramientas disponibles para poder testear un sistema que implementa la mencionada autoconfiguración.

2.1.Autoconfiguración DHCPv6

En un principio las computadoras que hacían uso de **TCP/IP** no eran muchas y además las mismas no eran muy portables. Incluso la gran mayoría de las computadoras eran compartidas por muchos usuarios y administradas por un grupo reducido de personas. Debido a todo esto es que en los comienzos no existía o era muy poca la motivación para hacer que la configuración de los dispositivos que usaban **TCP/IP** fuese automática.

En la actualidad nos encontramos en una realidad muy distinta, una organización puede tener cientos o miles de dispositivos IP en su red interna que van desde mainframes a PDAs, los cuales a diferencia de las computadoras del pasado si son portables. Esta portabilidad hace que muchos dispositivos actuales se conecten a diferentes redes durante el transcurso del día. Además las computadoras, por lo general, no son administradas por un grupo de personas si no configuradas por usuarios que no están familiarizados con detalles de **TCP/IP.**

De forma de poder manejar la operación de configuración de la red de una computadora de forma plug-and-play es que la **IETF** [IETF] desarrolló **DHCPv6** [RFC 3315].

Dynamic Host Configuration Protocol (DHCPv6) provee una forma de automatizar y manejar la configuración de red de una computadora y otros dispositivos que usan el protocolo **TCP/IP.** Por medio de **DHCPv6** el administrador de una red puede asignar direcciones, aplicar mascaras de subred y un router por defecto. **DHCPv6** está construido sobre una estructura cliente/servidor en la cual las computadoras (los clientes) contactan un servidor el cual provee los parámetros de configuración. El administrador provee al servidor la descripción de la infraestructura de la red junto a un conjunto de reglas de cómo asignar direcciones y otros parámetros de configuración.

En definitiva lo que **DHCPv6** permite hacer es construir una red en la cual los usuarios pueden añadir nuevas computadoras, sustituir computadoras existentes y mover las máquinas de lugar sin que el administrador o el propio usuario tengan que intervenir.

¿Por qué no basta con la autoconfiguración? La autonfiguración de las direcciones IPv6 permite como lo dice la palabra autoconfigurar una dirección IPv6 sin la necesidad de que ninguna persona y/o dispositivo externo intervenga pero con la posibilidad de recibir información básica (prefijos, tiempos relacionados a los estados de la dirección) desde un router. Pero esto no es suficiente si queremos formar una red en donde el dispositivo X tenga asignada la dirección Y, pertenezca al dominio Z y que consulte un servidor de DNS K, por ejemplo. Para esto es que existe la posiblidad de realizar una configuración por medio de DHCPv6 en donde se pueden asignar direcciones especificas a cada dispositivo, un dominio, servidores de DNS, NTP entre otras aplicaciones.

2.1.1.Del cuatro al seis

DHCPv4 [RFC2131] está basado en **BOOTP** [RFC951] de forma que mantiene el formato de los mensajes **BOOTP** y el funcionamiento de los relay agents de **BOOTP**, también comparte los puertos UDP que inicialmente fueron asignados a **BOOTP** (67 y 68). Esta compatibilidad con **BOOTP** permite que **DHCPv4** haga uso de los relay agents de **BOOTP** evitando así el requisito de un servidor **DHCPv4.**

DHCPv6 sólo está basado en la experiencia adquirida por el manejo de redes de quienes contribuyeron en su concepción, es un protocolo pensado y diseñado para cumplir con su objetivo, ayudar a configurar una red **IPv6**, ya que como se podrá apreciar a lo largo de este documento **DHCPv6** no es necesario para la configuración de la misma.

DHCPv4 es una forma de no tener que especificar manualmente cada una de las direcciones **IPv4** de una red mientras que **DHCPv6** es un protocolo que complementa a **IPv6 Stateless Address Autoconfiguration**, y puede ser usado separado o en conjunto con esta.

DHCPv4 y **DHCPv6** son independientes uno del otro, por ejemplo si se quieren configurar los nodos de una red dual se van a necesitar dos servidores DHCP, uno para cada protocolo.

En una red **IPv4** cada nodo debe de ser configurado de forma que sepa si debe o no hacer uso del servicio de **DHCPv4** mientras que en una red **IPv6** los mensajes **Router Advertisement** contienen opciones que indican al nodo si debe o no hacer uso del servicio **DHCPv6**.

Otra diferencia importante es como identificar a un nodo, **DHCPv6** usa identificador único denominado **DUID** (DHCP Unique Identifier), el cual no tiene equivalente en **DHCPv4**, mientras que este último usa la dirección **MAC** que si bien es usada en **DHCPv6** para llevar ciertas identificaciones de un nodo no es equivalente al **DUID**.

2.1.2.Configurando una dirección IPv6

Durante el proceso de configuración de una dirección, sin importar si es un proceso Stateless o Stateful, esta pasa por distintos estados, estos son:

- **Tentative** (**Tentativa**): Es cuando se está verificando la unicidad de la dirección, un nodo con una dirección en este estado no puede recibir mensajes unicast por esta dirección sin embargo puede procesar (recibir y enviar) mensajes muliticast del tipo **Neighbor Advertisement** que fueron enviados en repuesta a un mensaje **Neighbor Solicitation** enviado para detectar la duplicación de direcciones.

- **Valid** (**Válida**): Una dirección válida puede ser usada para enviar y recibir mensajes unicast. Además una dirección válida pude estar en dos sub-estados

    - **Preferred** (**Preferida**): Esto es cuando se comprobó la unicidad de la dirección y la misma puede ser usada para llevar cualquier tipo de comunicación sin ningún tipo de restricción.

    - **Deprecated** (**Desaprobada**): En este caso la dirección es válida y única pero no se recomienda su uso para establecer nuevas comunicaciones, las comunicaciones ya establecidas si pueden seguir haciendo uso de esta dirección.

    - Los tiempos que una dirección puede permanecer en cada uno de los sub-estados está determinado por el campo **Preferred Lifetime** que se encuentra dentro de la opción **Prefix Information** del mensaje **Router Advertisement**.

- **Invalid** (**Inválida**): La dirección ya no puede ser usada para ni para enviar ni recibir trafico unicast, una dirección entra en este estado luego de que el valid life time expira.

En la ilustración 1 se puede apreciar los estados por los cuales puede pasar una dirección **IPv6** durante su configuración.

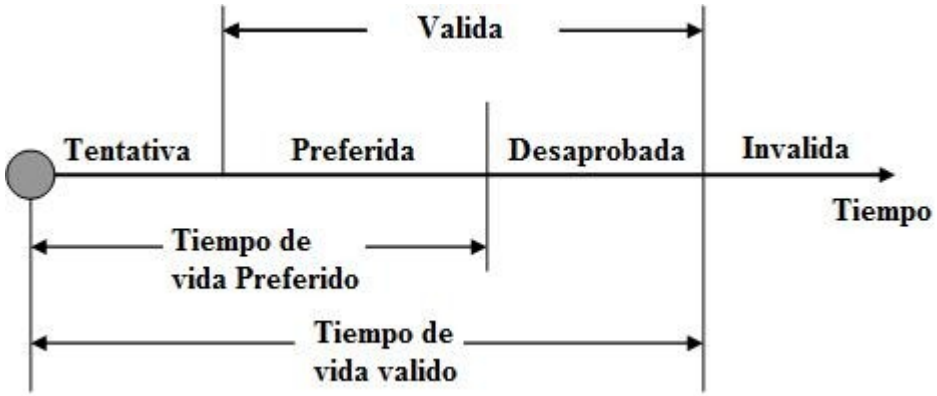

Ilustración 1: Estado de direcciones durante la configuración de las mismas

Si bien se muestra y explica cada uno de los estados por los cuales pasa una dirección que está siendo configurada aún no se ha mencionado como es que se llega a esto.

Todo el proceso inicia cuando se habilita la interfaz a partir de ahí se comienza a configurar la dirección link-local, a partir de esta configuración es que se envían mensajes del tipo **Neighbor Solicitation** y **Router Solicitation**. En respuesta a un **Router Solicitation** puede llegar o no un **Router Advertisement**, si no llega la configuración de direcciones termina con la configuración de la dirección link-local, pero este caso no es el que nos interesa ver a través de este documento.

Con la llegada del **Router Advertisement** pueden llegar o no prefijos con información de configuración, que según el valor que tenga la bandera **Autonomous Address-Configuration** deberá ser usada o no por la **Stateless Configuration**. Pero dentro de la información que nos llega dos banderas son de vital importancia a la hora de decidir qué tipo de configuración se debe de seleccionar para obtener las direcciones de la interfaz así como otras configuraciones necesarias. Estas banderas son la **Managed Address Configuration** (Flag M de ahora en más) y la **Other Stateful Configuration** (Flag O de ahora en más).

En caso de que ambas banderas tomen el valor cero entonces sólo se realizará la **Stateless Configuration**, en caso de que ambas banderas tomen el valor uno entonces se realizara una configuración **Stateful DHCP** y en el caso de que la Flag M tome el valor cero y la Flag O tome el valor uno se deberá de realizar una **Stateless DHCP Configuration**. Como se puede apreciar está quedando por fuera una posible combinación, Flag M con valor cero y Flag O con valor uno, esta combinación de valores no es válida.

En el proyecto IP4JVM esta configuración da inicio cuando se habilita una interfaz, lo cual se hace invocando al método ***enableInterface*** de la clase **IPv6IfaceParameters**, esto inicia el proceso **IPv6 Stateless Address Autoconfiguration.** Este proceso inicia configurando la dirección link-local para lo cual crea la clase **DupAddressTimer** la cual se encarga de enviar los mensajes **Neighbor Solicitation**, mediante la clase **ICMPv6NeighborSolicitation**, para poder comprobar la unicidad de la dirección link-local y poder hacer uso de la misma. Luego de lograr comprobar que la dirección no está duplicada entonces la clase **IPv6IfaceParameters** desde el método ***enableInterface*** realiza el envío de un **Router Solicitation,** lo cual logra hacer mediante la creación de la clase **RouterSolicitSender** la cual a su vez usa la clase **ICMPv6RouterSolicitation**.

Como ya se mencionó el envío de los **Router Solicitation** puede generar la recepción de **Router Advertisement** que dentro del procesamiento del stack de IP4JVM termina siendo procesado por la clase **NeighborDiscovery**. Dentro de la misma el método ***rcvICMPv6RouterAdvertisement*** es quien procesa el paquete recibido y es en donde se deberá modificar el proyecto IP4JVM para que pueda realizar la configuración **DHCPv6.**

2.1.3.Configuración DHCPv6

Como ya se mencionó anteriormente en este documento existen dos posibles configuraciones de **DHCPv6**

- **Stateful DHCPv6**

Esta configuración se inicia a partir de que en el procesamiento del **Router Advertisement** se advierte que los valores de las banderas O y M están seteados en uno. A partir de ahí se envía un mensaje de tipo **Solicit** a una dirección multicast quedando a la espera de que uno o varios servidores **DHCPv6** respondan mediante la recepción de mensajes del tipo **Advertise**.

Se selecciona uno de los servidores **DHCPv6** que hayan respondido y se le envía un mensaje de tipo **Request** al servidor seleccionado y se espera la confirmación del mismo mediante la recepción de mensajes del tipo **Reply**.

Luego del procesamiento de la información recibida en este mensaje se puede decir que la configuración **DHCPv6** ha terminado, pero por supuesto las direcciones que fueron asignadas mediante este medio no serán del dispositivo de por vida con lo cual a partir de que se termina la configuración comienza un ciclo en donde se utilizan estas direcciones mientras es posible para luego renovar su validez para poder usarlas nuevamente y así hasta que la interfaz es apagada o que no se puede llegar a renovar la dirección.

Este proceso se realiza mediante el envío de mensajes de tipo **Renew** por parte de los clientes, los cuales son respondidos por mensajes del tipo **Reply** por

los servidores. En caso de que estos no respondan luego de un tiempo configurado se pasa a realizar el envío por parte de los clientes de mensajes de **Rebind** los cuales también deberán de ser respondidos por los servidores con mensajes de tipo **Reply**.

Si todas las direcciones estuvieran a la espera de un **Reply** de un mensaje **Rebind** entonces se pueden tomar dos acciones, o decir que la configuración **DHCPv6** fallo o bien pasar a realizar todo el proceso de configuración desde el inicio.

En caso de no recibir respuestas a los mensajes de tipo **Solicit** o **Request** después de un determinado tiempo el proceso de configuración falla.

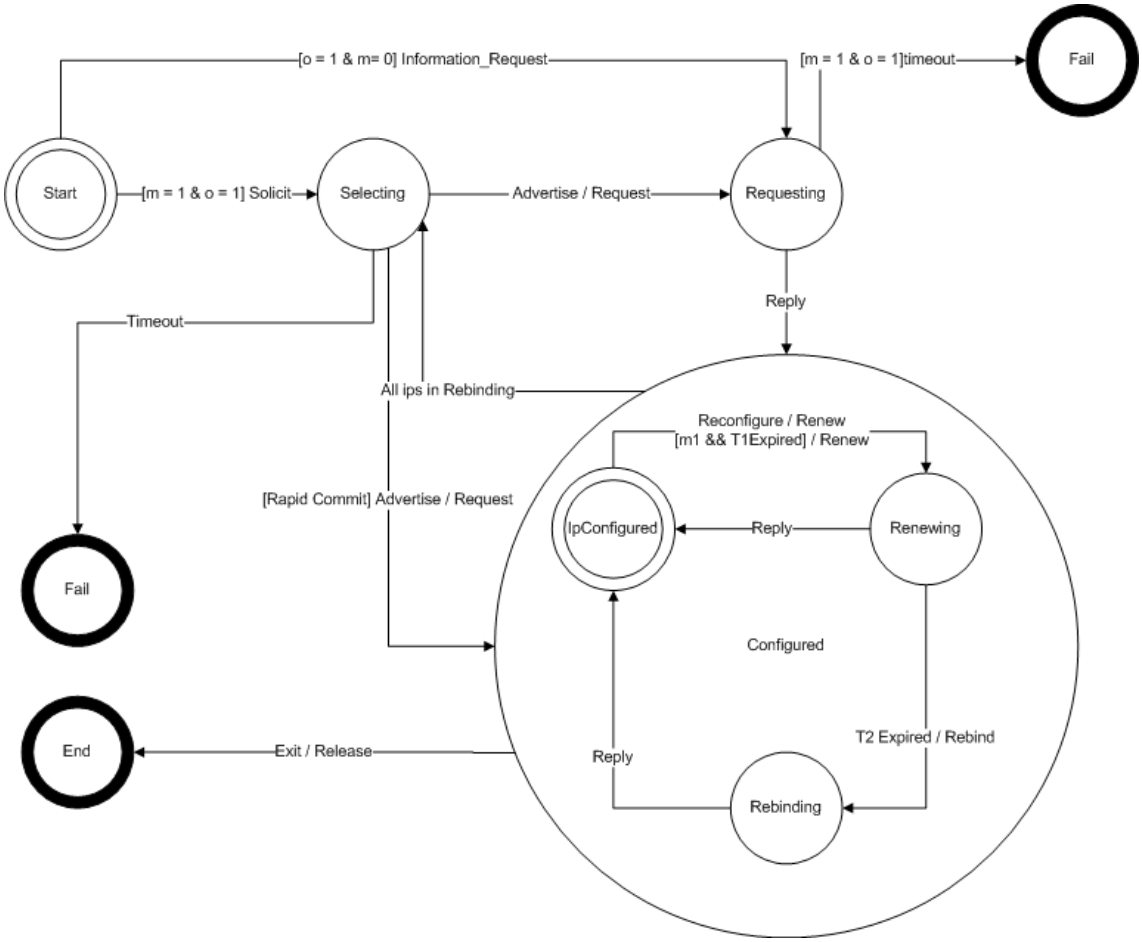

Ilustración 2: Estados de una configuración DHCPv6

- **Stateless DHCPv6**

Esta configuración **DHCPv6** al igual que la anterior se inicia a partir del procesamiento de un mensaje de tipo **Router Advertisement**, pero en este caso la bandera M toma el valor cero mientras que la bandera O toma el valor uno. A

partir de ahí se envía un mensaje de tipo **Information Request** y se queda a la espera de la respuesta por parte del servidor, la cual se da en forma de un menaje de tipo **Reply**. A partir de ahí se maneja la información de configuración recibida en el **Reply**.

En la ilustración 2 se pueden apreciar los estados por los cuales pasa una configuración **DHCPv6** y cómo es que puede ir de un estado a otro.

Como ya se mencionó para poder realizar todo lo descripto en este punto se deben realizar modificaciones sobre el método ***rcvICMPv6RouterAdvertisement*** de la clase **NeighborDiscovery.**

2.1.4.Formato de los mensajes

En este punto se presentan los formatos de los distintos mensajes a ser procesados y/o creados durante una configuración **DHCPv6**

- **Router Advertisement**

Estos mensajes son enviados periódicamente por los Routers en respuesta de un **Router Solicitation.**

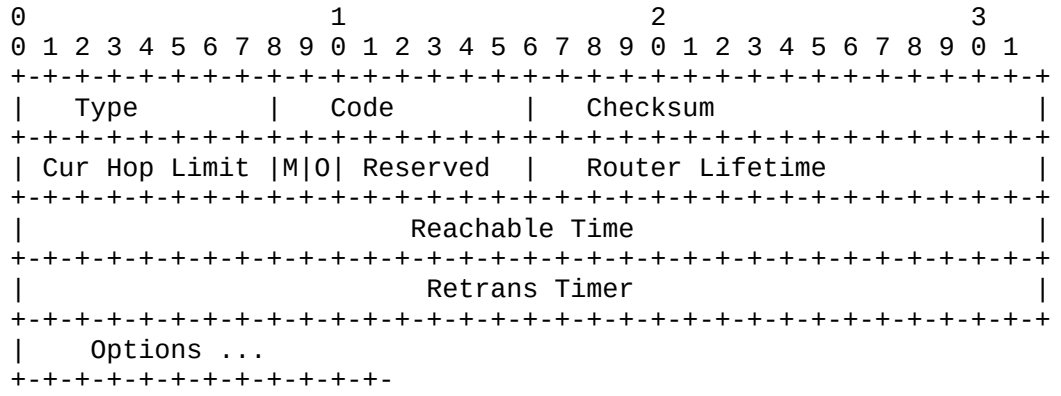

Ilustración 3: Formato de los mensajes

Campos IP:

- **Source Address**: Debe de ser una dirección link local asignada a la interfaz desde donde el mensaje es enviado.
- **Destination Address**: Típicamente la dirección IPv6 (Source Address) del router que ha recibido el Router Solicitation, también puede ser la dirección multicast correspondiente a todos los nodos de una red.
- **Hop Limit**: 255

Campos ICMP:

- **Type:** 134
- **Code:** 0
- **Checksum:** El checksum correspondiente a ICMP (Ver RFC 4443)
- **Cur Hop Limit:** entero sin signo de 8 bits. Corresponde al valor por defecto que debería de ser colocado en el campo **Hop Count** de un cabezal **IPv6** perteneciente a un paquete que va ser enviado. Si el valor es cero quiere decir que no está especificado (por el router que emitió el mensaje).
- **M:** Bandera de 1 bit, **Managed address configuration.** Cuando esta seteada se está indicando que las direcciones están disponibles por medio de **DHCPv6**
- **O: Bandera** de 1 bit, **Other Configuration.** Cuando esta seteada se está indicando que hay configuración disponible por medio de **DHCPv6.** Un ejemplo de esa información puede ser la información relacionada a DNS o bien información de otros servidores en la red.

Nota: Si ninguna de las dos banderas están seteadas, eso indica que no existe información disponible por medio de **DHCPv6.** En caso de que la bandera M

este seteada y no así la bandera O es tratado como que ninguna de las dos banderas lo estén.

- **Reserved:** campo de 6 bit que no es usado. Debe de ser inicializado en cero por quien envía y debe de ser ignorado por quien recibe.
- **Router Lifetime:** entero sin signo de 16 bit. Corresponde al tiempo de vida asociado el router por defecto, expresado en segundos. El campo puede contener valores hasta 65535 y quienes reciben el paquete deberían de poder manejar cualquier valor aunque las reglas de envío especificadas en el RFC 4861 pongan el límite de este campo en 9000 segundos. Un cero indica que el router no es un router por defecto que no debe de aparecer en lista de routers por defecto. La usabilidad de este valor sólo corresponde al router y no a la información contenida en otros campos o opciones. Las opciones que necesitan límites de tiempo incluyen sus propios campos de tiempo de vida.
- **Reachable Time:** entero sin signo de 32 bits. El tiempo en milisegundos en el cual el nodo asume que un vecino es alcanzado luego de haber recibido un confirmación de alcance. Este valor es usado por el algoritmo **Neighbor Unreachability Detection,** ver sección 7.3 de RFC 4861. Si el valor esta en cero quiere decir que esta sin especificar por el router.
- **Retrans Timer:** entero sin signo de 32 bits. El tiempo en milisegundos que debe de haber entre la retransmisión de dos mensajes **Neighbor Solicitation.** Es usado para la resolución de direcciones y por el algoritmo **Neighbor Unreachability Detection,** ver secciones 7.2 y 7.3 de RFC 4861.
- **Posibles Opciones**:
  - **Source link-layer address:** Es la dirección de capa de enlace de la interfaz desde la cual el **Router Advertisement** fue enviado. Sólo es usada en capas de enlace que tienen dirección. Un router puede omitir esta opción para poder compartir la carga de entrada entre varias direcciones de capa de enlace.
  - **MTU:** Debe de ser enviado en enlaces que tienen un MTU variable, aunque pueden ser enviado en otros tipos de enlaces.
  - **Prefix Information:** Estas opciones especifican los prefijos que están presentes en el enlace y/o que son utilizados en el protocolo **Stateless address autoconfiguration.** Un router debe de incluir todos los prefijos de enlace de forma que los host tengan información de los destinos posibles de cada uno de los links a los cuales están conectados. Si la información no está completa un host con múltiples interfaces quizás no pueda seleccionar la interfaz correcta cuando está enviando trafico a sus vecinos.

- **Mensajes Cliente/Servidor de DHCPv6**

El siguiente esquema ilustra el formato de los mensajes **DHCPv6** que son enviados entre clientes y servidores.

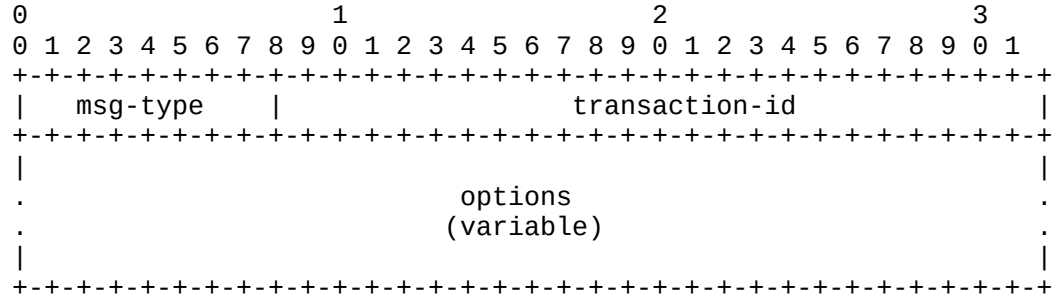

Ilustración 4: Mensajes Cliente/Servidor de DHCPv6

- **msg-type**: Identifica el tipo de mensaje **DHCPv6**
- **transaction-id**: El id de transacción involucrado en un intercambio de mensajes.
- **options**: opciones contenidas en el mensaje.

2.1.5.Mensajes DHCPv6

En este punto se explica cómo es que está formado cada uno de los mensajes involucrados en una comunicación Cliente/Servidor para realizar la configuración **DHCPv6**.

- **Solicit**

El cliente debe de setear el campo msg-type con el valor correspondiente al **Solicit** y generar un id de transacción e insertarlo en el campo transaction-id.

El cliente debe de incluir la opción **Client identifier** de forma de identificarse ante el servidor. El cliente debe de incluir **opciones IA** para todas las IAs a las cuales desea que el servidor le asigne direcciones. El cliente puede incluir direcciones en las IAs de forma de indicarle al servidor cuáles son sus preferencias.

El cliente usa opciones **IA_NA** para pedir que se le asignen direcciones no temporales y opciones **IA_TA** para pedir que se le asignen direcciones temporales. Estas opciones pueden ser enviadas en el mismo mensaje de **Solicit** si el cliente lo necesita.

El cliente debe incluir una opción **Option Request** para indicar las opciones que está interesado en recibir desde el servidor **DHCPv6.** Opcionalmente el cliente puede incluir instancias de las opciones especificadas en la opción **Option Request** de forma de indicar al servidor cuáles son sus preferencias.

El cliente incluye la opción **Reconfigure Accept** si está dispuesto a aceptar mensajes de reconfiguración por parte del servidor.

- **Advertise**

El servidor debe de setear el campo msg-type con el valor correspondiente al **Advertise** y copiar el transaction-id recibido en el **Solicit**.

El servidor debe de colocar la opción **Server Identifier** con el valor de su server identifier y copiar la opción **Client Identifier** recibido en el mensaje de **Solicit.**

El servidor puede incluir la opción **Preference** de forma que esta contenga el valor de para el mensaje de **Advertise.** También puede incluir la opción **Reconfigure Accept** si el servidor desea que el cliente acepte mensajes **Reconfigure.**

El servidor incluye las opciones que le podría enviar en el mensaje de **Reply.** De esta forma el cliente puede seleccionar entre todos los servidores que hayan respondido el mensaje **Solicit.**

Si el cliente incluyo una o más opciones **IA** en el mensaje de **Solicit**, entonces el servidor debe de incluir las opciones **IA** con las direcciones que podrían llegar a ser configuradas por el servidor. Si el servidor no va a asignar ninguna dirección a ninguna de las IAs entonces el cliente debe incluir únicamente una opción **Status Code** con el código ***NoAddresAvail***.

- **Request**

El cliente debe de setear el campo msg-type con el valor correspondiente al **Request** y generar un id de transacción e insertarlo en el campo transaction-id.

También debe de colocar la opción **Server Identifier** con el valor del servidor destino, dicho valor fue recibido en el mensaje **Advertise.**

Así como también debe de incluir la opción **Client identifier** de forma de identificarse ante el servidor. El cliente debe de incluir **opciones IA** para todas las IAs a las cuales desea que el servidor le asigne direcciones. El cliente puede incluir direcciones en las IAs de forma de indicarle al servidor cuáles son sus preferencias.

El cliente debe de incluir una opción **Option Request** para indicar las opciones que está interesado en recibir desde el servidor **DHCPv6.** Opcionalmente el cliente puede incluir instancias de las opciones especificadas en la opción **Option Request** de forma de indicar al servidor cuáles son sus preferencias.

El cliente incluye la opción **Reconfigure Accept** si está dispuesto a aceptar mensajes de reconfiguración por parte del servidor.

- **Reply**

El servidor debe de setear el campo msg-type con el valor correspondiente al **Reply** y copiar el transaction-id recibido en el **Request**.

El servidor debe de colocar la opción **Server Identifier** con el valor de su server identifier y copiar la opción **Client Identifier** recibido en el mensaje de **Request**.

El contenido de las opciones de un mensaje **Reply** dependen de a que tipo de mensaje este respondiendo, **Request, Information Request, Renew, Rebind** o **Solicit** con **Rapid Commit.**

Sin importar a qué tipo de mensaje de origen al **Reply** este puede contener opciones para configurar distintos servicios de capa de aplicación como pueden ser DNS, NIS o NTP, estas opciones son las únicas que están disponibles para un **Reply** que se origina a partir de un **Information Request**.

Si el cliente incluyo una o más opciones **IA** en el mensaje inicial, entonces el servidor debe de incluir las opciones **IA** con las direcciones que el cliente debe de configurar.

- **Confirm**

El cliente debe de setear el campo msg-type con el valor correspondiente al **Confirm** y debe de generar e insertar el valor del transaction-id.

El cliente debe de incluir la opción **Client identifier** de forma de identificarse ante el servidor. También debe de incluir **opciones IA** para todas las IAs asignadas a la interfaz para las cuales el mensaje de **Confirm** va a ser enviado.

- **Renew**

El cliente debe de setear el campo msg-type con el valor correspondiente al **Renew** y debe de generar e insertar el valor del transaction-id.

El cliente debe de colocar la opción **Server Identifier** con el valor del servidor a partir del cual obtuvo la configuración que desea renovar.

El cliente debe de incluir la opción **Client identifier** de forma de identificarse ante el servidor. El cliente debe de incluir **opciones IA** para todas las IAs a las cuales desea que el servidor le renueve las direcciones.

El cliente debe de incluir una opción **Option Request** para indicar las opciones que está interesado en recibir desde el servidor **DHCPv6.** Opcionalmente el cliente puede incluir instancias de las opciones especificadas en la opción **Option Request** de forma de indicar al servidor cuáles son sus preferencias.

- **Rebind**

El cliente debe de setear el campo msg-type con el valor correspondiente al **Rebind** y debe de generar e insertar el valor del transaction-id.

El cliente debe de incluir la opción **Client identifier** de forma de identificarse ante el servidor. El cliente agrega las opciones apropiadas incluyendo una o más opciones **IA**. El cliente debe de incluir la lista de direcciones que tiene actualmente están asociadas con la IAs en el mensaje de **Rebind.**

El cliente debe de incluir una opción **Option Request** para indicar las opciones que está interesado en recibir desde el servidor **DHCPv6.** Opcionalmente el cliente puede incluir instancias de las opciones especificadas en la opción **Option Request** de forma de indicar al servidor cuáles son sus preferencias.

- **Information Request**

El cliente debe de setear el campo msg-type con el valor correspondiente al **Information Request** y debe de generar e insertar el valor del transaction-id.

El cliente debe de incluir la opción **Client identifier** de forma de identificarse ante el servidor.

El cliente debe de incluir una opción **Option Request** para indicar las opciones que está interesado en recibir desde el servidor **DHCPv6.** Opcionalmente el cliente puede incluir instancias de las opciones especificadas en la opción **Option Request** de forma de indicar al servidor cuáles son sus preferencias.

- **Release**

El cliente debe de setear el campo msg-type con el valor correspondiente al **Release** y debe de generar e insertar el valor del transaction-id.

El cliente debe de colocar la opción **Server Identifier** con el valor del servidor a partir del cual obtuvo la dirección que desea liberar

El cliente debe de incluir la opción **Client identifier** de forma de identificarse ante el servidor. El cliente incluye opciones conteniendo las IAs para las cuales está liberando direcciones**.** Las direcciones a ser liberadas deben de estar incluidas en las IAs.

- **Decline**

El cliente debe de setear el campo msg-type con el valor correspondiente al **Decline** y debe de generar e insertar el valor del transaction-id.

El cliente debe de colocar la opción **Server Identifier** con el valor del servidor a partir del cual obtuvo la dirección que desea no usar.

El cliente debe de incluir la opción **Client identifier** de forma de identificarse ante el servidor. El cliente incluye opciones conteniendo las IAs para las cuales no quiere usar las direcciones**.** Las direcciones a ser liberadas deben de estar incluidas en las IAs.

2.1.6. Opciones DHCPv6

En este punto se explica cómo están formadas las opciones usadas en los mensajes de **DHCPv6**, sólo se explican las opciones definidas como parte de la especificación base de DHCPv6, pueden existir otras opciones.

- **Formato General**

Todas las opciones comparten un formato general y este se describe en este punto.

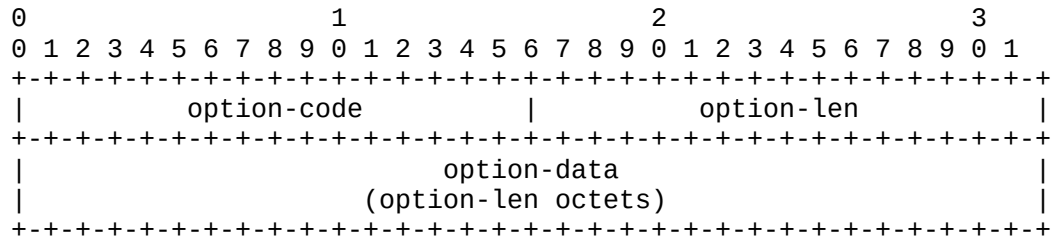

Ilustración 5: Formtaro general de las opciones DHCPv6

Campos:

- **option-code:** Entero sin signo que especifica el tipo de opción que se encuentra dentro de la opción.
- **option-len:** Entero sin signo que indica el largo, en octetos, del campo **option-data**
- **option-data:** La información de la opción, la forma de esta opción depende de la definición de la opción.

- **Client Identifier**

Esta opción es usada para contener el id del cliente el cual identifica al mismo entre un intercambio de mensajes Cliente/Servidor.

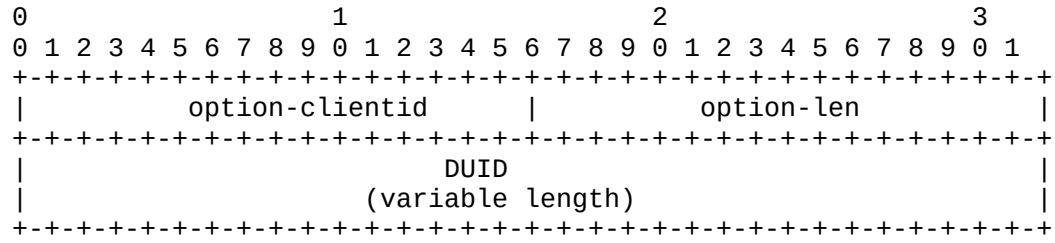

Ilustración 6: Formato del Client Indentifier

Campos:

- **option-clientid:** 1
- **option-len:** Largo del DUID, en octetos.
- **option-data:** El DUID del cliente.

- **Server Identifier**

Esta opción es usada para contener el id del servidor el cual identifica al mismo entre un intercambio de mensajes Cliente/Servidor.

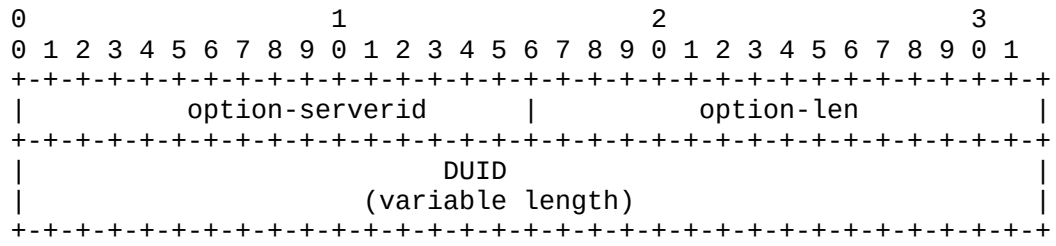

Ilustración 7: Formato del Server Identifier

Campos:

- **option-serverid:** 2
- **option-len:** Largo del DUID, en octetos.
- **option-data:** El DUID del servidor.

- **Identity Asociation for Non-temporary Addresses (IA_NA)**

Esta opción es usada para contener una IA_NA, los parámetros asociados con la IA_NA y las direcciones asociadas con la misma.

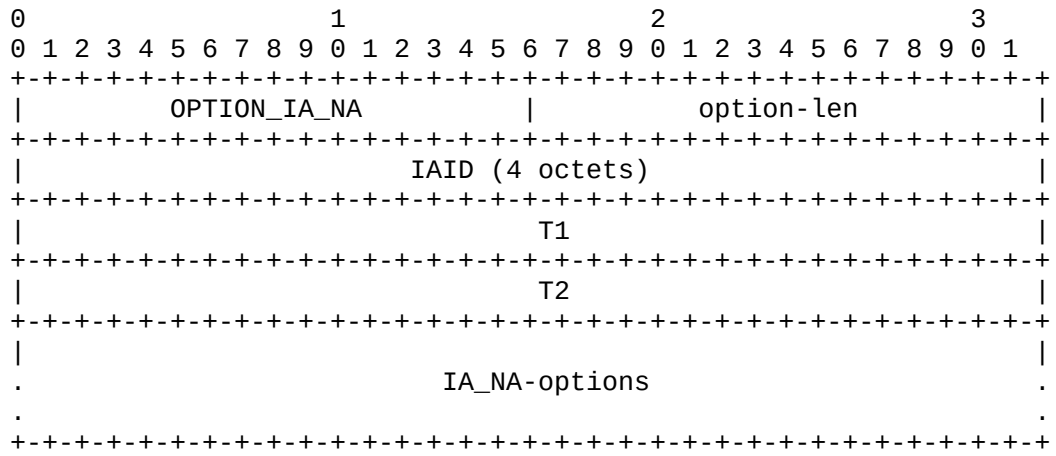

Ilustración 8: Formato de las IA_NA

Campos:

- **OPTION_IA_NA:** 3

- **option-len:** 12 + el largo del campo **IA_NA-options**.
- **IAID:** El identificador único para esta IA_NA, este debe de ser único entre todos los identificadores de las IA_NAs del cliente.
- **T1:** El tiempo en el cual el cliente debe de contactar al servidor desde el cual obtuvo las configuraciones para poder extender los tiempos de vida de las direcciones asignadas a la IA_NA, esta expresado en segundos y es relativo al momento actual.
- **T2:** El tiempo en el cual el cliente contacta cualquier servidor disponible para extender los tiempos de vida de las direcciones asignadas a la IA_NA, esta expresado en segundos y es relativo a el momento actual.
- **IA_NA-options:** Encapsula todas las opciones que son especificas para esta IA_NA, como por ejemplo todas las opciones **IA Address.**

- **Identity Asociation for temporary Addresses (IA_TA)**

Esta opción es usada para contener una IA_TA, los parámetros asociados con la IA_TA y las direcciones asociadas con la misma.

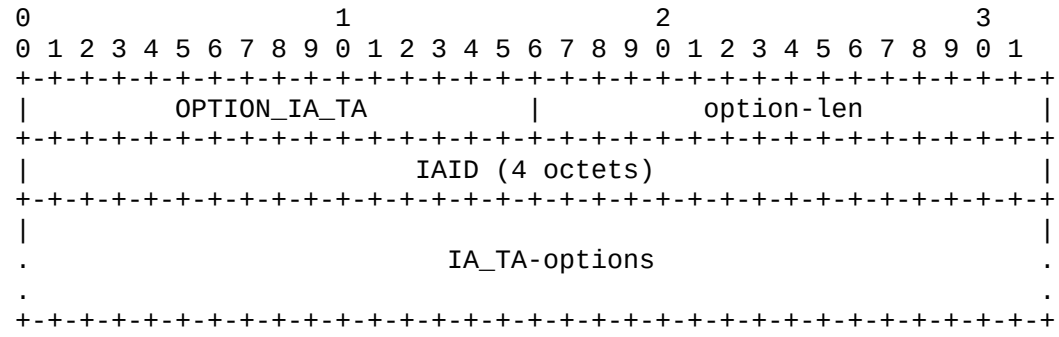

Ilustración 9: Formato de las IA_TA

Campos:

- **OPTION_IA_TA:** 4
- **option-len:** 4 + el largo del campo **IA_TA-options**.
- **IAID:** El identificador único para esta IA_NA, este debe de ser único entre todos los identificadores de las IA_NAs del cliente.
- **IA_TA-options:** Encapsula todas las opciones que son especificas para esta IA_TA, como por ejemplo todas las opciones **IA Address.**

- **IA Address**

Esta opción es usada para contener especificar direcciones **IPv6** asociadas con una IA_NA o una IA_TA.

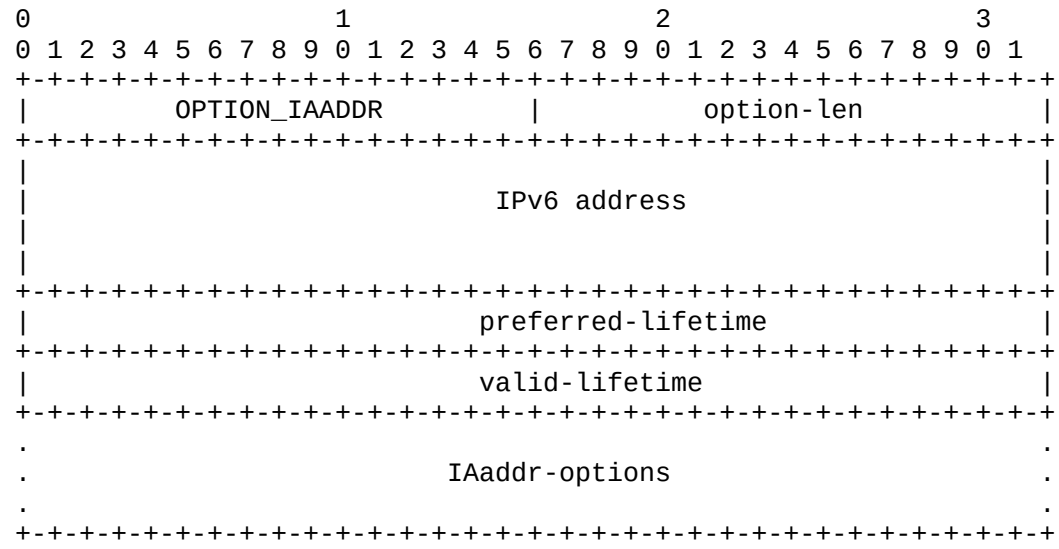

Ilustración 10: Formato de las IA Address

Campos:

- **OPTION_IAADDR:** 5
- **option-len:** 24 + el largo del campo **IAaddr-options**.
- **preferred-lifetime:** El preferred lifetime de la dirección **IPv6** en la opción, la unidad en la cual esta expresado es segundos.
- **valid-lifetime:** El valid lifetime de la dirección **IPv6** en la opción, la unidad en la cual esta expresado es segundos.
- **IAaddr-options:** Opciones asociadas con esta dirección.

- **Option Request**

Es usada para identificar una lista de opciones en un mensaje entre un Cliente y un Servidor.

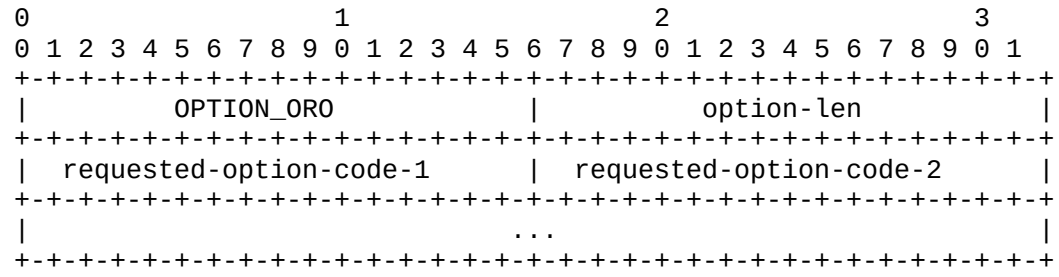

Ilustración 11: Formato del Option Request

Campos:

- **OPTION_ORO:** 6
- **option-len:** 2 * el número de opciones
- **requested-option-code-n:** El código de opción para una opción pedida por el cliente.

- **Preference**

Es usada por el servidor para afectar la selección del mismo por parte del cliente.

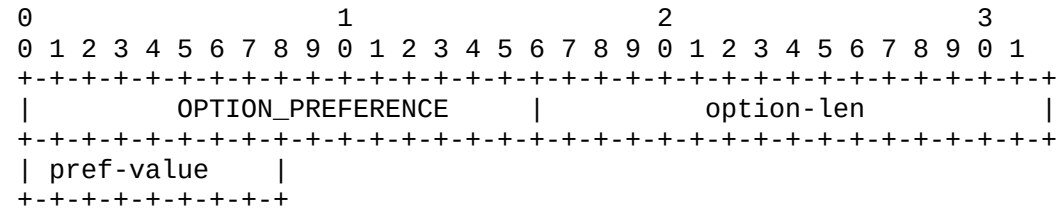

Ilustración 12: Formato del Preference

Campos:

- **OPTION_PREFERENCE:** 7
- **option-len:** 1
- **pref-value:** La preferencia del servidor.

- **Elapsed Time**

Es usada por el cliente para indicar el tiempo que ha transcurrido desde comenzó la configuración **DHCPv6**.

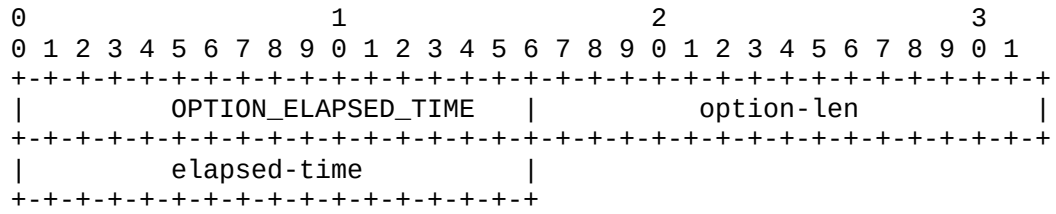

Ilustración 13: Formato Elapsed Time

Campos:

- **OPTION_ELAPSED_TIME:** 8
- **option-len:** 2
- **elapsed-time:** La cantidad de tiempo, expresada en milésimas de segundo, desde que el cliente inicio la configuración **DHCPv6**.

- **Server Unicast**

El servidor envía esta opción al cliente para indicarle que está permitida la comunicación con por medio de unicast con el servidor.

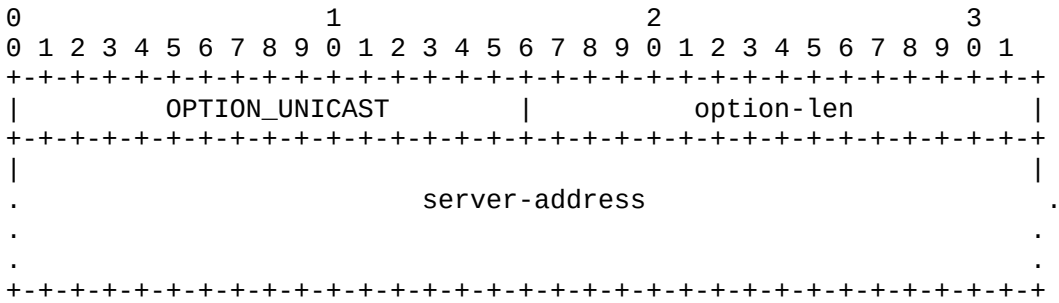

Ilustración 14: Formato Server Unicast

Campos:

- **OPTION_UNICAST:** 12
- **option-len:** 16
- **DHCP-relay-message:** La dirección a la cual el cliente debe de enviar los mensajes unicast para el servidor.

- **Status Code**

Esta opción contiene el estado relacionado al mensaje **DHCPv6**

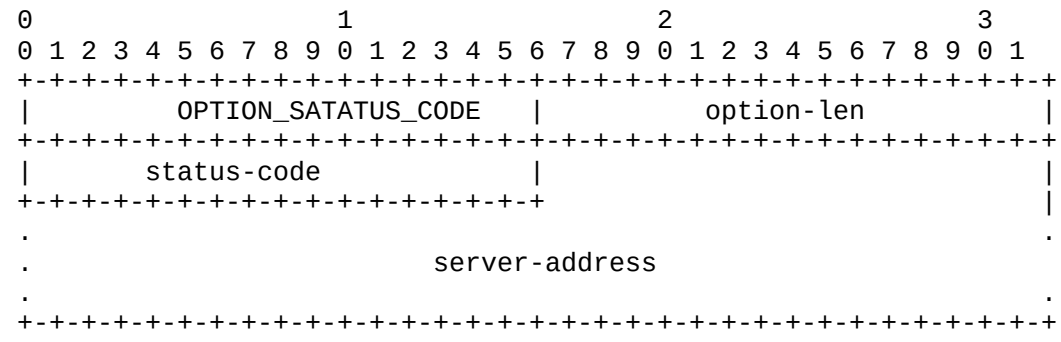

Ilustración 15: Formato Status Code

Campos:

- **OPTION_STATUS_CODE:** 13
- **option-len:** 2 + largo del status-message
- **status-code:** El código numérico para el estado.
- **status-message:** Texto en UTF-8 formateado de forma aceptable para ser desplegado a un usuario.

- **Rapid Commit**

Esta opción es usada para activar la asignación de dirección con sólo 2 mensajes.

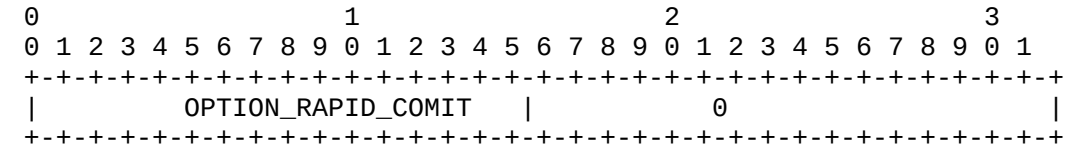

Ilustración 16: Formato Rapid Commit

Campos:

- **OPTION_RAPID_COMIT:** 14
- **option-len:** 0

- **Reconfigure Message**

Un servidor responde con esta opción para indicar al cliente que debe de reconfigurarse.

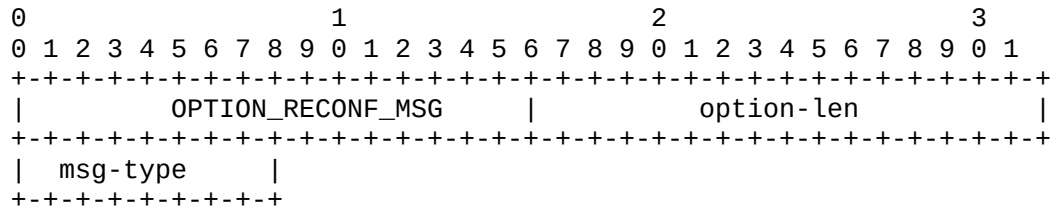

Ilustración 17: Formato Reconfigure Message

Campos:

- **OPTION_RECONF_MSG:** 19
- **option-len:** 1
- **status-code:** 5 para los mensajes de **Renew** y 11 para los mensajes de **Information Request.**

- **Reconfigure Accept**

Un cliente hace uso de esta opción para indicar que acepta mensajes de reconfiguración.

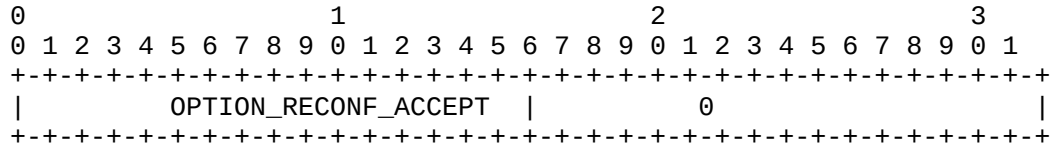

Ilustración 18: Formato Reconfigure Accept

Campos:

- **OPTION_RECONF_ACCEPT:** 20
- **option-len:** 0

2.2.Herramientas para testing en IPv6

Lo primero que se comenzó por estudiar y configurar fue el proyecto TAHI [TAHI]. Actualmente este proyecto está conformado por dos organizaciones, la universidad de Tokyo y el grupo Yokogawa Electric Group aunque inicialmente había una tercera organización, YDC Corp. Este proyecto tiene como objetivo el desarrollo y la verificación de la tecnología **IPv6** para lo cual investigan y desarrollan tests de conformidad e interoperabilidad.

Llevar adelante los tests que provee este proyecto implica realizar ciertas configuraciones y tener un buen entendimiento de que es lo que se quiere probar

ya que al correr el mencionado proyecto este puede ser o un nodo, o un router, o un servidor DHCPv6, o un relay agent, o sea es lo que uno necesita cuando desea probar su implementación **IPv6**.

Si bien el proyecto TAHI es muy útil para poder saber si una implementación **IPv6** cumple con una cierta calidad no es de mucha utilidad durante el desarrollo del sistema en sí. Algunas herramientas que si los son pueden ser las siguientes:

- VMWare Server [VARE], con esta herramienta se pueden crear maquinas virtuales en donde se pone en marcha nuestra aplicación sin el riesgo de dañar el sistema en donde estamos desarrollando. Además nos permite tener simultáneamente varios nodos sin la necesidad de tener varias maquinas físicas.

- Wireshark [WARK], con esta herramienta se pude capturar el trafico de una red, muy útil a la hora de corroborar si estamos enviando y recibiendo paquetes por la red y a su vez poder corroborar que los mismos están bien formados.

- radvd [RVD], demonio que juega el papel de un router IPv6. Disponible en Linux y de configuración muy sencilla. Envía los Router Advertisement de acuerdo a la especificación del RFC 2461, que si bien se encuentra obsoleto por el RFC 4861 al momento de la realización de este documento el formato de los mensajes no cambio de una especificación a la otra.

- Dibbler [DIBER], esta herramienta está disponible tanto para Windows como para Linux en sus versiones Cliente, Servidor y Relay Agent. En nuestro caso nos es muy útil para poder simular un servidor **DHCPv6** y fue seleccionada entre otras herramientas que cumplían las mismas funcionalidades debido a los resultados que obtuvo cuando se le corrieron los tests del proyecto TAHI.

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 IPv6, 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.

A partir de estos protocolos es que el proyecto brinda funcionalidades tales como el envió de paquetes **Router Solicitation** o **Neighbor Solicitation** o bien el poder procesar mensajes de **Router Advertisement** y **Neighbor Advertisement** los cuales permitían llevar a cabo el protocolo de **Stateless Autoconfiguration** y conforman la base para poder realizar las configuraciones de los nodos a partir de los protocolos **DHCPv6**.

2.4.Resumen

En esta etapa se realizó un estudio del proyecto y su funcionamiento. Se indagaron las funcionalidades que brindan las distintas herramientas con el fin de seleccionar las indicadas para poder forma un ambiente de trabajo que permitiera llevar adelante la implementación y testing de las nuevas funcionalidades a ser agregadas al proyecto IP4JVM.

Se realizó un estudio profundo del protocolo **DHCPv6** permitiéndonos saber que mensajes y que opciones del mismo se deben implementar para poder brindar el soporte **DHCPv6** al proyecto. También se comprendió la vinculación del protocolo **DHCPv6** con el de **Stateless Address Autoconfiguration** lo cual permitió aislar las modificaciones a realizar en el proyecto IP4JVM lo cual permitió trabajar con relativa comodidad con el grupo que aún se encontraba terminado el proyecto anterior.

3. Análisis de la solución

En este capítulo se presenta en forma genérica la solución implementada. Esta solución se presenta en función de sus componentes y se explican los principales puntos que componen su arquitectura y como estos se relacionan entre sí.

Se empieza por mencionar las distintas alternativas que se manejaron a la hora de encontrar una solución para después justificar y detallar la que se escogió y llevo adelante. De esta solución se presenta una descripción en alto nivel dando especial énfasis en los componentes de su arquitectura y su interacción.

3.1. Posibles Soluciones

Luego de culminado el estudio del estado del arte del proyecto IP4JVM, herramientas de testing y configuración **DHCPv6** se realizó un estudio de las distintas soluciones que resolvían el problema de incorporar la mencionada configuración al proyecto.

En primera instancia se estudio la posibilidad de realizar una implementación que reutilizara algunas de las clases con las que ya contaba el proyecto, con esto se buscaba la reutilización de código, no modificar la configuración actual del proyecto y no comenzar a distribuir el funcionamiento del mismo en distintos compontes.

Cuando se intentó comenzar la diagramación de esta solución se pudo constatar que la reutilización de código era escasa, que modificar la configuración actual del proyecto no era una tarea complicada de realizar y que cambiar dicha configuración no implicaría cambios para quienes hicieran uso del proyecto. Pero lo más importante que impulso la búsqueda de una solución alternativa fue detectar que la implementación del proyecto es una combinación de una programación orientada a objetos con una netamente orientada a redes, lo cual iba en contra de uno de los objetivos planteados. "*Obtener un diseño orientado a objetos que permita una sencilla incorporación de los protocolos que se configuran a partir de DHCPv6, como puede ser DNS*".

A continuación se puso el esfuerzo en encontrar una solución que permitiera cumplir con todos los objetivos planteados arribando a la solución que se presenta en los siguientes puntos de este capítulo.

3.2. Solución Implementada

En este punto se presenta la arquitectura que surge a partir de lo presentado en el punto anterior y del estudio del arte que se presento en el capítulo 2.

3.2.1. Descripción

La solución está compuesta por tres componentes, dos de los cuales interactúan entre si y un tercero que contiene los puntos en común entre los dos primeros. Estos componentes son el proyecto IP4JVM el cual se encarga de llevar adelante todo lo mencionado en su estudio del arte y de apoyarse en el componente **DHCPv6** para poder llevar a cabo las configuraciones **DHCPv6** cuando estas sean necesarias. Por último se cuenta con un componente COMMON el cual contiene los puntos mediante los cuales los dos primeros proyectos realizan sus comunicaciones.

3.2.2. Componente IP4JVM

Como ya se ha mencionado este componente es quien se encarga de intercambiar mensajes con vecinos y routers. Es quien mantiene los estados de las distintas interfaces configuradas así como las direcciones IP que estas poseen. Pero incorpora llamadas al componente **DHCPv6** en caso de ser necesaria una configuración **DCHPv6** ya sea Stateless o Stateful, dar aviso al mencionado componente en caso de que una interfaz deje de estar disponible o bien si alguna de las IPs que esta tiene configurada fuera dada de baja. Por último realizara la implementación de una interfaz mediante la cual el componente **DHCPv6** le advierte de las configuraciones realizadas por el mencionado protocolo.

3.2.3. Componente DHCP

Este componente realiza una implementación orientada a objetos del protocolo DHCPv6 haciendo uso de patrones de diseño como Template Method y State.

La comunicación con es este proyecto se da por medio de dos interfaces una para recibir peticiones de configuración de interfaces y otra mediante la cual da aviso del resultado de dichas configuraciones.

Este componente es quien lleva a cabo los distintos pasos de la configuraciones **DHCPv6** y quien mantendrá el estado de dichas configuraciones en el tiempo hasta que quien las solicito de por terminada su labor.

Otra característica importante a destacar de este componente es que si bien realiza la comunicación con la capa de red por medio del componente IP4JVM que es quien presta el servicio de comunicación con la mencionada capa, esto lo hace de forma totalmente transparente lo cual permite que este componente pueda ser utilizado por cualquier otro componente que brinde los mismos servicios que el componente IP4JVM y respete las interfaces de comunicación.

3.2.4.Componente COMMOM

Este componente es el punto de comunicación entre los dos compontes especificados previamente en este mismo capítulo. Es quien contiene las definiciones de las interfaces de comunicación que usaran los proyectos IP4JVM y **DHCPv6** para comunicarse entre si.

3.2.5.Diagramas de la solución

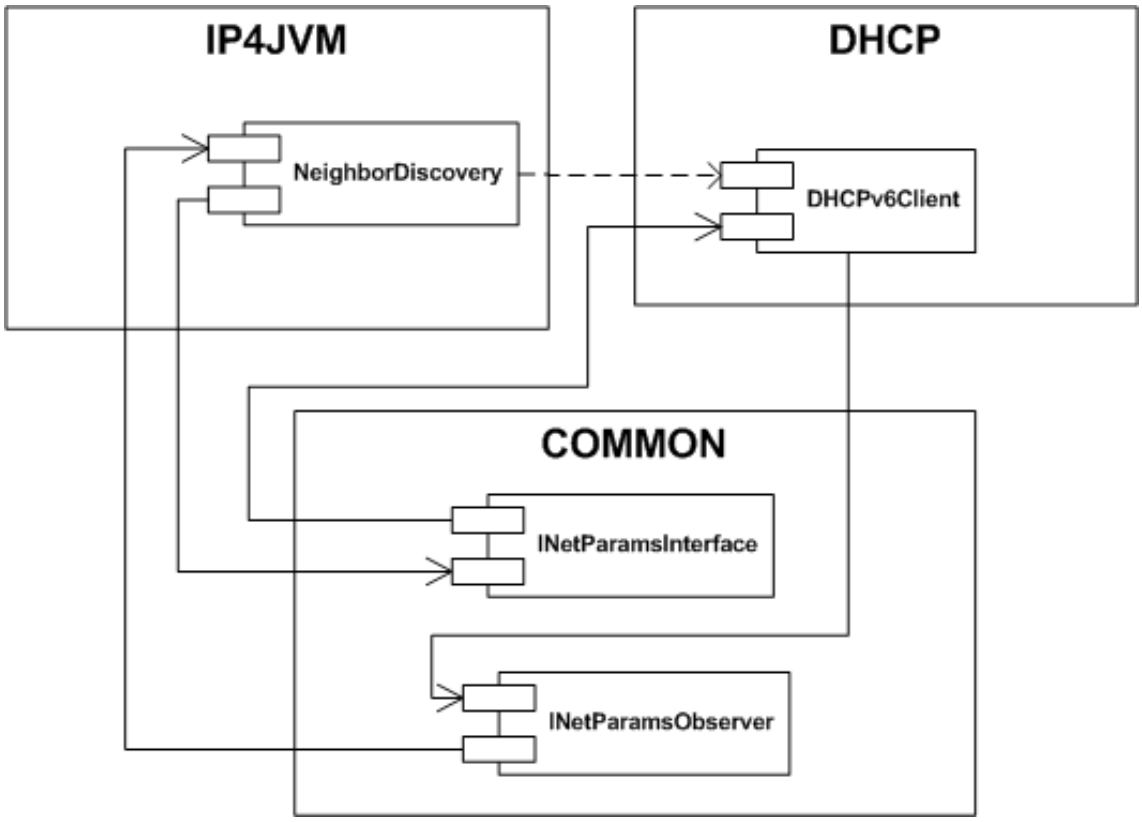

Ilustración 19: Diagrama de la solución

La ilustración 19 muestra el diagrama de los componentes que se describieron en los puntos previos.

La ilustración 20 muestra la implementación de los distintos estados por los que puede pasar la configuración **DHCPv6.**

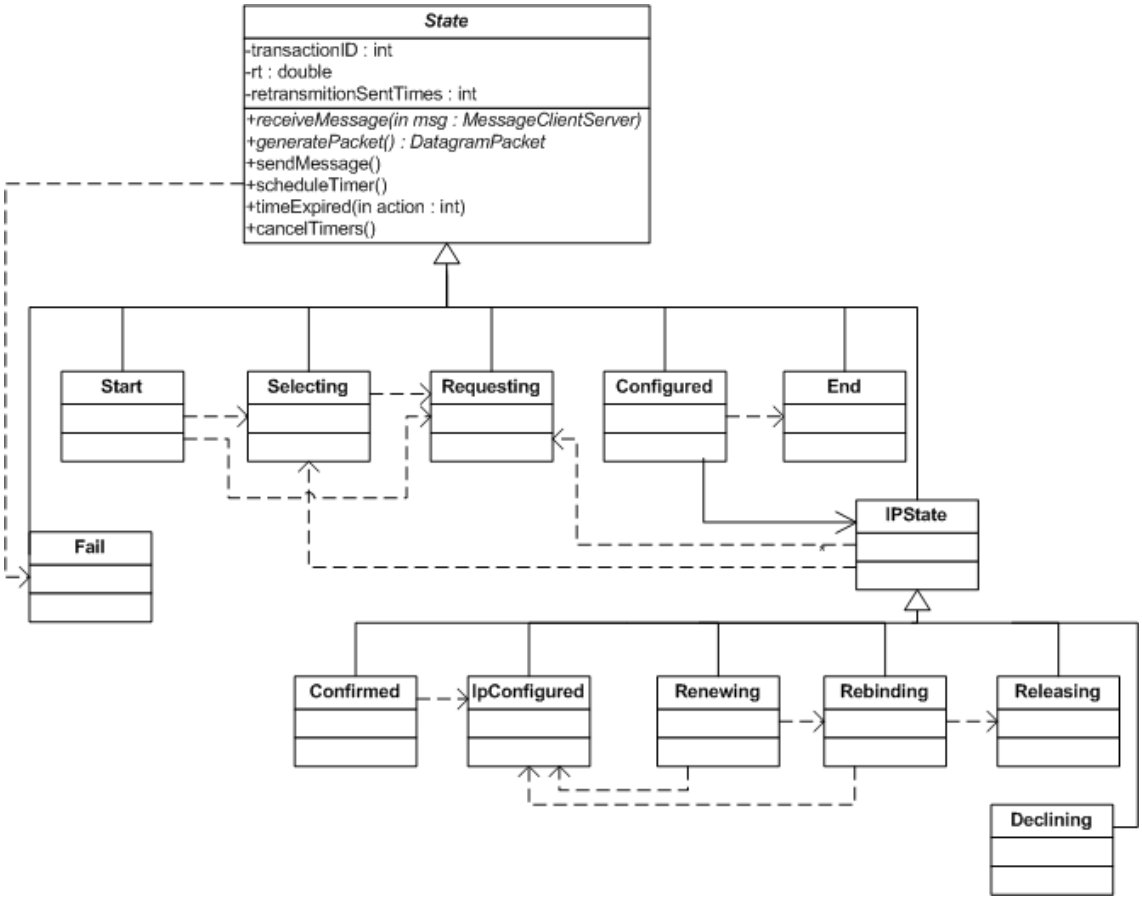

Ilustración 20: Diagrama de clases de los estados de la configuración de DHCPv6

3.2.6.Resumen

Se obtuvo una solución con una implementación orientada a objetos que permite una fácil extensión de sus funcionalidades. Cualquier desarrollador con conocimientos de redes y programación orientada a objetos podría realizar mejoras sobre la solución sin la necesidad de modificar el proyecto IP4JVM e incluso sin tener ningún tipo de conocimiento del mismo.

Debido a que la comunicación del componente DHCP con la capa de red se da por medio de una interfaz con el componente IP4JVM es que el primero puede ser usado por cualquier otro componente que desee realizar una configuración **DHCPv6** y brinde un soporte de comunicación con la capa de redes.

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 **DHCPv6**

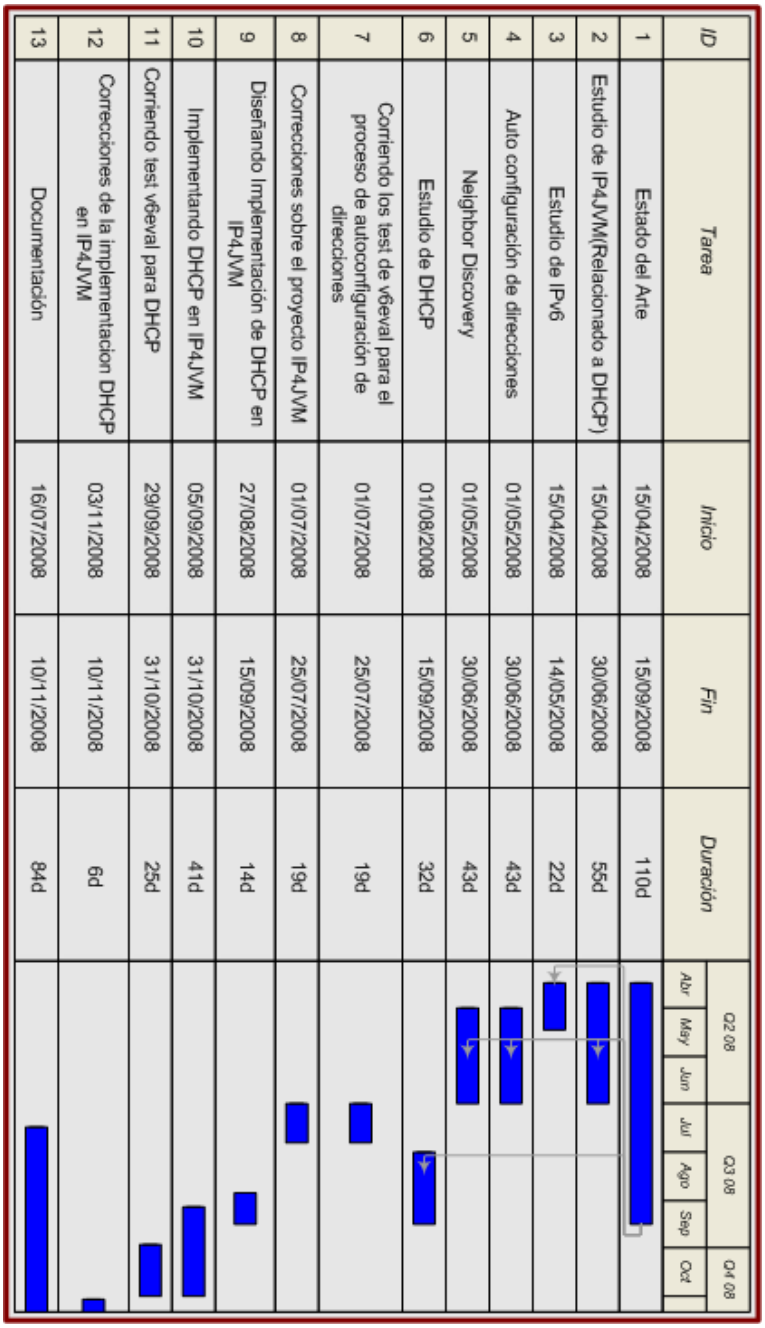

Ilustración 21: Gantt Tiempos Insumidos

El primer aspecto que debe mencionarse es que la implementación se dividió en cinco grandes tareas distintas:

•estudio del estado del arte,

•diseño de la solución,

•implementación de la solución,

•testing y correcciones y

•documentación

Como se puede ver en la tabla de tiempos estimados e insumidos, excluyendo la etapa de diseño todas las demás etapas tuvieron una desviación importante de lo inicialmente planificado. Al desglosar la tarea de Estudio del Arte en las subtareas, se aprecia que la mayor variación se vio en el estudio de IP4JVM el cual insumió bastante más tiempo del estimado, esto determinó que todas las tareas se fueran empezando más tarde o bien se fueran extendiendo en el tiempo mientras se las realizaba en paralelo con estudio de IP4JVM. El tiempo por el estudio de IP4JVM se debe a que la curva de aprendizaje del mismo, la cual en un principio requiere de muchas horas de estudio y muy poco avance.

Los tiempos de desviación correspondientes a la etapa de implementación se deben a una mala estimación inicial que se basó en que el reuso de código iba a ser mucho mayor a lo que realmente fue.

Durante la etapa de testing también se tuvo una desviación, en este caso el problema que se presentó fue que algunos de los tests que encontraron errores en la implementación estaban relacionados a la expiración de las configuraciones obtenidas mediante **DHCPv6**, estos tests tenían una duración de entre 10 y 20 minutos con lo cual realizar los mismos insumía bastante tiempo ya que requieren de la interacción de un usuario.

Tarea	Tiempo Estimado	Tiempo Insumido
Estudio del arte	13 semanas	**23 semanas**
Estudio de IP4JVM (Relacionado a DHCPv6)	2 semanas	**8 semanas**
Estudio de IPv6	3 semanas	3 semanas
Estudio de Autoconfiguración de direcciones	3 semanas	**6 semanas**
Estudio de Neighbor Discovery	3 semanas	**6 semanas**
Estudio de DHCPv6	4 semanas	**4,5 semanas**
Estudio de v6eval	1 semana	**2,5 semanas**
Diseño de la solución	2 semanas	2 semanas
Implementación de la solución	5 semanas	**7 semanas**
Implementación	4 semanas	**6 semanas**
Corrección de errores	1 semana	1 semana
Testing	2 semanas	**3,5 semanas**
Tests Básicos de e6eval (autoconfiguración de direcciones)	0,5 semanas	0,5 semanas
Tests v6eval para DHCPv6	1,5 semanas	**3 semanas**
Documentación	2 semanas	**3,5 semanas**
Total	20 semanas	**27 semanas**

Ilustración 22: Tabla de tiempos estimados e insumidos

5.Resultados y conclusiones

En este capitulo se presentan los resultados, limitaciones, mejoras y conclusiones a las cuales se llegaron luego de culminada esta etapa del proyecto.

5.1.Resultados

Se lograron cumplir con todos los objetivos planteados al inicio del proyecto luego de realizar un estudio del arte, evaluar distintas alternativas y por ultimo desarrollar una de estas.

Se realizo un estudio del arte de los pasos, metodologías y procesos a seguir para lograr la configuración de un nodo mediante el protocolo **DHCPv6** así como de las herramientas disponibles para poder crear un ambiente de trabajo. El resultado de este estudio fue la generación de ideas que llevaron a encontrar la solución implementada así como la selección de las herramientas que permitieron realizar los tests sobre la solución.

Luego de evaluar distintas soluciones se arribo a una que permite incorporar protocolos nuevos, vinculados a **DHCPv6**, de forma sencilla. Esto se debe a un diseño orientado a objetos el cual posee un bajo acoplamiento con el proyecto IP4JVM. Además esto permite poder llegar a hacer uso del componente implementado por otras implemtaciones del stack IPv6.

Con respecto al último objetivo planteado,realizar tests de conformidad e interoperabilidad, se obtuvieron muy buenos resultados en los tests realizados, ver **Anexo VI – Testing en IP4JVM**. Para poder llegar a la conclusión de que los resultados eran aceptables se compararon los mismos contra los resultados de otras herramientas disponibles que permiten realizar la configuración **DHCPv6**.

5.2.Limitaciones

Si bien la configuración DHCPv6 se puede realizar en todas sus formas, no todos los parámetros que pueden recibirse (pedírsele) desde(a) esta están soportados. Un ejemplo de esto es el DNS.

Otro ejemplo a considerar es la opción de delegación de prefijo [RFC3633], que permite a un router solicitar a un servidor **DHCPv6** un prefijo de direcciones a utilizar, para que luego el router, mediante un **Router Advertisement** o mediante **DHCPv6**, las asigne a los hosts de su propia red. Este es uno de los posibles modelos que pueden implementarse por parte de los ISPs para brindar conexión a Internet mediante el protocolo IPv6 a los suscriptores de DSL.

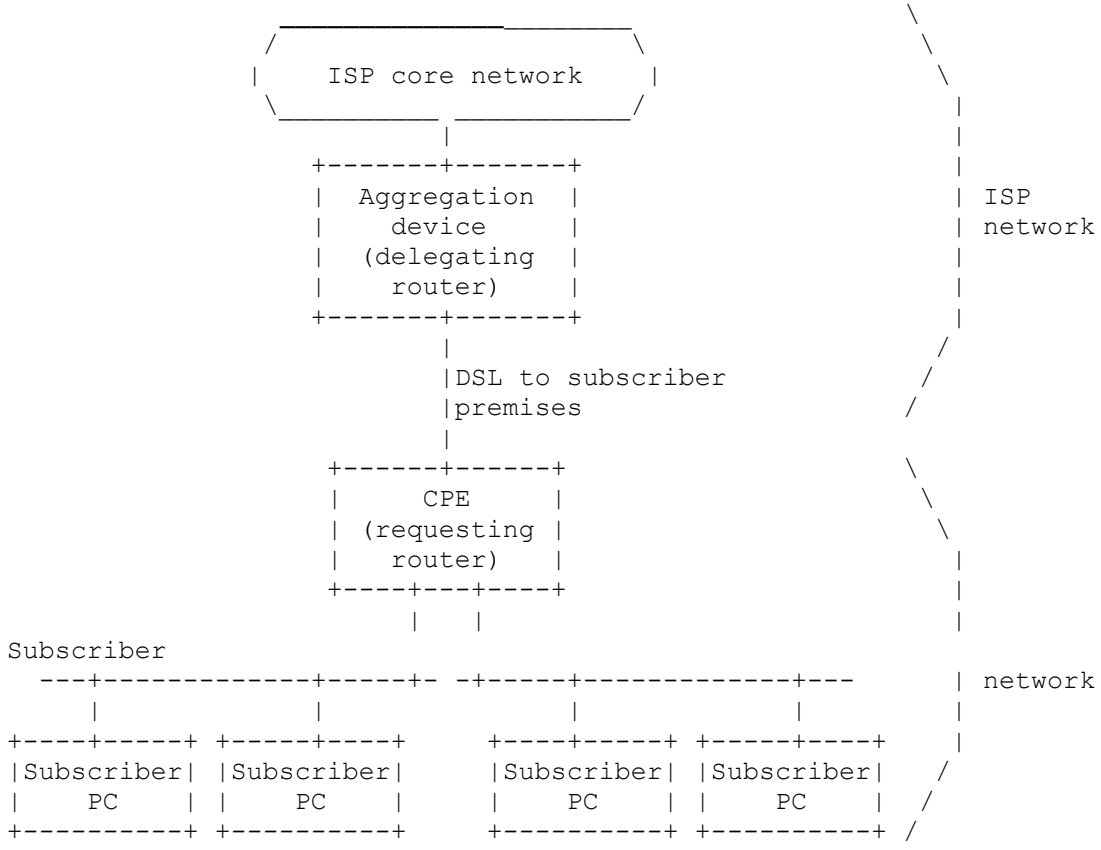

Ilustración 23: Ejemplo de deploy de DHCPv6 Prefix Discovery

Del esquema de la ilustración anterior, el Aggregation Device o Delegating Router, corresponde al router del ISP, mientras el CPE o requesting router corresponde al router del suscriptor de DSL. De está manera al conectarse a Internet el suscriptor el ISP le otorga un prefijo de direcciones para configurar sus PCs, tras lo cual el CPE configura las PCs de los suscriptores, esta configuración se puede realizar mediante Router Advertisement o DHCPv6. Luego si el ISP lo considera podría realizar un **reconfigure** para cambiar el prefijo de direcciones otorgadas al suscriptor y el CPE reconfigure también las direcciones de las PCs suscriptoras.

5.3. Trabajo futuro y posibles mejoras

Soporte para el todos los parámetros/información recibida en una configuración DHCPv6 como puede ser los parámetros referidos a la configuración de DNS. Otra posible implementación podria ser la de realizar la implementación de un servidor de DNS que interactue con el servidor DHCP.

6.Índice de ilustraciones