SERVICIOS INTERNET IPv6 PARA EL NODO DE LA UNIVERSIDAD DEL CAUCA EN EL 6BONE

Leidy Eliana Vivas Alzate
Fernando Andres Pérez Portilla

Director:
Ing. francisco Javier Terán

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICAIONES
DEPARTAMENTO DE TELECOMUNICACIONES
POPAYÁN
2006

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

INTRODUCCIÓN.

La tendencia tecnológica vislumbra un cambio obligado hacia una plataforma IPv6, debido entre otras razones a la masificación de Internet como una herramienta cotidiana, el surgimiento de nuevas tecnologías y aplicaciones que requieren un mayor rendimiento del sistema, como por ejemplo la convergencia de voz, video y datos en una sola infraestructura basada en el protocolo IP, la necesidad de mecanismos que provean seguridad a la información que circula por la red y el reto que supone la movilidad de los usuarios dado el gran auge de las comunicaciones móviles.

Como respuesta a dicha tendencia, en la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca, se desarrollaron trabajos en torno ha esta temática materializados en la implementación de un nodo IPv6 que utiliza como mecanismo de interconexión un túnel con la Universidad Nacional Autónoma de México, la cual se encuentra directamente conectada con la red de producción IPv6, lanzada el 6 de junio del 2006. Surge entonces, en la línea de evolución de la apropiación de esta tecnología, la imperiosa necesidad de capacitar al nodo IPv6 de la Universidad del Cauca con herramientas que permitan explotar, experimentar, y disfrutar plenamente a la red Ipv6 mundial.

Este documento de trabajo de grado detalla la implementación de un sistema que permite a un usuario de la Intranet interactuar con la red IPv6 mundial, así como también que la floreciente red Ipv6 de la Universidad del Cauca sea visible exteriormente, utilizando los mecanismos de migración creados durante el periodo de transición; cimentando de manera teórica y técnica unas bases que permitan a futuro la implementación total de este protocolo.

Para alcanzar los objetivos planteados, se realizó un trabajo sistemático en el cual inicialmente se recopiló y analizó la información relacionada con los Servicios Internet sobre los protocolos IPv4 e IPv6, seleccionando sus similitudes y diferencias en cuanto a parámetros de configuración, se llevó a cabo un proceso de apropiación de la tecnología que hasta el momento se había implementado en la Universidad para tener un punto de partida sobre el cual adherir nuevas capas de conocimiento, se realizó un estudio enfocado en los factores que influyen sobre las nuevas aplicaciones IPv6, finalmente se materializaron los conceptos con la implantación de los servicios canónicos, con base en una selección de herramientas que implementan las nuevas versiones de los protocolos de nivel de aplicación.

Toda la experiencia adquirida a través del desarrollo del proyecto se encuentra consignada en este documento de la siguiente forma: El capítulo 1 contiene un análisis de los requerimientos que las aplicaciones deben implementar para brindar soporte al protocolo IPv6 y se describen los posibles escenarios en los cuales puedan interactuar las dos pilas de protocolos existentes (IPv4 e IPv6), de esta manera se pretende ubicar al lector en el contexto de los diferentes tipos de aplicaciones para IPv6. Posteriormente en los capítulos 2 al 7 cada servicio implementado será abordado teniendo en cuenta una serie de ítems entre los que se encuentran: conceptos generales, nuevas especificaciones que estandarizan los protocolos de nivel de aplicación, las implementaciones que actualmente soportan dichos protocolos, configuración, puesta en marcha de cada servicio y pruebas de funcionalidad. Finalmente en el capítulo 8 se detalla la manera como fueron implementadas y desarrolladas las aplicaciones Web necesarias para el manejo y administración de los servicios, exponiendo los criterios por los cuales se desarrollo la aplicación “Interfaz web de autoconfiguración IPv6”.

1. GENERALIDADES.

1.1 TRANSICIÓN DE APLICACIONES A IPV6

A principios de los 90's se hizo una revisión del protocolo IP, para resolver problemas que evidentemente a futuro serian incontrolables. Como respuesta surgió el nuevo protocolo para Internet IPv6[1]. Este propuso una nueva arquitectura de direccionamiento, e incrementó las capacidades funcionales de IPv4[2]. En adición IPv6 ha sido diseñado para proveer los requerimientos que los servicios de nueva generación requieren, proveyendo Seguridad, Movilidad y Calidad de Servicios.

Muchas de las nuevas capacidades de IPv6 lo hicieron incompatible con IPv4, lo que ha dificultado la actualización de todos los nodos al nuevo protocolo. Por tanto el desarrollo de IPv6 ha sido planteado como un proceso de transición gradual, en el cual ambos protocolos podrán coexistir.

1.2 DEPENDENCIA DE LAS APLICACIONES AL PROTOCOLO IP.

A pesar de que en principio el cambio de protocolo debía afectar las funciones específicas relacionadas con el nivel IP como enrutamiento, y funciones de direccionamiento, la incompatibilidad entre los dos protocolos afectaron a las aplicaciones. Por tanto, cuando se piensa en una migración IPv4 a IPv6 se deben tener en cuanta las siguientes consideraciones:

- Formato de presentación para una dirección IP
- El API de nivel de transporte: Funciones para establecer una comunicación e intercambiar información.
- Resolución de nombres y direcciones: Funciones de conversión entre nombres de *host* y direcciones IP
- Dependencias específicas al protocolo IP: Dependencias más específicas como son la elección y almacenamiento de direcciones IP.
- Aplicaciones Multicast: Se deben realizar las equivalencias IP6 de las direcciones IP4 multicast, y usar las opciones de configuración socket correctamente.

1.2.1 Formato de presentación para una dirección IP.

El formato de presentación de direcciones IPv4 usa bloques decimales separados por puntos (“.”), para representar direcciones de 32 bits, mientras que IPv6 utiliza bloques hexadecimales separados por dos puntos (“:”) para representar direcciones de 128 bits. Existen dos problemas relacionados con la presentación de las direcciones:

1 Protocolo Internet versión 6.
2 Protocolo Internet versión 4.

- La memoria reservada para almacenar una dirección IPv4 como un *string* (cadena de caracteres), es insuficiente para almacenar una dirección IPv6, por lo cual las aplicaciones deben ser modificadas para prevenir un buffer overflow (desbordamiento de memoria).

- Las diferencias en la presentación de direcciones pueden acarrear problemas en las aplicaciones que requieran de éstas, un ejemplo particular se da en el formato de las URL, donde los dos puntos con que se suele separar el puerto del servidor al que se desea lanzar peticiones, pueden ser confundidos con los dos puntos de separación de los grupos de hexadecimales que componen la dirección IPv6, este caso es estudiado en profundidad en el capítulo 5 correspondiente a SERVIDOR WEB.

1.2.2 El API de nivel de transporte.

El API (*aplication program interface*) de *sockets* utilizado por las aplicaciones para intercambiar información con otras entidades hace que los detalles del protocolo IP sean visibles para estas.

1.2.2.1 Estructuras de dirección Socket.

El API de *sockets* provee a los programadores funciones que permiten el establecimiento de una comunicación, e intercambio de datos. Algunas de estas funciones requieren como parámetro de funcionamiento un tipo de dato complejo especial llamado sockaddr_in (Estructura de Direcciones Socket), el cual se requiere al momento de determinar el punto de acceso de comunicación al servicio especificado. La estructura usada en IPv4 utilizando el lenguaje C es la siguiente:

```
struct sockaddr_in {
     sa_family_t sin_family;             /* AF_INET */
     in_port_t sin_port;                 /* Numero de Puerto */
     struct in_addr sin_addr;            /* Direccion Internet
*/
     unsigned char sin_zero[sizeof (struct sockaddr) -
                            sizeof (sa_family_t) -
                            sizeof (in_port_t) -
                            sizeof (struct in_addr)];
};
```

Donde la estructura struct in_addr es:

```
struct in_addr  {

    in_addr_t s_addr;                    /*direccion IPv4 */
};
```

Como se puede observar esta estructura no es apropiada para IPv6, ya que está diseñada para almacenar direcciones de 32 bits. Entonces se diseñó una nueva estructura con una nueva familia de direcciones llamada AF_INET6 con capacidad de 128 bits:

```
struct sockaddr_in6 {

    sa_family_t sin6_family;    /* AF_INET6 */
    in_port_t sin6_port;        /* Puerto */
    uint32_t sin6_flowinfo;     /* Informacion de IPv6 */
    struct in6_addr sin6_addr;  /* Direccion IPv6 */
    uint32_t sin6_scope_id;     /* IPv6 scope-id */
};
```

Donde la estructura in6_addr corresponde a:

```
struct in6_addr {
    union {
        uint8_t u6_addr8[16];
        uint16_t u6_addr16[8];
        uint32_t u6_addr32[4];
    } in6_u;

    #define s6_addr                 in6_u.u6_addr8
    #define s6_addr16               in6_u.u6_addr16
    #define s6_addr32               in6_u.u6_addr32
};
```

Sin embargo, al utilizar cualquiera de las dos estructuras se restringe la portabilidad de las aplicaciones, por tanto es recomendable usar una que elimina las dependencias del protocolo en el código fuente, para ello se debe utilizar la estructura sockaddr_storage para almacenar tanto direcciones IPv4 como IPv6.

struct sockaddr_storage

```
{
    sa_family_t ss_family;      /* Address family */
    __ss_aligntype __ss_align;  /* Force desired alignment.  */
    char __ss_padding[_SS_PADSIZE];
};
```

De esta manera se oculta la estructura *socket* de direcciones específica que la aplicación esta utilizando.

1.2.2.2 Funciones Sockets.

A continuación se listan las funciones socket que requieren de las estructuras mencionadas en la sección previa, estas reciben o entregan las estructuras desde el Kernel, tablas 1 y 2 respectivamente.

Tabla 1. Funciones que entregan estructuras de direccion al Kernel

La estructura de dirección socket es entregada al kernel desde la aplicación		
Valor Retornado	***Nombre de la función***	***Parámetros requeridos***
Int	Bind	(int sockfd, struct sockaddr *my_addr, socklen_t addrlen)

La estructura de dirección socket es entregada al kernel desde la aplicación		
Int	Connect	(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen)
int	Sendto	(int s, const void *msg, size_t len, int flags, const struct sockaddr *to, socklen_t tolen)

Tabla 2. Funciones que reciben estructuras de dirección desde el Kernel

La estructura de dirección socket es entregada a la aplicacion desde el kernel		
Valor Retornado	***Nombre de la funcion***	***Parametros requeridos***
Int	Accept	(int s, struct sockaddr *addr, socklen_t *addrlen)
Int	Recvfrom	(int s, void *buf, size_t len, int flags, struct sockaddr *from, socklen_t *fromlen)
Int	Getpeername	(int s, struct sockaddr *name, socklen_t *namelen)
Int	Getsockname	(int s, struct sockaddr *name, socklen_t *namelen)

La tabla 3 muestra otras funciones proveídas por el API de sockets:

Tabla 3. Funciones proveídas por el API Sockets.

Valor Retornado	***Nombre de la función***	***Parámetros requeridos***
int	Socket	(int domain, int type, int protocol)
int	Listen	(int s, int backlog)
ssize_t	Write	(int fd, const void *buf, size_t count)
int	Send	(int s, const void *msg, size_t len, int flags)
int	Sendmsg	(int s, const struct msghdr

Valor Retornado	*Nombre de la función*	*Parámetros requeridos*
		*msg, int flags)
ssize_t	Read	(int fd, void *buf, size_t count)
int	Recv	(int s, void *buf, size_t len, int flags)
int	Recvmsg	(int s, struct msghdr *msg, int flags)
int	Close	(int fd)
int	Shutdown	(int s, int how)

1.2.2.3 Modificaciones requeridas al portar una aplicación.

Teniendo en cuenta lo expuesto hasta el momento, es claro que algunas modificaciones relacionadas con el API de sockets son necesarias para portar una aplicación IPv4 a IPv6. Se requieren tres modificaciones, relacionadas con los siguientes tópicos:

- Creación de un socket.
- Paso de la Estructura de direcciones socket desde la aplicación al kernel.
- Paso de la Estructura de direcciones socket desde el kernel a la aplicación.

1.2.2.3.1 Creación de un socket.

La creación de un *socket* se realiza mediante la función socket(). Esta función devuelve un descriptor de archivo normal que el sistema posteriormente asociará a una conexión de red.

La diferencia entre la creación de un socket IPv4 o IPv6, es el valor del argumento *family* al momento de la llamada.

Código fuente IPv4.

```
socket(PF_INET, SOCK_STREAM, 0);   /* socket TCP   */
socket(PF_INET, SOCK_DGRAM,  0);   /* socket UDP   */
```

Código fuente IPv6.

```
socket(PF_INET6, SOCK_STREAM, 0); /* socket TCP */
socket(PF_INET6, SOCK_DGRAM,  0); /* socket UDP */
```

1.2.2.3.2 Paso de la estructura de direcciones socket desde la aplicación al kernel.

Cuando se desea avisar al sistema operativo que se ha abierto un socket y que se desea que se asocie la aplicación a dicho socket, se hace un llamado a la función bind(). El sistema todavía no atenderá a las conexiones de clientes, simplemente anota que cuando empiece a hacerlo, tendrá que dar aviso. Esta función requiere la estructura de direcciones *socket* la cual es editada apropiadamente antes de llamar esta función.

Código fuente.

```
bind(sockfd,(struct sockaddr *)&addr, addrlen);
```

Donde struct sockaddr depende de si la conexión es IPv4 o IPv6 o mixto como se describe en la sección 1.2.2.1 (Estructuras de Dirección *Socket*).

1.2.2.3.3 Paso de la Estructura de direcciones socket desde el kernel a la aplicación.

Al momento de solicitar al sistema operativo pedir y aceptar las conexiones de clientes. Se realiza una llamada a la función accept(). Esta función le indica al sistema operativo que debe atender al siguiente cliente de la cola que se forma con las peticiones que llegan al sistema. Si no hay clientes se quedará bloqueada hasta que algún cliente se conecte.

Cuando se llama este tipo de funciones, la estructura de direcciones socket requerida, se forma con la dirección de la fuente.

Código fuente:

```
accept(sockfd,(struct sockaddr *)&addr, &addrlen);
```

Donde struct sockaddr depende de si la conexión es IPv4 o IPv6 o mixto como se describe en la sección 1.2.2.1 (Estructuras de Dirección Socket).

1.2.3 Resolución de nombres y direcciones: funciones de conversión entre nombres de host y direcciones IP.

Muchas aplicaciones requieren una direccion IP como argumento para establecer una comunicación, como por ejemplo en el modelo cliente/servidor el cliente debe conocer la dirección del servidor para conectarse a éste. Las aplicaciones de ser posible deberían usar FQDN (*Fully Qualified Domain Name*) en lugar de direcciones IP, debido a que los nombres son menos propensos a cambiar que las direcciones, evitando así, edición de código, al delegar el trabajo de obtención de la direccion IP al sistema de resolucion de nombres. Para las aplicaciones la resolución de nombres se lleva a cabo como un proceso de un sistema independiente. Las aplicaciones llaman a un sistema de librerías llamado *Resolver,* típicamente las funciones utilizadas son *gethostbyname* y *gethostbyaddr* Figura 1.

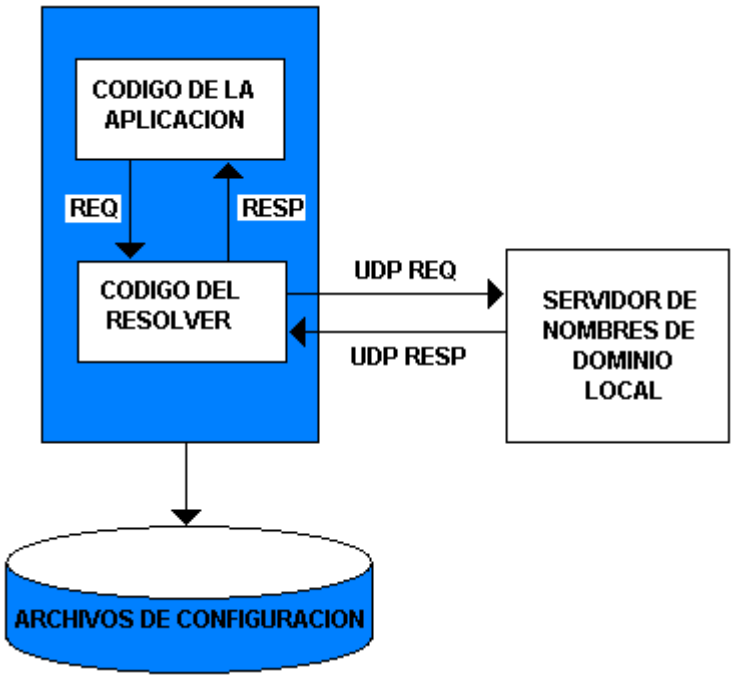


Figura 1. Funciones de conversión de direcciones

Para el protocolo IPv6 estas últimas deben ser reemplazadas por dos nuevas funciones *getaddrinfo()* y g*etnameinfo(),* la primera función retorna una lista de todas las direcciones configuradas en un *host*. Las peticiones pueden ser restringidas a una sola familia de protocolos IPv4 o IPv6, sin embargo es altamente recomendable que todas las direcciones configuradas sean obtenidas para permitir a las aplicaciones trabajar en cualquier tipo de escenarios (para más detalle sobre este punto consulte la sección 1.5 Escenarios de transición gradual). La segunda función retorna un nombre de *host* asociado a una dirección IP. **[1]**

1.2.4 Dependencias específicas al protocolo IP.

1.2.4.1 Elección entre múltiples direcciones IP.

A diferencia del modelo IPv4, IPv6 permite la configuración de múltiples direcciones IP por nodo. Esta nueva característica afecta directamente a las aplicaciones ya que estas solo pueden usar un par de direcciones destino/fuente. Escoger la dirección fuente y destino correctas, es un factor clave que determinara la ruta de los datagramas IP.

Cuando se selecciona una dirección destino, las aplicaciones usualmente preguntan por esta IP al *Resolver*. El *Resolver* devuelve una lista de direcciones IP válidas, para un nombre de *host*. A menos que la aplicación tenga una razón especifica para elegir una dirección en particular, esta debería tratar con cada elemento en la lista hasta que la comunicación se realice.

1.2.4.2 Almacenamiento de direcciones IP

Algunas aplicaciones suelen almacenar direcciones remotas para su funcionamiento, sin embargo, estas funciones deben ser editadas para IPv6, las razones para ello son:

- Las direcciones IP pueden cambiar a través del tiempo; por ejemplo después de un proceso de renumeración (Remítase al capitulo 2. Servidor DNS para mayor información al respecto).

- El mismo nodo puede alcanzar un *host* destino usando diferentes direcciones IP, posiblemente con diferentes versiones de protocolos.

Para contrarrestar estos problemas, las aplicaciones deberían almacenar nombres de *host* en lugar de direcciones IP. En este caso los programadores están limitados a especificar direcciones en tiempo de ejecución, o con una duración de un tiempo de vida de *cache*. Otro tipo de aplicaciones como por ejemplo sistemas masivos *peer-to-peer*, que cuenten con su propio *rendezvous*[3] y sus propios mecanismos de descubrimiento, pueden requerir de direcciones en *cache*, por razones de rendimiento, pero las direcciones en *cache* no pueden ser tratadas como información confiable permanente **[1]**.

1.2.5 Aplicaciones Multicast.

Cuando se hace uso de las facilidades multicast, se requiere realizar algunos cambios para soportar IPv6.

- Se deben remplazar las direcciones multicast IPv4, por sus equivalentes IPv6, las cuales se identifican por que inician con los siguientes dos octetos FF0X.

- Deben ser usadas las opciones socket multicast IPv6 apropiadas, para configurar parámetros necesarios para el envió de este tipo de paquetes. Estas opciones especializadas se listan en la tabla 4 **[2]**.

Tabla 4. Opciones socket multicast IPv6.

OPCION IPV6	***DESCRIPCION***
IPV6_MULTICAST_IF	Interfaz a utilizar para los paquetes multicast salientes.
IPV6_MULTICAST_HOPS	Numero de saltos límite para paquetes multicast.
IPV6_MULTICAST_LOOP	Control sobre el comportamiento loopback de los paquetes multicast.
IPV6_ADD_MEMBERSHIP	Adición a un grupo multicast.
IPV6_DROP_MEMBERSHIP	Renuncia a un grupo multicast.

1.3 PROBLEMAS CON LAS APLICACIONES IPv6 DURANTE EL PERIODO DE TRANSICIÓN.

Existen algunas razones por las cuales el periodo de transición IPv4 a IPv6 no es directo, entre las que se encuentran:

3 es una primitiva de sincronización asimétrica que permite a dos procesos concurrentes, el solicitante y el llamado, intercambiar datos de forma coordinada

1.3.1 Soporte desligado entre el Sistema Operativo y las Aplicaciones.

Las pilas de protocolos IPv4 e IPv6 seguramente tendrán que convivir por un largo tiempo **[3]**, periodo en el cual también se espera que las aplicaciones manejen los dos protocolos.

Muchas aplicaciones capaces de trabajar con ambos protocolos, deberán correr apropiadamente en nodos IPv4 (Si el protocolo IPv6 ha sido deshabilitado o no existe ningún tipo de conectividad IPv6), si esto no sucede estas aplicaciones no serán apetecibles, truncando así la posibilidad de ser explotadas.

1.3.2 Los servidores DNS no indican cual protocolo será usado.

Este problema radica en que un programa cliente no obtiene suficiente información por parte del servidor DNS sobre el protocolo con el que tendrá que trabajar en el extremo de la conexión. Por ejemplo cuando una aplicación servidor **no soporta** aún IPv6 pero corre en una máquina con ambas pilas de protocolos, y esta se encuentra asignado únicamente a una dirección IPv6 (como se discutirá a profundidad en el capítulo 2), la aplicación cliente fallará al intentar conectar con dicho servidor, esta falla se debe a la incompatibilidad entre la respuesta del DNS, en este caso una dirección IPv6 y la versión del protocolo IP manejada por el servidor, o sea IPv4.

1.3.3 El soporte de diferentes versiones de un programa puede resultar difícil.

Durante el periodo de transición de aplicaciones, los administradores de red podrían verse enfrentados a varias versiones de un mismo programa, por ejemplo una versión para IPv4 y otra para IPv6, lo cual aumenta el trabajo y por consiguiente la probabilidad de introducir errores humanos al sistema.

Por otro lado los usuarios de la red podrían tener problemas seleccionando la versión correcta de un programa para un tipo de dirección específico, ya que como se puede comprobar en la sección previa, es imposible deducir a partir de una consulta DNS el tipo de protocolo que un servicio en particular maneja.

1.4 MECANISMOS DE TRANSICIÓN.

Existen dos metodologías para solucionar el problema planteado, la primera es el uso de mecanismos de transición que permitan a las aplicaciones IPv4 comunicarse entre si usando IPv6. La segunda es modificar el código fuente de las aplicaciones para que estas usen IPv6 nativamente, la segunda opción es la más recomendable, siempre y cuando el código fuente esté disponible.

En el proceso de transición, las aplicaciones IPV4, podrán coexistir con las nuevas IPv6. El *dual-satck* es un mecanismo que permite la coexistencia de aplicaciones diseñadas para los diferentes protocolos. El *dual-stack* está basado en la instalación de las dos pilas de protocolos IP en un mismo nodo. Este usará la pila IPv4 cuando requiera comunicarse con nodos remotos IPv4, y de la misma manera utilizará IPv6 para intercambiar información con nodos IPv6 remotos.

1.4.1 Comunicaciones IPv6 sin adaptación del código fuente.

La instalación del *dual-stack* no garantiza la operación de las aplicaciones sobre ambas plataformas. Las aplicaciones IPv4/IPv6 corriendo en un *dual-stack* utilizarán IPv4 e IPv6 respectivamente, sin embargo, una aplicación IPv4 no podrá establecer directamente una comunicación IPv6, el código fuente de esta deberá ser cambiado. Si lo anterior no es posible, otras soluciones deben ser desarrolladas para permitir la coexistencia de ambas tecnologías.

Existen soluciones basadas en mecanismos de transición que se instalan en los nodos finales, en los cuales corren las aplicaciones IPv4. Esta metodología permite la coexistencia de las aplicaciones IPv4 existentes y en las nuevas redes IPv6 sin adaptar el código fuente. Actualmente se cuenta con dos mecanismos experimentales estudiados por la IETF, que permiten la traducción entre paquetes IPv4/IPv6: *Bump In the Stack* (BIS) y *Bump In the API* (BIA). BIS hace la traducción en la pila IP y BIA hace la traducción en el API.

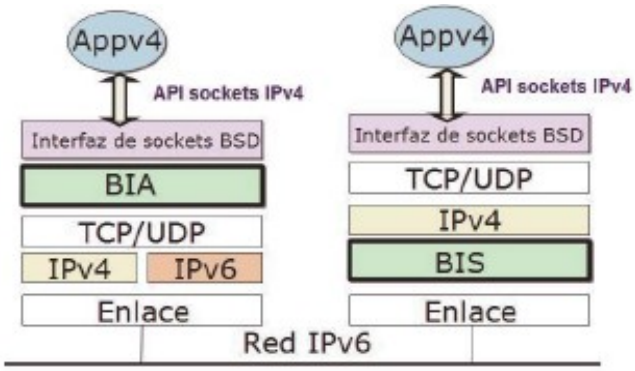


Figura 2. Mecanismos de transición BIA y BIS

Diferentes módulos específicos deben ser instalados en los nodos para ambos mecanismos. Dado que estos módulos no están disponibles para todos los sistemas operativos existentes, BIS y BIA no son siempre soluciones posibles.

1.4.2 Comunicaciones IPv6 por adaptación de código.

Los mecanismos de transición previamente descritos permiten a aplicaciones IPv4 intercambiar paquetes IPv6, sin embargo si el código fuente se encuentra disponible, estas deberían ser adaptadas para el soporte nativo de IPv6. En otras palabras sin traducción de protocolos. La adaptación del código fuente se puede dar en dos formas: Aplicaciones IPv6 puras y aplicaciones *dual-stack*.

1.4.2.1 Aplicaciones IPv6 Puras.

Son aquellas que han sido desarrolladas para trabajar usando únicamente IPv6, estas se deben correr en nodos con al menos la pila IPv6. La aplicaciones IPv4 pueden ser convertidas en aplicaciones IPv6 substituyendo las funciones API IPv4 por las IPv6 como se estudió en la sección 1.2.2.3 (Modificaciones Requeridas al portar una aplicación). El proceso de adaptación está basado en una exploración exhaustiva del código fuente a fin de cambiar las dependencias IPv4 a dependencias IPv6.

Existen herramientas de exploración automática que ayudan a los desarrolladores a encontrar las citadas dependencias IPv4 en el código fuente, entre las cuales se cuentan *scrubber* y *checkv4*, usando estas herramientas la adaptación de las aplicaciones es relativamente rápida y sencilla. Por ejemplo, Checkv4 muestra el número de línea y un mensaje con consejos acerca de cómo debería cambiarse el código. La figura 3 muestra un ejemplo de la salida de Checkv4.exe:

```
test.c(35) : gethostbyname : use getaddrinfo instead

test.c(40) : gethostbyaddr : use getnameinfo instead

test.c(48) : SOCKADDR_IN : use SOCKADDR_STORAGE instead, or use

SOCKADDR_IN6 in addition for IPv6 support

test.c(57) : AF_INET : use AF_INET6 in addition for IPv6 support

test.c(89) : inet_addr : use WSAStringToAddress or getaddrinfo
with

AI_NUMERICHOST instead
```

Figura 3. Salida del programa Checkv4

En las redes heterogéneas IPv4/IPv6 típicas del periodo de transición, las aplicaciones usarían ambos protocolos, por consiguiente serían necesarias dos versiones de la misma aplicación: las IPv4 puras y las IPv6 puras. Por tanto los usuarios necesitarían mecanismos para seleccionar la versión IP adecuada a cada caso.

1.4.2.2 Aplicaciones DUAL-STACK

Estas han sido desarrolladas para trabajar con ambos protocolos, dependiendo del nodo en el cual esté corriendo. En nodos IPv4 puros y nodos IPv6 puros, las aplicaciones dual-stack usarían IPv4 e IPv6 respectivamente. En nodos dual-stack, en los cuales están presentes ambas pilas, este tipo de aplicaciones son capaces de usar ambos protocolos, el uso de cualquiera de los dos depende de la red, del nodo remoto o de la aplicación remota.

Para producir aplicaciones *dual-stack*, un nuevo diseño de la aplicación es a menudo requerido para permitir las comunicaciones entre IPv4 e IPv6. Entonces es necesario un completo conocimiento de la arquitectura de la aplicación para decidir la mejor estrategia para lograr el cometido esperado **[3]**.

1.4.2.3 Consideraciones al portar código fuente.

Como se ha podido inferir a partir de lo expuesto hasta el momento, cuando se está portando una aplicación a IPv6, muchos de los cambios deben hacerse en el módulo de transporte de la aplicación, el cual es el encargado de establecer comunicaciones con nodos remotos. Por tanto es recomendable separar este módulo del resto de la aplicación. Esta separación hará a la aplicación independiente del sistema de red usado, ya que si el protocolo de red es cambiado, solamente el módulo de transporte deberá ser modificado. Este módulo proporcionará abstracción al canal de comunicación, con el uso de estructuras de datos genéricas para representar las

direcciones (como se expone en la sección 1.2.2.1 Estructuras de Dirección Socket) y operaciones de canal básicos. Esta abstracción puede ser instanciada en diferentes implementaciones, dependiendo del protocolo de red requerido en cada momento. La aplicación debería entonces, coexistir con este canal genérico de comunicación sin conocer el protocolo de red usado. Mediante este diseño en caso de adherirse un nuevo protocolo de red, los desarrolladores de la aplicación solo deben implementar una nueva instancia de la abstracción del canal, la cual manejará las características del nuevo protocolo.

1.5 ESCENARIOS DE TRANSICIÓN GRADUAL.

Desde que las versiones del IP son incompatibles, entidades usando diferentes versiones del protocolo no pueden establecer comunicaciones entre si, por tanto primero se debe establecer una comunión entre cliente y servidor para utilizar un protocolo en común. Por consiguiente durante el periodo de transición la elección de la versión de IP dependerá del escenario específico.

A continuación se listan los escenarios de coexistencia más comunes, durante el periodo de transición, y se definen algunas recomendaciones para mantener la interoperabilidad IPv4/ IPv6.

1.5.1 Aplicaciones IPv4 puras en un escenario dual-stack.

En este escenario el protocolo IPv6 ha sido adherido al nodo en cuestión, pero las aplicaciones IPv6 no han sido instaladas o no son disponibles. En principio las aplicaciones IPv4 solo podrán establecer comunicaciones IPv4. En este escenario el problema aparece cuando una aplicación necesita hacer una comunicación IPv6, la solución para este problema es la implementación de los mecanismos BIA o BIS, tratados en la sección “Comunicaciones IPv6 sin adaptación del código fuente”.

BIS y BIA, no trabajan con todos los tipos de aplicaciones, en particular con aplicaciones que intercambian direcciones, como ocurre en programas que realizan intercambio de datos, por ejemplo FTP. Este tipo de mecanismos provee una dirección IPv4 temporal a las aplicaciones, y localmente realiza la translación entre la comunicación IPv4 - IPv6. Sin embargo, esta dirección temporal IPv4 tiene valides únicamente dentro del nodo.

1.5.2 Aplicaciones IPv6 en un escenario dual-stack.

En este escenario las aplicaciones IPv6 corren en un nodo *dual-stack*. Los problemas de interoperabilidad aparecen cuando se requiere establecer una comunicación IPv4. Muchos nodos *dual-stack* permiten a aplicaciones IPv6 intercambiar paquetes IPv4. Este modo de operación se da cuando estas aplicaciones hacen uso de direcciones IPv6 especiales, estas son conocidas como direcciones IPv6 mapeadas, las cuales tienen el siguiente formato: "::FFFF:x.y.z.w", donde “x.y.z.w” es la dirección IPv4 del nodo remoto. Cuando una aplicación IPv6 usa este tipo especial de direcciones, el nodo *dual-stack* extrae la dirección IPv4 “x.y.z.w” de la dirección IPv6 mapeada y crea paquetes IPv4 usando la dirección de destino, de esta manera se resuelve el problema de interoperabilidad:

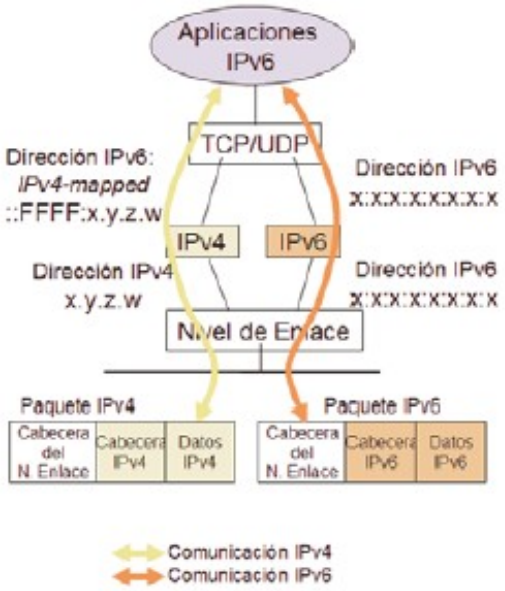


Figura 4. Dirección IPv6 mapeada

1.5.3 Aplicaciones Dual stack en nodos Dual-stack

Este caso de transición es el más recomendable. En este caso cuando las aplicaciones *dual-stack* corren en un nodo *dual-stack* se obtiene máxima interoperabilidad entre aplicaciones y redes, ya que la comunicación se realiza sin tener en cuenta la versión de la pila IP o el tipo de nodo remoto. Las aplicaciones *dual-stack* en su gran mayoría prefieren las comunicaciones IPv6 por defecto, sin embargo, si ésta no es posible se realizará un reintento automático con una comunicación IPv4.

Si el código fuente de la aplicación es escrito en una manera que resulte dependiente del protocolo, la aplicación soportará IPv4 e IPv6 usando dos sockets separados. Y por tanto se tendrán diferencias en la implementación, por ejemplo cuando se haga un llamado a la función *bind()* (como se puede comprobar en la seccion 1.2.2.3.2 Paso de la Estructura de direcciones socket desde la aplicación al kernel), primero se intentara hacer bind con la dirección IPv6 y luego con la IPv4. Un código fuente implementado de la forma anterior puede resultar demasiado complejo, por ello se recomienda utilizar funciones que implementen un dual-stack completo.

1.5.4 Aplicaciones dual-stack en un nodo IPv4 puro.

Una aplicación que ha sido adaptada a una *dual-stack* puede requerir correr en un nodo IPv4 puro, se pueden presentar inconvenientes de interoperabilidad si la aplicación no ha sido diseñada para correr en un nodo con la pila IPv6 presente. Por ejemplo supóngase que una aplicación lanza un comando socket(), primero éste hará un intento con AF_INET6 y luego con AF_INET. Sin embargo, si el Kernel del sistema no tiene soporte IPv6, el llamado resultará en un error EPROTONOSUPPORT o EAFNOSUPPORT. Típicamente errores como este conducirán a un loop de socket, y el AF_INET no será probado.

Este problema es solventado desarrollando aplicaciones *dual-stack* sin asumir un protocolo de operación obligatorio **[1].**

2. DNS SISTEMA DE NOMBRES DE DOMINIO EN IPV6

2.1 DEFINICION

El DNS (Sistema de nombres de Dominio) constituye una base jerárquica de datos distribuida en la cual se almacena información para realizar el mapeo de direcciones IP a nombres de máquina y viceversa, información de enrutamiento para el servicio de correo, y datos utilizados por las diferentes aplicaciones en Internet, permite que los usuarios localicen determinadas máquinas mediante el nombre asignado por el administrador, liberándolos así de la pesada tarea de recordar la dirección numérica de los recursos de red y además mantiene direcciones de otros servidores de nombres que puede emplear para actualizar su información.

El DNS almacena la información de acuerdo a una estructura en árbol, y cada nodo de éste, llamado Dominio, se identifica con una etiqueta; el nombre de dominio de cada nodo, es la concatenación de todas las etiquetas utilizadas para marcar la ruta desde el nodo final hasta el nodo raíz; del nodo raíz, que se representa de la forma ".", se desprenden todos los dominios primarios que indican el país, región o tipo de organización que utiliza ese nombre. Existen 2 tipos de dominios primarios, los genéricos y los geográficos. Los genéricos son conocidos como internacionales u organizacionales, entre los cuales están .org, .net, .edu, .com, .mil, y .gov; los geográficos son organizados por localidades. Así, por ejemplo el dominio unicauca.edu.co indica que su ubicación geográfica es Colombia (.co) y el tipo de organización es educativa (.edu).

2.2 COMPONENTES BASICOS

EL DNS y su estructura se basan en 4 componentes principales:

- El Espacio de Nombres de Dominio (Domain Name Space) es una estructura jerárquica con forma de árbol que clasifica los distintos dominios en niveles. Cada nivel del árbol puede representar una rama o una hoja del mismo, en el cual una rama representa un nivel donde se utiliza más de un nombre para identificar un grupo de recursos y una hoja representa un nombre único que se utiliza una vez en ese nivel para indicar un recurso específico.

- Los Registros de Recurso (Resource Records) se encargan de almacenar los datos asociados con cada Nombre de Dominio. Por lo general, la información es almacenada en archivos de texto simple con las respectivas etiquetas *resource records.*

- Los Servidores de Nombres (Name Servers), son programas encargados de mantener información actualizada acerca de la estructura en árbol del dominio, cada Servidor de Nombres, conoce las partes del árbol de dominio para las cuales tiene información completa y él se convierte en una Autoridad (AUTHORITY) para estas partes del espacio de nombres. Esta información se organiza en unas entidades llamadas Zonas, las cuales pueden ser automáticamente distribuidas en

aquellos servidores de nombre que utilizan redundancia para los datos de un dominio determinado.

- Los *Resolvers* son programas encargados de extraer información del Servidor de Nombres, en respuesta a la solicitud de un cliente. Estos programas utilizan la información del Servidor de Nombres para responder una solicitud directamente, o dar un seguimiento a la solicitud, utilizando referencias de otros Servidores de Nombres. Sus principales tareas son: Interrogar al Servidor de Nombres, Interpretar respuestas (provenientes de los registros de recurso) y devolver la información al programa que la ha solicitado. **[4]**

2.3 FUNCIONAMIENTO

El Sistema de Nombres de Dominio trabaja con unas entidades llamadas **Zonas de Autoridad**, en las cuales se almacena toda la información relacionada con un determinado dominio o subdominio (Registros de Recursos). Cada zona de autoridad abarca al menos un dominio y posiblemente sus subdominios, si estos últimos no son delegados a otras zonas de autoridad. Para efectos de disponibilidad, conectividad y tolerancia a fallos al resolver consultas, generalmente una red cuenta con un Servidor Primario o Maestro y uno o varios Secundarios. En el Servidor Primario se realizan todas las configuraciones iniciales y cambios correspondientes al dominio, y el Servidor Secundario copia dicha información, para utilizarla en el caso en que el Servidor Primario no pueda responder solicitudes. Al agregar un Servidor DNS Secundario a la red, éste debe realizar una transferencia inicial completa de la zona de autoridad para obtener y replicar una copia total de los registros de recursos de dicha zona, y cuando se realicen cambios de configuración en el Servidor Primario, debe llevarse a cabo de nuevo la transferencia de zona, para actualizar los registros que hacen parte de ella. Esta transferencia de zona es posible en los siguientes casos:

- Cuando el intervalo de actualización de una zona expira.
- Cuando el servidor primario le notifica al secundario que se presentaron cambios en la zona.
- Cuando el servidor secundario inicia el servicio.
- Cuando se realiza la transferencia de zonas de forma manual, desde el servidor primario.

La transferencia de información de zonas entre un servidor primario y uno secundario involucra los siguientes pasos y mensajes:

- Inicialmente el servidor secundario solicita al primario una transferencia total de la zona mediante un mensaje AXFR y éste le envía toda la zona.

- El registro de recurso de inicio de autoridad (SOA), su explicación se encuentra en la siguiente sección, contiene una bandera expresada en segundos, para establecer el intervalo de actualización e indicar al servidor secundario en que momento debe renovar la información de zona con el servidor primario y contiene un campo denominado Número de Serie, el cual se transfiere al servidor secundario en el momento del envío de la zona.

- Cuando expira el intervalo de actualización en el servidor secundario, éste envía una petición de inicio de autoridad solicitando una renovación de zona, el servidor

primario responde a ésta consulta con el número de serie del estado actual de la zona; si ese número de serie es igual al almacenado en el servidor secundario, se deduce que no existe modificación alguna, por lo cual, no es necesario realizar transferencia de zonas y el intervalo de actualización se reinicia. Si por el contrario, dicho número de serie es mayor al almacenado en el servidor secundario, se deduce que una transferencia de zonas es necesaria.

- El servidor secundario envía un mensaje IXFR al primario para solicitar una transferencia incremental de la zona, y éste último responde con una transferencia incremental o total, dependiendo del tipo de actualización que el secundario esté dispuesto a aceptar.

2.3.1 Resolución Directa

En el proceso de búsqueda de un nombre de *host* dada una dirección IP, intervienen 3 tipos de peticiones que el servidor debe responder:

- Un nombre de dominio DNS especificado, indicado como un nombre de dominio completo (FQDN).

- Un tipo de consulta especificada, que puede establecer un registro de recursos por tipo o una clase especializada de operación de consulta.

- Una clase especificada para el nombre de dominio DNS.

Cuando un cliente envía una petición de resolución, de acuerdo con la Figura 5 www.ejemplo.com, inicialmente dicha petición llega al cliente DNS para intentar ser resuelta con la información almacenada en la caché local. En la caché del cliente DNS, se encuentra almacenada temporalmente información correspondiente a consultas DNS anteriores y si un archivo *Host*s está configurado localmente, las asignaciones de nombre a dirección de ese archivo, se cargan previamente en la caché cuando se inicia el cliente DNS. Si se puede responder a dicha consulta con éste procedimiento, el proceso finaliza, de lo contrario, la petición debe pasar al Servidor DNS. El cliente con anterioridad ha seleccionado cuál es su DNS preferido, y éste DNS se encarga de recibir la consulta y comprueba si el nombre pertenece a un subdominio sobre el cual él tiene autoridad, es decir, si el nombre consultado coincide con un registro de recursos correspondiente en la información de zona local, si es así, el Servidor se encarga de enviar al cliente la respuesta a la solicitud planteada y el proceso finaliza; si el nombre no pertenece a ese subdominio, comienza el proceso de resolución mediante uno de los siguientes métodos:

- Recursivo: El Servidor pregunta a otros Servidores de Nombres por los datos necesarios para la resolución de la petición y almacena en la caché la información encontrada para sus futuras búsquedas.

- Iterativo: El Servidor de Nombres le envía al cliente la dirección del siguiente servidor, al cuál debe enviar su petición para realizar la resolución, con éste método, el Servidor no entra en contacto con otros Servidores DNS. En la mayor parte de las consultas iterativas, un cliente utiliza su lista de Servidores DNS configurada localmente para entrar en contacto con otros servidores de nombres a través del espacio de nombres, si su servidor DNS principal no puede resolver la consulta.

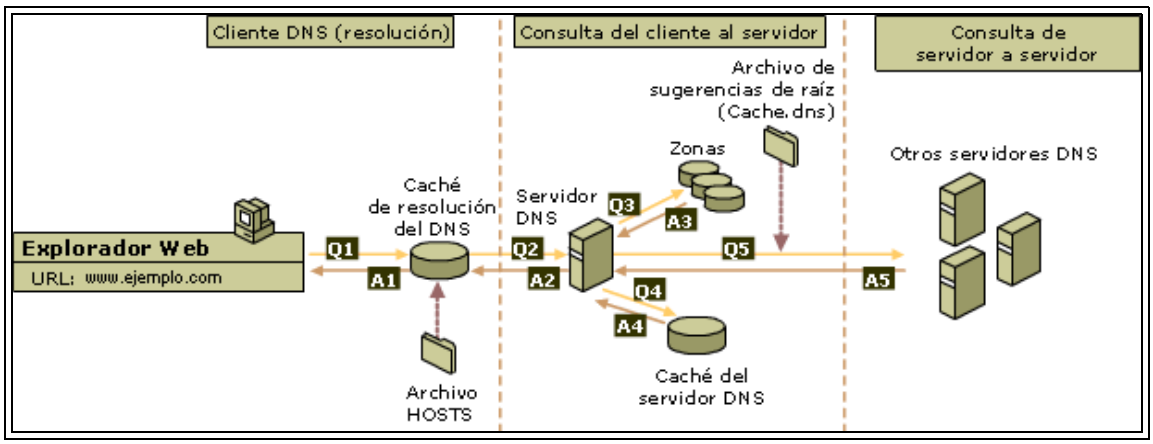


Figura 5.Resolución Directa

Para la resolución recursiva, el Servidor DNS necesita información útil de contactos acerca de los otros Servidores DNS del espacio de nombres de dominio que tienen autoridad sobre los diferentes dominios que hacen parte de los niveles del árbol de espacio de nombres, a los cuales pueda reenviar las peticiones de resolución.

La Figura 6 ilustra el proceso de resolución mediante el método recursivo, considerando que el Servidor DNS preferido no tiene almacenada la información necesaria para resolver la petición realizada por el cliente. El cliente desea localizar el *host* tx1.ejemplo.redX.com, el Servidor DNS preferido analiza el nombre completo y determina que necesita la ubicación del servidor con autoridad para el dominio de nivel superior "com". Una vez obtiene la ubicación de ese servidor, le envía una petición para obtener una referencia al servidor redX.com, finalmente el servidor redX.com proporciona una respuesta de referencia al servidor DNS para localizar "ejemplo.redX.com", DNS en el cual se encuentra almacenada la información correspondiente al nombre consultado. Una vez el servidor preferido recibe la respuesta a la consulta solicitada, éste la reenvía al cliente solicitante y se completa el proceso.

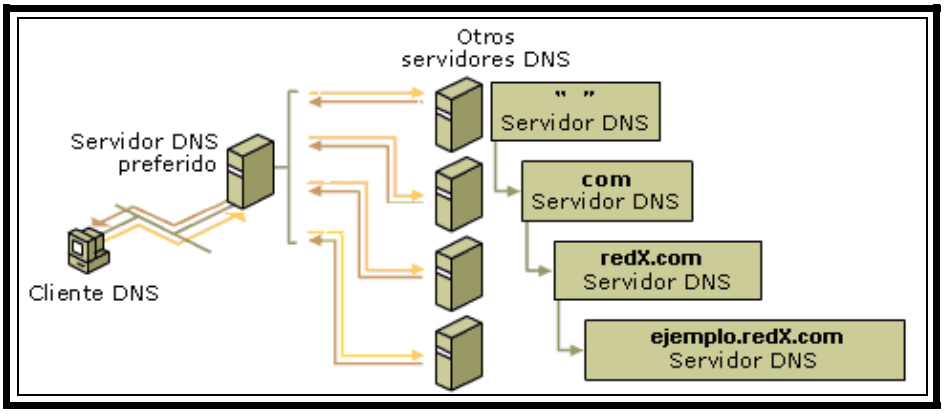


Figura 6. Método recursivo para la resolución de nombres

El almacenamiento en caché de los Registros de Recursos obtenidos en anteriores consultas, aumenta el rendimiento de la resolución DNS para futuras consultas y reduce el tráfico de las consultas relativas al DNS en la red.

2.3.2 Resolución Inversa

En este tipo de resolución, el cliente envía una solicitud para relacionar una dirección IP con su correspondiente nombre de máquina; la búsqueda se basa en la pregunta ¿puede decirme qué nombre de máquina corresponde el equipo que utiliza la dirección X.X.X.X ?

En el proceso de búsqueda inversa, un cliente envía a su servidor DNS preferido una consulta para conocer el nombre de máquina correspondiente a una dirección IP, por ejemplo, a la dirección 172.16.41.108; si ese servidor DNS tiene autoridad sobre la zona que incluye esa dirección IP, devolverá un registro PTR (Registro de Puntero) con el nombre de máquina y el nombre de dominio DNS para esa dirección IP específica. Si por el contrario, el DNS preferido no puede responder a esa consulta inversa, se utiliza la recursividad o la iteración, de igual forma que como ocurre en la búsqueda directa. En la Figura 7 se aprecia este proceso.

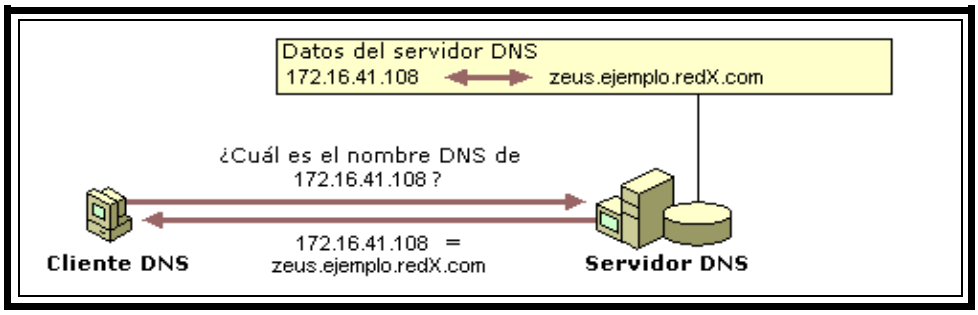


Figura 7. Resolución inversa

El DNS utiliza un formato general para todos los mensajes, tanto en las consultas directas como en las inversas, éste consta de un encabezado fijo, una sección de pregunta, que contiene la consulta, y tres secciones adicionales que pueden contener Registros de Recursos de diferentes tipos. La sección ***Answer*** (Respuesta) usualmente contiene registros relacionados directamente con la respuesta a determinada solicitud o consulta, la sección ***Authority*** (Autoridad) mantiene los nombres de los Servidores de Nombre que han enviado respuesta al cliente para la resolución de la consulta, y la sección ***Additional*** (Adicional) mantiene información extra acerca de los Servidores de Nombres que se han almacenado en la sección Autoridad, que puede llegar a ser muy útil para el cliente.

2.4 ESPECIFICACIONES DNS PARA EL PROTOCOLO IPV4

2.4.1 REGISTROS RECORD RESOURCES DEFINIDOS

En un Servidor de Nombres de Dominio debe almacenarse toda la información relacionada a un determinado dominio y a las zonas que debe gestionar. Una lista de los nombres de máquina con sus respectivas direcciones IP asociadas, una lista con las direcciones IP con sus respectivos nombres de máquina asociados y una lista de

los Servidores de Nombres mundiales para realizar consultas externas, son necesarios para llevar a cabo las funciones de resolución.

Toda esta información se almacena en los Registros de Recursos (RR) que permiten la descripción de los recursos asociados a un nodo. Los componentes de un RR, se explican a continuación:

- TYPE: Identifica el tipo de Registro de Recurso. Estos tipos se especifican más adelante.

- TTL: Especifica el intervalo de tiempo que el registro de recursos puede permanecer en caché, antes de consultar de nuevo la fuente de información.

- CLASS: Identifica una familia de protocolos o una instancia de un protocolo.

- RDATA: Describe el recurso, teniendo en cuenta información de los campos *Type* y *Class.*

- NAME: Identifica el nombre del nodo al cual pertenece el registro de recursos; existe la clase IN para redes basadas en TCP/IP (INTERNET), la CH para redes basadas en protocolos CHAOSNET y la clase HS para redes que utilizan Software HESIOD. La más utilizada es la IN.

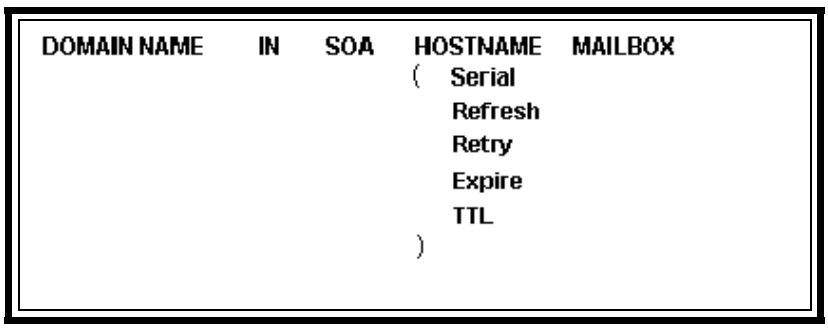
- RDLENGTH: Especifica la longitud en octetos del campo RDATA.

El orden de los elementos que constituyen los registros de recursos es el siguiente:

Name TTL Class Type Rdlength Rdata

Para la resolución de nombres de máquina a direcciones IP y viceversa utilizando el protocolo IPV4, los tipos (*type*) de RR más utilizados son:

- SOA: Identifica el inicio de autoridad de una zona e indica que los registros siguientes contienen información autorizada para el dominio; éste contiene la versión actualizada de la Base de Datos que conforma el DNS. Debe existir un solo registro SOA por cada dominio o subdiminio (para los casos en los cuales la red maneje una zona por cada subdominio) que conformen la red, ya que, cada registro representa una zona de autoridad. Su formato es el siguiente:



Domain Name: Indica el nombre de dominio al cual pertenece el registro SOA.
IN: Especifica la red Internet para el registro.
SOA: Es el tipo de registro de recurso.
Hostname: Contiene el nombre del *Host* que se desempeña como DNS.

Mailbox: Contiene la dirección de correo electrónico de la persona encargada de administrar el dominio, para el reporte de errores.
*Seria*l: Es un número que indica la versión actual de la base de datos del dominio y se utiliza para enterar a los DNS secundarios de las actualizaciones que se han realizado en el primario.
Refresh: Indica el intervalo de tiempo que debe transcurrir para preguntar a los DNS principales por los cambios realizados en la configuración de la zona.
Retry: Indica el intervalo de tiempo que debe esperar un servidor secundario para intentar reconectarse con el primario, si falla la conexión después del *Refresh.*
Expire: Contiene el número de segundos después de los cuales un DNS secundario debe considerar que los datos obtenidos del servidor primario han expirado.
TTL: Especifica el valor por defecto que tienen los registros de recursos en el Servidor de Nombres para responder a solicitudes DNS.

- NS: Identifica el Servidor de Nombres autoritativo, este registro contiene información completa sobre una parte del espacio del dominio y se dice que es una Autoridad en esa parte. Cada subdominio que tenga una representación diferente en el espacio de nombres de dominio, debe tener al menos un registro NS correspondiente. El formato para éste registro es el siguiente:

DOMANIN NAME IN NS HOSTNAME

El valor del registro NS es el nombre del servidor de nombres para ese dominio.

- ORIGIN: Nombre canónico del Servidor de Nombres primario para el dominio.

- A: Asocia direcciones IP con nombres de máquina, para cada máquina solo puede haber un registro A considerado el nombre oficial o canónico, cualquier otro nombre será un alias y debe estar incluido en un registro CNAME. A este registro debe ir asociado una dirección de 32 bits, separando los octetos por puntos. Por ejemplo, para la almacenar la información correspondiente a tx1.ejemplo.com, se tiene:

tx1.ejemplo.com IN A 172.16.41.108

- CNAME (Canonical Name): Establece un alias para un nombre ya definido. El siguiente fragmento ilustra el uso de los registros CNAME:

```
server1     IN    A        172.16.41.100
www   IN    CNAME    server1
ftp         IN    CNAME    server1
```

server1 es el nombre real del *host* que tiene como dirección IP 172.16.41.100, pero en ese mismo *host* se encuentran alojados los servicios www y ftp, por lo cual, mediante los registros CNAME es posible indicar que los alias www y ftp corresponden a server1 y por lo tanto a la dirección IP 172.16.41.100.

- PTR (Pointer): Se utiliza para obtener nombres de máquina a partir de direcciones IP (resolución inversa). Por cada Dirección IP debe existir solo un registro PTR.

- MX (Mail Exchange): Identifica un servidor de correo para el dominio. Este registro va acompañado de un número que establece la prioridad del servidor de correo, cuando en un dominio existe más de una máquina desempeñando esa función; de

esta forma, si el de mayor prioridad está ocupado, se pueden entregar los correos al dominio utilizando otros servidores. El formato para éste registro es el siguiente:

HOST.DOMAIN.NAME IN MX 10 HOST1.DOMAIN.NAME

Donde 10 indica la prioridad de ese servidor de correo. El registro con número de prioridad más bajo, tendrá la prioridad más alta.

- HINFO (Host Information): Proporciona información sobre el Hardware y Software de la máquina. Solo puede existir un registro HINFO por *host*. Por razones de seguridad estos registros rara vez son utilizados en DNS públicos.

- MB (MailBox): Registros de buzón, especifican el nombre de dominio del sistema principal que contiene el buzón de este objeto. El correo que se envía al dominio se dirigirá al sistema principal especificado en el registro MB. **[5]**

2.4.2 CARACTERÍSTICAS DE LA RESOLUCIÓN INVERSA

Para el soporte de la resolución inversa en IPV4, se definió y reservó un dominio especial en el espacio de nombres DNS de Internet llamado IN-ADDR.ARPA, con el fin de proporcionar un método garantizado para realizar el mapeo de direcciones IP a nombres de *hosts* y facilitar las consultas al ubicar todas las gateways en una red en particular. En este dominio los nombres están representados por cuatro etiquetas separadas por puntos, terminadas con el sufijo IN-ADDR.ARPA, cada etiqueta representa un octeto de la dirección IP, tomada en orden inverso; y se expresa como una cadena de caracteres por un valor decimal entre 0 y 255. Por ejemplo para la dirección 172.16.41.108, la representación en el dominio especial es 108.41.16.172.IN-ADDR.ARPA. El orden inverso de la dirección, un poco complicado para la lectura, se hace necesario, ya que, la lectura de las direcciones IP se realiza de izquierda a derecha, para ver inicialmente la información más general (una dirección IP de red) y posteriormente la más específica (una dirección IP de *host*) contenida en los últimos octetos, además esto permite a las zonas, ser delegadas exactamente en una red de espacio de direcciones. Por ejemplo, 30.IN-ADDR.ARPA puede ser una zona que contiene información para ARPANET, mientras que 45.IN-ADDR.ARPA puede ser una zona separada para MILNET.

2.5 ESPECIFICACIONES DNS PARA EL PROTOCOLO IPV6

La forma de almacenar las direcciones Internet y sus correspondientes nombres de máquina en el Sistema de Nombres de Dominio utilizada en el protocolo IPv4, no permite el soporte de direcciones IPV6, debido a que las aplicaciones asumen que las consultas retornan solamente direcciones IPV4 de 32 bits (Registro de Recurso A). Por esta razón y para brindar un soporte completo a la nueva especificación IPV6, se definen dos nuevos tipos de registros que soportan el almacenamiento de direcciones IPV6, los Registros de Recursos tipo **AAAA** y los **A6**; dos nuevos dominios, **IP6.ARPA** e **IP6.INT**, para realizar las búsquedas basadas en direcciones IP; un nuevo Registro de Recurso denominado **DNAME** y un nuevo **tipo de consulta**. Los demás tipos de Registros de Recursos, nombrados en la sección anterior, mantienen sus especificaciones tanto para el Protocolo IPv4 como para IPv6, debido a que su definición no depende de la longitud en bytes de la dirección IP asociada.

2.5.1 Resolución Directa

2.5.1.1 Registros AAAA

Se utilizan para mapear un nombre de *host* en su correspondiente dirección IP de 128 bits, esta dirección se codifica en el campo datos del registro de recurso AAAA, colocando primero el byte de mayor orden; de esta forma, una consulta de tipo AAAA para un dominio específico, retorna todos los registros AAAA asociados a la respuesta en la sección ANSWER (Respuesta). El tipo de consulta con RR AAAA, no origina procesamiento de Sección Adicional. La Sección Adicional se utiliza para mejorar el rendimiento del DNS y contiene los registros de recurso que están asociados con las respuestas, dada una consulta; pero no son estrictamente las respuestas de dicha consulta. **[6]**

El Registro de Recurso AAAA utiliza la representación textual de una dirección IPV6 para almacenar información en el campo datos; la representación textual de una dirección IPV6, se describe en el RFC 3513, donde se definen 3 formas convencionales para ello:

- La forma más utilizada es del tipo x:x:x:x:x:x:x:x, donde las x son los valores hexadecimales de los 8 segmentos de 16 bits que hacen parte de la dirección, cada segmento está separado por el símbolo ":" y abarca valores desde 0000 hasta FFFF. Por ejemplo: 2001:448:1024:1:0:0:0:10 es la representación textual de la dirección IPV6 del servidor DNS IPv6 para Unicauca. No es necesario escribir los 4 ceros para cada segmento, basta con colocar un numeral en cada campo.

- Teniendo en cuenta que muchas direcciones pueden tener largas cadenas de ceros, el segundo método se basa en la compresión de los bloques continuos de ceros, empleando el signo "::" para ofrecer una sintaxis más simple. El uso de "::" indica que en ese lugar van uno o más grupos de 16 bits compuestos por ceros, esta sintaxis solo puede ser utilizada una vez en cada dirección. Por ejemplo, la dirección 2001:448:1024:1:0:0:0:10 quedaría representada de la forma 2001:448:1024:1::10, indicando que entre 1024 y 10 hay 3 segmentos de ceros, que son los que hacen falta para completar los 128 bits o los 8 segmentos de la dirección IPV6.

- Una forma alternativa para la representación de direcciones, se presenta cuando en una red existen nodos IPV4 e IPV6, por lo cual las direcciones IPV4 deben representarse de la forma x:x:x:x:x:x:d.d.d.d, donde las x son los valores hexadecimales de los segmentos de 16 bit de mayor orden correspondientes a la dirección IP, y las d son los valores decimales de los 4 segmentos de 8 bits de bajo orden de la dirección IP (representación estándar IPV4). Por ejemplo, si se tiene la dirección IPV4 172.16.41.108, su correspondiente representación en IPV6 es 0:0:0:0:0:0:172.16.41.108 o mejor ::172.16.41.108, indicando que los 6 primeros segmentos de 16 bits están compuestos por ceros y los 2 segmentos siguientes (formando así los 128 bits), corresponden a los 32 bits de la dirección IPV4. **[7]**

2.5.1.2 REGISTROS A6

Al igual que el tipo de Registro de Recurso AAAA, el tipo A6 se utiliza para mapear un nombre de *host* en su correspondiente dirección IP de 128 bits, su creación y definición tienen como objetivo principal brindar soporte para la agregación de direcciones IPV6 (*IPv6 Address Aggregation*) y el *Renumbering* (re-numeración) en una red que utiliza el protocolo IPV6; la forma en la cual estos registros almacenan las direcciones IP, permite especificar un nuevo tipo de consulta a partir del tipo de consulta existente para localizar direcciones IPV4, que retorna direcciones IPV4 e IPV6 como parte del procesamiento de sección adicional.

Una dirección IPv6 se almacena en uno o más Registros de Recursos A6 y un Registro de Recurso A6 puede incluir una dirección IPV6 completa o un segmento de la dirección con información adicional de prefijos que conduzcan a la resolución de los segmentos faltantes. La información del prefijo comprende ***la longitud del prefijo y el nombre del DNS siguiente*** que contiene los Registros de Recursos A6 necesarios para construir una o más direcciones IPV6 completas. Esta forma de almacenamiento, tiene como característica fundamental agilizar los procesos de *renumbering* (re-numeración) y *multihoming* (multi-proveedor) en una red. **[8]**

Un cambio en la asignación de direcciones IP asociado a una subred, o a un *host* es conocido como **Re-numeración**. La Re-numeración puede ocurrir por varias razones, entre las cuales están:

- Trasladar la dirección IP de un *host* de una subred a otra, ocasionando cambios en la dirección IP del *host*.
- Fraccionar físicamente una subred a causa de la sobrecarga en el tráfico puede requerir re-numeración.
- Un cambio en el plan de direccionamiento en una organización, implica cambios tanto de las direcciones de *host*, como en los números de subredes.
- Cambiar de proveedor de servicios, lo cual origina un cambio en las direcciones IP que se habían asignado a la organización.
- La Asignación de un bloque de direcciones más largo por parte del proveedor de servicios, para contrarrestar el crecimiento de la información de enrutamiento en determinada red, requiere re-numeración de subred(es) y *host*(s). **[9]**

El *multihoming* se presenta cuando determinada red u organización cuenta con más de un proveedor de servicio (*Internet Provider Service ISP*) para realizar su conexión a Internet y se constituye en una técnica que permite incrementar la confiabilidad de dicha conexión, mejorar la tolerancia a fallos y ofrecer capacidades de ingeniería de tráfico. El *multihoming* puede presentarse en diferentes escenarios, entre los más importantes se encuentran:

- Un único enlace, con múltiples direcciones IP: Un *host* específico tiene múltiples direcciones Internet, por ejemplo, 2001:448:1024:1::10 y 2001:448:1024:1::12, pero hay un solo enlace físico *upstream*.

- Múltiples Interfaces y una única dirección IP por interfaz: Un *host* específico tiene múltiples interfaces y cada una de ellas cuenta con una única dirección IP, si uno de los enlaces falla la dirección asignada a ese enlace se convertirá en inalcanzable, pero el otro enlace seguirá funcionando.

- Múltiples enlaces y direcciones IP únicas: Con el uso de protocolos de enrutamiento, se anuncia el espacio de direcciones asignado a todos los enlaces,

cuando uno de los enlaces falla, el protocolo anuncia esta falla para que el tráfico no se envíe a través del enlace fuera de servicio.

- Múltiples enlaces y direcciones IP sin protocolo de enrutamiento: En este caso se utiliza un dispositivo especializado para balancear la carga del enlace entre el *firewall* y el enrutador de enlace, no se requiere una configuración especial en los enrutadores del ISP. Esta técnica permite utilizar todos los enlaces al mismo tiempo para incrementar el ancho de banda total disponible y detectar la saturación y las fallas de los enlaces en tiempo real para redireccionar de una forma eficiente el tráfico. **[10]**

Con el despliegue de IPV6 y el desarrollo de metodologías que permiten la fácil transición de IPV4 a IPV6, los conceptos de re-numeración y multi-proveedor se constituyen en un aspecto muy importante, y su implementación se hace necesaria, aunque no inmediatamente obligatoria, motivo por el cual, el estudio de los Registros de Recursos A6 y DNAME, se convierte en una muy buena base teórica y práctica para la definición de estrategias de direccionamiento.

El formato del Registro A6 es el siguiente:

Longitud de Prefijo de Dirección (1 OCTETO)	Identificador del Sufijo de Dirección (0-16 OCTETOS)	Nombre del Servidor DNS (0-255 OCTETOS)

- Longitud de Prefijo de Dirección: Corresponde al número de bits de la dirección IPV6 que el registro A6 NO resolverá, este campo está representado por un número entero entre 0 y 128; por ejemplo, si el presente registro A6 va a resolver 56 bits de la dirección total, éste campo debe tener un valor de 72 (128 - 56).

- Identificador del Sufijo de Dirección: Corresponde a la parte de la dirección IPv6 que será resuelta por el registro A6. Este campo debe contener en bits, un número igual a 128 (longitud de una dirección IPV6) menos el número de bits especificado en el campo Longitud del Prefijo. La longitud de este campo debe ser un número completo de octetos, para asegurar que esto se cumpla, después de efectuar la operación de resta se utiliza un relleno entre 0 y 7 bits; estos bits de relleno se fijan a cero, cuando se carga un archivo de zona y son ignorados en la recepción.

- Nombre del Servidor DNS: Representa el nombre del Servidor de Nombres de Dominio que contiene la resolución de los bits faltantes de determinada dirección IPV6, y por regla, éste nombre no debe estar comprimido. Si la Longitud del Prefijo es igual a 0, es decir que todos los bits de la dirección serán resueltos por un solo registro de recurso, el campo Nombre del Prefijo no estará presente; y si la Longitud del Prefijo es 128, el componente Sufijo de Dirección no estará presente.

Cuando un registro A6 apunte hacia otro registro A6 para resolver por completo una dirección, la Longitud de Prefijo del primero debe ser mayor que la Longitud de Prefijo del segundo, si se presenta el caso contrario, es decir, la Longitud del Prefijo del segundo mayor que la del primero, el RR A6 con la longitud de prefijo más grande será ignorado.

Para almacenar toda la información necesaria de una dirección IPV6 en el campo RDATA, la representación textual utilizada por el RR A6 comprende dos o tres campos separados con un espacio; estos campos corresponden a los 3 nombrados anteriormente que hacen parte del formato del registro A6. A continuación se enumeran algunos ejemplos para almacenar un nombre de dominio con registros A6.

A. Para el nombre equipo1.ejemplo.red.com cuya dirección IP es 2001:448:1024:1::2, su correspondiente registro A6 es:

Equipo1.ejemplo.red.com IN A6 96 2001:448:: tx1.ejemplo2.red.com

En el cual, Equipo1.ejemplo.red.com es el nombre a almacenar; A6 especifica el tipo de registro; 96 indica la longitud del prefijo, es decir que el registro resolverá 32 bits de los 128 en total; 2001:448:: indica la parte de la dirección que será resuelta por ese registro, es el identificador del sufijo y tx1.ejemplo2.red.com es el nombre del servidor DNS que contiene la otra parte de la dirección, los 96 bits faltantes.

B. Para éste mismo equipo se podrían realizar 2 tipos de configuraciones:

En el servidor DNS del dominio local:

Equipo1.ejemplo.red.com IN A6 0 2001:448:1024:1::2

En este caso, el nombre equipo1.ejemplo.red.com se almacena en un solo registro A6 el cual resuelve la dirección de una forma completa y la longitud de prefijo es 0 (cero), porque se están resolviendo 128 bits de 128 posibles y quedan cero por resolver. El campo Nombre del Servidor DNS no debe estar presente, ya que toda la dirección es resuelta solo por registro A6 y por un Servidor de Nombres.

En el servidor DNS del dominio local:

Equipo1.ejemplo.red.com IN A6 96 2001:448:: tx.red3.com

Y en el servidor DNS del próximo dominio:

tx.red3.com IN A6 0 1024:1::2

En este caso, el nombre equipo1.ejemplo.red.com, se almacena en dos registros A6, localizados en 2 servidores DNS diferentes. 32 bits (128 - 96) correspondientes a 2001:448 son resueltos por el DNS local y los restantes 96 son resueltos por el servidor de nombres tx.red3.com.

Para obtener la dirección o direcciones IPv6 pertenecientes a un nombre de máquina específico, el cliente DNS al realizar una petición para resolver un nombre de máquina dado, obtiene una o más cadenas completas de Registros de Recurso A6. Cada cadena de registros A6 debe empezar con un registro que contenga el nombre a resolver, incluir uno que contenga el nombre del prefijo; y terminar con un RR A6 cuya longitud de prefijo sea cero, indicando la finalización de la búsqueda. Una vez obtenida toda esta información, la dirección se construye tomando el valor de la posición de cada bit del primer registro A6 obtenido hasta el bit indicado por el campo Longitud de Prefijo. El conjunto de todas las direcciones IPv6 para el nombre de máquina especificado comprende las direcciones formadas a partir de las cadenas completas

de registros A6 que comienzan en ese nombre, descartando los registros que tienen Longitud de Prefijo inválido. **[8]**

Un ejemplo completo acerca de la utilización de los registros A6, ilustra las facilidades que estos proporcionan para aplicar la re-numeración y el multi-proveedor en los escenarios que así lo requieran, la arquitectura de este ejemplo se aprecia en la Figura 8 y su desarrollo se encuentra a continuación:

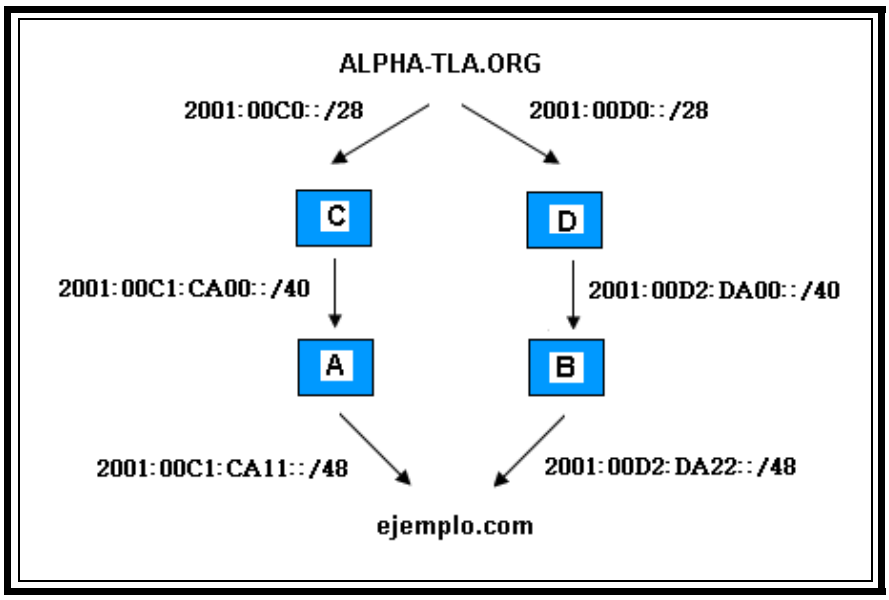


Figura 8. Arquitectura de direcciones para el manejo de RRs A6

Una red perteneciente al dominio ejemplo.com, cuenta con dos proveedores de servicios para su conexión a Internet, los proveedores A y B, éstos a su vez dependen de los proveedores C y D respectivamente, los cuales pertenecen al mismo TLA (top-level aggregate) con un identificador “2001”. La autoridad ALPHA-TLA.ORG asigna el prefijos NLA (next level aggregate) 2001:00C0::/28 al proveedor C y 2001:00D0::/28 al proveedor D.

El proveedor C asigna el prefijo NLA 2001:00C1:CA00::/40 a A, y D le asigna el prefijo 2001:00D2:DA00::/40 a B. A asigna el identificador de suscripción “11” a la red perteneciente al dominio ejemplo.com y B le asigna el identificador de suscripción “22”. De esta forma, los prefijos de dirección para la red de ejemplo serán:

*2001:00C1:CA11::/48 para rutas a través del proveedor C
*2001:00D2:DA22::/48 para rutas a través del proveedor D

En la red hay un nodo llamado N1.ejemplo.com perteneciente a la subred 1, al cual se le asigna un identificador de interfaz 1234:5678:9ABC:DEF0, sus direcciones IPV6 completas son:

*2001:00C1:CA11:0001:1234:5678:9ABC:DEF0
*2001:00D2:DA22: 0001:1234:5678:9ABC:DEF0

El nodo N1 pertenece al dominio ejemplo.com, los proveedores A, B, y D al dominio A.net, B.net, C.net y D.net respectivamente, por lo tanto el archivo de zona directa

para ejemplo.com, tiene las siguientes líneas para almacenar el nombre del nodo N1.ejemplo.com con registros A6:

```
$ORIGIN ejemplo.com
 N1                IN      A6      48      ::1:1234:5678:9ABC:DEF0   N2.A.net
 N1                IN      A6      48      ::1:1234:5678:9ABC:DEF0   N3.B.net
```

Este servidor resolverá los últimos 80 bits de las correspondientes direcciones IPv6 para el nodo N1, y se asigna la resolución de los 48 bits restantes a los servidores de nombre N2.A.net y N3.B.net. En los servidores de nombre de los dominios A y B quienes son proveedores directos de ejemplo.com, se especifican las siguientes líneas para seguir con el almacenamiento de la dirección para N1.ejemplo.com:

```
N2.A.net          IN      A6      40      11::      N4.C.net
N3.B.net          IN      A6      40      22::      N5.D.net
```

Estos registros resolverán los siguientes 8 bits de las direcciones IPv6, debido a que los identificadores de suscripción entregados a ejemplo.com por parte de los proveedores A y B son 11 y 22 respectivamente. Los siguientes servidores de nombres que continuarán con el almacenamiento son N4.C.net y N5.D.net, pertenecientes a los proveedores C y D que se constituyen en directos para A y B e indirectos para ejemplo.com. Estos servidores deben tener los siguientes registros:

```
N4.C.net          IN      A6      28      0001:CA00::   N6.ALPHA-TLA.ORG
N5.D.net          IN      A6      28      0001:DA00::   N7.ALPHA-TLA.ORG
```

Estos registros resolverán los siguientes 12 bits de las direcciones, y los servidores N6 y N7 se encargarán de los 28 bits restantes.

Y finalmente en ALPHA-TLA.ORG, la entidad que asignó los prefijos a los proveedores C y D deben estar presentes las siguientes líneas, indicando que el número de bits por resolver es cero, es decir, que ya se tienen las direcciones completas:

```
N6.ALPHA-TLA.ORG      IN      A6      0       2001:00C0::
N7.ALPHA-TLA.ORG      IN      A6      0       2001:00D0::
```

La forma de almacenar una dirección IPv6 explicada anteriormente, representa la mejor alternativa cuando de re-numeración y multi-homing se trata, ya que un cambio por ejemplo en el prefijo 2001:00C0 implicaría solo una reforma en el servidor de nombres ALPHA-TLA.ORG, dejando las otras configuraciones intactas. Pero esta alternativa también presenta un punto desfavorable y es el relacionado con la disponibilidad de todos los servidores de nombre para que el nombre de *host* N1.ejemplo.com sea resuelto, es decir, si uno de los servidores esta por fuera cuando una consulta de éste tipo se presente, no podrá ser resuelta, pues se necesita de cada uno de ello para ir formando la dirección.

Para este mismo ejemplo, los registros A6 ofrecen al administrador de una red una alternativa diferente, que consiste en resolver en el servidor de nombres ejemplo.com toda la dirección, para esto, los registros quedarían de la siguiente forma:

```
$ORIGIN ejemplo.com
N1          IN      A6      0       2001:00C1:CA11:0001:1234:5678:9ABC:DEF0
N1          IN      A6      0       2001:00D2:DA22:0001:1234:5678:9ABC:DEF0
```

Esta alternativa frente a una posible re-numeración o cambio de proveedor, presentará muchas complicaciones debido a la extensa longitud de las direcciones y al número de *host* que el servidor no solo de nombre sino de otro tipo de aplicación deba manejar.

Una solución, teniendo en cuenta los pros y contras de cada una de las alternativas, es utilizar la primera, pero sin tener que depender de tantos servidores, es decir, maximizar el número de bits que resuelve un servidor para dejar de depender de él.

2.5.1.3 Registros DNAME

El registro de recurso DNAME, permite asignar nombres alternativos a una porción del sub-árbol del espacio de nombres de dominio, en lugar de asignarlos a un único nodo (función desarrollada por el registro CNAME), logrando así que el sufijo de una consulta a un nombre de máquina sea sustituido por un nombre almacenado en el campo RDATA del registro DNAME.

La definición de éste nuevo tipo de registro, plantea una solución al problema del mantenimiento de la resolución de direcciones a nombres, en un contexto de re-numeración de red. Sin el mecanismo DNAME, un Servidor DNS debe reconfigurar la zona relacionada con la resolución inversa, cuando haya cambios en la numeración de la red; utilizando los registros DNAME, la zona puede ser definida de tal forma que no sea necesaria modificación alguna, cuando ocurra re-numeración o cambio de nombre en una unidad administrativa.

Para explorar todas las características que el mecanismo DNAME ofrece, los algoritmos de resolución de nombre son modificados levemente, tanto para los servidores como para los *resolvers.* Esta modificación incorpora un proceso de sustitución de un nombre, bajo el control de un registro DNAME llamado "*the DNAME substitution*".

A continuación, se citan algunos ejemplos, para los casos en los cuales, es importante el uso de los registros DNAME:

- Cambio de Nombre de la Organización

  Si una organización con un nombre de dominio servidor1.ejemplo, se convierte en parte del dominio servidor2.ejemplo, el proceso para actualizar la información en los archivos de zona, es muy sencillo, se debe agregar la siguiente línea:

  servidor1.ejemplo. DNAME servidor2.ejemplo.

  De esta forma, si se realiza una consulta a una aplicación del dominio anterior por ejemplo a www.servidor1.ejemplo, en la sección Answer se incluirá el registro DNAME descrito anteriormente, indicando que ahora el dominio es servidor2.ejemplo, y los RRs pertinentes para la resolución de la aplicación Web con el nuevo dominio, www.servidor2.ejemplo.

- Soporte para Re-numeración de Red.

  Este soporte ofrece la posibilidad de cambiar el dominio asignado a determinado espacio de direcciones, sin la necesidad de modificar la descripción e información utilizada en los archivos de zona del anterior dominio. Este soporte tiene mucha relación con la resolución inversa, ya que en los archivos de zona que contienen la asociación de direcciones IP a nombres de máquina, es posible especificar parte

de la dirección y delegar la otra a un dominio de más alto nivel, que puede ser el que maneja el proveedor de la red, de forma muy similar a lo expresado en el ejemplo con registros A6 anteriormente explicado. La asociación de registros DNAME debe especificarse con etiquetas Bit-String, su explicación se encuentra en la seccion 2.5.2.2. **[11]**

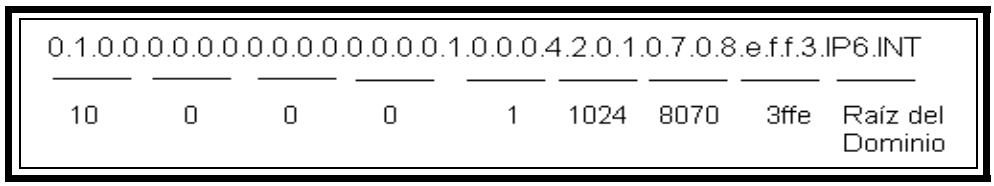
Los registros DNAME y A6 tienen una estructura un poco compleja cuando de implementación se trata, además no todos los escenarios de red requieren las características que estos registros ofrecen, en especial las relacionadas con multi-proveedor y re-numeración. Por esta razón estos registros han sido relegados a estado experimental según los RFC 3363 y 3364, ya que se considera más viable utilizar los registros AAAA en la transición de IPv4 a IPv6 para que el proceso sea más rápido y eficiente. Esta también es una de las razones por las cuales los desarrolladores de aplicaciones en sus últimas versiones de código, suprimieron el soporte para los registros A6 y DNAME, dando prioridad a los AAAA, debido a la falta de un estudio más profundo que defina la viabilidad de los registros A6 y DNAME.

2.5.2 RESOLUCIÓN INVERSA

2.5.2.1 Dominios Definidos para este tipo de resolución

Para las resoluciones inversas (conversión de dirección IP a nombre de máquina) dada una dirección IPV6, en el año de 1995 con el RFC 1886 se define un nuevo dominio de soporte para la resolución inversa basada en direcciones IPV6 llamado **IP6.INT**, enraizado en IP6.INT. Para representar una dirección IPV6 en este dominio, se especifica el formato *nibble*, que como su nombre lo indica, maneja una secuencia de nibbles separados por puntos con el sufijo “IP6.INT”. Un nibble es un conjunto de 4 dígitos binarios, y es utilizado por este dominio, gracias a que éste puede representar cualquier número hexadecimal, por ejemplo 1110 = E, 1100 = C, etc.

La secuencia de nibbles, que representen una dirección IP, deben asignarse en forma inversa, es decir, primero va el nibble de menor orden, seguido del nibble próximo de menor orden, y así sucesivamente. Por ejemplo, para 3ffe, su correspondiente en nibble es 3.f.f.e, y cumpliendo la norma del bit nibble más bajo primero, queda e.f.f.3 y la representación de la dirección completa 3FFE:8070:1024:1::10 en el dominio IP6.INT, es la siguiente:



Posteriormente en el año 2000, se define otro dominio para dar soporte a la resolución inversa basada en direcciones IPV6, el dominio **IP6.ARPA** enraizado en IP6.ARPA. La idea de crear este nuevo dominio nace de la necesidad de tener en el espacio de direcciones IPV6 una zona para el mapeo inverso análoga a la zona IN-ADDR.ARPA definida para el protocolo IPV4, teniendo en cuenta que según recomendaciones de la IAB (Internet Architecture Board) el dominio de alto nivel ARPA debe ser utilizado para crear sub-dominios y fue establecido como el dominio propio de Internet.

En este contexto, el uso del dominio IP6.INT entró en una fase de reducción progresiva (eliminación progresiva), se especificó que en las nuevas implementaciones

no era apropiado utilizar éste dominio, y finalmente en el año 2005 la IANA (Internet Assigned Numbers Authority) desaprueba totalmente el dominio IP6.INT para las normas que conforman la implementación del Protocolo IPV6 y oficializa el uso del dominio IP6.ARPA.

2.5.2.2 Representación de las direcciones en el dominio inverso

Con la definición del nuevo tipo de registro de recurso A6, también se define un nuevo esquema para la resolución de búsquedas inversas (conversión de dirección IP a nombre de *host*), basado en un nuevo tipo de etiquetas DNS llamadas "bit-string label" [BITLBL]. Este nuevo tipo de etiquetas hace posible que los registros de recurso sean almacenados en una sección binaria del árbol de Nombres de Dominio y ofrece una solución eficiente al problema relacionado con el almacenamiento de datos y la delegación de Autoridades cuando la estructura fundamental del espacio de nombres está en su mayor parte representada en binario.

BITLBL representa una secuencia de etiquetas "*One-Bit*", hasta 256 etiquetas "One-Bit" pueden ser agrupadas en una sola etiqueta "*bit-string*"; en éstas, los bits más significativos deben aparecer al comienzo, a diferencia de las etiquetas DNS, en las cuales los bits menos significativos van al principio.

El nuevo esquema para la búsqueda inversa utiliza un subconjunto de la sintaxis definida para las etiquetas BITLBL en el RFC 2673, denominado representación textual. Esta representación consta de un indicador base representado por "x", indicando que es base 16 (hexadecimal) y una secuencia de dígitos hexadecimales encerrados entre el delimitador "\[""]" y ordenados de más a menos significativos. La cadena de dígitos puede estar seguida de un "/" y un indicador de longitud decimal, si no se especifica el indicador de longitud, éste tomará un valor por defecto de 4 veces el número de dígitos hexadecimales encerrados entre los corchetes. **[12]**. Por ejemplo, si se desea relacionar un nombre de máquina a la dirección IPV6 3ffe:8070:1024:1::10, la representación debe ser la siguiente:

\[x3FFE80701024000100000000000000010/64].IP6.ARPA.

Como se observa entre "\[]" va la cadena de dígitos precedida de la letra x, que especifica la dirección IPV6, esta cadena debe ir sin puntos y se deben colocar todos los dígitos de cada uno de los octetos que forman la dirección, es decir, si la dirección tiene un octeto con el número 10, debe colocarse 0010.

Sin embargo, debido a que el tipo de representación utilizada en el dominio IN-ADDR.ARPA para IPv4 es muy similar al formato nibble, definido en el RFC 1886, este último es el más utilizado en la resolución inversa.

2.5.3 Modificación a los Tipos de Consulta existentes.

Los tipos de consulta existentes (los definidos para IPV4) para realizar procesamiento de sección adicional, son redefinidos de tal forma que permiten llevar a cabo el procesamiento de Registros de Recurso A, AAAA y A6; esta modificación consiste en que el Servidor de Nombres de Dominio puede agregar información de direcciones

tanto IPV4 como IPV6 accesibles localmente, a la Sección Adicional de la respuesta cuando se esté procesando una consulta de tipo Servidor de Nombre (NS), Intercambio de Correo (MX) o Buzón de Correo (MB). Esto se hace con el fin de facilitar las futuras consultas DNS, y de garantizar interoperabilidad entre redes que utilizan sólo el protocolo IPV4 ó IPV6, o que tienen configuradas aplicaciones dual-stack (IPV4 e IPV6).

2.6 IMPLEMENTACIONES DEL PROTOCOLO DNS CON SOPORTE IPV6

2.6.1 BIND

BIND (Berkeley Internet Name Domain) es el Servidor de Nombres de Dominio más popular, utilizado por la mayoría de los servidores de Internet y disponible bajo licencia BSD (Berkeley Software Distribution), se basa en una estable y robusta arquitectura sobre la cual una organización puede construir su arquitectura de nombres de dominio. La librería *resolver* que incluye la distribución de BIND contiene los API estándar para la interpretación entre nombres de dominio y direcciones IP, además permite realizar conexiones con aplicaciones que requieren el servicio DNS. Gracias a su popularidad, existe una gran cantidad de documentación, relacionada con su configuración y comportamiento. La distribución contiene 3 componentes principales:

- Un sistema de servidor de nombres de dominio NAMED.
- Un sistema de librerías *resolver*.
- Herramientas para verificar el correcto funcionamiento del Servidor DNS.

BIND en su versión 9 y posteriores ofrece soporte para la resolución de nombres de máquina y direcciones IP en redes que utilizan el Protocolo IPV6, según los estándares definidos para el servicio DNS bajo éste protocolo. BIND 9 soporta el uso de los registros AAAA (RFC 3596) y A6 (RFC 2874), el formato nibble utilizado en el dominio ip6.arpa para las búsquedas inversas y el formato de etiquetas binarias “bitstring”.

2.6.2 DJBDNS

En comparación con BIND, este paquete es relativamente nuevo y se basa en una estructura modular, es decir, pequeños programas llevan a cabo las diferentes tareas que un servidor de nombres realiza. Fue diseñado y escrito teniendo en cuenta como característica esencial la seguridad, factor que lo hace verdaderamente interesante, pero al no tener licencia *Open Source*, debido a sus bajos niveles de popularidad y poca documentación, este paquete no resulta tan atractivo para los Administradores de red a la hora de implementar y configurar un DNS.

Con respecto al soporte IPv6, esta implementación cuenta con un parche o librería adicional (librería extra al código fuente), que permite resolver consultas con registros AAAA, por lo cual no se garantiza su satisfactoria instalación en todos los sistemas operativos.

2.7 INSTALACIÓN Y CONFIGURACIÓN DEL SERVIDOR DNS SELECCIONADO

Para la configuración del Servidor de Nombres de Dominio se trabajará con BIND, de acuerdo a las características especificadas en la sección anterior y teniendo en cuenta que brinda soporte para nodos IPV4, IPV6 y *Dual stack* (IPV4 e IPV6) a partir de la versión 9 y posterior.

La arquitectura de funcionamiento de BIND se basa principalmente en el archivo *named.conf*, donde es posible especificar las zonas directas e inversas que harán parte del domnio a administrar y las opciones generales del servidor. Cada zona, tanto directa como inversa, tiene asignado un archivo en el cual se asigna la autoridad de dicha zona, los correspondientes mapeos y los parámetros de tiempo para el intercambio de mensajes con los cuales trabajará. A continuación se describe el proceso para instalar BIND y generar los archivos anteriormente mencionados.

2.7.1 Instalación

El equipo sobre el cual se implementaron los servicios es el mismo sobre el cual está configurado el túnel IPv6 en Fedora Core 4, razón por la cual la instalación del Servidor de Nombres de Dominio se realizó en éste Sistema Operativo.

A. En la URL http://www.isc.org/index.pl?/sw/bind/bind9.php se descarga el instalador con extensión .tar.gz más reciente para BIND y se instala ejecutando:

tar –xvzf Nombre del archivo.tar.gz
make
make install.

Para realizar las prácticas se instaló bind-9.3.1-4.

B. Con los Cds de instalación de Fedora Core 4 también es posible instalar el servidor DNS BIND, accediendo a Escritorio- Configuración del Sistema- Añadir/Eliminar Aplicaciones y seleccionando Servidores de Nombre DNS.

C. Una vez instalado el paquete, se verifica que en */etc/* se encuentre el archivo para realizar la configuración del servidor, *named.conf*.

D. Los comandos utilizados para iniciar y parar el servidor desde un terminal son */etc/init.d/named start* y */etc/init.d/named stop* respectivamente. Es necesario tener privilegios de administrador para ejecutar dichos comandos o para realizar cambios en los archivos de configuración.

2.7.2 Conceptos generales sobre los archivos de configuración

2.7.2.1 Archivo principal de configuración

El archivo de configuración de BIND es *named.conf*, cuya estructura se ilustra en la Figura 9.

```
Options         {
                };

controls        {
                };
zone "Nombre de la Zona Directa" IN {
      type XXXX;
      file "ruta del archivo de zona";
                };
zone "Nombre de la zona Inversa" IN {
      type XXXX;
      file "ruta del archivo de zona";
                };
```

Figura 9. Archivo de configuración de BIND

En ***options*** se configuran las opciones globales que serán utilizadas para que BIND funcione de manera correcta. Las opciones más utilizadas en esta parte de la estructura son:

- ***Directory***: Especifica el directorio en el cual se encuentran los archivos de zona que cargará la aplicación, una vez el comando start ha sido ejecutado.

- ***Statistics-file***: Especifica la ruta del archivo en el cual se almacenan las estadísticas que presenta el servicio. El valor por defecto para ésta opción es /var/named/data/named_stats.txt.

- ***Listen-on:*** Opción utilizada para indicar las interfaces y puertos por los cuales el servidor responderá a las consultas, cuando se trabaja sobre el protocolo IPV4. Si no se especifica el valor del puerto, por defecto tomará el 53.

- ***Listen-on-v6:*** Es la redefinición de la opción listen-on para consultas hechas a través del protocolo IPV6. Si esta opción no se encuentra en el archivo de configuración se debe adicionar. Su sintaxis es la siguiente:

  - **listen-on-v6 { lista de direcciones; }**

  Si el valor de la lista de direcciones es *{any;}*, el servidor no requerirá un socket por separado para cada dirección de interfaz IPV6, como lo hace para IPV4, si el sistema operativo cuenta con el API correcto para el soporte de IPV6. Si se especifican varias direcciones IPV6, el servidor escuchará sobre un socket separado en cada dirección especificada.

- ***Forwarders***: Le permite al servidor realizar peticiones a los servidores que en ésta opción se encuentran especificados, cuando éste no sea capaz de responder a una consulta realizada.

- ***Allow-v6-synthesis***: Esta opción fue introducida en la versión 9 de BIND para facilitar la transición de los registros AAAA a los registros A6 y de las etiquetas nibble a las etiquetas binary.

- ***Allow-query***: Contiene las direcciones de los *host* que pueden hacerle peticiones al DNS. Si no se especifica ésta opción, todos lod *host*s podrán realizar peticiones.

- ***Allow-transfer:*** Contiene las direcciones de los *hosts* que pueden recibir transferenncias de zonas desde el servidor.

- ***Query-source***: Se utiliza cuando el servidor no sabe la respuesta para determinada consulta y debe consultarla a otro Servidor de Nombres, y contiene el puerto y dirección IPV4 utilizados para tal fin. Para IPV6 ésta opción se convierte en Query-source-v6.

En ***controls***, van los parámetros de seguridad para la utilización de rndc, una herramienta que permite la administración de línea de comandos del demonio *named* desde el *host* local o desde un *host* remoto, para prevenir el acceso no autorizado a la herramienta BIND. La sintaxis que se utiliza para declarar parámetros en la sección controls es:

```
controls  {
          inet ( dirección IP) port ip_port allow { lista de direcciones }
           keys { lista de llaves };
           }
```

Donde inet es un canal de control que especifica un socket TCP para escuchar sobre un puerto (port ip) y una dirección IPV4 o IPV6.el parámetro *keys* se utiliza como mecanismo para autorizar la ejecución de comandos sobre el canal de control.

Zone permite identificar cada una de las zonas tanto directas como inversas que hacen parte del dominio que se desea administrar y especificar las características que cada una de ellas tendrá. Los parámetros que se pueden configurar aquí son:

- ***Type***: Como su nombre lo indica, especifica el tipo de zona, entre los cuales se encuentran:
  - ***Master:*** Se utiliza cuando el servidor está en capacidad de proveer respuestas autoritativas para esa zona. Contiene todas las configuraciones de dicha zona.
  - ***Slave***: Se utiliza cuando la zona es una una réplica de una zona tipo master, todos los cambios de configuración realizados en la zona master, serán transferidos a la zona slave para mantener una copia de ellos y servir como respaldo.
- ***Forward***: Este tipo de zona se constituye en una forma de de configurar un forwarders para una zona en especial. En este tipo de zonas se especifican los forwarders que se aplicarán para la resolución de peticiones dentro del dominio para ese nombre de zona.
- ***Hint:*** Especifica un conjunto de servidores de nombre raíces, que utilizará una vez se inicie el servidor, para obtener la lista más reciente de servidores de nombre.

- ***Options:*** las opciones que se pueden configurar en esta sección son entre otras *Allow-transfer y Allow-query*, las cuales ya fueron explicadas.

2.7.2.2 Archivos de Zona

2.7.2.2.1 Archivo de Zona Directa

Se define el nombre para el dominio y de acuerdo a éste se crea la zona directa en un archivo de zona que debe estar ubicado en el directorio */var/named/*, que es el directorio que por defecto utilizará el demonio *named* para cargar las zonas, para este propósito se puede usar como guía el archivo *db.local* creado al momento de la instalación del servidor, éste archivo contiene los siguientes parámetros básicos de configuración:

```
$ORIGIN.
@       IN      SOA

@       IN      NS
@       IN      MX

xxx     IN      A
yyy     IN      AAAA
zzz     IN      A6
nnn     IN      CNAME
ppp     IN      DNAME
```

$ORIGIN indica cual es el dominio para el cual ese archivo se encargará de responder de forma autoritativa. De ahí en adelante, no es necesario escribir el nombre completo del dominio, ya que éste será reemplazado por el símbolo @. El nombre del dominio debe terminar con un punto indicando el nodo raíz. Por ejemplo si el dominio definido es ejemplo.com, en el archivo de zona debe aparecer $ORIGIN ejemplo.com.

Posteriormente se indica el inicio de autoridad mediante el registro SOA, donde se especifica el equipo y el dominio donde se encuentra el servidor de nombres, y la dirección de correo del administrador encargado de la zona para notificación de posibles errores. Este registro involucra la definición de otras variables que controlan los tiempos relacionados con actualizaciones (Ver Numeral 1.4.1).

De aquí en adelante, se utilizan los registros de recursos (explicados en el numeral 1.4.1) necesarios para definir el servidor de nombres, nombres de máquinas con direcciones IPV4 o IPV6, servicios para el dominio, etc.

2.7.2.2.2 Archivo de Zona Inversa

Se crea la zona inversa para el mapeo de direcciones a nombres de máquina, cuyo archivo de zona debe estar en */var/named/.* para este propósito se puede usar como guía el archivo *db.127* creado al momento de la instalación del servidor. Los parámetros básicos para la configuración de éste archivo son:

```
$ORIGIN
@       IN      SOA

@       IN      NS
@       IN      MX
        IN      PTR     nombre de máquina
        IN      PTR     nombre de máquina
```

En éste archivo se utilizan los registros de recursos definidos para la resolución inversa, convirtiendo direcciones IPV4 e IPV6 en nombres de máquina. Los parámetros restantes se definen de igual forma que para el archivo de zona directa.

2.7.3 CONFIGURACIÓN DEL SERVIDOR DNS IPV6 UNICAUCA

Teniendo en cuenta que la Universidad del Cauca cuenta con el registro del dominio **unicauca.edu.co**, se decidió asignar la parte relacionada con los servicios IPV6 al dominio **IPv6.unicauca.edu.co** y tratarlo como un subdominio del dominio principal de Unicauca para tener autonomía en la asignación de servicios. La idea principal de la asignación de un subdominio consiste en delegar la autoridad de una parte del dominio a una máquina diferente a la que maneja el servidor de nombres principal. La Universidad del Cauca tiene asignado el bloque de direcciones de producción 2001:448:1024::/64 y todos los servicios IPV6 (DNS, Correo, WEB, FTP y SSH) correrán en la máquina con nombre **zeus.IPv6.unicauca.edu.co**, con dirección IPV4 real 208.195.214.50, cuya equivalente interna es 10.200.2.50 y con dirección IPv6 2001:448:1024:1::10.

Para realizar la delegación de IPv6.unicauca.edu.co en unicauca.edu.co, se agregan las siguientes líneas al archivo *named.conf* que define las zonas para unicauca.edu.co y se encuentra ubicado en el equipo dns1.unicauca.edu.co de la Red de Datos de la Universidad del Cauca.

```
Zone “IPv6.unicauca.edu.co” IN
{
        type forward-,
        forwarders {};
        forward only;
}
```

Éstas líneas le indican al servidor de nombres encargado de administrar el dominio principal unicauca.edu.co, que solamente para resolver consultas relacionadas con el subdominio IPv6.unicauca.edu.co se deben ignorar los *forwarders* definidos en la sección *options* de *named.conf.*

En el archivo de zona directa para unicauca.edu.co en la Red de Datos se adiciona lo siguiente:

```
IPv6                        IN    NS    zeus.IPv6.unicauca.edu.co.
zeus.IPv6.unicauca.edu.co.  IN    A     208.195.214.50
```

La primera línea indica que IPv6.unicauca.edu.co es un subdominio de unicauca.edu co y será administrado por el equipo zeus.IPv6.unicauca.edu.co. La segunda línea se utiliza para almacenar la dirección IPV4 real del equipo que servirá de servidor de nombres.

Una vez realizada la delegación del subdominio, en el equipo **zeus** se instala BIND 9.3.1-4 como se indicó anteriormente. El archivo *named.conf* para el servidor DNS IPV6 se aprecia en la Figura 10, aquí es indispensable especificar la opción *listen-on-v6*, para que BIND escuche peticiones IPV6. Como *forwarders* se definen los servidores de nombre de la Universidad del Cauca, para resolver consultas relacionadas con dominios diferentes a IPv6.unicauca.edu.co.

```
options {
        directory "/var/named";
        dump-file "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        listen-on-v6{any;};
        forwarders{ 172.16.255.183 ; 172.16.255.200;};

zone "1.0.0.0.4.2.0.1.8.4.4.0.1.0.0.2.ip6.arpa" IN {
        type master;
        file "ipv6unicaucainv.local";
};

zone "ipv6.unicauca.edu.co" IN {
        type master;
        file "ipv6unicauca.zone";
};
```

Figura 10. Archivo de configuración named.conf

Éste archivo contiene la definición de las 2 zonas encargadas de resolver peticiones DNS para el dominio IPv6; la declaración de zona IPv6.unicauca.edu.co se refiere a la resolución directa y especifica el nombre del dominio para el cual el servidor de nombres zeus va a tener autoridad, el archivo donde se almacenan todos los registros de recursos se encuentra en */var/named/IPv6unicauca.zone*; la zona 1.0.0.0.4.2.0.1.8.4.4.0.1.0.0.2.ip6.arpa hace referencia a la resolución inversa, la definición se encuentra en formato nibble e indica que se encargará de resolver peticiones para la red 2001:448:1024:1:: utilizando el dominio IP6.ARPA; el archivo que contiene los registros de recursos se encuentra en */var/named/IPv6unicaucainv.local.*

```
$ORIGIN ipv6.unicauca.edu.co.
@       IN      SOA     zeus.ipv6.unicauca.edu.co.   root.ipv6.unicauca.edu.co. (
                                        1               ; serial (d.adams)
                                        3H              ; refresh
                                        15M             ; retry
                                        1W              ; expiry
                                        1D )            ; minimum
                 IN NS          zeus.ipv6.unicauca.edu.co.
@                IN MX  0       zeus.ipv6.unicauca.edu.co.

dns6             IN      AAAA            2001:448:1024:1::10
www              IN      AAAA            2001:448:1024:1::10
www              IN      A               208.195.214.50
zeus             IN      AAAA            2001:448:1024:1::10
zeus             IN      A               208.195.214.50
ftp              IN      AAAA            2001:448:1024:1::10
```

Figura 11. Archivo de zona directa IPv6unicauca.zone

```
@       IN      SOA     ipv6.unicauca.edu.co. root.ipv6.unicauca.edu.co. (
                                        1               ; serial (d.adams)
                                        3H              ; refresh
                                        15M             ; retry
                                        1W              ; expiry
                                        1D )            ; minimum

@               IN      NS              zeus.ipv6.unicauca.edu.co.
@               IN      MX      0       zeus.ipv6.unicauca.edu.co.

0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0 IN      PTR     www.ipv6.unicauca.edu.co.
0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0 IN      PTR     dns6.ipv6.unicauca.edu.co.
0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0 IN      PTR     ftp.ipv6.unicauca.edu.co.
0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0 IN      PTR     zeus.ipv6.unicauca.edu.co.
```

Figura 12. Archivo de zona inversa IPv6unicaucainv.local

En el archivo de zona directa (Figura 11), $ORIGIN especifica el nombre del dominio y los registros NS y MX el equipo que hará las veces de servidor de nombre y servidor de correo respectivamente. De ahí en adelante, se encuentran los registros que asocian el nombre de máquina o servicio con una respectiva dirección IP. En éste caso solo se utilizaron RR AAAA para almacenar las direcciones IPV6 y teniendo en cuenta que el servidor debe ser capaz de resolver peticiones IPV4 también se utilizan los registros A.

En el archivo de zona inversa (Figura 12) también se especifican el nombre del dominio y el servidor de nombres y de correo. Debido a que en el archivo named.conf se definió la zona inversa para los primeros 64 bits del bloque de direcciones IPV6, es decir para 2001:448:1024:1, se debe completar la dirección para cada *host* o nombre de servicio en formato nibble y almacenarlo en un RR PTR, estos registros se encuentran definidos en el archivo de zona inversa. Por ejemplo, la dirección IPV6 para zeus.unicauca.edu.co es 2001:448:1024:1::10, la zona está definida hasta 2001:448:1024:1 mediante 1.0.0.0.4.2.0.1.8.4.4.0.1.0.0.2.ip6.arpa, por lo tanto en el archivo de zona es necesario especificar la parte de la dirección ::1mediante la representación 0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.

En el archivo */etc/hosts* se debe agregar el nombre y la direccion del Servidor DNS, específicamente para este caso se agrega la línea:

2001:448:1024:1::10 zeus zeus.ipv6.unicauca.edu.co

Las pruebas realizadas en el Servidor DNS IPV6 para comprobar su correcto funcionamiento tanto al interior como al exteriror de la red de la Universidad del Cauca se encuentran en el anexo A. Para llevar a cabo dichas pruebas es necesario configurar la dirección 2001:448:1024:1::10 como servidor DNS preferido. En Fedora Core 4 incluimos las siguientes líneas en el archivo /etc/resolv.conf:

Nameserver 2001:448:1024:1::10
Domain ipv6.unicauca.edu.co

Para el sistema operativo Windows, se debe especificar la dirección IPV4 del servidor DNS 172.16.41.108, debido a que el soporte para definir DNS preferidos con extensión de 128 bits, no funciona correctamente.

3. AUTOCONFIGURACION EN IPV6

La autoconfiguración es uno de los tópicos más importantes al abordar el tema IPv6, ya que este es un elemento fundamental para la implementación de la Internet de nueva generación a nivel global. Desde la concepción de IPv6 uno de los requerimientos fundamentales fue lograr que un *host* dotado con este protocolo tuviese la capacidad de conectarse a Internet, sin necesidad de configurar parámetros extras. Literalmente, autoconfiguración significa que un *host* automáticamente pueda descubrir y registrar parámetros que necesite para conectarse a Internet.

El proceso de autoconfiguración en IPv6 involucra la creación de una *dirección de enlace local* y la verificación de su unicidad en un enlace, determinando qué información debe ser autoconfigurable (dirección, otra información o ambas), y en el caso de direcciones se debe definir con que tipo de autoconfiguración se obtendrán.

IPv6 define dos mecanismos de autoconfiguración: **con control de estado (*stateful*)** y **sin control de estado (*stateless*)**. En el caso de sin control de estado, la configuración necesaria es nula en los nodos y muy sencilla en los servidores. Este mecanismo permite al *host* generar su propia dirección a partir de información local (el identificador de interfaz) e información anunciada por los *routers* IPV6 (el prefijo de red).

En el caso de autoconfiguración con control de estado, los *hosts* obtienen sus direcciones y otra información de configuración, como por ejemplo servidores DNS o domino, de algún servidor de autoconfiguración. Este servidor puede mantener un control preciso de que direcciones han sido asignadas a cada *host.*

La elección de uno u otro tipo de auto configuración depende de las necesidades propias de cada red, si por ejemplo en una red la prioridad es que los *hosts* obtengan direcciones válidas y encaminables sin preocuparse de quien las usa, debería usarse *stateless*, si por el contrario se requiere un control más rígido sobre la información anunciada, es recomendable el uso de *stateful.*

Es de notar que el uso de uno u otro tipo no es excluyente, IPv6 permite la combinación de los dos tipos de auto configuración para cubrir necesidades específicas, por ejemplo, una máquina puede usar configuración sin estado para configurar sus direcciones y usar configuración con estado para obtener el resto de información.

3.1 TIEMPO DE VIDA DE LAS DIRECCIONES.

Las direcciones IPv6 se asignan a una interfaz por un tiempo determinado (puede ser infinito). Cada dirección tiene un tiempo de vida asociado que indica por cuánto tiempo se asocia una dirección con una interfaz. Una dirección cuyo tiempo de vida ha expirado se convierte en inválida y no debería ser utilizada nuevamente como una dirección fuente por el equipo local, tampoco como una dirección destino por sus compañeros de red local (debería ser usada sólo por aquellas aplicaciones que han estado usándola y pudieran tener problema cambiándola sin interrumpir el servicio). Esto puede tener consecuencias muy severas en las comunicaciones. Por ejemplo, las conexiones TCP se establecen entre un par de direcciones de Internet, si una dirección ya no es válida, la conexión TCP será interrumpida **[13].** Para resolver este inconveniente la IETF ha introducido un mecanismo que hace más sencillo predecir cuando una dirección próximamente ya no será válida. Las direcciones IPV6 tienen 2 tiempos de vida: un tiempo de vida válido y un tiempo de vida recomendado. Una dirección cuyo tiempo de vida válido ha expirado no deberá utilizarse más, y una dirección cuyo tiempo de vida recomendado ha expirado ya no deberá ser utilizada para iniciar nuevas conexiones. Cuando un proceso TCP inicia una conexión, sabe la dirección destino, pero puede escoger cualquiera de las direcciones locales como su dirección fuente. El proceso TCP deberá elegir la dirección cuyo tiempo de vida recomendado sea el mayor o el mínimo y evitará utilizar cualquier dirección que ya se haya desaprobado o que pronto se desaprobará. **[14]**

El tiempo de vida válido y el recomendado pueden ser puestos en infinito, en este caso, la dirección nunca será invalidada o desaprobada, como es el caso de la dirección de red local la cual cumple con esta característica ya que por definición se encontrará permanentemente válida. Cabe hacer notar, que esta posibilidad puede ser revisada en nuevas versiones del estándar de IPv6. Existe una propuesta para reducir el valor infinito a otro valor largo, por ejemplo, 60 días, lo anterior para garantizar que todas las redes puedan ser renumeradas**.**

3.2 DETECCION DE DIRECCIONES DUPLICADAS.

Para asegurar que todas las direcciones configuradas sean únicas en un enlace dado, los nodos ejecutan un algoritmo de detección de dirección duplicada independientemente de si son obtenidas mediante autoconfiguración con o sin control de estado. Esta detección se realiza por medio de los mensajes: Solicitudes de Vecino y mensajes de Anuncio.

Una dirección en la que se aplica el proceso de Detección de Direcciones Duplicadas se denomina tentativa hasta que se completa el proceso con éxito. Una dirección tentativa no se considera "asignada a una interfaz" en el sentido tradicional, es decir,

la interfaz debe aceptar mensajes de Solicitud y de Anuncio de Vecino que contengan la dirección tentativa como dirección destino, pero procesará estos paquetes de manera diferente a aquellos cuya dirección destino concuerde con las direcciones asignadas a la interfaz. Otros paquetes dirigidos a la dirección tentativa deben ser descartados en silencio.

En el proceso de detección de direcciones duplicadas, las siguientes variables juegan un papel importante:

- **DupAddrDetectTransmits:** El número de mensajes de Solicitud de Vecino consecutivos que se envían mientras se hace una Detección de Dirección Duplicada para una dirección tentativa. Un valor de cero indica que la Detección de Direcciones Duplicadas no se ha hecho para esa dirección tentativa. Un valor de uno indica una sola transmisión sin retransmisiones detrás.
- **RetransTimer:** Especifica el retraso entre transmisiones consecutivas de Solicitudes de Vecino efectuadas durante Detección de Direcciones Duplicadas (si DupAddrDetectTransmits es mayor que 1), así como el tiempo que un nodo espera después de enviar la última Solicitud de Vecino antes de acabar el proceso de Detección de Direcciones Duplicadas.

La Detección de Direcciones Duplicadas se efectúa en direcciones unicast ANTES de asignarlas a un interfaz cuya variable DupAddrDetectTransmits sea mayor que cero. En primer lugar se envía un mensaje Solicitudes de Vecino a la direccion tentativa, si como replica se recibe un mensaje de Anuncio se puede concluir que la dirección ya esta en uso.

Si la Solicitud de Vecino es el primer mensaje enviado de un interfaz después de la (re)inicialización del mismo, el nodo debe retrasar el envío del mensaje un número aleatorio entre 0 y MAX_RTR_SOLICITATION_DELAY como se especifica en el RFC 2461 "Neighbor Discovery for IP Version 6 (IPv6)". Este sistema alivia congestiones cuando muchos nodos arrancan en un enlace a la vez, como ocurriría en un fallo de alimentación eléctrica, y evitar inconsistencias cuando más de un nodo intenta solicitar la misma dirección a la vez y asi aumentar la robustez del algoritmo de Detección de Direcciones Duplicadas.

La Detección de Direcciones Duplicadas DEBE tener en cuenta todas las direcciones unicast, con la excepción de los casos siguientes:

- NO SE DEBE EJECUTAR la Detección de Direcciones Duplicadas en direcciones anycast.
- Se debe comprobar la unicidad de todas las direcciones unicast, sin embargo cuando estas son formadas mediante la auto configuración sin control de estado, solo se debería comprobar la unicidad del identificador de interfaz, ya que si el prefijo de red es asignado correctamente, todas las direcciones generadas a partir de estos dos elementos serán únicas. En estos casos, la unicidad de la dirección de enlace DEBE SER COMPROBADA y, si no se detecta dirección duplicada, la implementación PUEDE elegir saltarse la Detección de Direcciones Duplicadas derivadas del mismo identificador de interfaz.

3.3 CONSIDERACIONES DE SEGURIDAD

La configuración de direcciones sin estado permite a una máquina conectarse a una red, configurar una dirección e iniciar una comunicación con otros nodos sin autenticarse o registrarse siquiera con la instalación local. Aunque esto permite a usuarios no autorizados conectarse y usar la red, la amenaza es inherente a la arquitectura de Internet. Cualquier nodo con una conexión física a una red puede generar una dirección que provea conectividad.

El uso de la Detección de Direcciones Duplicadas abre la posibilidad de ataques de denegación de servicio. Un nodo puede responder las Solicitudes de Vecino de direcciones tentativas, causando que el otro nodo rechace la dirección por ser duplicada. Este ataque es similar a otros ataques incluyendo el falseo de mensajes de Descubrimiento de Vecino, sin embargo puede ser evitado utilizando la autenticación de paquetes de Descubrimiento de Vecino[4]. **[14]**

3.4 AUTOCONFIGURACION SIN CONTROL DE ESTADO (STATELESS)

Como ya se mencionó, la configuración sin control de estado se utiliza cuando una entidad no se preocupa demasiado por las direcciones exactas que usa cada máquina mientras sean únicas y enrutables. La autoconfiguración sin estado se diseñó pensando en los siguientes objetivos:

No debería ser necesario configurar máquinas antes de conectarlas a la red. Consecuentemente, se necesita un mecanismo que permita a una máquina obtener o crear direcciones únicas para cada una de sus interfaces. La autoconfiguración de direcciones stateless asume que cada interfaz puede proveer un identificador único para esa interfaz (es decir, un "identificador de interfaz"). Este identificador de interfaz puede ser combinado con un prefijo para formar una dirección.

Redes pequeñas consistentes en un conjunto de máquinas compartiendo un enlace no deberían necesitar un servidor con estado o un *router* como prerrequisito para comunicarse. La comunicación conectar y usar *(plug and play)* se consigue mediante el uso de direcciones de enlace local. Las direcciones de enlace local tienen un prefijo conocido que identifica el (único) enlace compartido al que el conjunto de nodos se conecta. Una máquina forma una dirección de enlace local añadiendo su identificador de enlace al prefijo local de enlace.

Una entidad grande, con muchas redes y *routers* no debería necesitar la presencia de un servidor con estado. Para generar direcciones locales o direcciones globales, las máquinas deben determinar los prefijos que identifican las subredes a las que se conectan. Los *routers* generan Anuncios de *router* periódicos que incluyen opciones listando el conjunto de prefijos activos en un enlace.

La configuración de direcciones debería facilitar una correcta enumeración de las máquinas de una red. Por ejemplo, una instalación puede querer renumerar todos los nodos cuando cambia de proveedor de servicio. La renumeración se consigue a través de la cesión de direcciones a interfaces y la asignación de múltiples direcciones a la misma interfaz. El tiempo de cesión proporciona el mecanismo mediante el que una instalación desfasa prefijos viejos. La asignación de direcciones múltiples a un interfaz proporciona un periodo de transición durante el que tanto la dirección nueva como la desfasada funcionan simultáneamente**.**

4 Para mayor información del tema consultar RFC2402.

Los administradores de redes, deben contar con los elementos necesarios que les permita especificar que tipo de autoconfiguración se debe aplicar, con estado, sin estado o una combinación de las dos**. [14]**

3.4.1 Descripción del protocolo.

La autoconfiguración se ejecuta sólo en enlaces con capacidad de multicast, y comienza cuando una interfaz se activa, por ejemplo, en el arranque. Los nodos (tanto máquinas corrientes como *routers*) comienzan el proceso de autoconfiguración generando una dirección de enlace local para la interfaz. Una dirección de enlace local se forma añadiendo el identificador de red al prefijo del enlace local ya conocido.

Antes que puedan asignarse direcciones de enlace local a una interfaz y sean usadas, un nodo debe intentar verificar que esta dirección "tentativa" no está en uso ya por otro nodo del enlace. Para ello, envía un mensaje de Solicitud de Vecino que contenga la dirección tentativa como el destino. Si otro nodo ya está usando esa dirección, devolverá un Anuncio de Vecino afirmando su uso. Si otro nodo está intentando usar esa dirección, de igual forma se enviará una Solicitud de Vecino para el destino. El número exacto de veces que se (re)transmite la Solicitud de Vecino y el tiempo de retraso entre solicitudes consecutivas es específico del enlace y puede ser fijado mediante la gestión del sistema.

Si un nodo determina que su dirección de enlace local tentativa no es única, se detiene la auto configuración y se requiere la configuración manual del enlace. Para simplificar la recuperación en este caso, debe ser posible que un administrador pueda forzar un identificador de interfaz alternativo que sustituya al identificador por defecto, de manera que el mecanismo de auto configuración pueda ser aplicado al nuevo (supuestamente único) identificador de interfaz. De otro modo, las direcciones de enlace local (y otras) deberán ser configuradas manualmente.

Una vez que un nodo asegura que su dirección de enlace local tentativa es única, la asigna a la interfaz. En este momento, el nodo tiene conectividad a nivel IP con los nodos vecinos. Los pasos restantes de la auto configuración son ejecutados sólo por las máquinas; la configuración de *routers* debe ser realizada mediante otras técnicas.

La siguiente fase de la autoconfiguración implica obtener un Anuncio de *Router* (RA) o concluir que no hay *routers* alrededor. Si los hay, enviarán un RA que especifica qué tipo de autoconfiguración debería ser utilizado en dicha máquina, estos mensajes incluyen también el MTU del enlace, así como otra información como por ejemplo que datos esperar de una autoconfiguración con control de estado, ya sean direcciones u otra información de configuración de red, y el prefijo a ser usado. Si no hay *routers*, debe invocarse la autoconfiguración con control de estado.

Los *routers* envían RA periódicamente, pero no lo suficientemente consecutivos en el tiempo como para obtener una funcionalidad estilo conectar y funcionar (*plug and play*). Por eso mismo, cuando la interfaz es activada, el *host* envía una o más Solicitudes al *Router* (RS) al grupo multicast de 'Todos los *router*s.

Ya que los *routers* generan Anuncios de *Router* periódicamente, las máquinas recibirán nuevos anuncios continuamente. Las máquinas procesan la información contenida en cada anuncio como se describe arriba, añadiendo y refrescando la información recibida en anuncios anteriores**. [15]**

3.4.2 Implementaciones.

Existen múltiples implementaciones de *stateless*, propietarias a cada empresa productora de equipos de telecomunicaciones, lo que demuestra el auge de la Internet de nueva generación, algunas de ellas son:

- Cisco Systems lo soporta: a partir del IOS 12.2(2)T**.**
- HITACHI hace uso de múltiples productos del proyecto **Kame,** entre ellos una implementación *stateless.*
- Nortel Networks lo soporta a partir de la versión 12.0 de su software BayRS.
- Teldat, ofrece la auto configuración sin control de estado a partir de las versiones 1.0 Beta.

Por otro lado se cuenta con una implementación especial para sistemas Linux y BSD:

RADVD (*router advertisement daemon*): Este demonio es el encargado de enviar mensajes **RA**, y escuchar mensajes **RS**. Mediante una sencilla configuración, se anuncia el **prefijo** que puede ser usado para generar una dirección IPv6 válida.

3.5 AUTOCONFIGURACION CON CONTROL DE ESTADO (STATEFUL)-DHCPv6

Como respuesta a los requerimientos que plantea *stateful* se ha desarrollado una versión especifica de DHCP para IPv6. El protocolo de configuración dinámica de *host* IPv6 es un medio para ejecutar autoconfiguración *stateful,* permite a los servidores DHCP proveer parámetros de configuración como son direcciones de red a nodos IPv6, además de otra información de configuración adicional.

3.5.1 Terminología

Antes de abordar el tema DHCPv6, es necesario clarificar la terminología que será usada para una completa comprensión de este protocolo.

- **DUID** - Identificador Único DHCP: Cada Servidor y cliente tiene un DUID. Los servidores usan el DUID para identificar clientes en la selección de parámetros de configuración y en la asociación de IAs con clientes. Los clientes usan los DUID para identificar un servidor en mensajes donde estos requieran ser identificados.

  El DUID es almacenado en una opción debido a que este puede ser de longitud variable y adicionalmente por que el DUID no es requerido en todos los mensajes DHCP. El DUID está diseñado para ser único a través de todos los clientes y servidores DHCP, el DUID no cambiará nunca de ser posible, además este no debe superar los 128 bits de longitud.

  Existen diferentes tipos de DUID, y en el futuro se definirán nuevos tipos conforme el protocolo evolucione. La motivación para tener más de un tipo de DUID es que este debe ser globalmente único, y debe ser de fácil generación, por tanto la naturaleza del identificador puede diferir ampliamente entre los diferentes dispositivos que se pretendan conectar a la red. También, algunos dispositivos pueden no poseer almacenamiento persistente. Retener un DUID

generado en tales dispositivos no es posible. Por tanto el esquema del DUID debe ser flexible en estos casos.

Actualmente existen tres tipos de DUID:

DUID-LLT (1) - *Link-layer address plus time*- Este tipo de DUID consta de 2 bytes conteniendo el valor 1, 2 bytes con un código que indica el tipo de hardware, cuatro bytes con un valor de tiempo (representa el momento en el cual el DUID es generado, tomando como fecha de partida el 1 de Enero del 2000), seguidos por la dirección hardware de cualquier interfaz de red que este conectada al servidor DHCP al momento de la generacion del DUID. La figura 13 ilustra un DUID-LLT

1 Byte	1 Byte	1 Byte	1 Byte
1		Tipo de hardware (2Bytes)	
Valor de Tiempo(4 Bytes)			
Dirección hardware (Tamaño Variable)			

Figura 13. DUID-LLT

El tipo de hardware debe estar validado por la IANA como lo describe en el RFC 826[5]. La dirección hardware debe estar almacenada en el formato canónico establecido por la IANA en el RFC 2464[6].

Este es usado preferentemente por dispositivos de cómputo para propósitos generales como por ejemplo estaciones de trabajo, portátiles, impresoras, *routers* etc., que tengan algún tipo de almacenamiento no volátil.

DUID-EN (2) -*Vendor-assigned unique ID based on Enterprise Number*- Ese tipo de DUID es asignado por el proveedor del dispositivo, consiste en un número privado que la IANA asigna a las empresas productoras[7], almacenado como un numero de 32 bits sin signo, seguido de un identificador único asignado por dichas empresas, que es impuesto en el momento de la manufactura del dispositivo. La figura 14 sintetiza la estructura de un DUID-EN

1 Byte	1 Byte	1 Byte	1 Byte
2		Numero de la empresa	
Numero de la empresa			
Identificador (Tamaño Variable)			

Figura 14. DUIR-EN.

5 Plummer, D.C., "Ethernet Address Resolution Protocol: Orconverting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", STD 37, RFC 826, Noviembre 1982.

6 Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, Diciembre 1998.

7 IANA. Private Enterprise Numbers. http://www.iana.org/assignments/enterprise-numbers.html.

DUID-LL (3) - *Link-layer address* - Este DUID es basado en la dirección hardware y el tipo de hardware (Al igual que en los anteriores casos bajo especificaciones de IANA).

La figura 15 muestra un DUID-LL

1 Byte	1 Byte	1 Byte	1 Byte
3		Tipo de Hardware	
Dirección hardware (Tamaño Variable)			

Fig 15 DUID-LL

Es recomendado para dispositivos que tienen una interfaz de red permanentemente conectada, y con una dirección hardware.

- ***IA*** – Identidad de Asociación: Los clientes usan sus IA's para obtener información de configuración desde los servidores DHCP. Un cliente puede tener más de un IA asignado; Por ejemplo, uno para cada una de sus interfaces. Un IA se compone de un IAID e información de configuración asociada.

- ***IAID*** - Identificador de identidad de asociación: Un identificador para una IA, escogido por el cliente, el cual debe ser único entre todos los IAIDs. El IAID debe ser permanente a través del tiempo, mediante el uso de un almacenamiento no volátil, o por medio de algoritmos que lo generen consistentemente, mientras que la configuración del cliente no cambie.

- ***Prefix*** **-** Prefijo**:** Son el conjunto de bits iniciales de una dirección, o un grupo de direcciones.

- ***Link-local address*****:** Son direcciones con prefijo FE80::/10, que pueden ser usadas para alcanzar nodos vecinos adjuntos al mismo link, cada interface posee una dirección link-local.

- ***Cliente DHCP:*** Es un nodo que realiza peticiones en un link, para obtener información de configuración de uno o más servidores DHCP.

- ***Servidor DHCP:*** Es un nodo que responde a peticiones desde clientes, y que puede o no, encontrase en el mismo link que el cliente.

- ***Agente Relay:*** Es un nodo que actúa como un intermediario para el intercambio de mensajes DHCP entre servidores y clientes que no se encuentran en el mismo link. El agente *relay* se encuentra en el mismo link que el cliente.

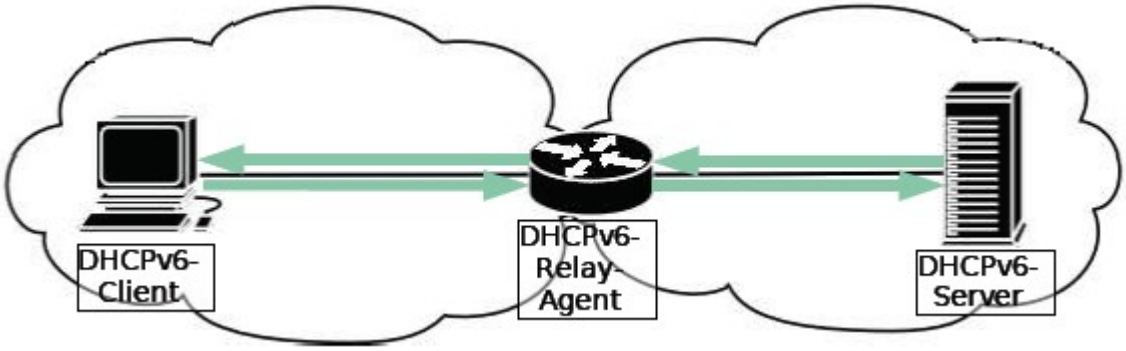


Figura 16. Agente Realy

3.5.2 Direcciones Multicast

Como ya se expuso anteriormente DHCP hace uso de nuevas características propias del protocolo IPv6, como son el uso de direcciones multicast "conocidas" es decir predefinidas:

- ***All_DHCP_Relay_Agents_and_*Servers (FF02::1:2)** una dirección multicast *link-scoped* (alcance dentro del enlace) usada por el cliente para comunicarse con vecinos (ej. on- link) agentes *relay* y servidores. Todos los servidores y agentes *relay* son miembros de ese grupo multicast.

- ***All_DHCP_Servers* (FF05::1:3)** una dirección multicast *scoped-site* (alcance dentro del sitio) usada por los agentes *relay* para comunicarse con servidores, se usa por que el agente *relay* quiere enviar mensajes a todos los servidores o por que este no conoce las direcciones unicast de los servidores. Note que para que el agente *relay* use esta dirección, este debe tener una dirección con suficiente alcance para ser accesible por los servidores. Todos los servidores dentro del sitio son miembros de este grupo multicast.

3.5.3 Puertos UDP.

Los clientes escuchan mensajes DHCP por el puerto UDP 546. Servidores y agentes *relay* escuchan mensajes por el puerto UDP 547.

3.5.4 Especificaciones Relacionadas.

Diversas especificaciones permitieron que se desarrollara este poderoso protocolo que se muestra como una herramienta con características que lo hacen imprescindible en ambientes IPv6.

- ***"IP Version 6 Addressing Architecture",*** La especificación de arquitectura de direccionamiento IPv6 define un rango de direcciones que pueden ser utilizadas en una implementación IPv6, y las diversas directivas de arquitecturas de configuración para diseñadores de redes con espacio de direcciones IPv6. Dos ventajas de IPv6 son: el soporte de multicast, y la capacidad de los nodos de crear direcciones link-local durante la inicialización. La disponibilidad de estas características hace que un cliente pueda usar esta

dirección link-local y una dirección multicast conocida para descubrir y comunicarse con servidores DHCP o agentes *Relay* en su link. **[16]**

- ***"IPv6 Stateless Address Autoconfiguration",*** Autoconfiguración de direcciones sin control de estado IPv6 **[14]** especifica procedimientos por los cuales un nodo puede autoconfigurar direcciones basándose en anuncios de *routers*, y el uso de un tiempo de vida valido para soportar renumeración de direcciones en la Internet.

- ***"Neighbor Discovery for IP Version 6 (IPv6)",*** Descubrimiento de vecinos IPv6 es el protocolo de descubrimiento de nodo en IPv6, el cual remplaza y enriquece funciones de ARP (*Address Resolution Protocol*). Para entender IPv6 y la autoconfiguración de direcciones *stateless*, es muy recomendado que los implementadores estudien a profundidad el descubrimiento de vecinos IPv6. **[17].**

- ***"Dynamic Updates in the Domain Name System (DNS UPDATE)",*** Actualización dinámica DNS **[18]** es una especificación que estandariza la actualización de registros DNS para IPv4 e IPv6. DHCP puede usar la actualización dinámica DNS para integrar direcciones y espacios de nombre, y de esta forma no solamente soportar autoconfiguración, sino también autoregistro en IPv6.**[19]**

3.5.5 Mensajes DHCPv6.

Los clientes y servidores intercambian mensajes DHCP usando UDP. Los servidores DHCP reciben mensajes desde los clientes mediante una dirección multicast link-scoped reservada. Un cliente DHCP transmite la mayor parte de los mensajes en esta dirección multicast reservada, así que este no necesita ser configurado con la dirección o direcciones de los servidores DHCP.

Para permitir que un cliente DHCP envié mensajes a un servidor DHCP que no esta conectado al mismo link, un agente *relay* DHCP en el link del cliente repetirá los mensajes entre el cliente y el servidor. La operación del agente *relay* es transparente al cliente.

Una vez el cliente tiene determinada la dirección de un servidor, este puede bajo ciertas circunstancias enviar mensajes directamente al servidor usando unicast.

Ahora bien la comunicación entre cliente y servidor DHCP puede ser configurada de tal forma que se haga en 2 o 4 mensajes.

3.5.5.1 Intercambios cliente-servidor involucrando dos mensajes.

Cuando un cliente no requiere la asignación de ninguna dirección IPv6, este puede obtener información de configuración tal como una lista de servidores DNS disponibles o servidores NTP a través de un único mensaje y respuesta con un servidor DHCP. Para obtener esta información el cliente primero envía un mensaje petición de información (*Information-request*) a la dirección multicast All_DHCP_Relay_Agents_and_Servers. Los servidores responden con mensajes conteniendo la información de configuración para el cliente.

Cuando un servidor tiene direcciones IPv6 y otra información de configuración obligatoria a un cliente, el servidor y el cliente pueden ser capaces también de

completar el intercambio usando tan solo dos mensajes, en vez de cuatro mensajes como se describe en la siguiente sección. En este caso, el cliente envía un mensaje solicitud (*Solicit*) a la dirección All_DHCP_Relay_Agents_and_Servers requiriendo la asignación de direcciones y otra información de configuración. Este mensaje incluye una indicación de que el cliente desea aceptar un mensaje de respuesta inmediato desde el servidor. El servidor que decide perpetrar la asignación de dirección al cliente, inmediatamente responde a la petición. La información de configuración y las direcciones en el mensaje de respuesta están entonces disponibles inmediatamente para ser usadas por el cliente.

Cada dirección asignada al cliente tiene una preferencia asociada y tiempo de vida válido especificado por el servidor. Para solicitar una extensión de los tiempos de vida asignados a una dirección, el cliente envía un mensaje Renovar (*Renew*) al servidor. El servidor envía un mensaje Replica (Reply) a el cliente con el nuevo tiempo de vida, permitiendo a éste que continué usando la dirección sin interrupción.

3.5.5.2 Intercambio Cliente-servidor involucrando 4 mensajes.

Para solicitar la asignación de una o mas direcciones IPv6, un cliente primero localiza un servidor DHCP y entonces solicita la asignación de direcciones y demás información de configuración desde el servidor. El cliente envía un mensaje de solicitud (*Solicit*) a la dirección All_DHCP_Relay_Agents_and_Servers para localizar servidores DHCP disponibles. Cualquier servidor que pueda responsabilizarse por los requerimientos del cliente, responde con un mensaje *Advertise* (anuncio). El cliente entonces escoge uno de los servidores y envía un mensaje *Request* (solicitud) al servidor preguntado por asignación confirmada de direcciones y otra información de configuración. El servidor responde con un mensaje *Reply* (respuesta) que contiene las direcciones confirmadas y la configuración.

Como se describió en la sección previa, el cliente envía un mensaje *Renew* (renovar) al servidor para extender el tiempo de vida asociado con la dirección, permitiendo al cliente continuar con uso de estas direcciones sin interrupción**. [19]**

3.5.5.3 Tipos de mensaje DHCP.

DHCP define los siguientes tipos de mensaje. La codificación numérica para cada mensaje se muestra entre paréntesis.

- ***SOLICIT (1):*** Un cliente envía un mensaje *Solicit* para localizar servidores.
- ***ADVERTISE (2):*** Un servidor envía un mensaje *Advertise* para indicar que está disponible para prestar servicio DHCP, en respuesta al mensaje *Solicit* recibido desde el cliente.
- ***REQUEST (3):*** Un cliente envía un mensaje *Request* para solicitar parámetros de configuración, incluyendo direcciones IP, a un servidor específico.
- ***CONFIRM (4):*** Un cliente envía un mensaje *Confirm* para cualquier servidor disponible, con el objeto de determinar si las direcciones que fueron asignadas son aun apropiadas para el link al cual el cliente esta conectado.

- ***RENEW (5):*** Un cliente envía un mensaje *Renew* al servidor que originalmente lo proveyó de la dirección IPv6 y los parámetros de configuración para extender el tiempo de vida de la dirección asignada al cliente y para actualizar otros parámetros de configuración.

- ***REBIND (6):*** Un cliente envía un mensaje *Rebind* a cualquier servidor disponible para extender el tiempo de vida en la dirección IPV6 asignada y para actualizar otros parámetros de configuración; este mensaje es enviado después de que el cliente no recibe respuesta a un mensaje *Renew*.

- ***REPLY (7):*** Un servidor envía un menaje *Reply* conteniendo las direcciones asignadas y parámetros de configuración en respuesta a mensajes *Solicit, Request, Renew, Rebind* recibidos desde el cliente. Por otro lado un servidor envía un mensaje *Reply* conteniendo parámetros de configuración en respuesta a un mensaje *Information-request*. También un servidor envía un mensaje *reply* en respuesta a un mensaje *Confirm* confirmando o negando que las direcciones asignadas a el cliente son apropiadas para el link al cual esta conectado. Por ultimo un servidor envía un mensaje *reply* para acuse de recibo de mensajes *Release* o *Decline.*

- ***RELEASE (8):*** Un cliente envía un mensaje *Release* hacia un servidor que asignó direcciones para indicar que este no usará una o más de las direcciones asignadas.

- ***DECLINE (9):*** Un cliente envía un mensaje *Decline* a un servidor que para indicar que un cliente ha determinado que una o mas direcciones asignadas por el servidor están ya en uso en el link al cual el cliente esta conectado.

- ***RECONFIGURE (10):*** Un servidor envía un mensaje *Reconfigure* a un cliente para informarle que dicho servidor tiene parámetros de configuración nuevos o actualizados, y que el cliente debe inicializar una transacción *Renew/Reply* o *Information-request/Reply* con el servidor a fin de recibir la información actualizada.

- ***INFORMATION-REQUEST (11):*** Un cliente envía un mensaje *Information-request* a un servidor para solicitar parámetros de configuración sin la asignación de ninguna dirección IP para el cliente.

- ***RELAY-FORWARD (12)*** Un agente *relay* envía un mensaje *Relay-forward* para retransmitir mensajes a servidores directamente o a través de otro agente *relay*.

- ***RELAY-REPLY (13)*** Un servidor envía un mensaje *Relay-reply* a un agente *relay* conteniendo un mensaje que el agente *relay* reenvía a un cliente. El mensaje *Relay-reply* puede ser transmitido por otros agentes *relay* hasta llegar a el agente relay destino. El servidor encapsula el mensaje como una opción en el mensaje *Relay-reply,* el cual el agente *relay* extracta y envía al cliente.

3.5.6 Parámetros de transmisión y retransmisión.

Esta sección presenta la tabla 5, en la cual se apreciar los valores usados para describir el comportamiento de los mensajes transmitido entre clientes y servidores.

Tabla 5. Valores en mensajes de transmisión.

PARAMETRO	VALOR POR DEFECTO	DESCRIPCION
SOL_MAX_DELAY	1 sec	Máximo retraso del primer Solicit.
SOL_TIMEOUT	1 sec	Tiempo fuera para el Solict inicial.
SOL_MAX_RT	120 secs	Máximo valor de tiempo fuera de un Solicit.
REQ_TIMEOUT	1 secs	Tiempo fuera para el Request inicial.
REQ_MAX_RT	30 secs	Máximo valor de tiempo fuera de un Request.
REQ_MAX_RC	10	Máximo número de intentos de Request.
CNF_MAX_DELAY	1 secs	Máximo retraso del primer Confirm.
CNF_TIMEOUT	1 secs	Tiempo fuera para el Confirm inicial.
CNF_MAX_RT	4 secs	Tiempo fuera máximo del Confirm.
CNF_MAX_RD	10 secs	Duración máxima de Confirm
REN_TIMEOUT	10 secs	Tiempo fuera para el Renew inicial.
REN_MAX_RT	600 secs	Valor de tiempo fuera máximo para Renew.
REB_TIMEOUT	10 secs	Tiempo fuera para el Rebind inicial.
REB_MAX_RT	600 secs	Máximo valor de tiempo fuera para Rebind.
INF_MAX_DELAY	1 secs	Máximo retraso del primer Information-request.
INF_TIMEOUT	1 secs	Tiempo fuera para el Information-request inicial.
INF_MAX_RT	120 secs	Máximo valor de tiempo fuera para Information-request.
REL_TIMEOUT	1 secs	Tiempo fuera Release inicial.
REL_MAX_RC	5	Máximo número de intentos de Release.
DEC_TIMEOUT	1 secs	Tiempo fuera para el Reconfigure inicial.
DEC_MAX_RC	5	Maximo numero de intentos de Decline.
REC_TIMEOUT	2 secs	Tiempo fuera para el Reconfigure inicial.
REC_MAX_RC	8	Máximo numero de intentos de Reconfigure.
HOP_COUNT_LIMIT	32	Máximo número de saltos en un mensaje Relay-forward.

3.5.7 Representación de valores de tiempo y valores de tiempo infinitos.

En DHCP se definen las siguientes constantes de tiempo, útiles para renovar los tiempos de vida de las direcciones.

T1: El tiempo en el cual el cliente debe contactar al servidor DHCP desde el cual sus direcciones actuales fueron obtenidas, para extender el tiempo de vida de estas (Renew).

T2: El tiempo en el cual el cliente debe intentar contactar a cualquier servidor DHCP para extender el tiempo de vida de sus direcciones (Rebind).

Estos tiempos están expresados en segundos.

Todos los valores de tiempo para tiempos de vida, son enteros sin signo. El valor 0xffffffff significa “infinito” cuando es usado como un tiempo de vida (como en RFC2461) en un valor para T1o T2.

3.5.8 Formato de mensajes Cliente/servidor.

Todos los mensajes DHCP enviados entre clientes y servidores comparten un formato de cabecera idéntico fijo y un formato de área variable para opciones.

Todos los valores en la cabecera del mensaje y en las opciones están en orden de bytes.

La figura 17 ilustra el formato de mensajes DHCP enviados entre cliente y servidor:

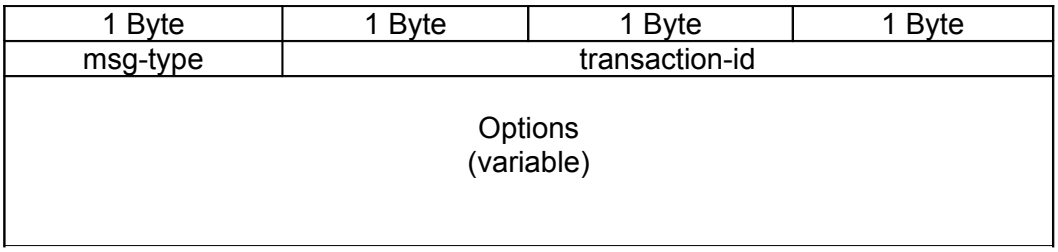


Figura 17. Formato de Mensajes DHCP.

- ***msg-type****:* Identifica el tipo de mensajes DHCP; los tipos de mensajes disponibles están listados en la sección 3.1.5.3.
- ***transaction-id:*** El ID de la transacción para este intercambio de mensajes.
- ***Options:*** Opciones transportadas en este mensaje.

3.5.9 Códigos de estado

La IANA ha registrado los códigos de estado definidos en la tabla 6. IANA manejará la definición de códigos de estado adicionales en el futuro.

Tabla 6. Códigos de Estado DHCPv6.

Nombre	Código	Descripción
Success	0	Éxito.
UnspecFail	1	Falla, razón no especificada; este código de estado es enviado por un cliente o un servidor para indicar una falla no explícitamente especificada
NoAddrsAvail	2	El servidor no tiene direcciones disponibles para asignar al o las IA (s).
NotOnLink	4	El prefijo para las direcciones no es apropiado para el link al cual el cliente esta conectado
UseMulticast	5	Enviado por un servidor a un cliente para forzar al cliente a enviar mensajes al servidor usando la dirección All_DHCP_Relay_Agents_and_Servers

3.5.10 Implementaciones.

De momento se cuenta con cuatro implementaciones:

- Dibbler por Tomasz Mrugalski (Disponible para sistemas Linux y Windows).
- Sourceforge DHCPv6 (Disponible para sistemas Linux).
- NEC DHCPv6 (Disponible para sistemas Linux).
- HP-UX (Disponible para sistemas HP-UX).

3.5.10.1 Dibbler

Esta es una solución DHCPv6 totalmente portable, ya que es la única implementación que puede ser implantada en sistemas Linux o Windows. Tiene la capacidad de trabajar tanto con autoconfiguración *stateful* y *stateless.* Dibbler cuenta con Servidor, Cliente y Agente Realy DHCPv6.

Dibbler es un software de código libre bajo la licencia GNU GPL, que ofrece las siguientes características.

- Descubrimiento por parte del cliente del servidor, Asignación de direcciones (mensajes SOLICIT, ADVERTISE, REQUEST y REPLY).
- Descubrimiento de Servidor DHCPv6 mejorado: Cuando un cliente recibe más de un mensaje ADVERTISE, proveniente de diferentes servidores, se escoge la mejor opción.
- Soporte Realy: Dibbler soporta comunicaciones indirectas vía Relay.
- Comunicación Unicast: Permite enviar mensajes desde cliente a Servidor mediante una dirección unicast, para aumentar la eficiencia, en situaciones donde no todos los servidores deben procesar estos mensajes.
- Renovación de direcciones (mensajes RENEW , REBIND y REPLY).
- Detección de direcciones duplicadas (mensajes DECLINE y REPLY).
- Falla de Poder/ soporte de emergencias (mensajes CONFIRM y REPLY), después de que el cliente se ha recuperado de una emergencia o una falla de poder, este puede tener asignado aún una dirección válida, por tanto el cliente debe hacer la reconfirmación de la misma mediante estos mensajes.
- Opción Rapid Commit: Por la cual Servidor y cliente pueden intercambiar únicamente dos mensajes, con lo que se logra un menor uso de la red y agilización del proceso de configuración.
- Permite la configuración de otra información como Servidores DNS, NTP, SIP, NIS.

3.5.10.1.1 Requerimientos

Dibbler puede correr en sistemas Linux con Kernel 2.4 o 2.6, con soporte para el protocolo IPv6, DHCPv6 usa puertos UDP por debajo de 1024, requiere privilegios de root, estos son necesarios para diversas tareas como borrar o adherir direcciones de red IPv6.

En sistemas Windows, Dibbler puede correr en Windows XP (con al menos el Service Pack 1) y 2003, para su instalación se requieren permisos de Administrador. Esta versión fue desarrollada por Marek Senderski usando microsoft Visual C++ 2002 edition.

3.5.10.2 Sourceforge DHCPv6.

Esta es una implementación de código abierto bajo la licencia BSD, disponible para sistemas Linux. Esta implementación ya esta disponible dentro de los paquetes opcionales a instalar de las distribuciones más actuales. Sourceforge DHCPv6 cuenta con Cliente, Servidor y Agente Realy.

Este software cumple con todas las especificaciones del RFC 3315, además implementa opciones adicionales como son:

- Configuración de prefijos IPV6.
- Opción configuración de DNS.
- Opción configuración de SNTP.
- Opción "*Time Configuration*".
- Permite el intercambio de 2 o 4 mensajes.

3.5.10.3 NEC Europe Ltd.

NEC implementa Cliente, Servidor y agente Relay DHCPv6. El servidor y cliente no se encuentran disponibles libremente, por otro lado el agente Relay se encuentra publicado bajo licencia pública GNU.

3.5.10.4 HP-UX.

HP ha implementado un avanzado Cliente, Servidor, Agente Relay DHCPv6, solo disponible para el sistema operativo HP-UX. Implementa completamente el RFC 3315. Además de otras opciones adicionales como la posibilidad de configurar en los clientes el uso de servidores DNS, SIP, NIS.

3.6 CONFIGURACION SERVIDOR DE AUTOCONFIGURACION.

La implementación de cada uno de los posibles sistemas de autoconfiguración requiere un tratamiento especial especificando paramentros diferentes en los archivos de configuración de las herramientas seleccionadas, en este caso Radvd, DHCPv6 de Sourceforce y Dibbler. Para una descripción detallada del proceso de configuración de dichos servicios remitirse al ANEXO B.

3.7 PRUEBAS REALIZADAS.

3.7.1 Autoconfiguración stateless.

Las pruebas realizadas para este servicio en particular se realizaron en un servidor con un sistema operativo Fedora Core 4, los clientes evaluados fueron sistemas Linux y Windows XP.

Como resultados de las pruebas realizadas se concluye que la autoconfiguración stateless es plenamente funcional, en todos los escenarios posibles.

Para información detallada de las pruebas realizadas consúltese Anexo A.

3.7.2 Autoconfiguración stateful.

Teniendo en cuenta que las aplicaciones que implementan el protocolo DHCPv6 son muy recientes, y dado que estas son del tipo IPv6 puras, y no existe experiencia previa en IPv4 con herramientas semejantes. Fue necesario hacer un conjunto de pruebas con las herramientas Dibbler y Surceforce, para determinar su comportamiento en diferentes escenarios.

Las múltiples posibilidades de interacción entre implementaciones Dibbler y Sourceforce, dieron como resultado los siguientes escenarios de prueba:

- Servidor Source Force – cliente Sourceforce.
- Servidor Sorceforce – cliente Dibbler Linux.
- Servidor Sorceforce – cliente Dibbler Windows.
- Servidor Dibbler – cliente Dibbler Linux.
- Servidor Dibbler – cliente Dibbler Windows.
- Servidor Dibbler – cliente Sourceforce.

Los parámetros evaluados fueron la asignación de direcciones IPv6, y la configuración de información adicional de configuración (Servidor DNS, Dominio). La tabla 7 resume los resultados obtenidos.

Tabla 7. Resultados pruebas a implementaciones de autoconfiguración stateful.

SERVIDOR CLIENTE	**SOURCEFORCE**	**DIBBLER**
SORCEFORCE	✓ Se presenta asignación de direcciones IPv6 ✓ Se presenta asignación de información adicional	✓ Se presenta asignación de direcciones IPv6. • No hay asignación de información adicional.
DIBBLER LINUX	✓ Se presenta asignación de direcciones IPv6 • No hay asignación de información adicional.	✓ Se presenta asignación de direcciones IPv6 ✓ Se presenta asignación de información adicional
DIBBLER WINDOWS	✓ Se presenta asignación de direcciones IPv6. • No hay asignación de información adicional.	✓ Se presenta asignación de direcciones IPv6 ✓ Se presenta asignación de información adicional

Se concluye en base a las pruebas realizadas que las dos implementaciones elegidas brindan un buen soporte al protocolo DHCPv6, sin embargo la interacción entre las dos es parcial, debido a las diferencias entre los formatos de las opciones adicionales de configuración, estas diferencias se presentan por que las citadas opciones no se encuentran estandarizadas en el RFC 3315.

Para información detalla de las pruebas realizadas, remítase al Anexo A.

3.7.3 Combinación stateless-stateful.

En este tipo de configuración busca, la interacción entre ambos tipos de autoconfiguración, el objetivo es que las direcciones sean obtenidas con stateless, mientras que stateful brinde información adicional de configuración a los clientes.

Mediante la experiencia obtenida en las pruebas detalladas en el apartado anterior, se dedujo la incompatibilidad entre las implementacines Dibbler y Sorceforce en la configuración de opciones adicionales, por tanto los posibles escenarios de pruebas son:

- Servidores Sourceforce + Radvd – cliente Sourceforce.
- Servidor Dibbler + Radvd – cliente Dibbler Linux.
- Servidor Dibbler + Radvd – cliente Dibbler Windows.

La tabla 8, presenta los resultados obtenido en las pruebas realizadas, para información detallada de dichas pruebas consúltese Anexo B.

Tabla 8. Pruebas combinación stateless-stateful.

SERVIDOR CLIENTE	**SOURCEFORCE + RADVD**	**DIBBLER + RADVD**
SORCEFORCE	✓ Se presenta asignación de direcciones IPv6 ✓ Se presenta asignación de información adicional	PRUEBA NO REALIZADA POR INCOMPATIBILIDAD
DIBBLER LINUX	PRUEBA NO REALIZADA POR INCOMPATIBILIDAD	✓ Se presenta asignación de direcciones IPv6 ✓ Se presenta asignación de información adicional
DIBBLER WINDOWS	PRUEBA NO REALIZADA POR INCOMPATIBILIDAD	✓ Se presenta asignación de direcciones IPv6 ✓ Se presenta asignación de información adicional

Se concluye que ambas implementaciones, prestan la funcionalidad de configuración adicional de información de configuración.

4. CORREO ELECTRONICO

El correo electrónico, es sin duda uno de los servicios más utilizados y de mayor importancia en la Internet actual. No solo por los millones de usuarios que diariamente utilizan este servicio, como se observa en la figura 18 **[20]**, sino también por que en muchos casos por medio de este, los usuarios hacen sus primeros acercamientos a esta tecnología, además actualmente es requisito poseer un correo electrónico para poder acceder a otros servicios.

Se hace por tanto totalmente indispensable en la Internet IPv6, que este servicio este plenamente disponible a los usuarios, de esta manera se podrá asegurar una penetración más eficaz entre los clientes finales, ya que es imposible pensar en captar usuarios para la nueva red sin ofrecer uno de los pilares sobre los que se cimienta la actual demanda de Internet.

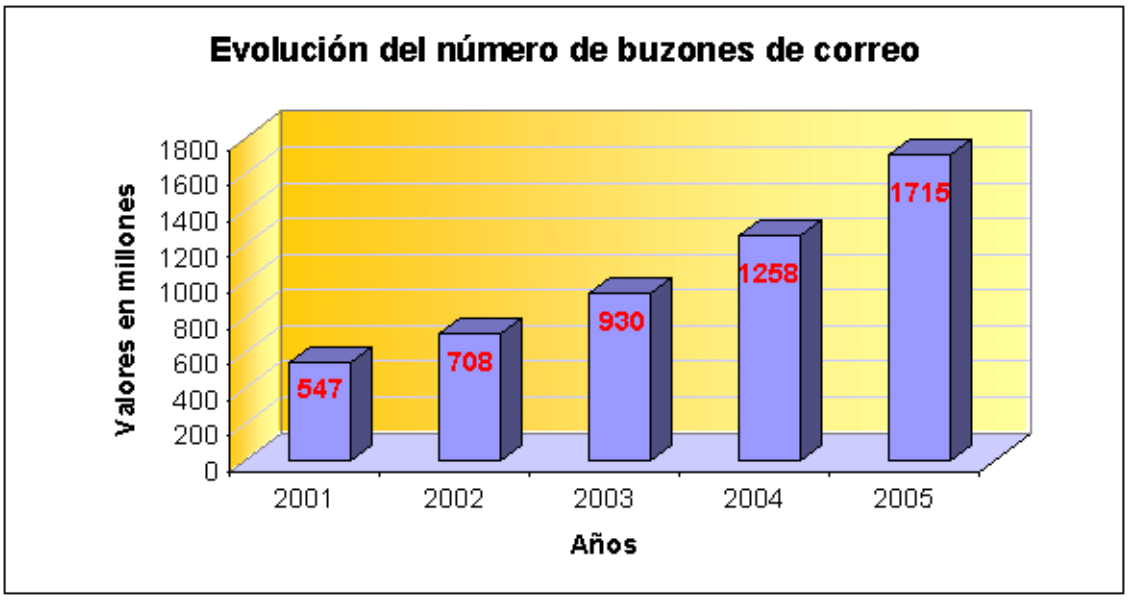


Figura 18. Evolución del número de buzones de correo.

4.1 PROTOCOLO SMTP

El SMTP (Protocolo simple de transferencia de correo) es uno de los Protocolos de Internet (IP) centrales diseñados para transferir correo electrónico de manera confiable y eficiente.

La idea que subyace a SMTP es bastante sencilla. Un usuario o una aplicación redacta un mensaje que contiene la dirección de correo electrónico del destinatario (por ejemplo, “bea@b.com") junto con el asunto y el contenido del mensaje.

La entrega del mensaje se inicia al transferir el mensaje a un servidor SMTP dedicado (Figura 19 paso 1), basándose en el nombre de dominio de la dirección de correo electrónico del destinatario (por ejemplo, "empresa.com"), el servidor SMTP inicia la comunicación con un servidor de nombres de dominio (DNS) (Figura 19 paso 2), que busca y devuelve el nombre de *host* del servidor SMTP de destino (por ejemplo, "mx.b.com") para ese dominio (Figura 19 paso 3).

Por último, el servidor SMTP de origen se comunica directamente con el servidor SMTP de destino a través del puerto 25 del Protocolo TCP (Figura 19 paso 4). Si el nombre de usuario de la dirección de correo electrónico del destinatario coincide con una de las cuentas de usuario autorizadas en el servidor de destino, el mensaje original se transferirá a dicho servidor, esperando que el destinatario recoja el mensaje mediante un programa cliente (Figura 19 paso 5).

En caso de que el servidor SMTP de origen no pueda comunicarse directamente con el servidor de destino, el protocolo SMTP dispone de mecanismos para transferir mensajes a través de uno o varios servidores SMTP intermedios de retransmisión. Un servidor de retransmisión recibirá el mensaje original e intentará entregarlo al servidor de destino o redirigirlo a otro servidor de retransmisión. Este proceso se repetirá hasta que se entregue el mensaje o hasta que transcurra un periodo de tiempo de espera designado.

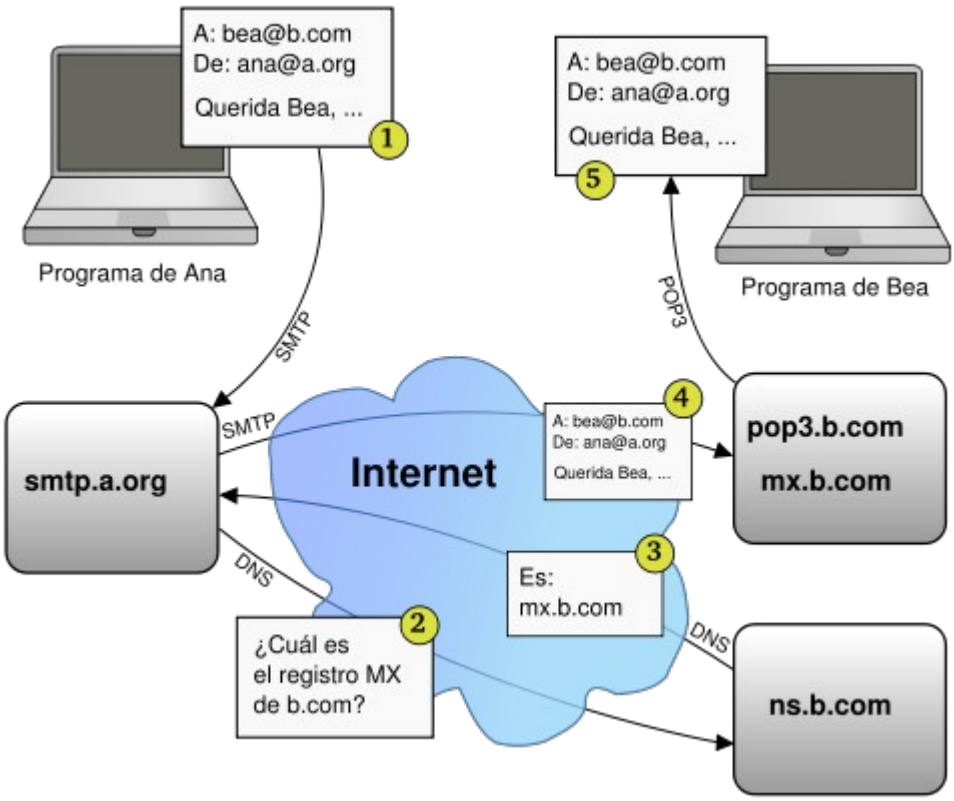


Figura 19. Proceso de envió de correo electrónico.

4.2 SMTP EN IPv6

La estructura fundamental de SMTP, reseñada en la sección anterior, no se ha visto modificada, sin embargo, como se observará más adelante, la nueva estructura de direcciones y la interacción con servidores DNS y sus registros especializados IPv6, introducirán particularidades determinantes para el correcto funcionamiento de este servicio.

Como ya se expuso anteriormente los mensajes Mail en Internet típicamente se envían basándose en el DNS. Primero se buscan los RR (Registros de Recursos) MX en los DNS para devolver los nombres de *hosts* corriendo MTAs (Agente de Transporte Mail) asociados con la parte del dominio de la dirección del correo electrónico. La búsqueda DNS usa la clase IN para IPv4 e IPv6, y de la misma manera registros IN MX son usados para el routing (enrutamiento) de correo electrónico en IPv4 e IPv6.

Como los servidores SMTP IPv6 están en desarrollo y las redes a nivel mundial experimentan un periodo de transición IPv4 – IPv6. Se han echo importantes unos tipos de configuraciones especiales de registros MX para la operación estable SMTP *dual-stack* (IPv4 - IPv6),

Por tal motivo la IETF se ha concentrado en publicar algunos lineamentos, que deberían ser tenidos en cuenta al momento de implementar este tipo de soluciones para la operación estable de SMTP IPv4/IPv6.

4.2.1 Configuraciones Transicionales IPv4-IPv6.

Un RR MX tiene 2 parámetros, un valor de preferencia y el nombre del *host* destino. El nombre del *host* destino se utiliza para buscar una dirección IP para inicializar una conexión SMTP.

Por ejemplo, un sitio IPv6 puro[8] podría tener la siguiente definición DNS:

```
ejemplo.org.          IN  MX 1      mx1.ejemplo.org.
                      IN  MX        10      mx10.ejemplo.org.
mx1.ejemplo.org.      IN  AAAA      2001:db8:ffff::1
mx10.ejemplo.org.     IN  AAAA      2001:db8:ffff::2
```

En el periodo de transición de IPv4 a IPv6 existen muchos sitios que disponen únicamente de IPv4, y muchos otros sitios no tienen interoperabilidad mail con sitios IPv6 puros para este periodo de transición, todos los dominios mail deberían tener registros MX apuntando a direcciones IPv4 e IPv6 existentes **[21]** Ejemplo:

```
ejemplo.org.          IN MX   1  mx1.ejemplo.org.
                      IN MX   10 mx10.ejemplo.org.
mx1.ejemplo.org.      IN AAAA 2001:db8:ffff::1
                      IN A    192.0.2.1
mx10.ejemplo.org.     IN AAAA 2001:db8:ffff::2
                      IN A    192.0.2.2
```

Pero como no todos los destinos MX soportan la operación *dual-stack* de protocolos IPV4 e IPV6, entonces se recomienda cierta configuración especial para *host*s que solo soportan entradas RRs A o RRs AAAA:

```
ejemplo.org.          IN MX   1  mx1.ejemplo.org.
                      IN MX   10 mx10.ejemplo.org.
mx1.ejemplo.org.      IN AAAA 2001:db8:ffff::1
mx10.ejemplo.org.     IN A    192.0.2.1
```

4.2.2 Algoritmo SMTP del emisor en un ambiente DUAL-STACK

En una ambiente *dual-stack*, los registros MX para un dominio se asemejan a lo siguiente:

```
ejemplo.org.          IN MX   1  mx1.ejemplo.org.
                      IN MX   10 mx10.ejemplo.org.
mx1.ejemplo.org.      IN A    192.0.2.1       ;          dual-stack
                      IN AAAA 2001:db8:ffff::1
mx10.ejemplo.org.     IN AAAA 2001:db8:ffff::2 ;         IPv6 puro
```

El algoritmo para un remitente SMTP *dual-stack* es básicamente el mismo que para un remitente IPv4, pero este ahora incluye operaciones de búsqueda de registros MX para transferencias SMTP sobre IPv6. Los Destinos *dual-stack* IPv4/IPv6 deberían ser tratados como destinos multihomed, caso en el cual el Domain Name Resolver

8 Sitios que disponen únicamente de Ipv6.

devuelve una lista de direcciones IP alternativas. Cuando no hay registros de direcciones de destino encontradas como por ejemplo, cuando el MTA remitente es IPv4 puro[9] y no hay registros A disponibles, el caso debería ser tratado como registros MX sin una dirección asignada, y las transferencias fallarán.

El siguiente ejemplo ilustra un caso en el cual el MTA emisor es IPv4 puro, un correo electrónico enviado **a.ejemplo.org** debería devolver el mismo error que al intentar una transferencia a **b.ejemplo.org**.

```
a.ejemplo.org.          IN MX   1  mx1.a.ejemplo.org.
mx1.a.ejemplo.org.      IN AAAA 2001:db8:ffff::1 ; IPv6-puro

b.ejemplo.org.          IN MX   1  mx1.b.ejemplo.org. ; Sin dirección
```

4.2.3 Asegurando la accesibilidad para ambas versiones de protocolos

Cuando se registran MX en un DNS en un ambiente *dual-stack*, la accesibilidad entre *host* MX debe ser considerada cuidadosamente. Supóngase que todos los correos que arriban son almacenados en el *host* MX primario, “mx1.ejemplo.org.”:

```
ejemplo.org.    IN MX   1   mx1.ejemplo.org.
                IN MX   10  mx10.ejemplo.org.
                IN MX   100 mx100.ejemplo.org.
```

Si “mx1.ejemplo.org.” es un nodo IPv6 puro, y los otros son nodos IPv4 puros, no existe accesibilidad entre el *host* primario MX y los otros *host* MX. Cuando un mail llegue a uno de los *host*s MX de más bajo valor de preferencia, este no podrá ser reenviado al *host* MX primario basándose en mecanismos de preferencia. Por consiguiente, mx1.ejemplo.org no estará capacitado para colectar todos los correos (a menos que existan otros mecanismos de transporte entre los *Host* MX con menor valor de preferencia y mx1.ejemplo.org).

```
ejemplo.org.    IN MX   1   mx1.example.org.    ; IPv6-only
                IN MX   10  mx10.example.org.   ; IPv4-only
                IN MX   100 mx100.example.org.  ; IPv4-only
```

La anterior es una configuración problemática, ningún MX secundario puede alcanzar mx1.ejemplo.org.

La solución más simple es configurar el *host* MX primario como un nodo *dual- stack*. Haciendo así, que los *host*s MX secundarios no tengan problemas alcanzando el *host* MX primario.

```
example.org.    IN MX   1   mx1.example.org.    ; dual-stack
                IN MX   10  mx10.example.org.   ; IPv4-only
                IN MX   100 mx100.example.org.  ; IPv6-only
```

Esta configuración trabaja bien, los *host*s MX secundarios están habilitados para enviar mensajes al *host* MX primario sin ningún problema.

9 Sitios que disponen únicamente de IPV4.

4.2.3 Experiencia Operacional.

Muchos de los MTAs IPv6 en operación existentes trabajan de la forma documentada en la sección 3. (Algoritmo SMTP del remitente en un ambiente *dual-stack*)

Hubo, sin embargo, casos donde MTAs IPv6 en operación fueron confundidos por servidores DNS mal configurados. Cuando intentaron obtener un nombre de *host* canónico, algunos servidores DNS con fallos devuelven SERFAIL (RCODE 2), un error temporal en búsquedas basadas en registros AAAA. Bajo este error temporal, el mail se pone en cola para un ensayo posterior **[20]**. Para lograr una completa interoperabilidad IPv4/IPv6, es necesario entonces una perfecta configuración de los Servidores DNS[10].

4.3 IMPLEMENTACIONES

SENDMAIL

Sendmail es probablemente el agente de transporte de Correo más conocido y ampliamente utilizado en Internet.

Sendmail ha soportado nativamente IPv6 desde la version 8.10 usando las interfaces descritas en RFC 2553 “Basic Socket Extensions for IPv6”, la cual fue liberada en el año 2000.

A través del tiempo Sendmail IPv6 ha madurado alcanzando el punto de ofrecer todas las características soportadas para IPv4 en las versiones IPv6, por ejemplo transferencia hacia y desde servidores o *host* habilitados con IPv6, filtraje y redirección.

Las razones por las cuales es tan utilizado son su potencia, escalabilidad y compatibilidad con los estándares de Internet. Además Sendmail es altamente configurable, permitiendo controlar cada aspecto de cómo se gestiona un correo.

POSTFIX

Fue ideado por Wietse Venema como una alternativa a Sendmail. Postfix pretende ser rápido, fácil de administrar y seguro. Soporta nativamente IPv6 desde la version 2.2.

EXIM

Exim es un MTA desarrollado por la Universidad de Cambridge bajo licencia pública GNU, trabaja sobre sistemas Unix.

4.4 CONFIGURACION CORREO ELECTRONICO IPV6.

El MTA elegido para la configuración del correo electrónico IPv6 fue SENDMAIL. La razón para ello fue su capacidad IPv6 anidada a la interacción que se puede lograr con el servidor Dovecot, que ofrece el manejo de los protocolos IMAP y POP3 con soporte IPv6, y con el Webmail Squirrelmail.

10 Para obtener mas información respecto a fallos en servidores DNS y sus efectos negativos consultar Morishita, Y. and T. Jinmei "Common Misbehavior against DNS Queries for IPv6 Addresses".

El paquete de instalación de Sendmail generalmente es incluido en las distribuciones de Linux recientes. Este paquete es instalado en Fedora Core 4 por defecto al solicitar un servidor de correo electrónico, al momento de la instalación del sistema operativo.

Por otro lado se puede descargar el paquete de instalación de:

http://download.fedora.redhat.com/pub/fedora/linux/core/4/i386/os/Fedora/RPMS/

Y se debe ejecutar el siguiente comando para su instalación:

#rpm –i sendmail-8.13.4-2.i386.rpm

4.4.1 Configuracion Sendmail.

La configuración de Sendmail, implica la modificación de algunos archivos, para lograr la interoperabilidad IPv4 –IPv6, y una correcta configuración del Servidor de nombres de dominio:

1. Se edita el archivo */etc/mail/local-host-names,* y se agrega el dominio, en este caso:
   IPv6.unicauca.edu.co

2. Se adhiere en el archivo */etc/mail/acces* la dirección IP del servidor de correo y agregamos las subredes que podrán enviar y recibir correo. Este archivo es en realidad una base de datos donde se definen las máquinas que pueden acceder al servidor de correo y el tipo de acceso que pueden tener. Las máquinas se listan junto con las opciones *OK, REJECT, RELAY* o simplemente junto con un mensaje de error que se entrega a la rutina de gestión de excepciones de Sendmail. Las máquinas que se listan junto con la opción *OK*, que es el valor por defecto, tienen permiso para enviar correo desde el servidor de correo siempre y cuando la dirección de correo de destino sea la misma máquina enviante, en otras palabras permite únicamente correo local. Las máquinas listadas junto con la opción *REJECT* tienen el acceso prohibido a conexiones de correo electrónico con el servidor. Por último las máquinas que poseen la etiqueta *RELAY* tienen permitido enviar correo para cualquier destino a través de la máquina servidora de correo. En este caso, permitirá un acceso de tipo RELAY a toda la red IPv6 de la Universidad del Cauca:

   2001:448:1024:1::10 RELAY
   2001:448:1024:1::RELAY

3. Se crea el archivo *relay-domains* y se ubica en */etc/mail*, y en el agregamos el dominio y el nombre de la maquina donde corre el servidor:

   ipv6.unicauca.edu.co
   zeus.ipv6.unicauca.edu.co

4. Una vez estas modificaciones han sido echas dentro del directorio */etc/mail/* se ejecuta el comando: # *make*. Este paso se debe realizar cada vez que el archivo access es modificado.

5. Ahora se debe editar el archivo de configuración principal de Sendmail, el *sendmail.cf* ubicado en */etc/mail/.* Dicho archivo es demasiado complejo como

para ser manipulado con facilidad, sin embargo este se puede construir a partir de m4[11], es decir, macros que se utilizan para definir características y comportamientos específicos de Sendmail, para tal fin como primer paso se edita el archivo */etc/mail/sendmail.mc* ,luego en necesario comentar la linea: DAEMON_OPTIONS(`Port=smtp, 127.0.0.1, Name=MTA') anteponiendole dnl, Lo anterior para eliminar el uso exclusivo de IPv4, en su lugar se habilita la línea que indica a Sendmail usar los dos protocolos:

DAEMON_OPTIONS(`Name=MTA-v4, Family=inet, Name=MTA-v6, Family=inet6')

Además se agrega el dominio mediante la línea:

LOCAL_DOMAIN(`IPv6.unicauca.edu.co')

Para que los cambio realizados en sendmail.mc, se repliquen en sendmail.cf ejecutamos el comando:

m4 etc/mail/sendmailmc > /etc/mail/sendmail

6. Se requiere descomentar la última línea del archivo */etc/aliasses (#root),* e ingresamos un correo para el administrador de este sistema:

   root: adminIPv6 , luego ejecutamos el comando new aliases.

El último paso es en este momento, inicializar el servidor: */etc/init.d/sendmail start*

4.4.2 Configuración POP3 e IMAP.

Una vez el servidor SMTP se encuentra en un estado plenamente funcional, es necesaria, la instalación de aplicaciones que permitan la comunicación entre éste y los usuarios, a fin de que estos últimos puedan acceder a su buzón de correos, mediante el uso de protocolos IMAP o POP3. Como ya se mencionó con anterioridad, el servidor DOVECOT es un servidor IMAP y POP3 de excelentes prestaciones y código abierto para sistemas Linux y Unix. Realizado pensando en la seguridad como meta, soporta nativamente IPv6 a partir de la versión 0.99.10.6.
Para el correcto funcionamiento del sistema se requiere editar el archivo de configuración de Dovecot: dovecot.conf, ubicado en */etc/*

1. En la línea *protocols*, se deben ingresar los protocolos que Dovecot manejará en este caso:
   protocols = imap pop3

2. Se requiere especificar las direcciones por las cuales el servidor Dovecot recibirá conexiones, si se desea que escuche todas las direcciones IPv4 se usará "*", por otro lado mediante [::], el servidor escuchará tanto direcciones IPv6, como direcciones IPv4. Por tanto en este caso estas dos líneas deben ser adheridas:
   imap_listen = [::]
   pop3_listen = [::]

11 m4 copia los archivos dados expandiendo en el proceso las macros que contengan. Estas macros pueden ser internas o definidas por el usuario y pueden tomar cualquier número de argumentos

3. Por último, se deshabilita el soporte SSL:
   # Disable SSL/TLS support.
   ssl_disable = yes

   La razón por la cual se deshabilita el soporte SSL, el cual permite que dovecot trabaje con llaves públicas y privadas para encriptar la información sensible de correo, radica en que los métodos utilizados por los usuarios para acceder a un buzón de correos (mediante una aplicación Web implemetada sobre HTTPS o mediante acceso remoto por SSH) garatizan la seguridad de los datos manejados en una sesión. Para mayor información al respecto consúltese capitulos 5 y 7.

En este punto se debe inicializar Dovecot: */etc/init.d/dovecot start*

4.4.3 Pine (*Program for Internet news and e-mail*).

Es una herramienta que permite enviar, leer y administrar mensajes de correo electrónico, fué desarrollado en la Universidad de Washington por *Computing & Communications,* como una alternativa al comando *mail.*

Esta herramienta puede acceder a un buzon de mensajes específico tanto de manera local como remota, y está disponible para sistemas Linux, Unix y Windows. En el caso particular de este proyecto se instaló el paquete pine-4.64-1.2.fc4.rf.rpm en el Servidor SMTPv6, ejecutando el comando *#rmp i pine-4.64-1.2.fc4.rf.rpm*, considerando la posibilidad que un usuario que accede por SSH pueda visualizar su buzon de mensajes ejecutando el comando *#pine.*

4.4.4 Webmail.

El acceso de los clientes finales debe ser lo más transparente y sencillo posible, una forma efectiva de lograr este cometido es instalar una aplicación Web sobre el servidor HTTPS que permita a los clientes acceder a su buzón de mensajes de forma segura, haciendo uso de los protocolos IMAP o POP3. Esta aplicación es SQUIRREALMAIL. Es un software *open source*, escrito totalmente en el lenguaje PHP, que puede interactuar plenamente con SMTP e IMAP. Este proyecta goza de gran reconocimiento debido a su estabilidad, potencia y versatilidad.

Los requisitos para la instalación de Squirrelmail son:

- Sistemas Linux/Unix o Windows (sobre sistemas Linux/Unix se obtiene mejor rendimiento, y mas funcionalidades).
- IMAP4rev1 server (Por ejemplo uw-imap, courier-imap, cyrus-imap, hMailServer, Binc IMAP, CommuniGate, MS Exchange Server, MercuryMail 32, Dovecot).
- Servidor Web con PHP instalado.
- Instalación de PERL (Este requisito no es indispensable, mediante PERL, la configuración de este Webmail es más sencilla).

Para detalles sobre la instalación de Squirrelmail consultar el ANEXO C.

4.4.5 Pruebas Realizadas.

Se realizaron dos tipos de pruebas: Una directamente sobre la consola Linux haciendo uso de una conexión ssh y pine, y otra mediante el servicio Web de correo electrónico, de esta manera la evaluación cubre tanto el sistema Web, como los Servidores SMTP y DOVECOT (IMAP, POP3).

Las pruebas tuvieron por objeto evaluar las configuraciones propuestas por el RFC 3974, buscando un correo electrónico *dual-stack*, que interactuase con una configuración especial en el DNS, como se expuso en la sección 4.2.1 (Configuraciones transicionales IPv4-IPv6).

La configuración evaluada puede ser observada en la figura 11 del capitulo 2 Servidor DNS.

Después de realizar las pruebas detalladas en el ANEXO C, se concluye que el sistema evaluado, cumple con las metas planteadas y es plenamente funcional, logrando la interoperabilidad IPv4 –IPV6 esperada.

5. SERVIDOR WEB EN IPv6.

Un servidor Web es un programa que implementa el protocolo HTTP (*hypertext transfer protocol*). Este protocolo está diseñado para transferir lo que se conoce como hipertextos, páginas Web o páginas HTML (*hypertext markup language*): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos.

Un servidor Web se encarga de mantenerse a la espera de peticiones HTTP llevadas a cabo por un cliente HTTP que se suele conocer como navegador. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita. A modo de ejemplo, al teclear www.IPv6.unicauca.edu.co en un navegador, éste realiza una petición HTTP al servidor de dicha dirección. El servidor responde al

cliente enviando el código HTML de la página; el cliente, una vez recibido el código, lo interpreta y lo muestra en pantalla. Como se puede apreciar, el cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página; el servidor tan sólo se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma.

Sobre el servicio Web clásico se puede disponer de aplicaciones Web. Éstas son fragmentos de código que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP. Hay que distinguir entre:

Aplicaciones en el lado del cliente: el cliente Web es el encargado de ejecutarlas en la máquina del usuario. Son las aplicaciones tipo javascript: el servidor proporciona el código de las aplicaciones al cliente y éste, mediante el navegador, las ejecuta. Es necesario, por tanto, que el cliente disponga de un navegador con capacidad para ejecutar aplicaciones (también llamadas scripts). Normalmente, los navegadores permiten ejecutar aplicaciones escritas en lenguaje javascript y java, aunque pueden añadirse más lenguajes mediante el uso de *plugins*.

Aplicaciones en el lado del servidor: el servidor Web ejecuta la aplicación; ésta, una vez ejecutada, genera cierto código HTML; el servidor toma este código recién creado y lo envía al cliente por medio del protocolo HTTP.

Las aplicaciones de servidor suelen ser la opción por la que se opta en la mayoría de las ocasiones para realizar aplicaciones Web, auque es de notar que las aplicaciones del cliente pueden ser usadas en combinación, para obtener funcionalidades adicionales. La razón es que, al ejecutarse éstas en el servidor y no en la máquina del cliente, éste no necesita ninguna capacidad adicional. Así pues, cualquier cliente dotado de un navegador Web básico puede utilizar este tipo de aplicaciones. Algunos lenguajes de este tipo relacionados con las aplicaciones Web son: PHP, ASP, Perl, CGI, .NET, JSP (Tecnología Java)

5.1 FUNCIONAMIENTO SERVICIO WEB EN IPV6

En esta sección se expone la metodología por la cual un cliente Web puede acceder a una pagina Web, a través de IPv6.

1. Un usuario en cualquier parte del mundo requiere visitar un portal IPv6, por ejemplo www.IPv6.unicauca.edu.co por tanto lo primero que hace es escribir esta dirección en la barra de direcciones de su navegador.
2. El cliente Web (Navegador), realiza una petición como cliente DNS de la dirección a visitar. Una vez que ha recibido la traducción del servidor DNS, en este caso una dirección de 128 bits como por ejemplo 2001:448:1024:1::20, se entabla un diálogo HTTP con el servidor Web que tenga la IP concreta. Sin embargo, para que esto sea posible el navegador debe estar capacitado para interactuar con las nuevas direcciones IPv6, de no ser así, la combinación de esta dirección y el puerto formarán un *socket* no válido, para el navegador y se reportará un error.
3. Ahora intervienen un cliente y un servidor Web, en una conversación cuyo idioma es HTTP. HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores, ahora bien el cliente hará una solicitud al servidor, y este intentará devolver el recurso solicitado. El formato tanto del mensaje como de la respuesta es el siguiente:

<Línea inicial>

Header-1: value-1
...
Header-n: value-n

<Cuerpo del mensaje (Opcional)>

La línea inicial es diferente en las solicitudes y en las respuestas. En las solicitudes van tres campos separados por un espacio en blanco:

"Método recurso versiónDelProtocolo".

Por ejemplo: "GET /path/to/file/index.html HTTP/1.0".

La línea inicial de una respuesta tiene tres campos separados por un espacio:

"versiónDelProtocolo códigoRespuesta Mensaje".

Por ejemplo: "HTTP/1.0 200 OK" o bien "HTTP/1.0 404 Not Found".

Los encabezados están normalizados en el protocolo, e incluyen, en el caso de una solicitud, información del navegador y eventualmente del usuario cliente; En el caso de una respuesta, información sobre el servidor y sobre el recurso. El cuerpo del mensaje contiene el recurso a transferir o el texto de un error en el caso de una respuesta. En el caso de una solicitud, puede contener parámetros de la llamada o archivos enviados al servidor.

Analizando la información anterior se puede concluir que el protocolo HTTP no debe realizar ningún cambio para trabajar sobre IPv6, ya que los mensajes que intercambian cliente y servidor no brindan información referente a protocolos de capas inferiores, el fin del diálogo HTTP es devolver el código de la página o aplicación Web solicitada, si la respuesta es afirmativa, en este caso www.IPv6.unicauca.edu.co, o un código de error en caso de una respuesta negativa.

Sin embargo, es claro que el servidor Web también debe tener capacidad para interactuar con el nuevo modelo de direcciones, no solo por funciones propias del sistema en las cuales se requiere conocer la dirección IPv6 dentro de su configuración, sino también debido a que las aplicaciones que este aloja, pueden requerir de la capacidad IPv6 del servidor para funcionar correctamente.

5.2 FORMATO DE DIRECCIONES IPv6 EN URL'S

Para usar una direccion literal IPv6 en una URL, o sea para escribir en la barra de direcciones de un cliente Web una dirección literal, esta debe ser encerrada entre los caracteres "[" y "]", la razón fundamental para esto es que los dos puntos que separan los bloques de las direcciones IPv6, son también utilizados para identificar el puerto del servidor Web pretendido . Por ejemplo las siguientes direcciones literales:

- FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
- 1080:0:0:0:8:800:200C:4171

- 3ffe:2a00:100:7031::1
- 1080::8:800:200C:417A
- ::192.9.5.5
- ::FFFF:129.144.52.38
- 2010:836B:4179::836B:4179

Serían representadas como las siguientes URL's

- http://[FEDC:BA98:7654:3210:FEDC:BA98:7654:321]:80/index.html
- http://[1080:0:0:0:8:800:200C:417A]/index.html
- http://[3ffe:2a00:100:7031::1]
- http://[1080::8:800:200C:417A]/foo
- http://[::192.9.5.5]/ipng
- http://[::FFFF:129.144.52.38]:80/index.html
- http://[2010:836B:4179::836B:4179]

Este formato ha sido ampliamente implementado en conocidos clientes Web como Microsoft Internet Explorer, Mozilla, y Lynx. **[22]**

5.3 HTTPS (HTTP Over SSL)

El protocolo HTTPS es la versión segura del protocolo HTTP. El sistema HTTPS utiliza un cifrado basado en las *Secure Socket Layers* (SSL) para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP. El puerto estándar para este protocolo es el 443. Es importante notar que el uso del protocolo HTTPS no impide en caso alguno que se pueda utilizar HTTP, por lo que la mayoría de los clientes Web advierten cuando una página tiene elementos que no son seguros en entornos HTTPS, como también advierten cuando se invoca un protocolo distinto al de la página actual: (HTTP -> HTTPS o HTTPS->HTTP) **[23].**

5.3.1 Elementos básicos en una comunicación segura basada en SSL

Supóngase que un usuario desea realizar una transferencia de información sensible (por ejemplo una clave de tarjeta de crédito, un número de cuenta corriente), como resulta obvio se requiere que esta operación se realice de una manera segura, confiable y privada. Para tal fin es posible utilizar algoritmos de cifrado.

5.3.2 Criptografía basada en llaves públicas y privadas

Existen múltiples tipos de cifrado, SSL utiliza particularmente el método de llaves públicas y privadas, las cuales son complementarias. Es decir un mensaje cifrado con una de las dos llaves solo puede ser interpretado con la ayuda de la otra. Como su nombre lo indica la llave pública se encuentra a disposición de todo el mundo, mientras que la privada solo la posee su dueño. De esta manera un mensaje cifrado con la llave pública solo podrá ser leído por la persona o entidad que posea la llave privada, asegurándose así que el mensaje llegara únicamente al destino deseado; o un mensaje cifrado con la llave privada solo podrá ser leído mediante el uso de la llave pública, legitimando de esta manera la autenticidad del remitente.

5.3.3 Message Digest.

Aunque sea posible encriptar un mensaje para hacerlo privado, mediante el uso de una llave pública, todavía existe la posibilidad de que alguien pueda modificarlo o sustituirlo, para esto se hace necesario crear un mecanismo que permita garantizar la integridad del mensaje, este mecanismo se llama *Digest* y consiste en crear un pequeño resumen del mensaje que será enviado al destinatario, de esta manera, cuando este recibe el mensaje creará su propio *Digest* y lo comparará con el que envió el remitente, si ambos coinciden significa que el mensaje fue recibido intacto.

Un *Digest* también recibe el nombre de "*one way function*" o "*hash function*", ejemplos de implementaciones son MD5 o SHA, se utilizan para crear pequeñas representaciones de largo fijo para grandes mensajes de largo variable, estos algoritmos aseguran un valor único para distintos mensajes, por lo que es imposible que 2 mensajes tengan el mismo valor *Digest*, además se hace muy difícil reconstruir el mensaje basándose solo en el valor *Digest.*

Aun se tiene un problema, se requiere que el *Digest* sea enviado en forma segura, una manera de hacerlo es incluirlo en una *digital signatura* (Firmas digitales).

5.3.4 Digital Signatures.

Cuando un remitente envía un mensaje al destinatario, este último necesita asegurarse de que el mensaje realmente proviene de la persona o entidad correcta y no de un intruso que esta utilizando sus datos para transferir datos incorrectos o viciados.

Una firma digital se crea encriptando el Digest del mensaje junto con otra información importante (como el numero de secuencia del mensaje) y para esto se utiliza la llave privada de quien envía el mensaje. Aunque cualquier persona puede desencriptar la firma digital (cualquiera que posea la llave publica) solo el firmante conoce la llave privada, y por tanto como se observó en una sección previa se legitimiza la identidad del remitente.

Al incluir el Digest en la firma digital se asegura que nadie más pueda cambiar el Digest y volver a firmar el mensaje.

Por otro lado se debe proveer una posible intercepción del mensaje, para el uso posterior de la firma por parte de un intruso, este punto se ve contrarrestado mediante el uso de un único número de secuencia.

5.3.5 Certificados.

Aunque se logre enviar un mensaje firmado, y con mecanismos que aseguren la integridad del mismo, aun queda un punto por resolver , y es saber si el destinatario es realmente la persona o entidad que dice ser. Para ello se utiliza la llave pública que confirmara la identidad de este último, esta llave pública debe estar firmada por una agencia creíble *(trusted agency)* que legitimise la identidad del destino mediante un certificado.

Un certificado asocia una llave pública con la identidad real de un individuo, de un servidor o de cualquier otro tipo de entidad, que se conoce como sujeto. La información del sujeto contiene el nombre distintivo, la llave pública, el periodo de vigencia del certificado como también la identificación de la entidad certificadora que emitió el certificado.

5.3.6 SSL.

El protocolo SSL es un protocolo que se sitúa entre el protocolo de transporte TCP y un protocolo de la capa de aplicación (ej: HTTP). Habitualmente, sólo el servidor es autentificado (es decir, se garantiza su identidad) mientras que el cliente se mantiene sin autentificar; la autentificación mutua requiere un despliegue de infraestructura de claves públicas (o PKI) para los clientes. Los protocolos permiten a las aplicaciones cliente-servidor comunicarse de una forma diseñada para prevenir intromisiones, la falsificación de la identidad del remitente y mantener la integridad del mensaje.

SSL implica una serie de fases básicas:

- Negociar entre las partes el algoritmo de cifrado que se usará en la comunicación:
- Intercambio de claves públicas y autenticación basada en certificados digitales.
- Encriptación del tráfico basado en cifrado simétrico, para mayor información al respecto consúltese el capítulo 7, sección 7.1.

5.3.7 Estableciendo una sesión SSL.

Una sesión SSL se establece por medio de un *handshake* entre el cliente y el servidor, esta secuencia puede variar dependiendo de si el servidor entrega un certificado o solicita el certificado al cliente:

- El cliente envía un mensaje *ClientHello* especificando una lista de cifrados, métodos de compresión y la versión del protocolo SSL más alta permitida. Éste también envía bytes aleatorios que serán usados más tarde (llamados Challenge de Cliente o Reto). Además puede incluir el identificador de la sesión.

- Después, recibe un registro *ServerHello*, en el que el servidor elige los parámetros de conexión a partir de las opciones ofertadas con anterioridad por el cliente.

- Cuando los parámetros de la conexión son conocidos, cliente y servidor intercambian certificados (dependiendo de las claves públicas de cifrado seleccionadas). Estos certificados son actualmente X.509[12], pero hay también un borrador especificando el uso de certificados basados en OpenPGP (PGP = Pretty Good Privacy).

- El servidor puede requerir un certificado al cliente, para que la conexión sea mutuamente autenticada.

12 X.509 especifica, entre otras cosas, formatos estándar para certificados de claves públicas y un algoritmo de validación de ruta de certificación

- Cliente y servidor realizan una negociación para determinar una clave secreta común llamada *master secret*, posiblemente encriptando una clave secreta con una clave pública que es desencriptada con la clave privada de cada uno. Todos los datos de claves restantes son derivados a partir de este *master secret* (y los valores aleatorios generados en el cliente y el servidor), que son pasados a través una función seudo aleatoria cuidadosamente elegida.

5.3.9 OpenSSL.

OpenSSL es un proyecto de software desarrollado por los miembros de la comunidad *Open Source* para libre descarga y esta basado en SSLeay desarrollado por Eric Young y Tim Hudson. Consiste en un paquete robusto de herramientas de administración y librerías relacionadas con la criptografía, que suministran funciones criptográficas a otros paquetes como OpenSSH y navegadores Web (para acceso seguro a sitios HTTPS). Estas herramientas ayudan al sistema a implementar el *Secure Sockets Layer* (SSL), así como otros protocolos relacionados con la seguridad, como el *Transport Layer Security* (TLS). Este paquete de software corre sobre máquinas Linux. OpenSSL también permite crear certificados digitales que podrían aplicar a un servidor, por ejemplo Apache.

5.4 IMPLEMENTACIONES IPv6.

En la actualidad muchos servidores ofrecen soporte IPv6, por tanto al momento de implementar este servicio, es necesario estudiar las particularidades del sistema donde será implementado, y de esta manera elegir la mejor solución.

Apache
El servidor Apache es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1. Cuando comenzó su desarrollo en 1995 se basó inicialmente en el código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que originalmente Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA.

La Primera versión con soporte IPv6 fue Apache 1.3.12 que mediante parches ofrecía esta funcionalidad, cabe destacar que las versiones 1.3.x son las más utilizadas actualmente en la Web. Apache ofrece a partir de la versión 2.028 soporte IPv6 nativo, mediante el uso de IPv6 *listening sockets* por defecto. Además, las directivas Listen, *NameVirtualHost*, y *VirtualHost* soportan direcciones IPv6 numéricas (por ejemplo, "Listen [2001:db8::1]:8080").

La arquitectura del servidor Apache es muy modular. El servidor consta de un sección *core* y mucha de la funcionalidad que podría considerarse básica para un servidor Web es provista por módulos. Algunos de estos son:

mod_perl - Páginas dinámicas en Perl.
mod_php - Páginas dinámicas en PHP.
mod_python - Páginas dinámicas en Python.
mod_aspdotnet - Páginas dinámicas en .NET_de_Microsoft.

Tiny/turbo/throttling HTTP Server
Este es un Servidor HTTP pequeño, simple, potable, rápido y seguro. Soporta IPv6 a partir de la versión 2.20c.

Webfs

Webfs es un servidor HTTP básico para contenido puramente estático. Puede ser usado por ejemplo para mostrar el contenido de un servidor FTP vía HTTP. Soporta IPv6 a partir de la versión 1.19.

Publicfile

Soporta IPv6 a partir de la versión 0.52 pero solo mediante parches adicionales. Publicfile provee archivos a clientes a través de HTTP y FTP.

Bozohttpd

Bozohttp es un servidor HTTP pequeño y seguro. Su principal característica es precisamente la ausencia de funciones, reduciendo el tamaño código y mejorando la verificablilidad del mismo. Soporta CGI/1.1, HTTP/1.1, HTTP/1.0, HTTP/0.9, *hosting* virtual. Además de soportar el protocolo IPv6 en su versión 1.1. Otra característica resaltable es no poseer archivos de configuración por diseño.

Leahhttpd

El proyecto leahhttpd apunta al desarrollo de un Servidor Web eficiente y rápido. Eficiente al transferir rápidamente datos, alto rendimiento, Además se desea el soporte para tantas arquitecturas y plataformas como sea posible y soporte de nuevas tecnologías y estándares (como IPV6). Leahttpd no tiene la intención de ser fácilmente configurable. Pero si poseer muchas características especiales que permitan un pleno desarrollo del mundo CGI.

IIS
Internet Information Services (o Server), IIS, es una serie de servicios para los equipos que funcionan con Windows. Originalmente era parte del *Option Pack* para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS.

Ahora bien como ya se definió previamente, es de vital importancia que los clientes Web posean soporte para IPv6, el siguiente es un listado de algunos de ellos:

MOZILLA

Mozilla un cliente Web *open-source* de clase mundial que soporta todos los Standard de Internet en una variedad de plataformas incluyendo Windows, Linux, Mac OS X, OS/2, Solaris; y muchas mas. Cuenta con poderosas características como bloqueo de pop-up y manejo de tabs. (pestañas). Mozilla provee también una sofisticada plataforma para desarrollo Web y aplicaciones intranet tecnologías de vanguardia

como XML, SOAP y XSLT. Mozilla 1.4 no soporta plenamente IPv6 en plataformas Win32.

FIREFOX

Firefox es un navegador *open-source* basado en Mozilla. Soporta IPv6 desde la versión 1.5

KONKEROR

El cliente Web Konkeror HTTP/FTP está incluido en el ambiente escritorio KDE, soporta IPv6 desde la versión de KDE 2.2.

LYNX

Lynx es un cliente Web de texto para el Word Wide Web funciona en Unix, VMS, WINDOWS 95/98/NT (no en 3.1x), en DOS (386 o superior) y OS/2 EMX.

OPERA

Es un buen cliente Web, famoso por su velocidad, reducido tamaño, seguridad, adaptabilidad, soporte de Standard. Desde la versión 7.20 beta, Opera soporta IPv6 nativamente en plataformas Linux, BSD y Windows.

5.5 INSTALACION SERVIDOR WEB IPv6.

El servidor instalado para albergar las aplicaciones Web IPv6 es **Apache**, ya que éste ofrece todas las características necesarias para realizar la migración Web a IPv6 al manejar el API Sockets IPv6, por otro lado ofrece una gran cantidad de opciones de configuración que se traducen en flexibilidad para la personalización del sistema, y adicionalmente permite la inserción de funcionalidades extra mediante módulos externos al servidor.

La versión de Apache instalada es httpd-2.0.54-10, sobre un sistema Linux Fedora Core 4. Este paquete está disponible en la sección de descargas de la página oficial del proyecto Apache, mientras que en la página http://rpm.pbone.net, se puede obtener una versión con extensión .rpm, que se instalará ejecutando el comando:

#rpm –i httpd-2.0.54-10.rpm

Después de realizar estos pasos el servidor apache estará listo para correr. Entonces se generará un archivo de configuración llamado httpd.conf. Particularmente en Fedora Core 4, este esta ubicado en */etc/httpd/conf/*

Este archivo debe ser modificado, para lograr una configuración personalizada.

La configuración del servidor Web se encuentra dividida en dos partes, una configuración estándar para las aplicaciones que no requieren de un nivel de seguridad adicional, como por ejemplo el portal www.IPv6.unicauca.edu.co, y otra especial utilizada para brindar soporte seguro a aplicaciones que hacen intercambio

sensible de información, como son las aplicaciones de Webmail, e Interfaz de control de autoconfiguración.

Para esta segunda configuración se debe instalar un modulo especial que implementa el protocolo SSL, el paquete descargado para tal fin fué openssl-0.9.7f-7, se debe ejecutar el comando:

#rpm –i openssl-0.9.7f-7.

Después de realizar esta acción y si el proceso de instalación es exitoso el servidor Apache podrá hacer uso de las herramientas proveídas por Openssl, y así brindar un nivel de seguridad adecuado a la información sensible. En este punto aparecerá un archivo llamado ssl.conf en el directorio */etc/httpd/conf.d/* el cual controla el comportamiento de https. La configuración detallada de estos archivos y creación de certificados de autenticidad, se encuentra documentada en el ANEXO D.

5.6 PORTAL WEB UNICAUCA IPV6

Dentro de los objetivos del proyecto se contempló la posibilidad de poner en alta un portal Web, que hiciese las veces de puente gráfico entre los potenciales clientes de la nueva red y los servicios que a su disposición fuesen configurados, en este caso a través del portal Web **www.IPv6.unicauca.edu.co,** se debe tener la posibilidad de ingresar a los servicios de correo electrónico, ftp, y la aplicación interfaz gráfica para Autoconfiguración. El contenido que manejará esta página es netamente científico-técnico relacionado con la temática IPv6.

Para que un usuario pueda acceder al portal www.IPv6.unicauca.edu.co, es necesaria una configuración en el DNS IPv6, que incluya registros de recursos A y AAAA, que almacenen las direcciones IPv4 e IPv6 respectivamente, para garantizar el acceso de clientes IPv4 e IPv6 indistintamente, tal como lo ilustra la figura 11 del capítulo 2.

Para la creación y administración de la página www.IPv6.unicauca.edu.co fue utilizado un software administrador de contenidos Web, dado que de esta manera la administración del portal se hace más práctica. El administrador de contenido elegido fue **Mambo** entre las razones para su elección destacan: Es un software escrito en PHP, recomendado para funcionar sobre Apache, cuenta con una intuitiva y amigable interfaz grafica que permite gestionar contenidos y secciones, así como también usuarios en el sistema, este es un software opensource con gran reconocimiento al haber ganado el premio al mejor software libre en el reino unido en el año 2004, el premio a la mejor solución opensource en el año 2005 según la publicación LinuxWord, y en abril del 2006 fue nominada por LinuxWord Australia como el mejor software opensource.

Los requerimientos para su instalación son:

Apache 1.3.19 o superior.
Mysql 3.2 o superior.
PHP 4.2 o superior.

La instalación detallada puede consultarse en el ANEXO D.

6. FTP PROTOCOLO DE TRANSFERENCIA DE ARCHIVOS

6.1 DEFINICION

FTP es una especificación para el protocolo Internet, en la cual se definen los métodos por los cuales es posible intercambiar archivos entre estaciones de trabajo a través de Internet. Este protocolo es ampliamente utilizado para la transferencia de archivos alojados en páginas Web y para descargar programas y otro tipo de archivos desde los servidores FTP hacia las estaciones de trabajo destino. Es posible copiar, borrar, actualizar, mover y renombrar los archivos alojados en el servidor FTP, desde el mismo servidor o desde un cliente FTP. Los objetivos principales del protocolo FTP son:

- Promocionar el uso compartido de Archivos (programas y/o datos).
- Incentivar al uso indirecto o implícito (a través de programas) de servidores remotos.
- Hacer transparente al usuario las variaciones entre la forma de almacenar archivos en diferentes computadores.

- Transferir datos fiable y eficientemente.

Este protocolo se basa en el esquema cliente/servidor. El cliente FTP se conecta a un servidor FTP (que es la máquina remota) y una vez establecida la conexión, se inicia una sesión entre las dos máquinas basada en peticiones, respuestas y transmisión de información mediante un canal de datos creado para tal fin. Las peticiones realizadas por el cliente se denominan comandos FTP.

Para la transferencia de archivos mediante el protocolo FTP, se utilizan 2 tipos de conexiones, la conexión de control y la conexión de datos. La conexión de control sirve para enviar subcomandos desde el cliente al servidor y recibir las correspondientes respuestas a esos subcomandos. La conexión de datos es utilizada para transferir los archivos que el cliente haya solicitado. Conectividad TCP/IP, existencia de un servidor FTP configurado apropiadamente, tener el nombre de *host* del sistema remoto y contar con nombre de usuario y contraseña en el sistema remoto, son los elementos necesarios en el establecimiento de una sesión FTP.

Para que un usuario pueda conectarse con un servidor FTP, es necesario tener un usuario (login) y una contraseña (password), debido a que los servidores controlan el acceso de los usuarios a su sistema de carpetas, pero en los casos en los cuales los servidores FTP contienen información que puede ser vista por cualquier tipo de usuario, es posible utilizar una forma de acceso no autentificada, es decir que no utiliza contraseñas, denominada *usuario anonymous*. Existen 3 tipos de usuarios FTP:

- Usuario Anónimo: Puede acceder a los archivos del servidor, sin importar si se encuentra registrado en él, cualquiera que esté conectado a Internet puede establecer una conexión y realizar transferencia de archivos, del tipo Servidor-Cliente. Está específicamente orientado para trabajar con archivos de contenido variado (fotos, texto, software) y la transferencia se puede realizar entre computadores con distintos sistemas operativos y entre distintas redes. Sus privilegios son restringidos.

- Usuario Autenticado: Necesita un nombre de usuario y una contraseña para acceder al servidor y poder enviar o recibir archivos desde y hacia los directorios del servidor y convertirlos en públicos o privados.

- Usuario Embebido: Forma parte del usuario Anónimo, las descargas de archivos se realizan desde enlaces predispuestos en páginas Web, y es la forma de acceso a servidores FTP más popular hoy en día.

6.2 COMPONENTES

El protocolo está compuesto por 5 entidades que permiten la comunicación entre un usuario y un servidor FTP:

- Servidor PI[13] (Intérprete del protocolo): Escucha sobre un puerto determinado para realizar la conexión desde el usuario y establecer una conexión de control de comunicación. También se encarga de recibir comandos FTP, enviar respuestas y controlar el servidor DTP.

13 Protocol Interpreter.

- Servidor DTP[14] (Proceso de transferencia de datos): Establece el canal de conexión de datos con la información del puerto por el cual está escuchando, configura los parámetros necesarios para transferir y almacenar archivos, y transfiere comandos desde y hacia el servidor PI.

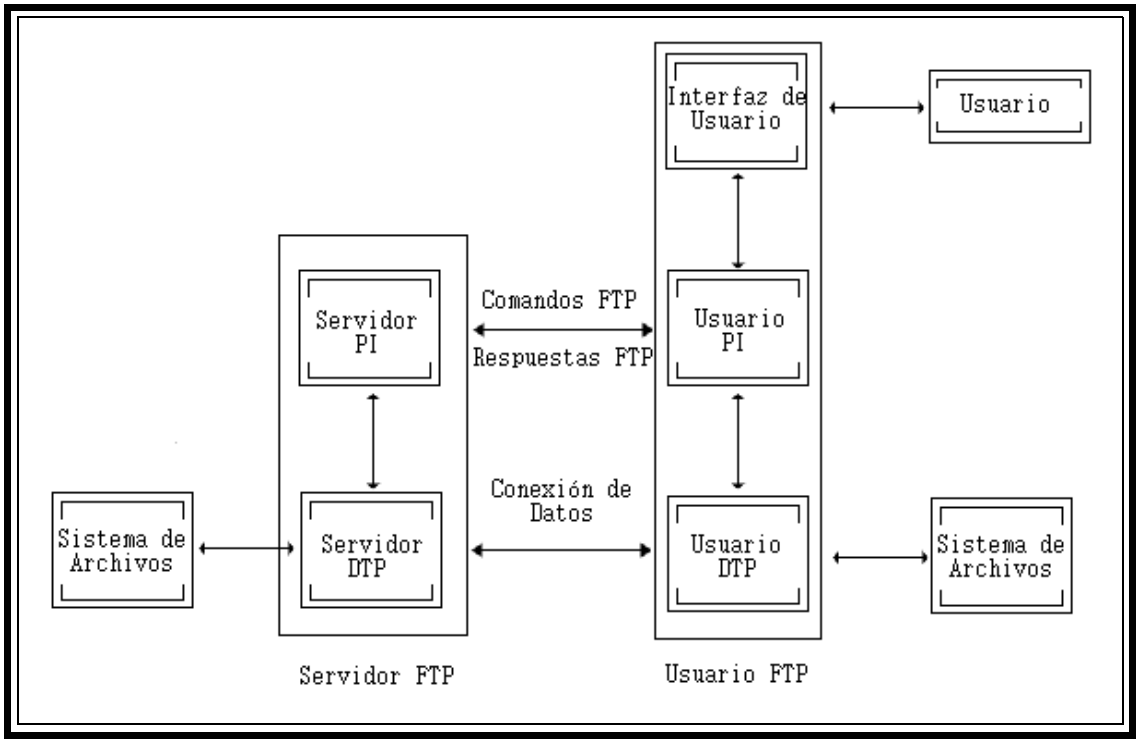
- Usuario DTP: Espera que el servidor inicie la conexión con el puerto de datos asignado y transfiere los datos en función de los parámetros que se hayan especificado.

- Usuario PI: Inicializa la conexión de control desde un puerto determinado, inicializa los comandos FTP y controla al usuario DTP si dicho proceso forma parte de la transferencia de archivos.

- Interfaz de Usuario: Permite que un lenguaje local sea utilizado en los diálogos que sostienen el servidor y el cliente.

La Figura 20 ilustra el proceso para la transferencia de archivos utilizando las entidades descritas anteriormente.

El usuario PI inicia la conexión de control, genera comandos FTP estándar y los transmite al servidor mediante dicha conexión. En respuesta a estos comandos el servidor PI envía acuses de recibo (ACK) positivos o negativos a través de la conexión de control, dependiendo de la recepción de los mensajes por parte del servidor. Los comandos FTP especifican el puerto de datos, el modo de transferencia, el tipo de representación y la estructura que se constituyen en los parámetros necesarios para la conexión de datos. El usuario DTP debe empezar a escuchar por el puerto de datos especificado y el servidor es quien inicia la conexión de datos y la transferencia de ellos de acuerdo con los parámetros especificados. La conexión de datos se utiliza simultáneamente para enviar y recibir, solo mediante ésta, es posible la transferencia de archivos.



14 Data Transfer Process.

Figura 20. Componentes del protocolo FTP

En el caso en el que un usuario quiera realizar una transferencia de archivos entre dos *hosts*, ese usuario configura las conexiones de control de los dos servidores y establece la conexión de datos entre ellos. La información de control se transfiere al usuario PI, pero los datos se intercambian entre los procesos de transferencia de datos de los servidores. El diagrama que ilustra el flujo de información entre las entidades, se observa en la figura 21.

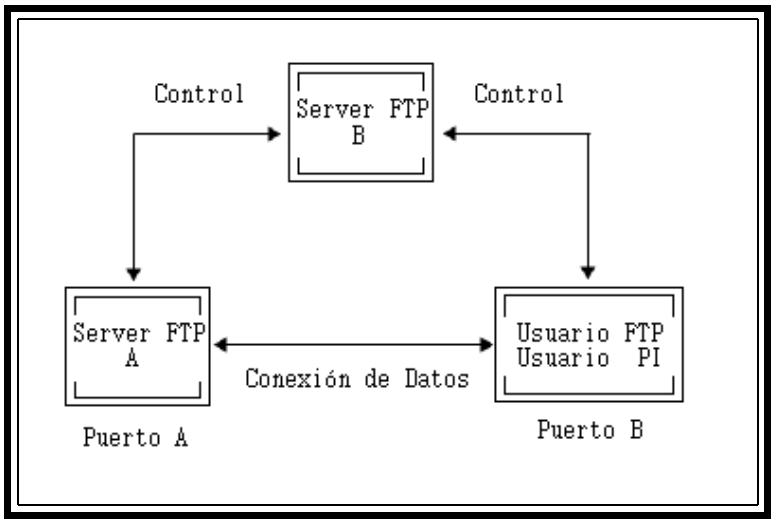


Figura 21. Flujo de información FTP

Como se mencionó anteriormente, FTP utiliza el protocolo Telnet sobre la conexión de control, para lo cual existen dos formas en las cuales estos protocolos pueden interactuar: En la primera, el usuario PI o el servidor PI implementan directamente las reglas para el protocolo Telnet dentro de sus procedimientos, generando una comunicación eficiente e independiente; y en la segunda el usuario o servidor PI hacen uso de los módulos propios de Telnet en el sistema, lo que se traduce en una fácil implementación, programación modular y reutilización de código. **[24]**

6.3 REPRESENTACION Y ALMACENAMIENTO DE LOS DATOS

A menudo, los datos almacenados en el *host* que envía y los almacenados en el *host* que recibe, tienen una representación diferente, por lo que se hace necesaria una transformación de éstos cuando se presenta una transferencia de archivos. La representación de los datos es definida por el usuario, cuando especifica el tipo de representación, entre los cuales están:

Tipo ASCII: Es el tipo de representación por defecto y debe ser aceptado por todas las implementaciones FTP. Este tipo de representación fue definida primordialmente para la transferencia de archivos de texto. El equipo que envía los datos, los convierte de una representación de caracteres interna a la representación estándar NVT-ASCII (esquema de codificación de comandos) de 8 bits. El receptor convertirá los datos del tipo estándar a su propia forma interna. Utilizar la representación estándar NVT-ASCII implica que los datos se deben interpretar como bytes de 8 bits.

Tipo EBCDIC: Se utiliza para la transferencia de información en la que intervienen *host*s que utilizan como representación interna de caracteres EBCDIC. Para la transmisión, los datos son representados como caracteres EBCDIC de 8 bits.

Tipo IMAGE: En este tipo de representación, los datos se envían como bits contiguos, el transmisor empaqueta los datos en bytes de transferencia de 8 bits y el receptor los almacena como bits contiguos. Para conseguir un correcto tamaño de las palabras o bloques de palabras, se utilizan bits de relleno (solo ceros) hasta el límite pertinente, los cuales solo pueden ir al final del archivo y debe existir una forma de identificarlos para eliminarlos en el momento en que se reciba el archivo. Esta representación garantiza un eficiente almacenamiento y recuperación de los archivos, y es muy utilizada para la transferencia de datos binarios.

Tipo LOCAL: Los datos se envían en bytes lógicos del tamaño especificado por el parámetro Byte Size. Éste valor debe ser un entero decimal, y debe ser especificado obligatoriamente, ya que, no existe valor por defecto para él. En el receptor, los datos se transforman de forma independiente al tamaño del byte lógico.

La transferencia de datos debe terminar con un end-of-file (EOF fin de archivo), el cual puede ser expresado implícita o explícitamente por el cierre de la conexión de datos. El end-of-line (EOL fin de línea) en un archivo con tipo de representación ASCII o EBCDIC se debe indicar con los caracteres <CRLF> y <NL> respectivamente.

6.4 MODOS DE TRANSMISION

Existen 3 modos para realizar la transferencia de datos, definidos en el protocolo FTP:

- Modo STREAM: Los datos se transmiten como un flujo de bits, no hay restricciones para el uso del tipo de representación y se permiten estructuras de registro.

- Modo BLOCK: El archivo se envía como una serie de bloques de datos, precedidos por uno o más bytes de cabecera. Los bytes de cabecera contienen un campo contador que indica la longitud total en bytes del bloque de datos, y un código descriptor que define el último bloque del archivo (EOF), el indicador de reinicio o los datos sospechosos, por ejemplo cuando se trabaja sobre una conexión no confiable.

  La cabecera contiene 3 bytes, 24 bits para la información del encabezado, 16 para representar el contador y 8 para representar el código descriptor.

- Modo COMPRESSED: Es utilizado para incrementar el ancho de banda en transmisiones de red muy grandes, reduciendo el tamaño de los archivos.

6.5 COMANDOS FTP

Los comandos FTP se dividen en 3 tres grupos, de acuerdo a las operaciones a realizar: comandos para control de acceso, para la transferencia de parámetros y para el servicio FTP.

6.5.1 Comandos para el Control de Acceso:

- USER: Identificación de usuario requerida por algunos servidores para el acceso a sus archivos. Usualmente, este es el primer comando transmitido por el cliente una vez la conexión de control se ha establecido.

- PASS: Especifica la contraseña de usuario y debe estar presente inmediatamente después del comando USER, complementa la identificación del usuario para el control de acceso.

- ACCOUNT: Identifica una cuenta de usuario. Los códigos devueltos a éste comando dependen de los siguientes casos: Si la información de cuenta es utilizada para efectos de validación la respuesta al comando contiene el código 332; si la cuenta no es utilizada para validación, se obtiene un código 230 y si la cuenta y si la cuenta se requiere para una orden enviada más tarde, se obtiene una respuesta con código 332 o 532 dependiendo de si se almacena o se descarta la orden.

- CWD (CHANGE WORKING DIRECTORY): Este comando le permite al usuario trabajar en directorios diferentes con la misma sesión, es decir, sin alterar su información de acceso (login).

- CDUP (CHANGE TO PARENT DIRECTORY): Es un caso especial de CWD, y se utiliza para la transeferencia de directorios entre sistemas operativos que utilizan sintaxis diefrentes para nombrar al directorio raíz.

- SMNT (STRUCTURE MOUNT): Permite montar un sistema de archivos diferente sin alterar la información de validación del usuario.

- REIN (REINITIALIZE): Se utiliza para inicializar todos los parámetros de la sesión y eliminar toda la información de cuenta.

- LOGOUT: Finalizar la conexión FTP, si la transferencia de un archivo está en progreso cuando se envía este comando, la conexión se mantiene abierta hasta que el servidor envíe una respuesta con el resultado de la transferencia, si no hay un archivo en proceso de transeferencia, el servidor cierra la conexión de control.

6.5.2 Comandos para la transferencia de parámetros

- PORT: Especifica el puerto que será utilizado para la conexión de datos, en este comando se debe incluir una dirección IP de 32 bits y una dirección de puerto TCP de 16 bits.

- PASV: Indica al servidor DTP que debe escuchar por un puerto de datos y esperar para recibir una conexión. Como respuesta a éste comando se obtiene la dirección del puerto y la dirección IP.

- TYPE: Especifica el tipo de representación a utilizar, los posibles tipos se encuentran enumerados en la sección Representación y Almacenamiento de los datos. El tipo de representación por defecto es ASCII.

- MODE: Este comando especifica el modo de transferencia de los datos, los posibles valores son:
  S – Método Stream
  B – Método Block
  C – Método Compressed
  El valor por defecto para este comando es S.

6.5.3 Comandos para el servicio FTP

- STOR: Habilita al servidor DTP para aceptar los datos transferidos a través de la conexión de datos y almacenarlos como un archivo en el servidor.

- APPE: Este comando hace que el servidor DTP reciba la información a través de la conexión de control y los guarde en un archivo en el servidor; si el archivo especificado existe, la información se anexa, de lo contrario se crea un nuevo arhivo en el servidor.

- ALLO (ALLOCATE): Se utiliza cuando un servidor necesita reservar sufucuente espacio de almacenamiento para reecibir los datos de un archivo.

- REST: Este comando define un marcador a partir del cual la transferencia del archivo será reiniciada y debe ir seguido de un comando que habilote de nuevo la transferencia.

- ABOR: Este comando pide al servidor que interrumpa la orden de servicio FTP previa y cualquier transferencia de datos asociada.

- DELE (DELETE): Este comando se utiliza para borrar un archivo en una ruta específica del servidor.

- MKD (MAKE DIRECTORY): Utilizado para la creación de directorios, dada una ruta específica en el servidor.

- PWD: Despliega el nombre del directorio actual de trabajo. **[24]**

6.6 MODIFICACIONES A LA ESPECIFICACIÓN FTP PARA INCLUIR SOPORTE IPV6

El protocolo para la transferencia de archivos según lo especificado en el RFC 959 solo posibilita la comunicación de información en redes con conexión de datos IPv4, debido a que éste asume que las direcciones de red tendrán una longitud de 32 bits; por lo cual se hace necesario realizar algunos cambios para que el protocolo FTP acepte conexiones de datos tanto para IPv4 como para IPv6.

Estos cambios consisten en reemplazar los comandos FTP para la transferencia de parámetros PORT y PASV, ya que éstos incluyen en su definición el manejo de la longitud de direcciones IP. El comando PORT es reemplazado por el comando EPRT, permitiendo así especificar una dirección IP extendida para la realización de la conexión de datos y el comando PASV es reemplazado por EPSV.

6.6.1 Comando EPRT

El formato del comando EPRT es el siguiente:

EPRT - NET PTR - NET ADDR – TCP PORT

En el argumento NET PTR se especifica un número asociado al protocolo de red que se vaya a utilizar para realizar la conexiòn de datos. La asignación de dicho número según el protocolo se indica en la tabla 9:

Tabla 9. Valores para NET PTR.

Número AF	***Protocolo***
1	Protocolo IPv4
2	Protocolo IPv6

En el argumento NET ADDR se especifica la representaciòn textual de la dirección de red. Se aceptan dos tipos de formatos para la representaciòn de las direcciones, dependiendo del protocolo especificado. En la tabla 10 se encuentran los formatos de direcciones que acepta este comando.

Tabla 10. Formato para el campo NET ADDR.

Número AF	***Fornato de Dirección***	***Ejemplo***
1	Decimales separados por puntos	172.16.41.108
2	Representación textual con compresiòn de bloques continuos de ceros	3ffe:8070:1024::5

El argumento TCP PORT indica el número del puerto TCP sobre el cual el *host* estará escuchando. A continuación se enuncian dos ejemplos acerca de cómo utilizar el comando EPRT para establecer una conexión con el protocolo IPv4 sobre el puerto 6275 y otra con IPv6 sobre el puerto 5282 respectivamente.

EPRT |1| 172.16.41.108 |6275|
EPRT |2| 3ffe:8070:1024:1::10 |5282|

Ante un comando EPRT el servidor retornará el código 200 para indicar que no existieron errores, los códigos 500 y 501 en caso contrario y el código 522 para especificarle al cliente que él no soporta los requerimientos del protocolo que se desea utilizar. Cuando un servidor envíe el código 522 también debe enviar una porción de texto en la que se detalle para cual protocolo el servidor tiene soporte.**[24]**

6.6.2 Comando EPSV

El comando EPSV solicita a un servidor que escuche sobre un puerto de datos y espere por una conexión, el servidor ante éste comando responde con los códigos 500 y 501 para manejo de errores o con el código 229, un mensaje de texto indicando que el servidor está entrando en modo pasivo y el puerto tcp cuando el comando fue bien procesado. Un ejemplo de respuesta es:

Entering Extendes Passive Mode (| | | 2185|)
Donde 2185 es el puerto TCP.

Para el caso en que el servidor le retorne al cliente que no soporta el protocolo que el cliente está utilizando, éste último debe ejecutar el comando ABOR para permitir que el servidor cierre definitivamente la conexión de escucha. Posteriormente el cliente podrá enviar un nuevo comando especificando el protocolo que si soporta ese servidor mediante la expresión **EPSV NET PTR**.

El protocolo de red y dirección IP utilizados para la conexión de datos deben ser los mismos que los utilizados para la conexión de control. **[25]**

6.7 IMPLEMENTACIONES DEL PROTOCOLO FTP

6.7.1 PROFTPD

ProFTPD es un servidor FTP con licencia GPL que se ejecuta en una variedad de plataformas UNIX, sus principales cualidades son su seguridad y facilidad de configuración. El estilo de configuración es similar al Apache, es altamente configurable, incluyendo soporte para múltiples servidores FTP virtuales, FTP anónimo, visibilidad de directorios basada en permisos, etc.

El archivo de configuración principal para este servidor es /etc/proftpd y se puede ejecutar como INETD (demonio que atiende las solicitudes de conexión que llegan al equipo, y está a la espera de todos los intentos de conexión que se realicen) o un servidor autónomo.

Cuenta con soporte IPv6 nativo adicionando el soporte para los comandos EPSV y EPRT a partir de la versión 1.2.9rc2.

6.7.2 VSFTPD (Very Secure FTP Daemon)

VSFTPD es un servidor FTP con licencia GPL para sistemas UNIX, incluyendo Linux. Es estable seguro y extremadamente rápido. Entre sus características más importantes se encuentran: Configuración para *Host* virtuales y usuario virtuales, operación automática o con INETD, regulación del ancho de banda, soporte para encriptar a través de integración SSl, entre otros.

Cuenta con soporte IPv6 nativo a partir de la versión 1.2.0.

6.8 INSTALACIÓN Y CONFIGURACIÓN DEL SERVIDOR FTP SELECCIONADO

VSFTPD y PROFTPD tienen un comportamiento similar, y ofrecen buenos niveles de seguridad al permitir configurar restricciones a nivel de usuario, además ambas cuentan con soporte nativo IPv6, por lo tanto para realizar las pruebas se trabajará con los dos servidores para comprobar su compatibilidad con los diferentes clientes FTP.

6.8.1 Instalación

6.8.1.1 Instalación VSFTPD

A. De la página http://rpm.pbone.net para Fedora Core 4, se descarga el archivo vsftpd-2.0.3-1.rpm y se ejecuta el comando **rpm –i** vsftpd-2.0.3-1.**rpm** para realizar la instalación.

B. Una vez instalado el paquete, éste genera el archivo de configuración en */etc/vsftpd/* con el nombre vsftpd.conf.

C. Para iniciar y para el servidor se ejecutan los comandos */etc*/init.d/vsftpd start y stop respectivamente. Para comprobar el acceso al servidor FTP desde un *host* remoto, se deben observar los logs generados por el servidor en la ruta */var/log/xferlog*.

6.8.1.2 Instalación ProFTPD

A. Si se quiere utilizar el soporte IPv6 que ofrece este paquete, éste debe ser habilitado en la instalación. Se descarga el archivo proftpd-1.3.0.tar.gzde la URL ftp://ftp.proftpd.org/distrib/source, se descomprime con el comando **tar –xvzf** y para instalarlo se debe ejecutar:

#./configure –enable-IPv6
#make
#make install

B. Para el caso específico de Fedora Core4, el paquete proftpd-1.3.0-6.fc4.rpm ya viene con la opción IPv6 habilitada, por lo cual solo es necesario bajar el paquete de http://rpm.pbone.net y ejecutar **rpm –i proftpd-1.3.0-6.fc4.rpm.**

C. Para iniciar el servidor de tal forma que se puedan observar los logs, ingresamos el comando /usr/sbin/proftpd –n, para iniciar y pararlo en modo nomal se ejecuta /etc/init.d/start y stop respectivamente.

D. La instalación generará el archivo de configuración para este servidor en /etc/ con el nombre proftpd.conf.

6.8.2 Configuración

6.8.2.1 Configuración VSFTPD

El archivo de configuración para el servidor VSFTPD se puede apreciar en la Figura 22.

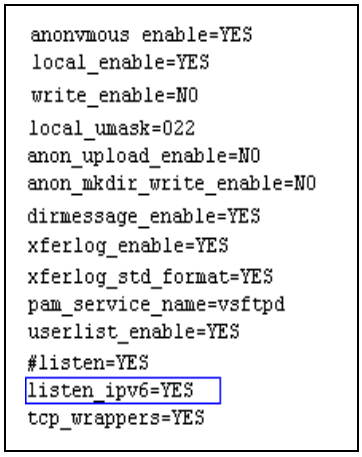


Figura 22. Archivo de Configuración para VSFTPD

Anonymous enable permite que los usuarios anónimos se conecten al servidor VSFTPD, *local_enable* indica si los usuarios locales del sistema puden acceder al servidor, en éste caso si es posible, *write_enable* y *anon_upload_enable* en conjunto le permiten al usuario anónimo ejecutar comandos de escritura, debe estar deshabilitado por seguridad, *local_umask* especifica el valor de UMASK para la creación de archivos, el valor por defecto debe estar en octal incluyendo un prefijo cero, *anon_upload_enable* se habilita para permitir a los usuarios anónimos subir archivos al directorio por defecto del servidor FTP, por seguridad esta directiva debe estar deshabilitada y las directivas xferlog_enable y xferlog_std_format se relacionan con los logs del servicio. Para permitir que el servidor VSFTPD atienda solicitudes IPv6 es necesario agregar la línea *listen_IPv6* = YES, cabe aclarar que ésta línea no viene incluida en el archivo de configuración que se genera después de la instalación, se debe agregar y se debe comentar la línea *listen* = YES, ya que no es posible utilizar estos dos parámetros al tiempo. Con *listen_IPv6* también se está indicando que el servidor se ejecutará en modo autónomo.

6.8.2.2 Configuración ProFTPD

El archivo de configuración para el servidor ProFTPD como se mencionó anteriormente tiene una estructura muy similar a la manejada por el servidor Web Apache, ya que permite especificar los parámetros del servidor con etiquetas muy similares y la creación de *host* virtuales. Para trabajar con el soporte IPv6 que este paquete ofrece, el archivo de configuración puede quedar de las formas ilustradas en las figuras 23 y 24.

```
ServerName                      "ProFTPD Default Installation"
ServerType                      standalone
DefaultServer                   on
# Port 21 is the standard FTP port.
Port                            21
# Umask 022 is a good standard umask to prevent new dirs and files
# from being group and world writable.
Umask                           022
User                            nobody
Group                           nobody
AllowOverwrite          on
<Anonymous ~ftp>
  User                          ftp
  Group                         ftp
  UserAlias                     anonymous ftp
  MaxClients                    10
  <Limit WRITE>
    DenyAll
  </Limit>
</Anonymous>
<VirtualHost 2001:448:1024:1::10>
ServerName      "ftp.ipv6.unicauca.edu.co"
ServerAdmin     "root@ipv6.unicauca.edu.co"
</VirtualHost>
```

Figura 23. Archivo de configuración para ProFTPD con *VirtualHost*

ServerName almacena el nombre del servidor FTP, *Server Type* indica si correrá de forma automática (standalone) o con INETD, si se habilita DefaultServer se está especificando que éste será el servidor FTP por defecto. Debido a que el servidor FTP servirá para descargar archivos que cualquier usuario pueda leer, se define el grupo y el usuario anónimo, que aceptará conexiones con los nombres de usuario anonymous y ftp. *Maxclients* especifica el número máximo e usuarios permitidos, se deja el valor por defecto que trae el archivo de configuración.

Con respecto a IPv6 es posible definir la dirección por defecto del servidor como una dirección IPv6 en éste caso 2001:448:1024:1::10 tal y como lo muestra la figura 24, en éste caso el servidor entenderá direcciones IPv6 y relacionará las peticiones con la dirección IPv6 especificada. También es posible crear un Virtual*Host* en el archivo de configuración e indicar cual es el servidor de nombres para él. Al utilizar el parámetro <VirtualHost >, para IPv4 se puede incluir una dirección o un nombre de máquina, para IPv6 solo es posible especificar la dirección IPv6, ya que el software no está en capacidad de interpretar los registros AAAA que devuelve el servidor cuando se consulta por un nombre de máquina cuya dirección es IPv6, Ver figura 23.

Las pruebas realizadas con los diferentes clientes FTP en Windows y Linux, se encuentran en el Anexo E.

```
ServerName                      "ProFTPD Default Installation"
ServerType                      standalone
DefaultServer                   on
DefaultAddress                  2001:448:1024:1::10
# Port 21 is the standard FTP port.
Port                            21
# Umask 022 is a good standard umask to prevent new dirs and files
# from being group and world writable.
Umask                           022
User                            nobody
Group                           nobody
AllowOverwrite          on
<Anonymous ~ftp>
  User                          ftp
  Group                         ftp
  UserAlias                     anonymous ftp
  MaxClients                    10
  <Limit WRITE>
    DenyAll
  </Limit>
</Anonymous>
```

Figura 24. Archivo de configuración para ProFTPD con servidor por defecto

7. SSH SECURE SHELL – ACCESO REMOTO

7.1 DEFINICIÓN

SSH es un protocolo basado en la arquitectura cliente/servidor, permite que un usuario se conecte a un *host* remotamente o ejecute procesos remotos utilizando el puerto 22, que ha sido registrado por la IANA y oficialmente asignado para SSH, en redes que utilizan TCP/IP. A diferencia de los protocolos FTP y TELNET, que también facilitan el acceso remoto, SSH se encarga de encriptar la información de registro, es decir, los datos utilizados para la validación de sesión, reduciendo así los riesgos de seguridad tanto para el sistema cliente como para el sistema remoto, cuando se envía información a través de una red insegura. SSH también es ampliamente utilizado para asegurar el uso de protocolos inseguros.

Las técnicas de cifrado que utiliza SSH para la manipulación de la información, permiten asegurar que alguien que no sea un legítimo destinatario acceda a información que no le corresponde, que la información no pueda ser alterada en el tránsito desde el emisor hacia el destinatario y que tanto el emisor como el receptor puedan confirmar la identidad de la otra parte involucrada en la comunicación. Éstos tipos de cifrado se dividen en dos grupos: los de cifrado simétrico y los de cifrado asimétrico.

La técnica de cifrado simétrico, consiste en el uso de una clave que es conocida por el emisor y el receptor involucrados en la comunicación. Cuando el emisor desea enviar un mensaje al receptor, utiliza un algoritmo de cifrado simétrico y la clave que tienen en común, generando así el nuevo mensaje que será transmitido. El receptor utilizando la clave y el algoritmo inverso utilizado por el emisor, obtiene el mensaje original. El algoritmo de cifrado utilizado debe contar con las técnicas necesarias para garantizar que sea difícil de descifrar, sin embargo, este sistema presenta vulnerabilidades, ya que es necesario que ambas partes conozcan la clave. **[26]**

La técnica de cifrado asimétrico consiste en el uso de 2 llaves: una pública y una privada. La clave pública como su nombre lo indica se puede hacer pública y se entrega a aquellos receptores que nos vayan a enviar mensajes cifrados, la clave privada se debe mantener en secreto. El emisor tiene 2 llaves, por ejemplo JyK, para denotar la pública y la privada respectivamente; cuando éste desea transmitir un mensaje que nadie, excepto el receptor específico debe conocer, con el uso de un algoritmo de cifrado asimétrico y su llave privada, genera un nuevo mensaje ya cifrado y lo transmite. Posteriormente en el destino, se utiliza un algoritmo de cifrado inverso al que utilizó el emisor y la clave pública del emisor (la cual ha sido enviada al servidor con anterioridad) para obtener el mensaje original. Si bien las dos llaves están fuertemente relacionadas entre sí, no es posible calcular la primera a partir de los datos de la segunda, ni tampoco a partir de los documentos cifrados con la clave privada. Este tipo de cifrado opera de tal forma que la información cifrada con una de las claves sólo puede ser descifrada con la otra; es decir, si un usuario cifra determinada información con su clave privada, cualquier persona que conozca su clave pública podrá descifrar la misma. Por lo tanto, si un receptor logra descifrar un mensaje utilizando la llave pública del emisor, puede afirmarse que el mensaje lo generó dicho emisor utilizando su clave privada, validación de la identidad del cliente y servidor.

7.2 VERSIONES DEL PROTOCOLO

En la actualidad existen dos versiones del protocolo SSH: SSH1.x (ssh1.3 y ssh1.5) y SSH2. SSHv2 nace para corregir algunas especificaciones de la versión 1, debido a

que ésta última presenta una vulnerabilidad de seguridad relacionada con la posibilidad de que un usuario pueda insertar datos en el flujo de comunicación.

En la tabla 11 se enuncian las principales diferencias y características de las 2 versiones del protocolo SSH.

Tabla 11. SSHv1 vs SSHv2

SSH Versión 1	***SSH Versión 2***
Diseño monolítico (Integrado)	Separación en capas de las funciones de autenticación, conexión y transporte.
Integridad via CRC32 (no seguro)	Integridad via HMAC (cifrado hash)
En la negociación permite utilizar únicamente cifrado simétrico, en la identificación de sesión se utiliza una clave única en los dos lados de la comunicación.	En la negociación se pueden utilizar métodos como cifrado simétrico, asimétrico, llave pública, etc. En la identificación se utiliza una clave de sesión independiente en ambos lados.
Sólo es posible utilizar el algoritmo de llave pública RSA.	RSA y DSA son los algoritmos definidos para la creación de llaves públicas.
La clave de sesión es válida para toda la sesión.	La clave de sesión se renueva, despues de cierta cantidad de datos enviados.
Existe un único canal por sesión.	Existen ilimitado número de canales por sesión.

Las dos versiones del protocolo SSH no son compatibles entre sí, sin embargo, las actuales implementaciones software del protocolo son diseñadas de tal forma que las dos versiones puedan correr en un mismo computador, haciendo que la transición de la vieja pero bien establecida versión 1 hacia la más segura y flexible versión 2, sea más rápida y sencilla.

7.3 MÉTODOS DE AUTENTICACIÓN

Para la autenticación de los usuarios, el protocolo SSH especifica dos métodos:

1. Autenticación con contraseña: En este método el programa cliente SSH solicita al usuario el ingreso de una contraseña que se enviará al servidor SSH para realizar la validación. El usuario deberá ingresar la contraseña cada vez que desee conectarse al servidor.

2. Autenticación con clave pública: Para éste método de autenticación, el usuario debe tener una llave pública y una privada, y la llave pública debe estar almacenada en el servidor al cual desea conectarse. Una vez establecida la conexión, el servidor genera un número aleatorio que se cifra con esa llave pública utilizando el algoritmo RSA o DSA (la explicación matemática de estos algoritmos se puede apreciar en http://es.tldp.org/Tutoriales/doc-ssh-intro/intro-ssh/node34.html). Este número aleatorio cifrado se envía al cliente, quien debe descifrarlo con su llave privada y enviarlo de nuevo al servidor, para garantizar que el usuario es realmente quien dice ser. Este método presenta una ventaja muy notable frente a la autenticación con contraseña, debido a que, no es necesario que el usuario ingrese la contraseña para establecer una conexión y la contraseña no es enviada al servidor en ningún momento, lo que se manejan son las llaves del usuario.

7.4 FUNCIONAMIENTO

El funcionamiento del protocolo SSH, en lo referente a la ejecución de procesos remotos inseguros, se basa en la construcción de un túnel mediante el cual viajarán los datos de una manera segura ("tunneling"). En cada uno de los extremos del túnel se encuentra las aplicaciones estándares y ssh se encarga de recoger todos los datos que el cliente desea enviar y los reenvía por el túnel o canal seguro hacia el servidor, función que se denomina "port-forwarding".

Con respecto al acceso remoto, cuando el cliente intenta acceder al servidor para establecer un canal seguro entre ellos, se realizan las siguientes tareas:

- Intercambio de claves.
- Determinación del algoritmo de encriptación de la clave pública.
- Determinación del algoritmo de la encriptación simétrica o asimétrica.
- Determinación del algoritmo de autenticación de mensajes.
- Determinación del algoritmo de hash (Firmas de comprobación aleatoria) que se va a utilizar. La firma digital se transforma a través de una función hash en un número único que es descifrado en el otro extremo del mensaje utilizando el mismo algoritmo y posteriormente verificado con el agente autentificador que almacena estos datos

Durante el intercambio de llaves, el servidor utiliza una llave de *host* para demostrarle al cliente que él es el servidor que dice ser.

A continuación se enumeran cada uno de los pasos realizados por un cliente y un servidor para hacer uso del protocolo SSH en su intecambio de información:

- El cliente abre una conexión TCP mediante el puerto 22 del servidor.
- De acuerdo a la configuración y capacidades tanto del cliente como del servidor, se establece la versión del protocolo que se va a utilizar.
- El servidor le envía al cliente la llave pública y el cliente compara esa clave con la que él tiene almacenada para verificar la autenticidad del servidor al cual se va a conectar.
- El cliente selecciona un algoritmo de cifrado y genera una clave para la sesión que está por establecerse; ésta información la almacena en un mensaje y utilizando la clave pública del servidor, lo cifra y envía.
- Durante toda la sesión se utilizará el algoritmo seleccionado por el cliente y la clave de sesión.
- Posteriormente se realiza la autenticación del usuario con alguno de los métodos enunciados anteriormente. Cabe resaltar que antes de la autenticación de usuario ya se ha establecido un canal cifrado seguro por lo cual la información que se intercambie en esa sesión no puede ser fácilmente descifrado.
- Se inicia la sesión entre el cliente y el servidor o emisor y receptor.
- Después que una cierta cantidad de datos ha sido transmitida, con el algoritmo que seleccionado anteriormente, ocurre otro intercambio de claves que genera un nuevo secreto compartido y se repiten los mencionados para la supervisión de la comunicación. **[27]**

7.5 COMPONENTES

SSHv2 está construido en base a 3 protocolos principales: Protocolo de Transporte, de Autenticación y de Conexión.

7.5.1 Protocolo de nivel de transporte

Proporciona autenticación, confidencialidad e integridad a la información y normalmente opera sobre TCP/IP. Además permite la compresión de datos, lo que acelera la transmisión de la información

Cuando la conexión ha sido establecida, tanto el emisor como el receptor envían una cadena llamada "identificación", para especificar la versión del protocolo SSH que soportan. La versión 2 del protocolo es la única que tiene actualmente documentación oficial, pero las versiones anteriores a ésta (1.3, 1.5, etc) son ampliamente conocidas y muchas aplicaciones todavía las utilizan, razón por la cual en el "identificador", pueden especificarse las viejas y nuevas versiones del protocolo.

El servidor envía su "identificación" y no debe enviar datos hasta recibir la identificación del cliente para determinar la versión del protocolo que éste soporta. El servidor contiene una bandera que le indica si puede o no tener compatibilidad con versiones anteriores, de ésta forma, si un cliente con una versión antigua intenta conectarse con un servidor que tiene implementada una versión reciente y la bandera tiene como valor ON (acepta versiones antiguas), dicho servidor empieza a trabajar con la versión que el cliente tiene.

Para el caso contrario, en el que un cliente con soporte de la última versión desea conectarse a un servidor con soporte para versiones antiguas, el cliente una vez identifica la versión del protocolo en el servidor, cierra la conexión y vuelve a conectarse pero utilizando la versión antigua del protocolo.

Los paquetes que maneja el protocolo de transporte siguen el siguiente formato:

- Packet_Length: Indica la longitud del paquete en bytes sin incluir el campo MAC.
- Padding_Length: Especifica en bytes la longitud del relleno utilizado.
- Payload: Carga últil del paquete.
- Random Padding: Indica la longitud arbitraria del relleno, para lograr que la longitud total del paquete (packet_length+padding_length+payload+ random padding) sea un múltiplo de 8 o del tamaño del bloque , se elige el que sea mas largo. La máxima cantidad de relleno permitida es 255 bytes.
- MAC: Código del mensaje de autenticación.

El campo "mac" garantiza la integridad de los datos, ya que, es calculado utilizando un secreto compartido, el número de secuencia del paquete y el contenido del paquete. Inicialmente su longitud es cero y durante el intercambio de llaves entre cliente y servidor es negociado el algoritmo para encriptar el paquete. **[28]**

El tamaño mínimo de un paquete es 16 bytes más la longitud del campo MAC y el máximo es 35.000 bytes incluyendo el campo MAC.

Si se va a aplicar compresión de la información, el campo "payload" se comprime utilizando el algoritmo negociado previamente, y los campos "packet_length" y "mac"

serán calculados a partir de la longitud del payload ya comprimido. El único tipo de compresión que actualmente se encuentra definido es ZLIB, cuyo formato de datos es portable para cualquier plataforma.

Los mensajes y correspondientes números que maneja éste protocolo son:
- 1 – 29 Nivel de transporte genérico, por ejemplo, desconectar, ignorar, etc.
- 20 – 29 Algoritmos de negociación.
- 30 – 49 Métodos específicos para el intercambio de llaves.

7.5.2 Protocolo de autenticación de usuario

Se encarga de autenticar al usuario ante el servidor, una vez el protocolo de transporte ha sido establecido. Funciona sobre el protocolo de nivel de transporte y asume que los protocolos que se encuentran por debajo de él, proveen protección para la integridad y confidencialidad. El servidor puede ser configurado para que acepte varios tipos de autenticación y el cliente será quien decida en que orden intentará utilizar dichos métodos; el nombre del servicio para éste protocolo es “ssh-userauth”.

Las peticiones de autenticación ante un servidor se envían en mensajes denominados SSH_MSG_USERAUTH_REQUEST. El formato de estos mensajes contiene los siguientes campos:

- *User Name*: Identifica el nombre de usuario que se utilizará para la conexión, si este campo no se especifica, la autenticación no será aceptada.

- *Service Name*: Especifica el servicio que se va a correr una vez la autenticación haya ocurrido, si el servicio no existe la autenticación no será aceptada.

- *Method Name*: Indica el método de autenticación a utilizar, los valores definidos para este campo son:

    1. Publickey = Autenticación requerida
    2. Password = Opcional
    3. Hostbased = Opcional
    4. None = No recomendado

  Si el servidor rechaza la petición de autenticación realizada, éste responde con mensajes SSH_MSG_USERAUTH_FAILURE, los cuales están compuestos por los campos:

- *Authentication that can continue*: En este campo se almacena una lista de nombres, separados por coma, con los valores “method name” que son capaces de continuar con el diálogo de autenticación.

- *Partial Success*: El valor de este campo debe ser TRUE si la petición de autenticación para la cuál ésta es una respuesta fue exitosa; y debe ser FALSE si la petición no fue exitosamente procesada.

Puede darse el caso en el cual un cliente envía varias peticiones de autenticación antes que la respuesta para cada una de ellas se haya efectuado. En este caso, el servidor debe procesar cada petición por completo y si hay peticiones fallidas, envía mensajes SSH_MSG_USERAUTH_FAILURE antes de procesar las próximas peticiones. **[29]**

Una vez el servidor acepta la autenticación, responde al cliente con un mensaje SSH_MSG_USERAUTH_SUCCESS, el proceso finaliza y el servidor corre el servicio solicitado por el cliente. Este mensaje debe ser enviado solo una vez, si posterior a esto existe alguna petición de autenticación, dicha petición debería ser ignorada según el RFC según el RFC 4252.

Todos los mensajes utilizados por el protocolo de autenticación tienen asignado un número dentro del rango 50 – 79. Los mensajes con número 80 y superior, son reservados para ejecutar protocolos por encima del protocolo de transporte, así que si un mensaje de éste tipo se recibe antes que la autenticación haya sido completada ocurrirá un error, a lo cual el servidor debe responder desconectándose. Los mensajes para autenticación se agrupan de la siguiente forma:

- 50 – 59 Mensajes genéricos de autenticación de usuario.
- 60 – 79 Métodos específicos para la autenticación de usuarios

Los mensajes genéricos corresponden a los explicados anteriormente, su equivalencia en número es:

*SSH_MSG_USERAUTH_REQUEST 50
*SSH_MSG_USERAUTH_FAILURE 51
*SSH_MSG_USERAUTH_SUCCESS 52

7.5.3 Protocolo de conexión

Una vez realizada la autenticación, éste protocolo multiplexa los datos encriptados en varios canales lógicos y cada canal maneja la conexión para diferentes sesiones de terminal. Tanto clientes como servidores pueden crear un canal nuevo. Cuando el cliente intenta abrir un nuevo canal, los clientes envían el número del canal junto con la petición, esta información es almacenada por el servidor y utilizada para dirigir la comunicación a ese canal. Esto se hace para que diferentes tipos de sesión no afecten una a la otra y así cuando una sesión termine, su canal pueda ser cerrado sin interrumpir la conexión SSH primaria. Una sesión es una ejecución remota de un programa, ya sea, un shell, una aplicación, un comando del sistema o algún subsistema incorporado.

Todo lo referente a peticiones de conexión utiliza los siguientes campos para formar el mensaje SSH_MSG_GLOBAL_REQUEST:

- Request Name
- Want Reply
- Request – Specific data follows

La respuesta al mensaje SSH_MSG_GLOBAL_REQUEST puede ser un mensaje SSH_MSG_REQUEST_SUCCESS o un mensaje SSH_MSG_REQUEST_FAILURE cuando el valor del campo “Want Reply” es TRUE, si por el contrario el valor del campo es FALSE no se enviará ninguna respuesta a la petición.

Cada canal se identifica con un número, el cual hace referencia a que el canal puede ser diferente en cada lado de la sesión; Cuando uno de los extremos que va a ser parte de la conexión (emisor o receptor) desea abrir un canal, este asigna un número local para el canal y posteriormente envía un mensaje con ese número y el tamaño

inicial de la ventana al otro extremo de la conexión. El mensaje se denomina SSH_MSG_CHANNEL_OPEN y contiene los siguientes campos:

- Channel Type: Indica el tipo de canal, entre los cuales están “session” para operaciones shells remotas, “direct-tcpip” para conexiones cliente – servidor, “forwareded-tcpip” para conexiones servidor – cliente y “x11” para operaciones realizadas a través de X11 (Protocolo de comunicaciones para sistemas de ventanas)

- Sender Channel: Es un identificador local para el canal, asignado por la entidad que inicia la petición, es decir, la que envía el mensaje.

- Inicial Window Size: Este campo especifica cuantos bytes pueden ser enviados al extremo que envió el mensaje sin ajuste de ventana.

- Maximum Packet Size: Especifica el tamaño máximo de un paquete de datos individual, que puede ser enviado al extremo que generó el mensaje. **[29]**

El extremo remoto de la conexión, una vez recibe el mensaje SSH_MSG_CHANNEL_OPEN, puede responder con un mensaje SSH_MSG_CHANNEL_OPEN_CONFIRMATION si todo ha salido bien y el canal puede establecerse o SSH_MSG_CHANNEL_OPEN_FAILURE si el extremo que recibe el mensaje no soporta el tipo de canal especificado en él.

El mensaje SSH_MSG_CHANNEL_OPEN_CONFIRMATION contiene los siguientes campos:

- Recipient Channel: Es el número de canal asignado en la petición de conexión original.
- Sender Channel: Es el número de canal asignado por el extremo que envió la petición.
- Initial Window Size
- Maximun Packet Size
- Channel Type Specific Data Follows

Los valores correspondientes a los tres últimos campos son iguales a los enunciados para el mensaje SSH_MSG_CHANNEL_OPEN.

El mensaje SSH_MSG_CHANNEL_OPEN_FAILURE contiene los campos:

- Recipient Channel: Similar al mensaje explicado anteriormente.

- Reason Code: Contiene un valor en formato decimal que indica cuál fue la razón por la cual el canal no se pudo establecer entre los extremos. Los posibles valores para este campo se especifican en la Tabla 12.

Tabla 12. Mensajes de error SSH.

Tipo de Mensaje	*Valor del campo*	*Razón de la Falla*
SSH_OPEN_ADMINISTRATIVELY_PROHIBITED	1	Se prohíbe abrir el canal
SSH_OPEN_CONNECT_FAILED	2	Conexión Fallida
SSH_OPEN_UNKNOWN_CHANNEL_TYPE	3	Tipo de Canal desconocido
SSH_OPEN_RESOURCE_SHORTAGE	4	Ausencia de Recursos

- Language Tag: Especifica el lenguaje que se debe utilizar para el envío de mensajes en respuesta a una petición original.

De igual forma cuando un extremo de la conexión no desea enviar más datos por el canal que se ha establecido con anterioridad, deberá enviar un mensaje SSH_MSG_CHANNEL_EOF. Este mensaje bloque el envío de información solo en una dirección, es decir, el canal permanecerá abierto y será posible enviar datos desde el extremo que no ha decidido cerrar el canal hacia el extremo que envió el mensaje de cierre. Si lo que requiere algún extremo es terminar con el canal, debe enviar un mensaje SSH_MSG_CHANNEL_CLOSE y el canal quedará cerrado en un sentido, el extremo que envió la petición de cierre del canal puede reutilizar el número de canal. **[31]**

El rango de números del 80 al 89 se reservan para mensajes genéricos del protocolo de conexión y el rango entre 90 y 127 se utiliza para los mensajes relacionados a cada canal.

7.6 TIPOS DE DATOS UTILIZADOS POR EL PROTOCOLO SSH

- Byte: Representa un valor de 8 bits, la longitud de los datos se representa como un arreglo de bytes de la forma byte[n], donde n es el número de bytes en el arreglo.
- Boolean: Este tipo de datos se almacena como un solo byte, el valor 0 (cero) representa FALSO y el valor 1 representa VERDADERO. Las aplicaciones no deben trabajar con valores diferentes a 1 o 0.
- Uint32: Representa un entero sin signo de 32 bits almacenado en 4 bytes, en los cuales va primero el más significativo.
- Uint64: Representa un entero sin signo de 64 bits almacenado en 8 bytes.
- String: Es utilizado para almacenar texto.

El funcionamiento, tipos de datos utilizados y componentes del protocolo SSH explicados anteriormente, se mantienen para las redes que utilizan tanto el protocolo IPV4 como el IPV6, debido a que las especificaciones para SSH son muy recientes. Las implementaciones software del protocolo son quienes deben realizar el manejo de los diferentes tipos de sockets para el manejo de las aplicaciones bajo el protocolo IPV4 e IPV6 y brindar soporte a las 2 clases de direcciones, 32 bits y 128 bits.

7.7 IMPLEMENTACIONES DEL PROTOCOLO SSH

Con respecto a implementaciones que se utilizan como servidores del protocolo SSH las más representativas son:

OpenSSH: Es la implementación del protocolo SSH para la plataforma Unix, bajo licencia BSD, desarrollada por OpenBSD. Soporta las versiones 1 y 2 del protocolo, pero dependiendo de cual versión se utilice el soporte para los métodos de autenticación cambia; si un cliente se conecta con la versión 1 del protocolo, la autenticación por clave pública solo soporta el uso del algoritmo RSA, si se conecta con la versión 2 se soportan los algoritmos RSA y DSA.

OpenSSH permite establecer conexiones remotas, crear sesiones seguras con servidores smtp, pop3,etc. y tiene soporte para el protocolo IPV6 desde el año 2000, sin necesidad de utilizar parche o librería externa alguna, el soporte ya viene incluido en el paquete de instalación. OpenSSH ha sido uno de los primeros paquetes que desarrolló el soporte IPV6 y además soporta también SCP y SFTP.

SSH secure shell: Es la implementación del protocolo SSH para la plataforma Windows, desarrollada por SSH Communications security. Es ampliamente utilizada para la administración de la seguridad de un sistema, transferencia segura de archivos y conectividad segura de aplicaciones, su soporte para el protocolo IPv6 no se encuentra ampliamente explorado.

7.8 INSTALACIÓN Y CONFIGURACIÓN DEL SERVIDOR SSH SELECCIONADO

Teniendo en cuenta que el nodo IPV6 de la Universidad del Cauca en el cual estará alojado el servidor SSH, cuenta con sistema operativo Linux Fedora Core 4, se seleccionó OPENSSH para realizar las respectivas pruebas. En lo referente a clientes SSH, se exploraron diferentes alternativas, tanto para Windows como para Linux.

7.8.1 Instalación

A. El paquete utilizado es openssh-server-4.0p1-3. En la dirección http://rpm.pbone.net se busca el paquete openssh con extensión rpm para fedora 4 y se descarga. Posteriormente ejecutamos el comando rpm –i openssh-server-4.0p1-3.rpm para instalar el paquete en el PC.

B. Con los CDs de instalación de Fedora Core 4, también es posible instalar el servidor SSH.

C. Una vez instalado el servidor SSH, se comprueba que en la ruta */etc/ssh/* se encuentre el archivo de configuración para este servicio sshd_config.

D. Los comandos utilizados para iniciar y para el servidor son */etc/init.d/sshd Start* y */etc/init.d/sshd stop* respectivamente.

7.8.2 Configuración

La estructura del archivo de configuración sshd_config, se puede apreciar en la figura 25. Éste archivo contiene dos parámetros muy importantes en lo que al Protocolo Internet se refiere, **ListenAddress 0.0.0.0 y ListenAddress :: ,** si se activa el primero, es decir se quita el comentario #, SSH solo escuchará peticiones realizadas con direcciones IPv4, las realizadas con IPv6 las desechará, si por el contrario está habilitada la directiva ListenAddres :: se escucharán peticiones con direcciones IPv6.

```
Port 22
#Protocol 2,1
Protocol 2
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile     .ssh/authorized_keys
```

Figura 25. Archivo de configuración SSH

Teniendo en cuenta que muchas redes en el mundo tienen únicamente soporte para el protocolo IPv4 y otras cuentan con servidores *Dual-stack*, se hace necesario que el servidor SSH para el nodo IPv6 de la Universidad sea *Dual-stack*, logrando interconexión con otros nodos IPv4 e IPV6. Para lograr esto, en el archivo de configuración sshd_config se deben dejar comentados los parámetros ListenAddres 0.0.0.0 y :: , para que acepte conexiones de los dos tipos.

7.8.3 Pruebas

Para probar el funcionamiento de este servidor, se realizaron pruebas con clientes SSH Windows y Linux. Se encontró que los clientes Windows SSH con soporte IPv6 son PuTTY y AXESSH, para Linux el más popular es el proyecto OPENSSH.

7.8.3.1 Cliente PuTTY

A partir de la versión 0.58, este cliente SSH soporta IPv6, gracias al proyecto UNFIX.ORG, quien desarrolló el parche para habilitar este soporte y posteriormente lo integró con el código original de PUTTY. El archivo ejecutable se puede descargar de las siguientes direcciones:
http://unfix.org/projects/IPv6/
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Al descargar PuTTy.exe solo se debe hacer doble clic sobre él y se debe desplegar una ventana igual a la de la Figura 26.

Una vez el servidor SSH se encuentre corriendo, se podrían realizar loa accesos con éste cliente. La Figura 27 ilustran una de las pruebas, en la cual se desea conectarse remotamente a la dirección 2001:448:1024:1::10 en la cual se encuentra alojado el servidor SSH. En la figura 28 se observa el uso del cliente PuTTy para acceder a un equipo cuyo nombre de máquina se encuentra almacenado en el servidor de nombres con un registro AAAA, en este caso el equipo zeus.IPv6.unicauca.edu.co.

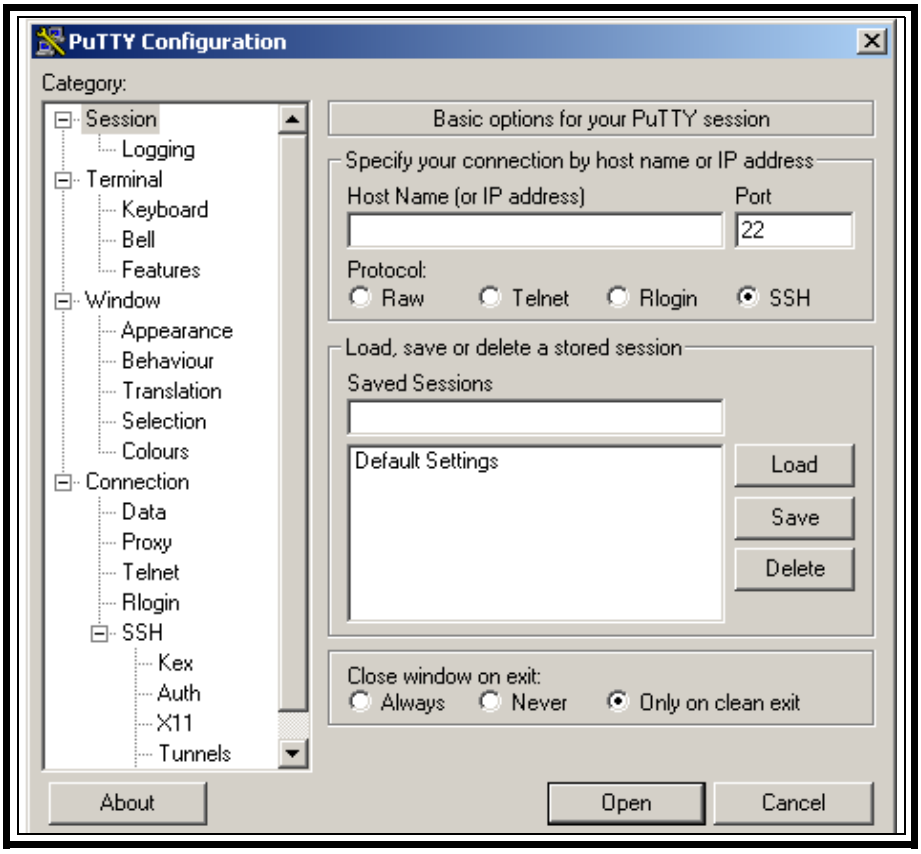


Figura 26. Cliente PuTTY

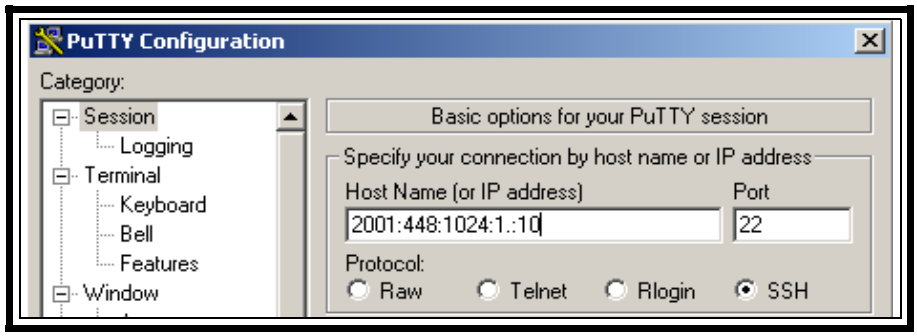


Figura 27. Prueba Cliente PuTTY con dirección IPv6

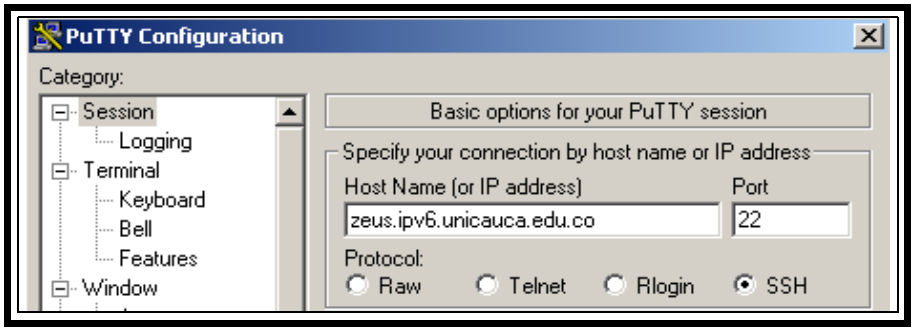


Figura 28. Prueba Cliente PuTTy con nombre de máquina

7.8.3.2 CLIENTE AXESSH

Este software no es libre, tiene licencia shareware y la versión de prueba para este cliente se puede descargar de http://www.softpedia.com/progDownload/Axessh-

Windows-SSH-Client-Download-23408.html. Se debe descargar la versión más reciente, ya que ésta es la que contiene soporte IPv6, en este caso la versión 3.3. Su instalación se realiza de igual forma que un paquete de Windows. Se realizaron pruebas de interconectividad con direcciones IPv6 y con nombres de máquinas, las pruebas fueron satisfactorias. Uno de los resultados se ilustra en la Figura 29.

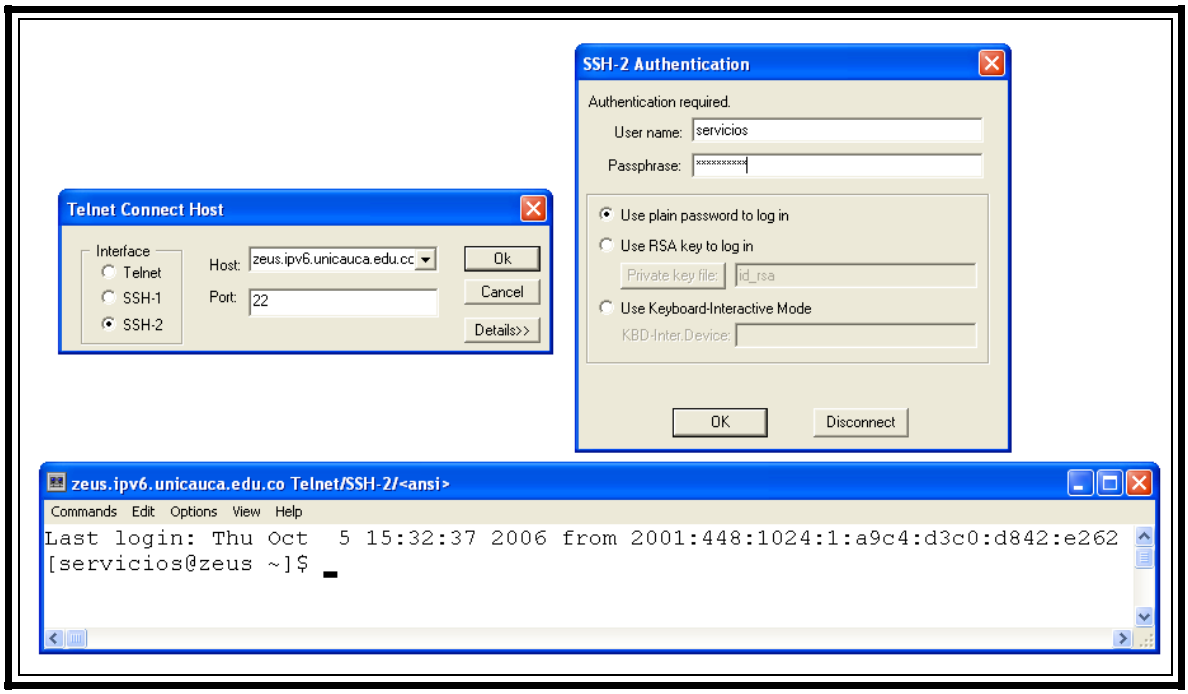


Figura 29. Prueba Cliente AXESSH con nombre de máquina

7.8.3.3 CLIENTE OPENSSH

El cliente OPENSSH con soporte IPv6 para realizar las pruebas fue openssh-clients-40p1-3.rpm, descargado de la misma dirección de descarga del servidor. Para iniciar este cliente y forzarlo a utilizar el protocolo IPv6 se ejecuta el comando:

ssh -6 -l usuario nombre de *host* ó ssh -6 -l usuario dirección del *host*

La correcta interacción de éste cliente con direcciones IPv6 se aprecia en la Figura 30. Es posible especificar una dirección IPv6 o un nombre de máquina.

```
[root@zeus ~]# ssh -6 -l servicios 2001:448:1024:1::10
servicios@2001:448:1024:1::10's password:
Last login: Thu Oct  5 15:22:52 2006 from zeus.ipv6.unicauca.edu.co
[servicios@zeus ~]$
```

Figura 30. Cliente OpenSSH

8. APLICACION WEB PARA LA GESTION DE AUTOCONFIGURACION IPV6

Las aplicaciones Web ocupan hoy día un lugar de suma importancia en la Internet, dada la facilidad con la cual son alcanzadas, ya que en general para acceder a ellas

solo es necesario un cliente Web y una conexión a Internet, por tanto cada vez se cuenta con mas software orientado al servicio Web, el cual es utilizado masivamente por millones de usuarios, como es el caso de los Webmails mas populares o los servicios de mensajería instantánea, o también existen aplicaciones personalizadas que pueden dar solución a problemas puntuales, como la aplicación "Interfaz Web para la autoconfiguración IPv6" implementada para el nodo IPv6 de la Universidad del Cauca.

Para el proyecto "Servicios Internet IPv6 para la Universidad del Cauca en el 6bone", fueron identificados los siguientes requisitos para las posibles aplicaciones Web a implementar.

- El software debe estar escrito plenamente en PHP. Eliminando así la necesidad de nueva infraestructura, ya que de esta manera todas las aplicaciones se alojarán en el servidor Apache. Lo anterior teniendo en cuenta que todos los servicios y aplicaciones de enrutamiento están alojadas en un único equipo.

- Se debe procurar el uso de nombres de máquina por encima de direcciones IP, lo anterior resulta particularmente útil teniendo en cuenta la etapa floreciente del proyecto IPv6 en la Universidad del Cauca, ya que al migrar estas aplicaciones, como puede suceder al incrementarse la demanda de la red, resultará menos traumático dado cualquier cambio inesperado, ya que solo será necesaria una reconfiguración en el servidor DNS.

8.1 MODELADO DE LA APLICACIÓN INTERFAZ WEB PARA LA AUTOCONFIGURACIÓN IPV6

8.1.1 Modelado de la Organización

La nueva estructura de las direcciones en el protocolo IPv6 que entre sus particularidades destaca una longitud de 128 bits, aporta nuevas características que ofrecen soluciones a problemas que actualmente registra el protocolo IPv4, sin embargo esto conlleva a que la configuración de los parámetros de red en un *host* sea más compleja.

Teniendo en cuenta lo anteriormente expuesto, la autoconfiguración se convierte en un ítem fundamental para la masificación de IPv6 en una red basada en éste protocolo, ya que mediante este mecanismo se libera al usuario final no especializado, de la tarea de configurar los parámetros mencionados. De esta manera la responsabilidad del buen funcionamiento de la red se centraliza en la gestión realizada por los administradores de red, quienes deben hacer buen uso de las herramientas que este nuevo protocolo les provee (como la autoconfiguración stateless y stateful), para lograr este cometido.

Como ya se especifico en el capitulo 3 (Autoconfiguración IPv6), existen tres tipos de autoconfiguración: con control de estado, sin control de estado y la combinación de las dos anteriores, cada una de estas implica procesos de configuración y administración diferentes. Dependiendo de las políticas con las cuales se maneja una red estos tipos de configuraciones pueden cambiar con el paso del tiempo. La conjunción de estos elementos puede provocar un periodo de inestabilidad para la red en la transición de un método de autoconfiguración a otro; o en el caso que la opción elegida sea la

combinación stateless y stateful, puede inducir errores humanos por cuenta del manejo de múltiples herramientas al tiempo. En la figura 31 se ilustra el diagrama de casos de uso inicial de la organización, en el cual se especifican las funciones que un administrador de red debe realizar para seleccionar un tipo de autoconfiguración.

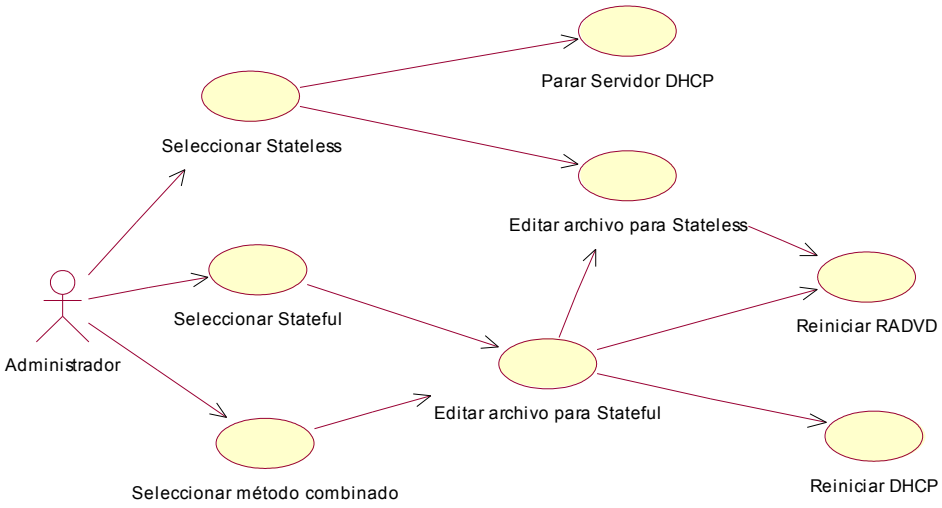


Figura 31. Diagrama de casos de uso inicial

8.1.2 Captura de requisitos.

Como respuesta al problema planteado se requiere la implementación de una aplicación que sirva de apoyo a los administradores de red en la gestión de los diferentes métodos de autoconfiguración. La aplicación debe cumplir con los siguientes lineamientos:

- Debe contar con un control de acceso que verifique la identidad del administrador de red encargado.
- El sistema ofrecerá la posibilidad de elegir la autoconfiguración con la cual trabajara la red.
- Dependiendo de la elección de la administración en cuanto al tipo de autoconfiguración, el sistema manipulara debidamente los archivos de configuración adecuados para poner en marcha el servicio elegido.
- El sistema debe presentar información relevante para la correcta administración del tipo de autoconfiguración elegida.
- El administrador debe tener la capacidad de manipular los parámetros de configuración mediante una interfaz grafica Web.
- El sistema debe contar con una gestión de errores, que alerte ante comportamientos no deseados o gramática mal utilizada en los archivos de configuración.
- El sistema debe garantizar la privacidad de los datos (contraseñas, comandos) manejados en las diferentes operaciones de la aplicación.

8.1.3 Casos de Uso de Alto Nivel

La figura 32. Muestra los diagramas de casos de uso de más alto nivel mediante los cuales se hace una descripción inicial de las interacciones y funcionalidades que el sistema debe ofrecer.

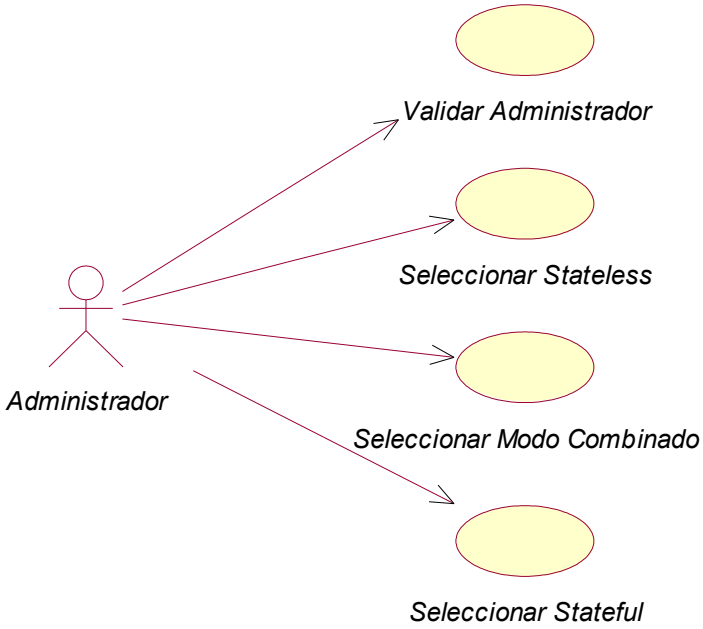


Figura 32. Diagrama de casos de uso de alto nivel.

Descripción Inicial de los escenarios de los Casos de Uso.

Caso de uso: Validar Administrador.
Actor: Administrador**.**
Tipo: Primario.
Descripción: El caso de uso empieza cuando el administrador desea ingresar al sistema, este solicita al cliente su login y password, y verifica su identidad como un usuario valido.

Caso de uso: Seleccionar Stateless.
Actor: Administrador**.**
Tipo: Primario.
Descripción: El caso de uso empieza cuando el administrador decide utilizar el tipo de autoconfiguración sin control de estado, el sistema verifica que el archivo de configuración radvd.conf se encuentre apropiadamente acondicionado, y le devuelve al usuario un resumen con las propiedades del servicio.

Caso de uso: Seleccionar Stateful.
Actor: Administrador**.**
Tipo: Primario.
Descripción: El caso de uso empieza cuando el administrador decide utilizar el tipo de autoconfiguración con control de estado, el sistema verifica que los archivos de configuración radvd.conf y dhcp6s.conf se encuentren apropiadamente acondicionados, y le devuelve al usuario un resumen con las propiedades del servicio.

Caso de uso: Seleccionar modo combinado.
Actor: Administrador**.**
Tipo: Primario.

Descripción: El caso de uso empieza cuando el administrador decide utilizar el tipo de autoconfiguración sin control de estado para entregar direcciones IPv6 y autoconfiguración con control de estado para anunciar parámetros adicionales de configuración.

8.1.4 Casos de uso extendidos.

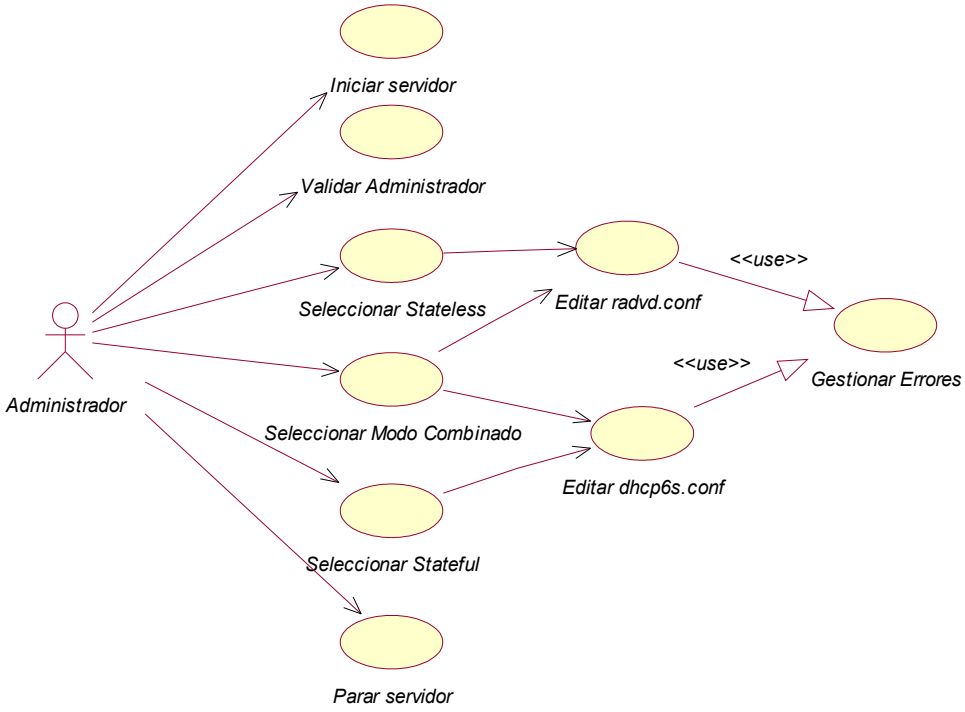


Figura 33. Diagrama de casos de uso extendido.

Descripción extendida de los casos de uso.

Caso de uso: Iniciar Servidor.
Actor: Administrador**.**
Tipo: Primario.
Descripción: Este caso de usa se inicia cuando el administrador decide correr los demonios asociados a un tipo específico de autoconfiguración, o cuando se presenta una actualización en los archivos de configuración. El sistema envía los comandos necesarios para subir los servidores, y si se ha presentado algún error durante éste proceso, desplegará ante el administrador un mensaje indicando cual fue la razón por la cual no se pudo iniciar el servicio.
Precondiciones: El sistema debe contar con los datos necesarios para establecer cuál servidor es el que debe iniciar.

Caso de Uso: Validar Administrador
Actor: Administrador.
Tipo: Primario.

Descripción: El caso de uso empieza cuando el administrador desea ingresar al sistema, este solicita al cliente su login y password, y verifica su identidad como un usuario valido. Si el proceso de validación es exitoso, el sistema le desplegará al administrador opciones mas avanzadas, de lo contrario no se tendrá acceso al sistema.
Precondiciones: El sistema debe contar con la información apropiada para realizar la validación del administrador (login y password).

Caso de Uso: Seleccionar Stateless
Actor: Administrador.
Tipo: Primario.
Descripción: El caso de uso empieza cuando el administrador decide utilizar el tipo de autoconfiguración sin control de estado, el sistema verifica que el archivo de configuración radvd.conf se encuentre apropiadamente acondicionado, envía comandos para iniciar el servicio, y le devuelve al usuario un resumen con las propiedades de éste.
Precondiciones: El sistema debe tener acceso al archivo de configuración del demonio Radvd y éste a su vez no debe presentar errores gramaticales.

Caso de Uso: Seleccionar Modo Combinado
Actor: Administrador.
Tipo: Primario
Descripción: El caso de uso empieza cuando el administrador decide utilizar el tipo de autoconfiguración sin control de estado para entregar direcciones IPv6 y autoconfiguración con control de estado para anunciar parámetros adicionales de configuración, el sistema verifica que los archivos de configuración radvd.conf y dhcp6s.conf se encuentren apropiadamente acondicionados, y le devuelve al usuario un resumen con las propiedades del servicio.
Precondiciones: El sistema debe tener acceso a los archivos de configuración del demonio Radvd y del servidor DHCPv6.

Caso de uso: Seleccionar Stateful.
Actor: Administrador**.**
Tipo: Primario.
Descripción: El caso de uso empieza cuando el administrador decide utilizar el tipo de autoconfiguración con control de estado; el sistema verifica que los archivos de configuración radvd.conf y dhcp6s.conf se encuentren apropiadamente acondicionados, y le devuelve al usuario un resumen con las propiedades del servicio.
Precondiciones: El sistema debe tener acceso a los archivos de configuración del demonio Radvd para efectos de enrutamiento y del servidor DHCPv6 para configuración de parámetros.

Caso de uso: Parar Servidor.
Actor: Administrador**.**
Tipo: Primario.
Descripción: Este caso de usa se inicia cuando el administrador decide detener los demonios asociados a un tipo específico de autoconfiguración. El sistema envía los comandos necesarios para parar los servidores.
Precondiciones: El sistema debe contar con los datos necesarios para establecer cuál servidor es el que debe detener.

Caso de uso: Editar radvd.conf.
Actor: Administrador**.**
Tipo: Primario.

Descripción: Este caso de uso se inicia cuando el Administrador decide modificar, eliminar o adherir parámetros de configuración para el servicio de autoconfiguración stateless o combinado. Utiliza el caso de uso gestionar errores, para evaluar los datos ingresados por el usuario.
Precondiciones: El sistema debe contar con los datos que el administrador desea modificar.

Caso de uso: Editar dhcp6s.conf.
Actor: Administrador**.**
Tipo: Primario.
Descripción: Este caso de uso se inicia cuando el Administrador decide modificar, eliminar o adherir parámetros de configuración para el servicio de autoconfiguración stateful o combinado. Utiliza el caso de uso gestionar errores, para evaluar los datos ingresados por el usuario.
Precondiciones: El sistema debe contar con los datos que el administrador desea modificar.

Caso de uso: Gestionar Errores.
Actor: Administrador**.**
Tipo: Primario.
Descripción: Este caso de uso se inicia cuando el Administrador realiza cambios en un archivo de configuración o inicia un servicio específico, el sistema una vez realiza la edición de los archivos, verifica que la gramática utilizada sea la apropiada.
Precondiciones: El sistema debe tener acceso a los archivos en los cuales se almacenan los logs del sistema.

8.1.5 Diagrama de Paquetes del Sistema

El sistema se divide en 6 subsistemas, agrupando la relación de elementos:

- Interfaz de Usuario
- Control
- Acceso a Datos
- Librerías de Desarrollo
- Funciones de Gestión de Configuración
- Conexión SSH.

El subsistema Interfaz de Usuario contiene todos los elementos relacionados con las vistas del sistema, a través de las cuales el usuario puede interactuar con la aplicación; el Control contiene todas las clases y componentes que le permiten a la aplicación llevar una secuencia lógica de los procesos a realizar; el Acceso a Datos asocia las funciones que permiten la comunicación con la Base de Datos para realizar la autenticación del administrador y con los archivos de configuración a manipular; el subsistema Librerías de Desarrollo asocia todas las clases relacionadas con la conexión SSH y la gestión de los archivos de configuración. En la figura 34 se ilustra el diagrama de paquetes.

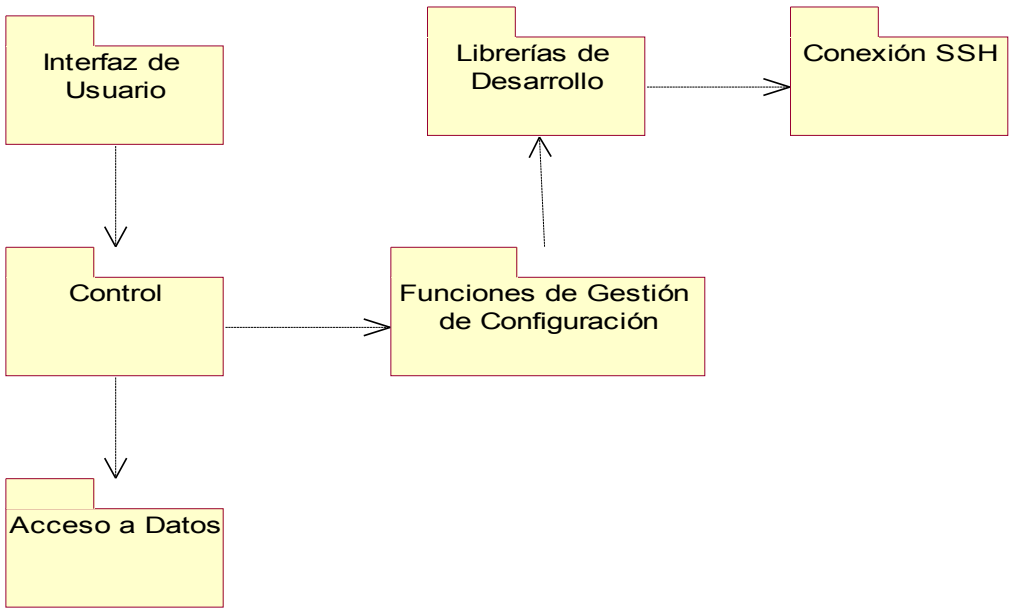


Figura 34. Diagrama de Paquetes

8.1.6 Diagrama de Clases

En la figura 35 se aprecia el diagrama de clases del sistema con sus correspondientes atributos y funciones, a continuación se realiza una breve descripción de cada una de las clases que componen el sistema:

Clase SSH: Su función es ejecutar comandos Linux en el Servidor de autoconfiguración, lo anterior mediante la realización de una conexión SSH, en la cual el sistema Web se valida como un usuario root, frente al Servidor remoto.

Clase Editar: Esta clase tiene por responsabilidad realizar cambios en archivos de texto, entre sus funcionalidades se cuenta: Buscar frases, editar líneas, editar palabras, borrar , adicionar y editar líneas.

Clase Imprimir: Esta clase es la encargada de mostrar las respuestas que el sistema genera frente a peticiones del administrador al sistema Web.

Clase GestionarErrores: Su función es revisar los logs del sistema, cada vez que la clase SSH inicia un servicio buscando fallos introducidos por la manipulación de los archivos de configuración, para luego reportarlos al sistema.

Clase Realizar Cambios: Esta clase coordina las funciones de Clase SSH, clase Editar, y clase Imprimir, para materializar los cambios que el administrador desea introducir en un archivo de configuración en particular.

Clase GestionarEntarda: La función de esta clase es editar apropiadamente los archivos de configuración al iniciar un tipo en particular de autoconfiguración.

Clase Control: Esta clase maneja la navegación del sistema, dependiendo de las entradas del usuario Control redirige el navegador hacia las diferentes instancias de la aplicación.

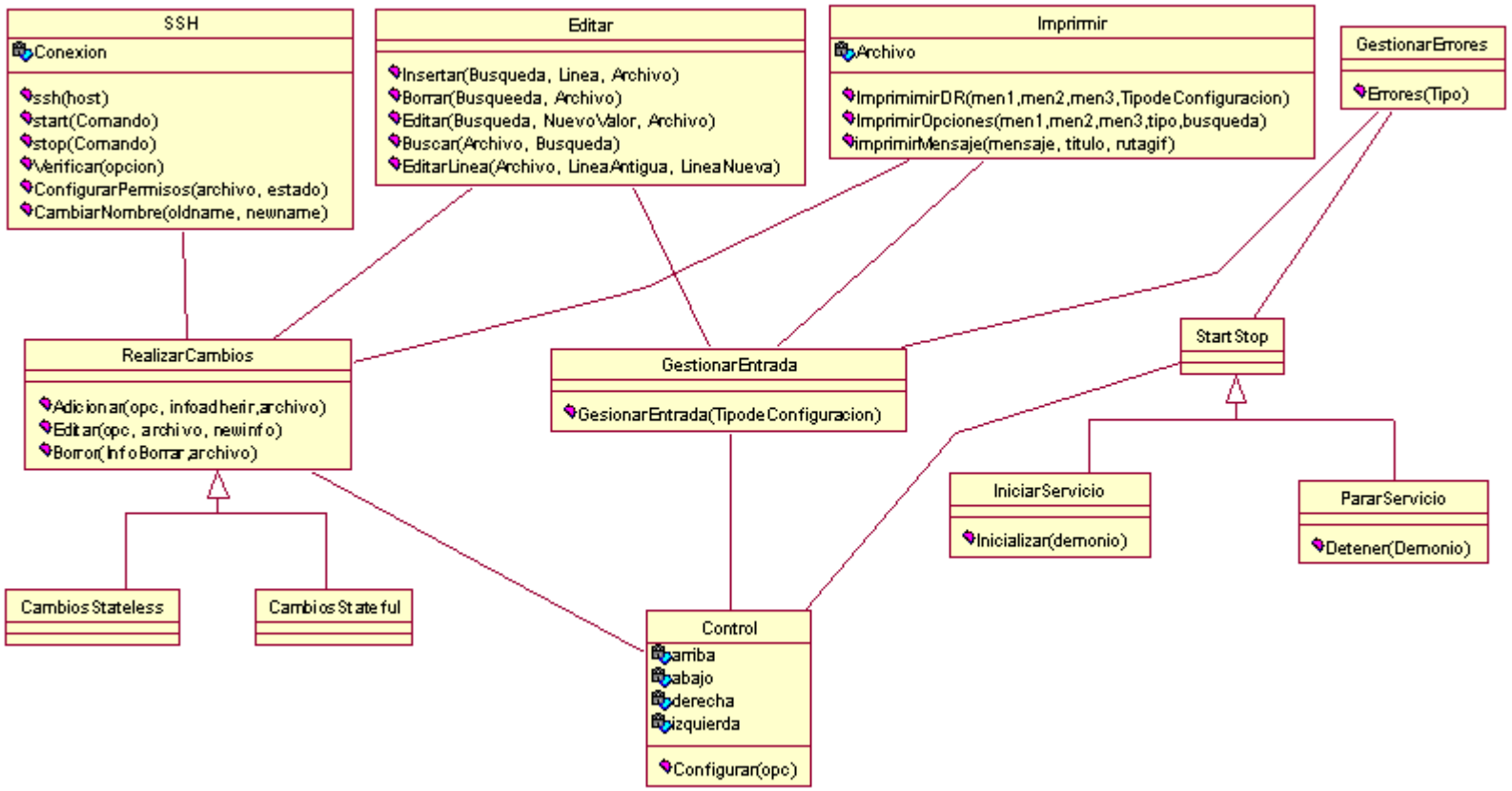


Figura 35. Diagrama de Clases

8.1.7 Diagramas de secuencia para los casos de Uso Esenciales del Sistema.

Casos de uso iniciados por el Administrador:

Las figuras 36, 37, 38 y 39 ilustran los diagramas de secuencias para los casos de uso Validar Administrador, Seleccionar stateless, seleccionar stateful y Seleccionar modo combinado respectivamente.

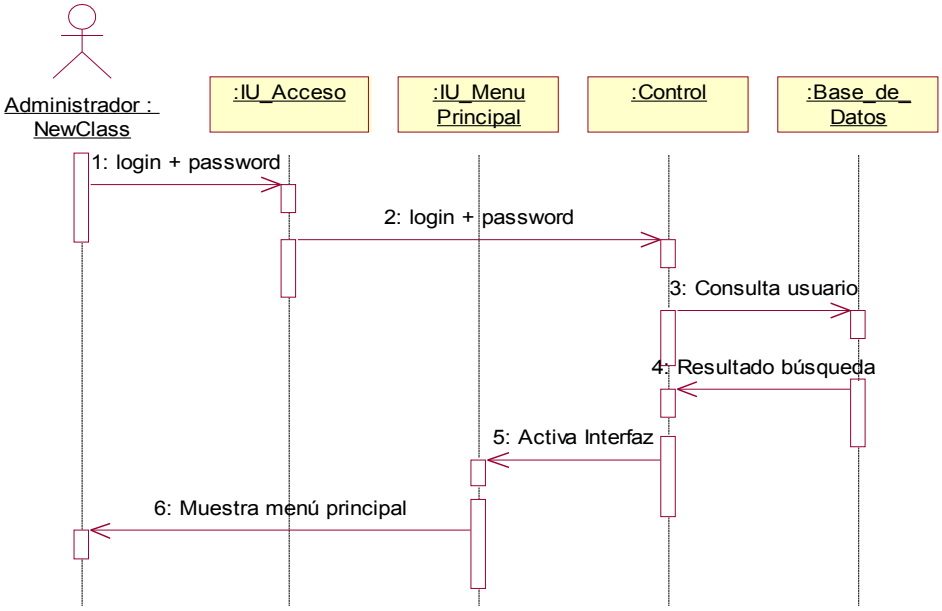


Figura 36. Diagrama de secuencias validar administrador.

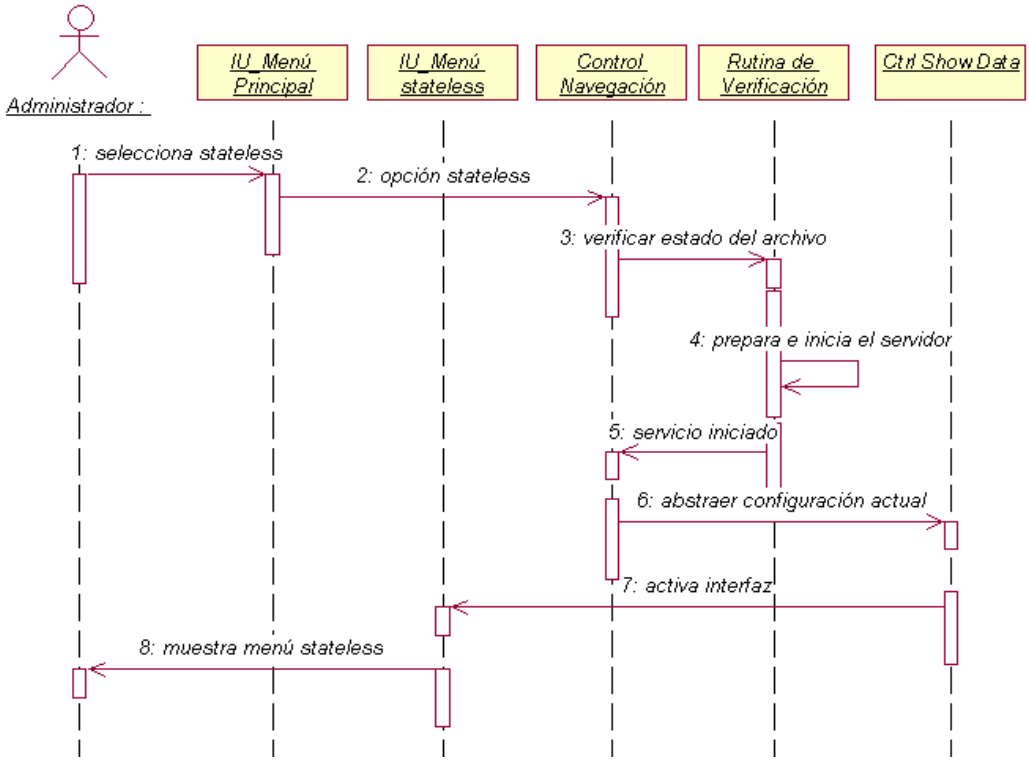


Figura 37. Diagrama de secuencias seleccionar stateless

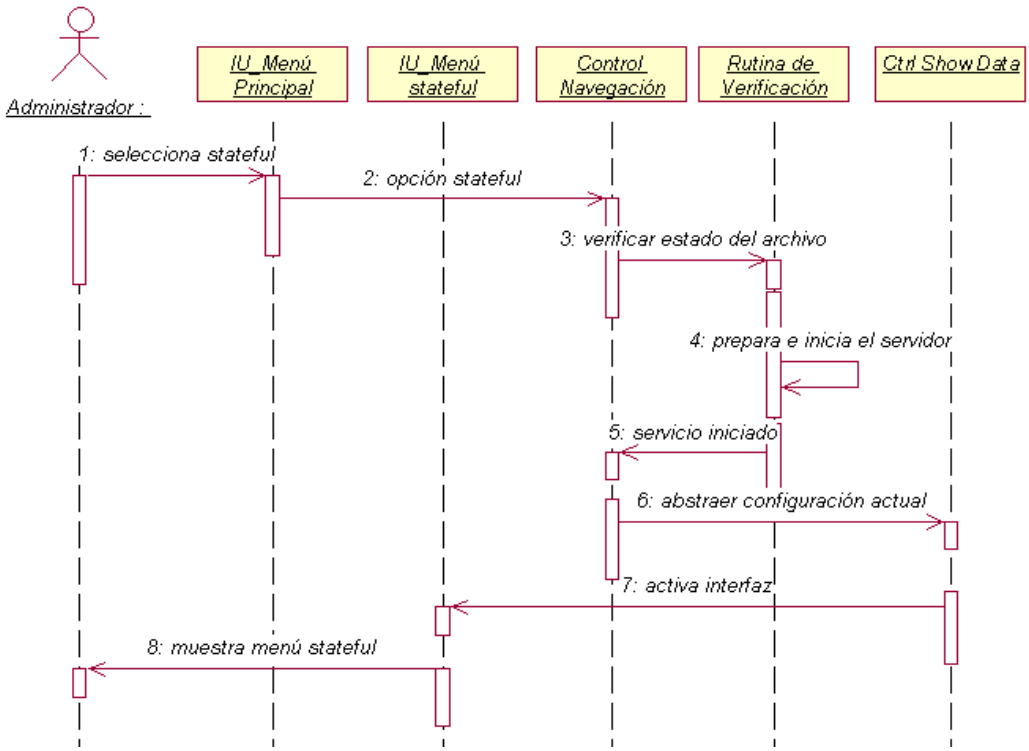


Figura 38. Diagrama de secuencias seleccionar stateful

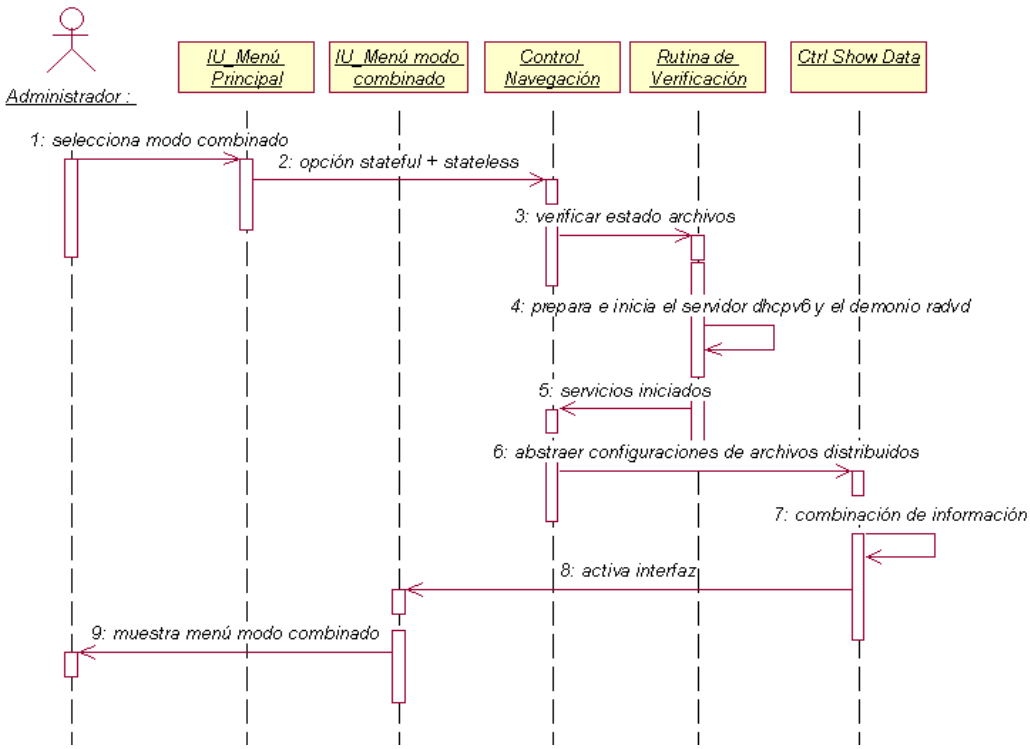


Figura 39. Diagrama de secuencias seleccionar modo combinado

8.1.8 Diagramas de Estados

En las figuras 40, 41, 42 y 43, se pueden apreciar los diagramas de estado para las clases GestionarEntradas, IniciarServicio, PararServicio y GestionarErrores, debido a su naturaleza dinámica.

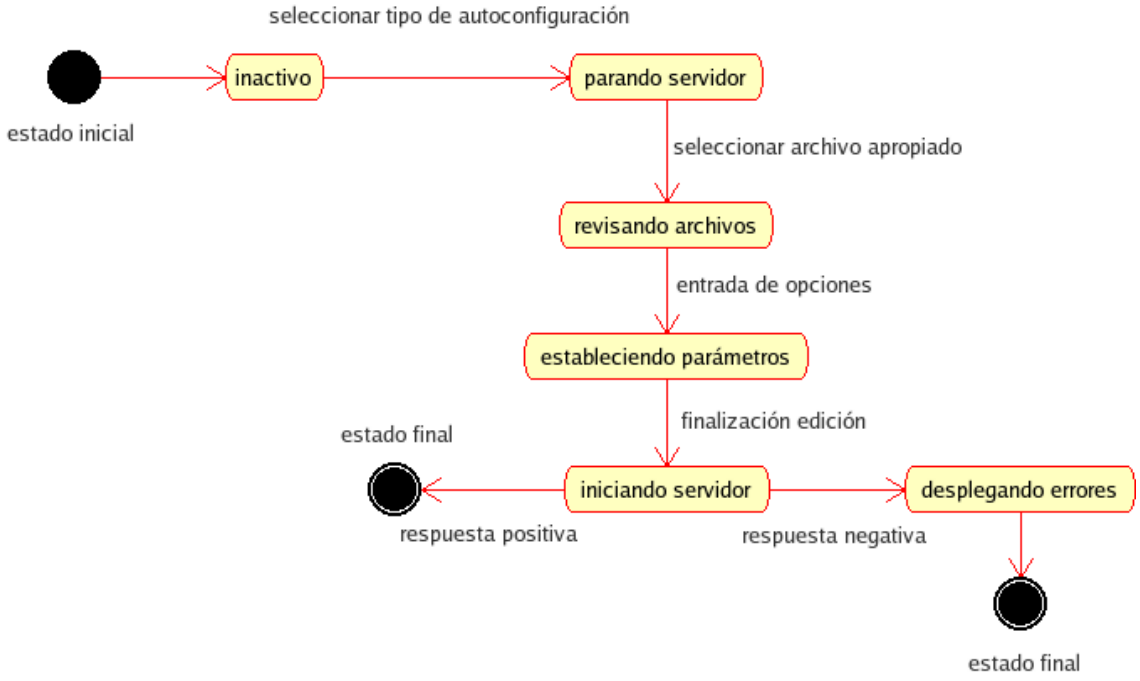


Figura 40. Diagrama de estados GestionarEntradas

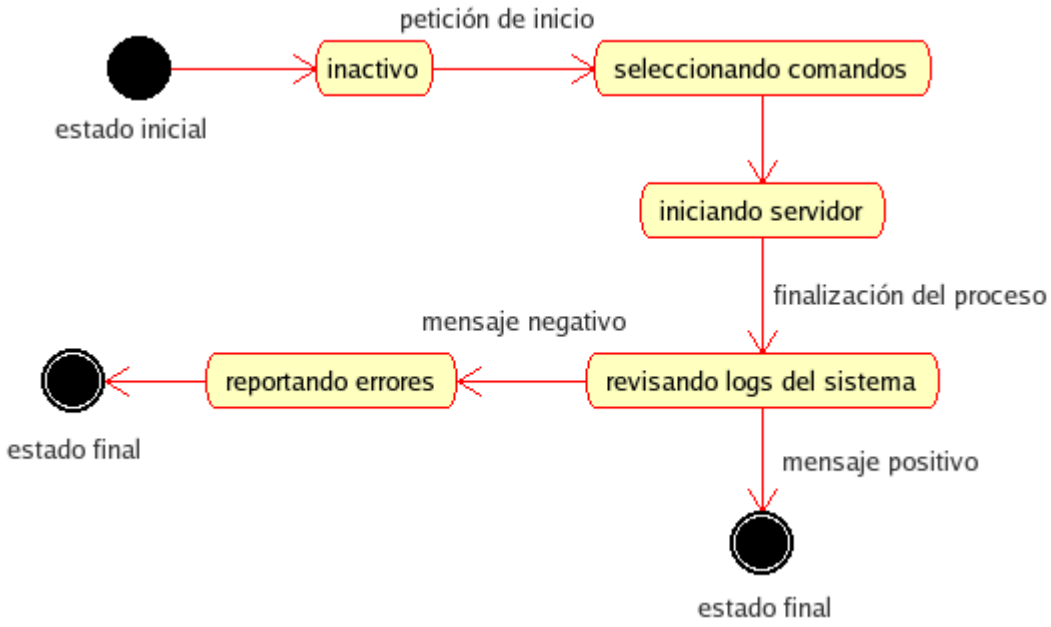


Figura 41. Diagrama de estados IniciarServicio

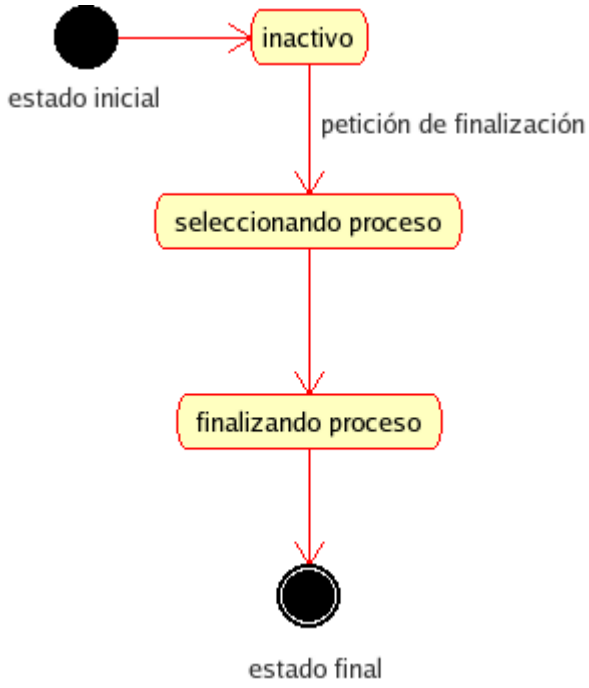


Figura 42. Diagrama de estados ParарServicio

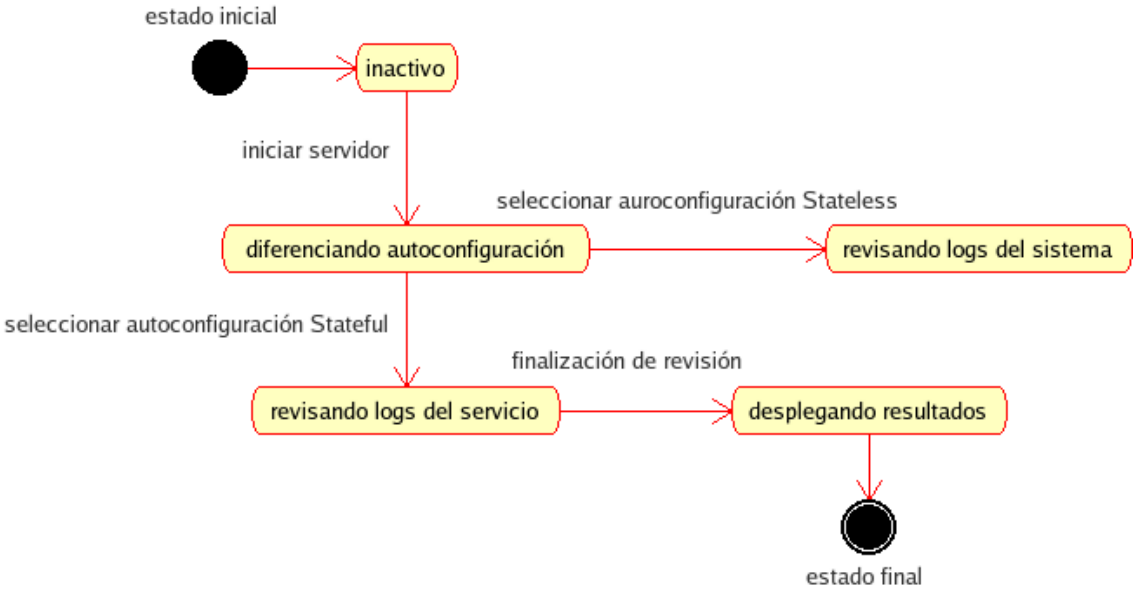


Figura 43. Diagrama de estados GestionarErrores

8.1.9 Arquitectura del Sistema

En la figura 44 Se presenta la arquitectura del sistema. En ella se puede apreciar como el administrador se vale de las páginas cliente para interactuar con el servidor Web, a través del protocolo seguro HTTPS. Las páginas clientes ejecutan código Javascript para garantizar la validez de la información introducida por el administrador antes de ser enviada al servidor, de esta manera, la interacción de dicho usuario con

los formularios que el sistema presenta a éste producirán datos de entrada al sistema con una baja tasa de errores. Por otro lado el servidor Apache alberga módulos PHP que realizan funciones específicas:

- El módulo de validación hace una conexión con una base de datos MySQL para garantizar la identidad del súper usuario del sistema.

- El módulo SSH mantiene un canal seguro SSH con el servidor DHCP y RADVD, este canal permite enviar órdenes a los servidores como parar e iniciar, además solicita información de que servicios están funcionando y cuales no y cambiar permisos de escritura sobre los archivos de configuración al momento de editarlos y al final de éste proceso.

  Para que la contraseña de root no sea revelada por el código, el sistema Web se identifica ante los servidores de autoconfiguracion con una llave privada, previa configuración de su llave publica en dicho host remoto, es decir, utiliza el mecanismo de autenticación por clave pública, la empliación de éste tema se encuentra en el capítulo 7 sección 7.3.

- El modulo gestión de errores, se encarga de revisar los logs del sistema operativo, para mantener al sistema enterado de eventualidades que sean reportadas por los demonios Radvd o DHCP.

- El módulo de edición actúa cuando los permisos de escritura de un archivo de configuración son cambiados a un estado permisivo, permitiendo operaciones de eliminación, adición y edición de palabras o frases dentro del documento; una vez estos procesos concluyen los archivos son configurados con unos permisos de escritura restrictivos.

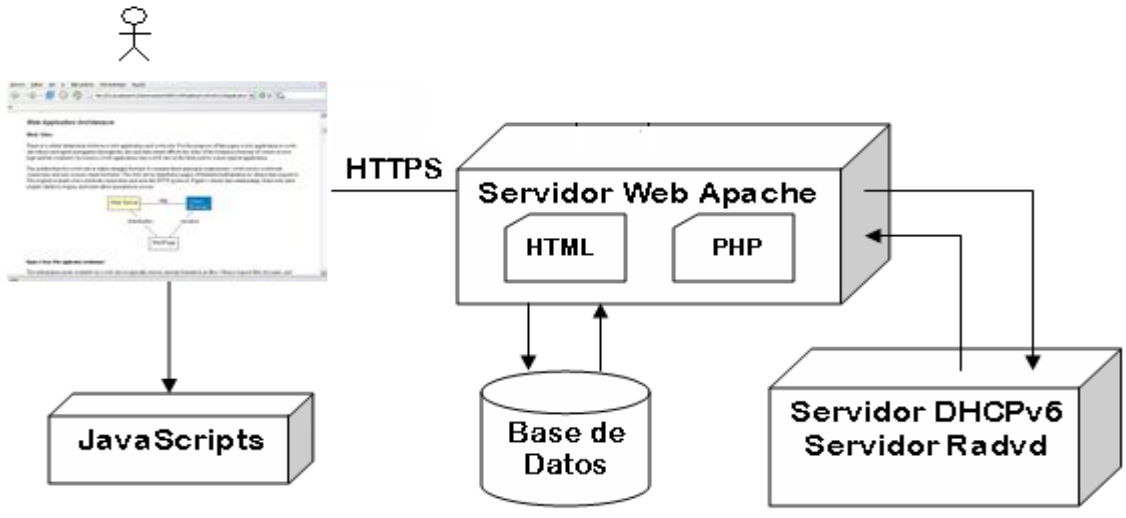


Figura 44. Arquitectura del sistema.

La arquitectura modular de éste sistema está orientada en la gestión genérica de control de errores del sistema, la ejecución de cualquier tipo de comando y la edición de archivos de texto; razón por la cual puede ser adaptada para la gestión de configuración de otros servicios cuyo comportamiento se base en archivos de

configuración mediante la adaptación de las funciones que implementan las librerías y la configuración de las variables globales.

8.1.10 Vistas del Sistema

En la figura 45 se puede apreciar la interfaz gráfica que permite hacer la validación del usuario; la figura 46 ilustra un menú donde el usuario puede seleccionar el tipo de autoconfiguración a aplicar al sistema, la figura 47, 48 y 49 ilustran el menú y la vista principal de los servicios autoconfiguración con control de estado, sin control de estado y la combinación de los dos respectivamente.

Figura 45. Vista validar

Figura 46. Vista menú principal

Figura 47. Vista autoconfiguración con control de estado

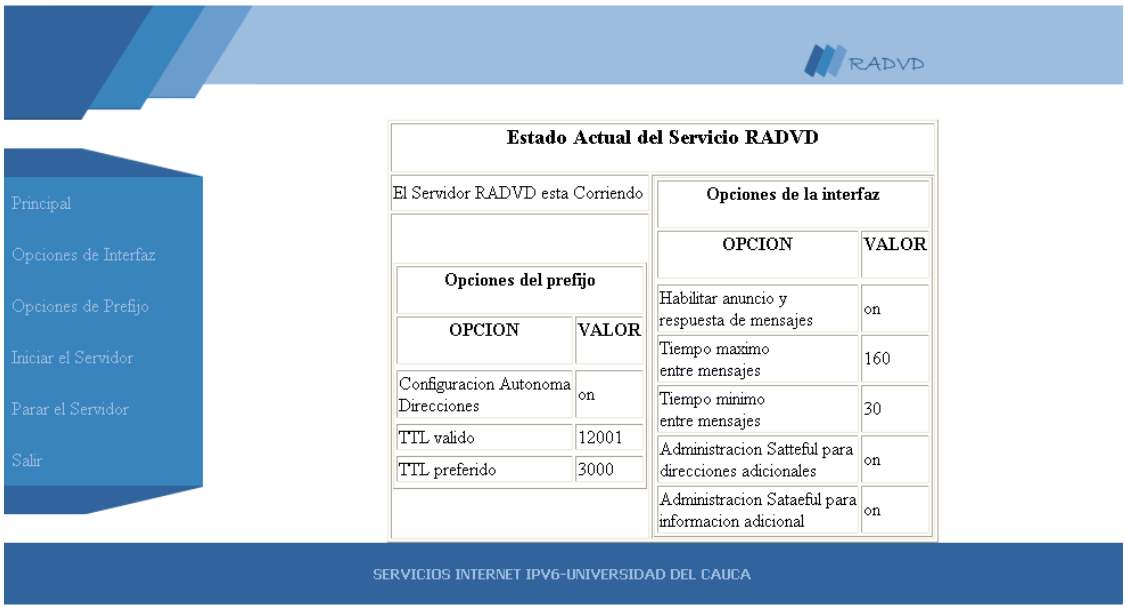


Figura 48. Vista autoconfiguración sin control de estado

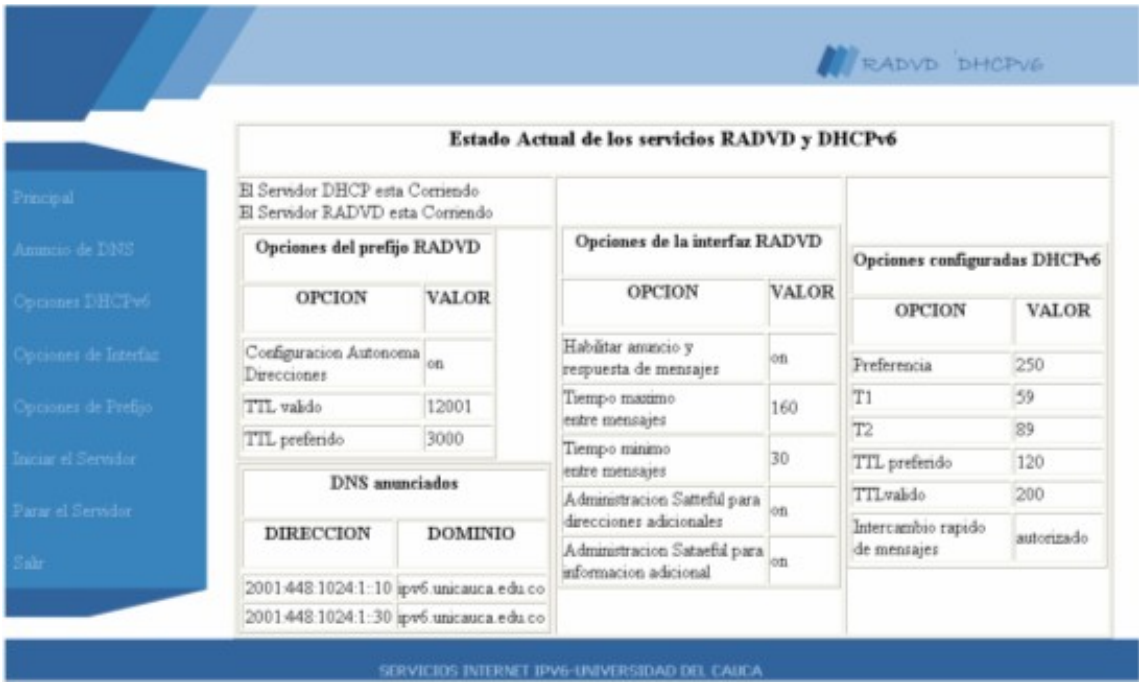


Figura 49. Vista autoconfiguración Híbrida

CONCLUSIONES Y RECOMENDACIONES

Los objetivos planteados en el anteproyecto de este trabajo de grado fueron cabalmente cumplidos, la Universidad del Cauca además de contar con conectividad a nivel IPv6, tiene implementados los servidores necesarios para brindar los servicios básicos Ipv6: DNS, Autoconfiguración (*Stateless, Stateful* y combinación *Stateless – Stateful*), Correo electrónico, FTP, Acceso remoto y servidor HTTP. Adicionalmente se implantaron dos aplicaciones Web que se integraron a los servicios configurados, para brindar funcionalidades extras a los usuarios, Squirrelmail y Mambo. Por último ante la imperiosa necesidad de administrar de forma adecuada la autoconfiguración de direcciones IPv6 dentro de la red, se desarrolló un aplicativo Web que interactúa con los servidores encargados de ofrecer estas funcionalidades, logrando que el proceso de configuración de parámetros sea intuitivo y rápido a través de la Web. La unión de todos los componentes mencionados anteriormente, integran un sistema de administración para la red experimental IPv6 en la Universidad del Cauca.

Conclusiones respecto a los servicios:

- Los registros AAAA se constituyen en la mejor alternativa al momento de implantar un DNS IPv6, debido a la similitud que en formato y funcionamiento tienen con los registros A, abonando así la ventaja de experiencia operativa que con éstos últimos se ha obtenido desde la definición del Sistema de Nombres de Dominio. Por otro lado cabe destacar las características adicionales que ofrecen los registros A6 y DNAME en un entorno en el cual una re-numeración pueda significar un proceso laborioso dada la cantidad de hosts y servidores que formen parte de la red, o en organizaciones que trabajen con topologías multi-proveedor, donde un cambio en un DNS de alto nivel dentro de la jerarquía, pueda significar cambios traumáticos en los niveles más bajos de ésta. El estado de estos registros es actualmente experimental, sin embargo, vale la pena seguir estudiándolos para abarcar todo el potencial que éstos pueden llegar a brindar al desarrollar mecanismos que permitan enlazar las diferentes respuestas provenientes de registros A6.

- Durante el periodo de transición Ipv4 a Ipv6, es importante contar con servidores de nombres de dominio Dual-Stack, para garantizar la conectividad entre escenarios de redes heterogéneas Ipv4–IPv6, La tendencia Dual-Stack se observó a lo largo de todo el proyecto, al verificar esta condición en un sin número de servidores DNS externos.

- El buen funcionamiento de todos los Servicios Internet, dependen de una óptima configuración del Servidor de Nombres de Dominio.

- La autoconfiguración es una característica fundamental en el protocolo IPv6, y su implantación resulta definitiva para el desarrollo de esta tecnología, existen múltiples metodologías para asignar parámetros de autoconfiguración a un nodo cliente (*Stateful, Staeless*, combinación), particularmente para la red de la Universidad del Cauca, se debe realizar un estudio por sectores, para definir las necesidades propias de cada uno de ellos y seleccionar así el método de autoconfiguración más apropiado.

- La autoconfiguración libera al usuario de la tarea de configurar los parámetros de la red para obtener conectividad y minimiza los posibles errores de direcciones duplicadas al utilizar un algoritmo creado para tal fin.

- Utilizando la autoconfiguración sin control de estado, es posible que cualquier dispositivo que se conecte a la red tenga una dirección IPv6 válida, lo cual podría disminuir el nivel de seguridad en una red, dado el bajo nivel de control en la asignación de las direcciones a usuarios anónimos.

- El correo electrónico funciona óptimamamente en un ambiente dual-stack, siempre y cuando se sigan los lineamientos propuestos en este documento. Un servidor de correo Dual-Stack esta en capacidad de enviar y recibir desde y hacia MTA IPv6 puros y MTA IPV4.

- Los nuevos comandos adheridos a las especificaciones FTP para dar soporte al protocolo IPv6, le imprimen flexibilidad, dado que es posible utilizarlos indistintamente del protocolo IP, gracias al formato que manejan.

Conclusiones respecto al software que implementa los protocolos del nivel de aplicación

- El desarrollo de las aplicaciones IPv6 requiere consideraciones especiales, dado que el API de comunicaciones para el desarrollo cambia para soportar el protocolo IPv6, sin embargo, la mejor opción es construir aplicaciones independientes al protocolo sobre el cual van a trabajar.

- Teniendo en cuenta los lineamientos para la construcción de aplicaciones orientadas hacia IPv6 es posible desarrollar software que explote las características principales que este nuevo protocolo ofrece.

- A diferencia de Linux, en los sistemas operativos con los cuales trabaja la Universidad del Cauca Windows 2000 y XP, en lo referente a la configuración de un cliente DNS, no existen los mecanismos necesarios para realizar dicha configuración, pues gráficamente se aceptan direcciones solo de 32 bits y mediante consola aparentemente se pueden agregar los de 128 bits, mas sin embargo, su funcionamiento no es el esperado.

- Existen gran cantidad de aplicaciones que implementan este protocolo, aunque es de notar que la mayoría son producidas para el sistema operativo Linux, entre los que se encuentran no solo paquetes fundamentales como los de servicios canónicos sino también de aplicaciones multimedia como video conferencia, chat, y otros servicios de valor agregado, lo cual muestra el auge y el momento clave por el que se esta atravesando.

- Existen mecanismos de coexistencia entre aplicaciones que manejan diferentes pilas de protocolos que pueden ser de gran utilidad en el periodo de transición.

- Gracias a los esfuerzos de las empresas productoras y desarrolladores de software de comunicaciones, la transición para la Universidad del Cauca no debería suponer un gran trauma, dado que la mayoría de las aplicaciones utilizadas por la red de datos, cuentan con soporte IPv6.

Conclusiones con respecto a la tecnología

- La etapa experimental del protocolo IPv6 finalizó, prueba de ello es la desaparición del 6Bone que fue una red experimental creada para realizar pruebas de los estándares e implementaciones del nuevo protocolo IP, los prefijos 3ffe asignados por esta red a las diferentes entidades interesadas en la investigación y desarrollo del protocolo ya no son válidos, razón por la cual a la Universidad del Cauca le fue asignado un nuevo bloque de direcciones IPv6 2001:448:1024::/48 en reemplazo al asignado cuando el 6Bone estaba vigente 3ffe:8070:1024::/48. Este nuevo bloque es considerado de producción, para brindar servicios a los usuarios de una red indicando que las nuevas implantaciones del protocolo deben ser de carácter formal, mostrando la madurez que la tecnología ha alcanzado.

- La universidad del Cauca puede convertirse en una isla IPv6 con funcionamiento dual-stack; para alcanzar este objetivo se requiere fundamentalmente un equipo humano preparado e informado, la voluntad institucional e inversión de tiempo.

- Se concluye que la implementación de esta tecnología en la Universidad del Cauca resulta muy viable y altamente productiva, ya que, los usuarios finales se podrán beneficiar de los nuevos servicios y características que IPv6 ofrece; por otro lado la Facultad de Ingeniería Electrónica y Telecomunicaciones tendría un nuevo campo para explorar.

RECOMENDACIONES

- Se recomienda la conformación de un grupo institucional que propenda por la implementación total del protocolo IPv6 en la Universidad del Cauca, utilizando como herramienta base la conjunción de todo el conocimiento adquirido en cada una de las entidades investigadoras que en torno a esta temática han aportado en la Universidad.

- Es necesaria la migración de los servicios implementados, en este trabajo de grado, a equipos que ofrezcan la capacidad técnica, para brindar soporte de dichos servicios a toda la comunidad universitaria, dado que actualmente todos ellos se encuentran alojados en un mismo equipo.

- El desarrollo de este proyecto de grado deja abierta la posibilidad de futuros trabajos relacionados con esta tecnología, enfocados en el análisis e implementación de servicios que exploten las nuevas características que brinda el protocolo IPv6, como los servicios de tiempo real, Calidad de servicio y Mobile IPv6, al igual que el desarrollo de herramientas propietarias de la Universidad del Cauca, para que de esta manera la tecnología no sea solamente adquirida, sino que también sea generada.

- Debido a que en Colombia ya se consolidó el grupo de trabajo IPv6 denominado TASK FORCE COLOMBIA que tiene como objetivo coordinar esfuerzos para lograr una eficaz y pronta adopción del protocolo IPv6, es de gran importancia el acercamiento de la Universidad del Cauca a este grupo de trabajo para

intercambiar experiencias e impulsar a nivel nacional, la adopción de esta tecnología.

ACRÓNIMOS.

API	Interfaz de programación de aplicaciones (Aplication Program Interface).
BIND	Servidor de Nombres de Dominio de Berkeley (Berkeley Internet Name Domain).
BSD	Distribución Software Berkeley (Berkeley Software Distribution).
DHCP	Protocolo de Configuración Dinámica de equipos (Dynamic Host Configuration Protocol).
DNS	Sistema de Nombres de Dominio (Domain Name System).
DTP	Proceso de Transferencia de datos (Data Tranfer Process).
FQDN	Dominio completamente expresado (Fully Qualified Domain Name).
FTP	Protocolo de Transferencia de Archivos (File Transfer protocol).
GPL	Licencia Pública General (General Public License).
HTML	Lenguaje de Etiquetas de Hipertexto (Hypertext Markup Language).
HTTP	Protocolo de Transeferencia de Hipertexto (Hypertext Transfer Protocol).
HTTPS	Protocolo Seguro de Transeferencia de Hipertexto (Hypertext Transfer Protocol Secure).
IANA	Autoridad para la Asignación de Números Internet (Internet Assigned Numbers Authority).
IIS	Servidor de Información de Internet (Internet Information Server).
IMAP	Protocolo de Acceso a Mensajes Internet (Internet Message Access Protocol).
IPv4	Protocolo Internet Versión 4 (Internet Protocol 4).
IPv6	Protocolo Internet Versión 6 (Internet Protocol 6).
JSP	Páginas de servidor de Java (Java Server Pages).
MAC	Código del Mensaje de Autenticación (Message Authentication Code)
MTA	Agente de Transporte de Correos (Mail Transport Agent).

MTU	Unidad Máxima de Transferencia (Maxium Transfer Unit).
NLA	Agregación de siguiente Nivel (Next Level Aggregate).
PHP	Preprocesador de Hipertextos (Personal Home Page).
PI	Intérprete del Protocolo (Protocol Interpreter).
PKI	Infraestructura de Llaves Públicas (Public Key Infrastructure).
POP3	Protocolo de oficina de Correos Versión 3(Post Office Protocol 3).
RR	Registros de Recursos (Resource Record).
SMTP	Protocolo Simple de Transferencia de Correo (Simple Mail Transfer Protocol).
SOA	Inicio de Autoridad (Stara of Authority).
SSH	Consola Segura (Secure SHell).
SSL	Capa de Conexión Segura (Secure Socket Layers).
TCP	Protocolo de Control de Transporte (Transport Control Protocol).
TLA	Agregación de Nivel Superior (Top Level Aggregate).
TLS	Capa de Transporte Segura (Transport Layer Security)
UDP	Protocolo de Datagrama de Usuario (User Datagram Protocol).
URL	Localizador uniforme de recurso (Uniform Resource Locator).

REFERENCIAS BIBLIOGRAFICAS

[1] CASTRO, Eva., SAVOLA, P., HAGINO J., HONG Y-G., M-K. SHIN, Ed. RFC 4038. Application Aspects of IPv6 Transition. Marzo de 2005.

[2] CASTRO, Eva M. Porting applications to IPv6 HowTo. URL: http://jungla.dit.upm.es/~ecastro/IPv6-web/ipv6.html. Consulta Agosto de 2006.

[3] CASTRO, Eva M., DE MIGUEL, Tomas P., PAVON, Santiago. Transition of Applications to IPv6. Junio de 2005.

[4] MOCKAPETRIS, P. RFC 1034. Domain names – Concepts and Facilities. Noviembre de 1987.

[5] NUÑEZ, Zuleta Jose Vicente. Descripción del servicio DNS. URL: http://es.tldp.org/htmls/manuales.html. Consulta Mayo de 2006.

[6] THOMSON, S., HUITEMA, C., KSINANT, V., SOUISSI, M. RFC 3596. DNS Extensions to Support IP Version 6. Octubre de 2003.

[7] HINDEN, R., DEERING, S. RFC 3513. Internet Protocol Version 6 (IPv6) Addressing Architecture. Abril de 2003.

[**8]** CRAWFORD, M., HUITEMA, C. RFC 2874. DNS Extensions to Support IPv6 Address Aggregation and Renumbering. Julio de 2000.

[9] FERGUSON, P., BERKOWITZ, H. RFC 2071. Network Renumbering Overview. Enero de 1997.

[10] ABLEY, J., BLACK, B. RFC 3582. GILL, V. Goals for IPv6 Site-Multihoming Architectures. Agosto de 2003.

[11] CRAWFORD, M. RFC 2672. Non-Terminal DNS Name Redirection. Agosto de 1999.

[12] CRAWFORD, M. RFC 2673. Binary Labels in the Domain Name System. Agosto de 1999.

[13] RODRIGUEZ, Francisco. Ciudad de Mexico, 2001.Implantación del Protocolo IPv6 para Aplicaciones de Internet2 en el ITAM. Tesis (Ingeniero en Telecomunicaciones). Instituto Tecnológico Autónomo de México.

[14] THOMSON, S., NARTEN, T., RFC 2462. IPv6 Stateless Address Autoconfiguration. Diciembre 1998.

[15] PERALTA, Luis. Protocolo IPv6 URL: http://www.si.uji.es/bin/docs/projectes/ipv6/ipv6p.pdf, Consulta Abril de 2006.

[16] HIDEN, R., DEERING S. RFC 2373., IP Version 6 Addressing Architecture. Julio de 1998

[17] NARTEN, T., NORDMARK, E., SIMPSON W. ., RFC 2461. Neighbor Discovery for IP Version 6. Diciembre de 1998.

[18] VIXIE, P., THOMSON Ed., REKHTER, S., BOUND, J., RFC 2136. Dynamic Updates in the Domain Name System (DNS UPDATE). Abril de1997.

[19] DROMS, Ed. R., BOUND, J., LEMON, T., PERKINS, C., CARNEY, M., RFC 3315. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). Julio de 2003.

[20] Estadisticas uso de correo electrónico 2001-2005 URL: http://www.radicati.com, Consulta Marzo de 2006.

[21] HAGINO, J., NAKAMURA, M., RFC 3974. SMTP Operational Experience in Mixed IPv4/v6 Environments. Enero de 2005.

[22] MASINTER, L., HINDEN**,** R., CARPENTER, B., RFC2732. Format for Literal IPv6 Addresses in URL's., Diciembre de 1999.

[23] MACINE WEBLOG, Https, ssl y comunicación segura. URL: http://macine.epublish.cl/articles/epublish_httpssslycomunicacionsegura.html. Consulta Julio de 2006.

[24] POSTEL, J., REYNOLDS, J. RFC 959. FILE TRANSFER PROTOCOL (FTP). Octubre de 1985.

[25] ALLMAN, M., OSTERMANN, S., METZ, C. RFC 2428. FTP Extensions for IPv6 and NATs. Septiembre de 1998.

[26] SMALDONE, Javier. Introducción a Secure Shell, versión 0.20. URL: http://es.tldp.org/htmls/tutoriales.html. Consulta Junio de 2006.

[27] YLONEN, T., LONVICK, C. RFC 4251. The Secure Shell (SSH) Protocol Architecture. Enero de 2006.

[28] YLONEN, T., LONVICK, C. RFC 4253. The Secure Shell (SSH) Transport Layer Protocol. Enero de 2006.

[29] YLONEN, T., LONVICK, C. RFC 4252. The Secure Shell (SSH) Authentication Protocol. Enero de 2006.

[30] INOUYE, Alan S. Items from security-related news (E66.May-2005). URL: http://www.ieee-security.org/Cipher/Newsbriefs/2005/170505.ListWatch.html. Consulta Junio de 2006.

[31] YLONEN, T., LONVICK, C. RFC 4254. The Secure Shell (SSH) Connection Protocol. Enero de 2006.

[32] HIEKATA, Kazuo. Manual dhcp6s.conf. 1 cd-rom Fedora core 4. Consulta Mayo de 2006.