ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL.

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"ANÁLISIS Y ESTUDIO DEL PLANO DE CONTROL DE UN OPEN *ROUTER* BASADO EN EL SISTEMA OPERATIVO LINUX Y EN EL OPEN SOURCE SOFTWARE XORP"

TESIS DE GRADO

Previa a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentada por

Olga María Jaramillo Ortiz

Guayaquil - Ecuador

2007

AGRADECIMIENTO

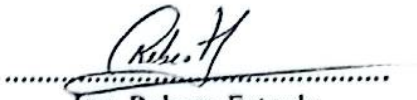
Ing. Rebeca Estrada, Directora de Tesis, por su gran apoyo durante el desarrollo de este trabajo.

DEDICATORIA

A mis queridos padres, Olga y Luis, por el amor y apoyo incondicional que siempre me han brindado.

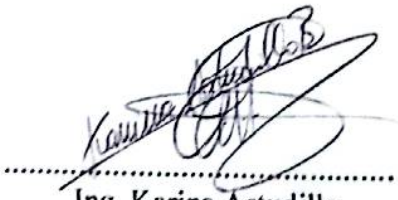
TRIBUNAL DE GRADUACION

Ing. Holger Cevallos
Sub-Decano de la FIEC



Ing. Rebeca Estrada
Directora de Tesis

Ing. Boris Ramos
Miembro Principal



Ing. Karina Astudillo
Miembro Principal

DECLARACION EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL"

.....

Olga Taramillo Ortiz

RESUMEN

El presente trabajo consiste en el "Análisis y Estudio del plano de control de un Open Router basado en el Sistema Operativo Linux y en el Open Source Software Xorp".

Para el desarrollo de este proyecto nos hemos focalizado en analizar la estructura del Open Router, conocer sus elementos arquitecturales, sus funciones, familiarizarnos con las plataformas sobre las que funciona, para desarrollar estudios que analicen el perfomance de este tipo de aparatos.

El Open Router dispone de una plataforma Linux con un kernel 2.6. La plataforma de enrutamiento que emplearemos es el proyecto Xorp versión 1.3, soporta los protocolos OSPF y BGP. La simulación de las diferentes configuraciones de redes para el desarrollo de nuestra tesis recae sobre el Router Tester Agilent 900 que nos permitirá simular redes de 400 nodos, routers que se encuentren en sistemas autónomos diferentes, rupturas de enlaces para el análisis del cambio de camino y la captura del tráfico de la red para el posterior análisis de información.

Una vez estudiados los experimentos a realizar procedemos a ejecutarlos, uno de los parámetros a medir es el tiempo de convergencia del aparato al producirse cambios en la topología de red. Los resultados obtenidos dan una visión del desempeño del Open Router, demuestran la capacidad de reacción al producirse un cambio en la métrica de un link y al anunciarse el estado de baja de un camino.

Los resultados son analizados y comparados con índices de prestación de aparatos de arquitectura cerrada y demuestran que los aparatos abiertos con el continuar del desarrollo de nuevas plataformas y mejoras en el código representan una óptima opción en el campo de la investigación y una buena alternativa a bajo costo.

INDICE GENERAL

Pág.

ABREVIATURAS

ABR	Router de Borde de Área (Area Border *Router*)
ARP	(Address Resolution Protocol)
ARPA	Agencia de Proyectos de Investigacion Avanzada (Advanced Research Projects Agency)
AS	Sistemas Autónomos (Autonomous System)
ASN	Número de Sistema Autónomo (Autonomous System Number)
BDR	Router Designado de Respaldo (Backup Designated Router)
BGP	(Border Gateway Protocol)
BN	Redes Broadcast (Broadcast Network)
CIDR	(Classless Inter-Domain Routing)
DDP	Paquete de Descripción de la Base de Datos (Database Description Packet)
DR	Router Designado (Designated Router)
DUT	Dispositivo bajo prueba (Device Under Test)
EGP	(Exterior Gateway Protocol)
FEA	(Forwarding Engine Abstraction)
FTP	Protocolo de Transferencia de Archivos (File Transfer Protocol)
HP	Paquete Hello
ICANN	Entidad que asigna las direcciones IP (Internet Corporation for Assigned Names and Numbers)
ICMP	Protocolo de Control de Mensajes (Internet Control Message Protocol)
IETF	(Internet Engineering Task Force)
IGP	(Interior Gateway Protocol)
IGRP	(Interior Gateway Routing Protocol)
IP	Protocolo de Internet (Internet Protocol)
LSA	Anuncio de Estado del Enlace (Link State Advertisement)

LSAP	Paquete de Anuncio de Estado del Enlace (Link State Acknowledgment Packet)
LIS	Subredes IP lógicas (Logical IP Subnet)
LSP	Paquete de Estado de Enlace (Link State Packet)
LSRP	Paquete de Requerimiento de Estado del Enlace (Link State Request Packet)
LSUP	Paquete de Actualización de Estado del Enlace (Link State Update Packet)
MTU	Unidad Máxima de Transferencia (Maximum Transfer Unit)
OR	Router Abierto (Open Router)
OSPF	(Open Shortest Path First)
RARP	(Reverse Address Resolution Protocol)
RIB	Base de Información de enrutamiento (Routing Information Base)
RIP	Protocolo de Información de Enrutamiento (Routing Information Protocol)
SMTP	Protocolo de Transferencia de Correo (Simple Mail Transfer Protocol)
SNMP	Protocolo Simple de Gestión de Redes (Simple Network Management Protocol)
SO	Sistema Operativo
SPF	Primero el camino más corto (Shortest Path First)
SUT	Sistema Bajo Prueba (System Under Test)
TCP	Protocolo de Control de Transmisión (Transmission Control Protocol)
TCP/IP	Arquitectura de reglas de comunicación (Transmission Control Protocol/Internet Protocol)
TOS	Tipo de Servicio (Type of Service)
UDP	(User Datagram Protocol)

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION

La sucesión de eventos, proyectos, ideas y protagonistas que, en el curso de treinta años, han llevado al nacimiento de Internet y a la evolución de su forma actual, constituye un capítulo importante en la historia del desarrollo tecnológico. Parte del encanto se debe al papel determinante que esta tecnología ha desarrollado y está todavía desarrollando en la llamada "revolución digital"

Los orígenes de Internet se remontan a los años 60, en plena Guerra Fría, como proyecto del Departamento de la Defensa estadounidense para el desarrollo de una red telemática descentralizada. Sucesivamente estuvo disponible para fines no militares, conectando primero los principales centros universitarios para luego difundirse y convertirse en la "red de redes".

La tecnología de red que ha tenido la mayor difusión es la suite protocolar TCP/IP, nacida para el transporte de datos y hoy candidata a convertirse la tecnología de referencia para futuras redes globales.

El triunfo obtenido por Internet proviene en gran parte del hecho que nació como una suite protocolar abierta: todos los protocolos, las arquitecturas y las estructuras han sido creadas y descritas públicamente.

Lo contrario a que casi todos los aparatos comerciales que son caracterizados por sus arquitecturas "cerradas". En otras palabras los detalles arquitecturales no son

de dominio público, protegiendo así la inversión industrial de proyectos en el mismo dispositivo (aparato).

En los años ochenta los programadores comenzaron a limitar los derechos de uso de su software y cobrar por cada copia vendida. El flujo de comunicación interna a la categoría de los desarrolladores fue interrumpido por las exigencias legales de los derechos que los respectivos propietarios vanagloriaban sobre el software de nueva producción, haciendo de esta manera imposible realizar, analizar y validar nuevos mecanismos y protocolos sobre aparatos comerciales. Fue de esta manera que gran parte de estos grupos focalizaron el propio interés sobre proyectos de código abierto (*open-source*) basados en el sistema operativo Linux.

Hoy en día muchos de estos proyectos han logrado un nivel de maduración elevado para constituir un completo y confiable soporte a las funcionalidades de enrutamiento IP. El conjunto de estos instrumentos open-source constituye una arquitectura de red que está en grado de competir con la mayor parte de los protocolos comerciales. Esta arquitectura se llama Open Router.

A pesar de los muchos elementos arquitecturales del Open Router que ya han sido descritos, todavía faltan, estudios que analicen el perfomance obtenido en este tipo de equipos.

El objetivo de esta tesis ha sido el de realizar un análisis de prestaciones de realización open source pertenecientes al plano de control utilizando los protocolos de enrutamiento BGP y OSPF para su posterior confrontación con las características de un ruteador comercial.

Para tal objetivo usamos el Open Router que dispone de una plataforma Linux equipada con un kernel 2.6, y del uso de instrumentos de medida y emulación de altas prestaciones (Agilent Router Tester 900). La selección del uso de la

plataforma de enrutamiento open source recae sobre Xorp, proyecto en veloz evolución y altamente flexible.

El presente trabajo se divide en 5 capítulos; en el primer capítulo se procede a una introducción de la arquitectura TCP/IP, en particular a la descripción de las funcionalidades típicas de un ruteador IP y de los protocolos más significativos que gobiernan el comportamiento.

En el capítulo 2 se presentan en el detalle las arquitecturas hardware y software del Open Router, con los particulares resultados del estudio y de las realizaciones del código de networking interna al kernel Linux.

El capítulo 3 presenta los protocolos de enrutamiento utilizados, examinando en el detalle los protocolos OSPF y BGP, a los cuales se ha dado particular atención durante el desarrollo de este trabajo.

El Capítulo 4 trata del estudio de Xorp, analizando en particular su arquitectura modular, los protocolos de enrutamiento hasta el momento implementados y la comunicación interna entre procesos, examinando con un nivel de detalle elevado el código que realiza el procedimiento de recálculo de los caminos óptimos.

El capítulo 5 exhibe los diferentes tests propuestos con los resultados obtenidos en el desarrollo de este trabajo.

Capítulo 1 : Router IP

En este capítulo describiremos los principios fundamentales de la transmisión de datos utilizando un servicio no orientado a la conexión, siendo Protocolo Internet (*Internet Protocol*, IP) el principal protocolo usado en el ámbito de networking. Presentaremos el formato del datagrama IP indicando como representa la base de la comunicación en Internet. Además describiremos el proceso de enrutamiento y los principales protocolos utilizados para tal propósito.

1.1 Origen de Internet

A mitad de los años 70 la agencia americana ARPA (*Advanced Research Projects Agency*), bajo el control del DoD (*Department of Defense*), comenzó a trabajar en un proyecto de internetworking, especificando la actual arquitectura de protocolos, denominada TCP/IP, alrededor de 1978/79. El verdadero nacimiento de Internet [1] fue en 1980, cuando ARPA comenzó a convertir todos los aparatos conectados a su red de investigación al nuevo protocolo TCP/IP. La adopción de la nueva tecnología Internet viene realizada en 1983, cuando el Departamento de Defensa americano ordenó que todas sus computadoras conectadas a la red geográfica usaran TCP/IP.

Al mismo tiempo la red fue dividida en dos segmentos, uno de investigación, llamado ARPANET, y uno militar, llamado MILNET.

Para promover la utilización de esta nueva tecnología a las universidades, ARPA hizo pública una implementación del TCP/IP. Dado que la mayor parte de los departamentos de informática de las universidades de la época utilizaban el BSD Unix de Berkeley, entonces ARPA decidió financiar a la sociedad BBN para que implementara TCP/IP bajo Unix, y a la Universidad de Berkeley para que integrara esta implementación en la distribución BSD. De esta manera obtiene el 90% de los departamentos de informática de las universidades.

La implementación TCP/IP de Berkeley también integró en los servicios de red el soporte a una serie de aplicaciones típicas de Unix (por ejemplo RPC) proporcionando una nueva abstracción – *socket* – para permitir a las aplicaciones el acceso a las funcionalidades de comunicación proporcionadas por los protocolos de red.

El éxito de TCP/IP y de Internet en la investigación informática universitaria llevó a otros grupos a la decisión de adoptarlo. En 1986 la *National Science Foundation* (NFS) financió un nuevo backbone Internet, llamado NFSNET, que conectaba no solo a los departamentos de informática, sino también a todos los principales centros de investigación americanos.

A partir del final de los años 80 Internet comenzó a difundirse también en las universidades y centros de investigación de Europa y en las grandes compañías americanas. Al inicio de los años 90 tanto las compañías pequeñas como grandes de América y del mundo comenzaron a conectarse a Internet.

El número de sistemas informáticos conectados a Internet ha pasado de 213 en agosto del 81, a 313000 en octubre del 90, 3500000 en julio del 94 y a casi

60000000 a mediados del 99, con un ritmo de crecimiento estimado de más de 4000 nuevos hosts al día [2].

La tecnología TCP/IP también fue adoptada al interno de las redes privadas empresariales (*intranet*), sin que estas se conecten necesariamente a Internet.

Los protocolos TCP/IP pertenecen a una tecnología abierta, en que sus especificaciones han sido publicadas en modo de ser implementadas por cualquier persona, a diferencia de los protocolos de comunicación propietarios, como DECNET, SNA, IPX, etc.

Internet, por lo tanto, es una tecnología compuesta de una arquitectura y de reglas de comunicación (protocolos), llamadas *TCP/IP suite*. La arquitectura DoD, desarrollada con el fin de crear una interconexión universal entre redes (*internetworking*) y con el conocimiento de que ninguna otra red está en grado de servir eficazmente a cada tipo de cliente, se distingue de un estándar "*de jure*" como por ejemplo la ISO-OSI porque:

- Usa una estructuración más flexible;
- Da más importancia al internetworking;
- Considera con mucha atención los servicios no orientados a la conexión.

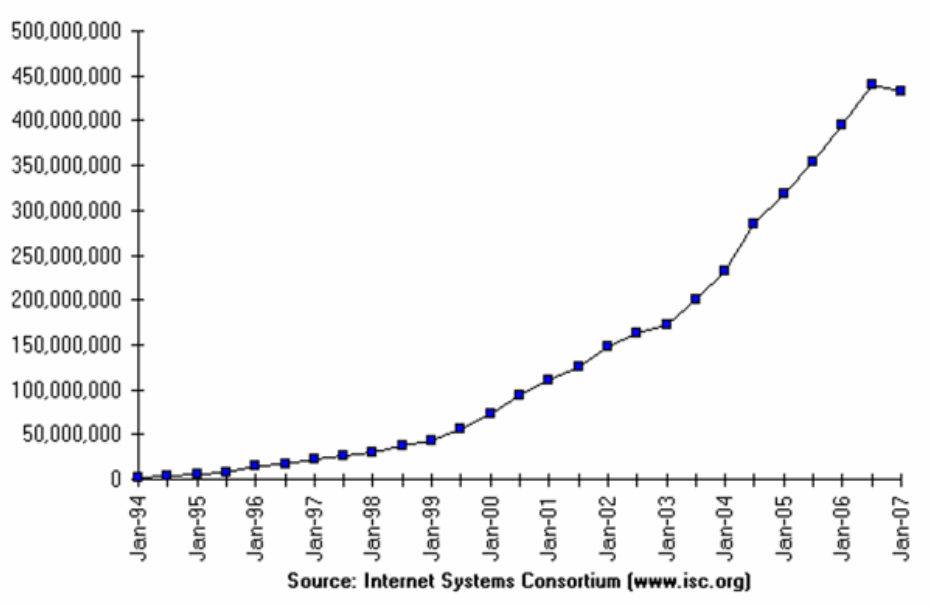


Fig. 1.1: Número estimado de hosts conectados a Internet en los últimos 10 años.

La verdadera fuerza de la arquitectura *TCP/IP suite*, respecto a las varias redes y protocolos propietarios existentes, es el internetworking. De hecho, gracias a los enlaces de conexión entre las varias redes propietarias (llamadas *gateway*) y a la definición de los protocolos, se pueden conectar redes entre sí.

El TCP/IP es, por lo tanto, un conjunto de reglas públicas, o bien, un sistema abierto (*open system*), que permite la interconexión, independiente de la tecnología usada, de redes. Sus principales ventajas son: la independencia de tecnologías de las pequeñas redes interconectadas, la posibilidad de transmitir confirmaciones de recepción (*acknowledgement*) directamente del destinatario al remitente (*end to end*), y sobretodo el soporte de una notable cantidad de protocolos aplicativos para cualquier fin posible.

Como confirmación de todo aquello que ha sido indicado, se puede ver en la figura 1.2 las diferencias de estructuración de la arquitectura DoD respecto a la OSI: los niveles bajos (es decir la capa Física y la capa de Enlace de Datos) de la arquitectura TCP/IP no son especificados, debido a que han sido creados para funcionar eficazmente sobre cada tecnología de red. De hecho, es el nivel de red, con el protocolo IP, que "esconde" a los niveles superiores los detalles de funcionamiento de las redes atravesadas, además de desarrollar las funciones de conmutación y enrutamiento de paquetes.

En el nivel de red, la red Internet puede ser vista como un conjunto de redes más pequeñas o de Sistemas Autónomos (*Autonomous System*, AS) interconectadas entre sí. La función del protocolo IP, en este contexto, es la de proveer un medio de tipo best effort (y por tanto no garantizado) para transportar los paquetes de información datagrama desde el origen hasta el destino, sin tener en cuenta la posición de ambas terminales (que pueden encontrase en diferentes redes), y de la virtual presencia de redes intermedias.

El servicio que provee este protocolo de red es llamado *unreliable*, es decir no confiable, por lo que no da ninguna garantía de que el paquete llegue efectivamente al destino. En segundo lugar es también llamado *connectionless*, es decir sin conexión directa, por lo que la transmisión no se realiza directamente hacia el destinatario, el mensaje es enviado a la red dejando a ésta la tarea de llevarlo al destino usando la dirección IP del host de destino. En fin se habla del envío del paquete con el mayor esfuerzo posible (*best effort delivery*) pero sin ninguna garantía, en cuanto a que la red hace todo lo posible para llevar el paquete al destino.

Arriba de los niveles de interconexión entre redes está el nivel de transporte. Conceptualmente es a este nivel que se encuentran TCP y UDP.

El Protocolo de Datagrama del Usuario (*User Datagram Protocol*, UDP) es un protocolo muy difundido. Su encabezado es simple y está formado del número de puertas del origen, del destino, de la longitud del mensaje UDP, de los datos, y del checksum para verificar la integridad de los datos. De hecho, también UDP, como IP, es un protocolo llamado no confiable. Esto quiere decir que un mensaje UDP puede perderse, ser duplicado, o llegar en el orden equivocado. La fiabilidad de la comunicación es de hecho dada a los protocolos de nivel superior.

El Protocolo de Control de Transmisión (*Transmission Control Protocol*, TCP), en general, se ocupa de controlar que la información pasada por los niveles superiores llegue correctamente al destino. Esto ha sido definido para que funcione sobre cualquier sistema de transmisión de datos a paquete (por lo tanto funciona sin IP).

El TCP realiza un servicio orientado a la conexión (*end to end*) y logra proveer a los niveles superiores una visión de los datos de tipo data stream, es decir los datos son recibidos en secuencia y en el mismo orden con el cual han sido transmitidos. En este nivel, el usuario del TCP envía los datos como un simple flujo de bytes y en el mismo modo los recibe.

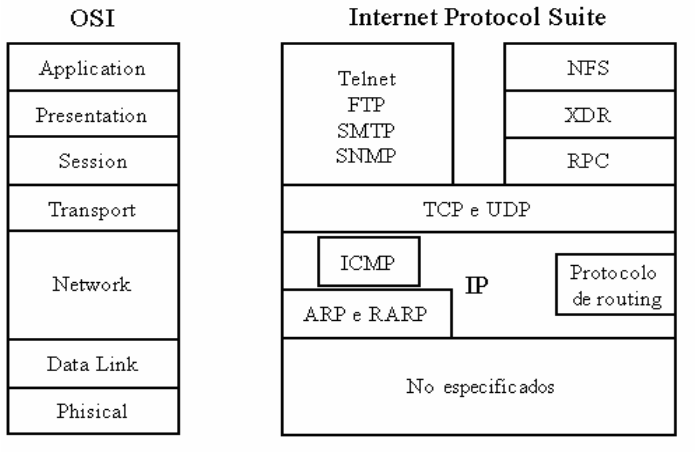


Fig. 1.2: Modelo de referencia ISO-OSI e Internet Protocol Suite.

Todos los protocolos de nivel superior al de Transporte, son especializados en realizar servicios específicos; el Protocolo de Transferencia de Archivo (*File Transfer Protocol*, FTP), por ejemplo, permite interactuar con un computador remoto, el Protocolo Simple de Transferencia de Correo (*Simple Mail Transfer Protocol*, SMTP), sirve para transferir el correo electrónico, el Protocolo Simple de Administración de red (*Simple Network Management Protocol*, SNMP), para administrar y monitorear el estado de la red.

1.2 Protocolo IP

El protocolo IP que provee el mecanismo de entrega de paquetes, arriba descrito, es el Internet Protocol (IP) [3], protocolo fundamental sobre el cual se basa la arquitectura TCP/IP, ayudado de otros protocolos de soporte.

IP es un protocolo simple, de tipo datagrama, no orientado a la conexión, y junto a TCP constituye el núcleo original del Internet Protocol Suite.

Entre las funciones principales implementadas en este protocolo, podemos citar el enrutamiento, la fragmentación y reensamblaje de los mensajes y la revelación de los errores.

El elemento base de transporte del protocolo IP es el paquete IP (*IP datagram*) que, como las tramas de la red Ethernet, se compone de una cabecera y de una área de datos (*payload*). El formato de la cabecera es mostrado en la figura 1.3.

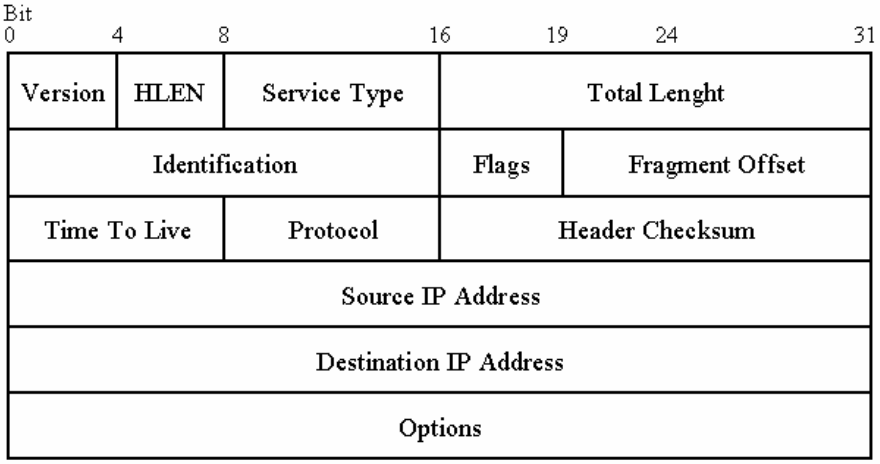


Fig. 1.3: Esquema del encabezado IP.

El encabezado tiene una longitud variable partiendo de un mínimo de 20 bytes, contiene algunos campos cuya posición al interno del paquete es fija. El detalle de la estructura es la siguiente:

bit 0-3 *Version:* indica la versión del protocolo IP usado, actualmente es la 4, pero ya existen servicios que están usando la nueva versión 6 (IPv6 [4])

bit 4-7 *Header Length*: contiene la longitud total del encabezado, que varía según el contenido del campo *Options*

bit 8-15 *Type of Service*: inicialmente utilizado para describir el contenido del paquete, ahora considerado de gran importancia en el soporte a la calidad de servicio.

bit 16-31 *Total Length*: longitud total del datagrama (*header + payload*)

bit 32-47 *Identification*: en caso de fragmentación indica el datagrama al que pertenece el fragmento en cuestión

bit 48-50 *Flags*: administran la fragmentación

bit 51-63	*Fragment Offset*: indica en que posición del datagrama completo va colocado el fragmento en cuestión
bit 64-71	*Time to live*: es un contador numérico utilizado para limitar la permanencia en red de los paquetes y así evitar bucles infinitos y tráfico inútil.
bit 71-79	*Protocol*: indica que protocolo de transporte esta siendo utilizado.
bit 80-95	*Header Checksum*: garantiza el control de la integridad del *header*
bit 96-127	*Source IP Address*: dirección IP del host que ha transmitido el paquete
bit 128-159	*Destination IP Address*: dirección IP del host de destino
bit 160-…	*Options*

El datagrama IP es encapsulado en una trama física, para hacer eficiente la transmisión a través de una red física real. Sin embargo, cada red local permite una dimensión mínima y máxima (indicada como *Maximum Transfer Unit*, MTU) para los paquetes que la atraviesan, y por lo tanto es obvio, visto que los datagramas IP deben ser encapsulados en una trama, no puedan prescindir de éstas dimensiones por otras variables según el tipo de red; la celda de las redes ATM, por ejemplo, es más pequeña (48 bytes fijos) que la del paquete IP (64 bytes mínimo).

Entonces para protegerse de la desigualdad de las redes, en lugar de estructurar el datagrama en modo excesivamente ajustado a los vínculos físicos de los varios productos, se ha decidido elegir una dimensión conveniente para los datagramas IP y utilizar un método (*segmentation and reassembly*) para dividirlos en pequeños pedazos llamados fragmentos de modo que pueden ser aceptados desde una red con cualquier pequeño MTU y, obviamente, reensamblados a la salida.

1.2.1 Otros protocolos del nivel de red

Junto al protocolo IP trabajan otros protocolos con funciones de soporte, los más importantes son ARP y RARP (que veremos con detalle en el párrafo 1.4.2) que se encargan de administrar la correspondencia entre las direcciones físicas de los hosts y las direcciones IP, el protocolo ICMP [5] que se ocupa del intercambio de información y de mensajes entre hosts, y los protocolos de enrutamiento que trataremos con profundidad más adelante en este capítulo.

1.3 Direccionamiento IP

Para que el sistema de comunicación hasta el momento descrito funcione, cada host conectado en red debe tener una dirección unívoca que lo distinga, estructurada de manera tal que facilite las operaciones de enrutamiento y la identificación de los hosts mismos. En realidad, la dirección no viene asignada al host directamente pero sí a su propia tarjeta de red, así un host con algunas interfaces tendrá igual número de direcciones.

La dirección IP es una dirección binaria de 32 bits y usa una estructura sustancialmente diferente a la de las direcciones físicas de las tarjetas de red (dirección MAC). Las direcciones MAC están compuestas de 48 bits divididos en 2 partes principales, una identifica al productor del hardware de la tarjeta de red, y la otra parte identifica a la tarjeta de red. La idea base de la dirección IP es aquella de poder identificar de manera unívoca los hosts sobre la red, pero a diferencia de las direcciones MAC, la dirección IP viene asignada según la ubicación topológica del host a la cual ha sido asignada.

Para hacer que estas direcciones sean fácilmente reconocidas, han sido subdivididas en grupos de 8 bits las cuales son representadas con una notación decimal separada por puntos (*Dotted Decimal Notation*). Para garantizar la univocidad de las direcciones IP su asignación ha sido administrada por un ente *no profit*: *Internet Corporation for Assigned Names and Numbers* (ICANN).

Cada dirección IP representa una pareja *net-id – host-id*, donde *net-id* identifica la red a la cual ha sido conectado el host, mientras *host-id* identifica al host mismo al interno de la red, como se muestra en la figura 1.4.

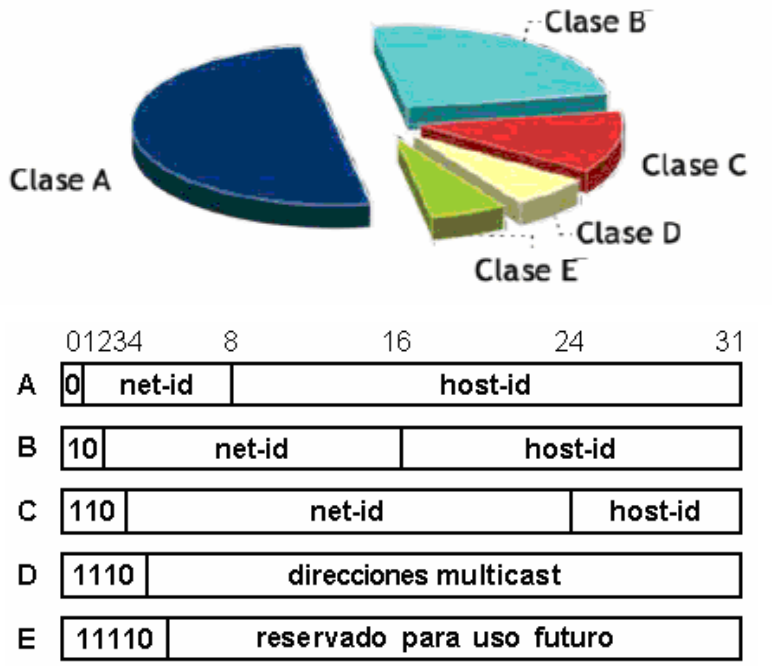


Fig. 1.4: Subdivisión en clases de las direcciones IP.

Las direcciones IP inicialmente estaban divididas en cinco clases, de las cuales las tres primeras (A, B, C) son distinguidas por los tres bits de orden más alto.

Clase A: redes con más de 2^{16} (65536) hosts, dedica 7 bits para *net-id* y 24 para *host-id*
Clase B: redes entre 2^{8} (256) y 2^{16} (65536) hosts, dedica 14 bits para *net-id* y 16 para *host-id*
Clase C: redes con menos de 2^{8} (256) hosts, dedica 21 bits para *net-id* y 8 para *host-id*
Clase D: reservada para la distribución multicast
Clase E: reservada para uso futuro

Tab. I: Número de hosts de las clases de direccionamiento IP

En la tabla II están escritos los rangos correspondientes a cada clase de las direcciones IP; algunos valores son reservados para fines específicos (127.0.0.0 está reservado para el host local).

Clase	Dirección menor	Dirección Mayor
A	0.1.0.0	126.0.0.0
B	128.0.0.0	191.255.0.0
C	192.0.1.0	223.255.255.0
D	224.0.0.0	239.255.255.255
E	240.0.0.0	247.255.255.255

Tab. II: Rango de las clases de direccionamiento IP

1.3.1 Subredes

La división inicial en clases se convirtió rápidamente insuficiente para la administración eficaz del número de direcciones totales a disposición, en cuanto a la asignación de clases enteras a grandes empresas e ISPs causaba un desperdicio de un gran número de direcciones no utilizadas, además, con el crecimiento de las topologías de red tanto en dimensión como en complejidad, se

hacía cada vez más difícil administrar las tablas de enrutamiento que crecían exponencialmente.

Por lo tanto, se optó por un sistema más flexible de división de redes en subredes (*subnet*) llamado *Classless Inter-Domain Routing* (CIDR [6]), que provee el uso de la dirección IP junto a una máscara (llamada *netmask*), compuesta también de 32 bits, para separar *net-id* de *host-id*. Este método permite dividir las redes de manera jerárquica en más subredes, permitiendo así administrar de manera más eficiente el enrutamiento de paquetes. De hecho, por medio del mecanismo de subnetting[1], las decisiones para el enrutamiento vienen tomadas por los niveles más altos del ruteador que no necesariamente deben conocer las direcciones de todos los hosts de la red, sólo se limitan a sus propias tablas de enrutamiento que contienen la información de los hosts que se encuentran a su mismo nivel.

Para obtener la *net-id* de una dirección IP basta realizar un AND lógico entre la dirección misma y la máscara de red que está compuesta, iniciando de la derecha, de tantos bits a ceros que sirven para indicar el número de hosts de una subred.

Para entender mejor el procedimiento veamos un ejemplo: tres grandes compañías necesitan direcciones IP para sus redes informáticas respectivamente de 2000, 1000 y 4000 hosts. La entidad designada asigna los siguientes tres bloques de direcciones creando 2 subredes al interno de una entera clase B (figura 1.5):

[1] Subnetting es un proceso que consiste en dividir las clases de direcciones de red completas en partes de menor tamaño; es útil para una mejor administración de las direcciones IP.

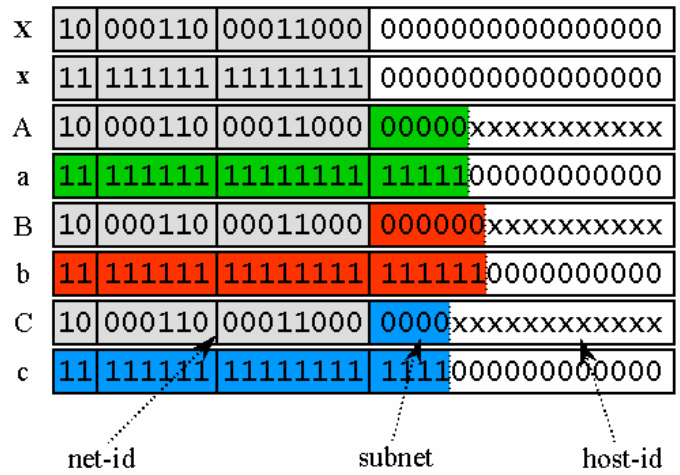


Fig. 1.5: Subdivisión CIDR de las direcciones IP de una clase B con sus respectivas máscaras de red.

A el bloque desde 134.24.0.0 hasta 134.24.7.255 (2048 direcciones)
B el bloque desde 134.24.8.0 hasta 134.24.11.255 (1024 direcciones)
C el bloque desde 134.24.16.0 hasta 134.24.31.255 (4096 direcciones)

Y sus respectivas máscaras de red:

A 255.255.248.0 (11111111.11111111.11111000.00000000)
B 255.255.252.0 (11111111.11111111.11111100.00000000)
C 255.255.240.0 (11111111.11111111.11110000.00000000)

Se pueden describir los bloques de direcciones y las relativas máscaras de red (*netmask)* mediante la notación: *net-id*/longitud-máscara donde longitud de la máscara quiere decir el número de bits a 1. En nuestro ejemplo indicaremos con 134.24.0.0/21 al primer bloque de direcciones, con 134.24.8.0/22 al segundo y con 134.24.16.0/20 al tercero.

En el siguiente párrafo veremos las relaciones dirección/máscara que son usadas en el enrutamiento.

1.4 El Ruteador IP

En un sistema de conmutación de paquetes, la operación de enrutamiento se refiere al mecanismo con el cual es elegido el camino para el enrutamiento de paquetes, y el ruteador es el encargado de tomar dicha decisión. El enrutamiento se realiza a niveles altos; por ejemplo, al interno de una WAN interconectada a algunas redes físicas mediante un switch, es la red misma que se encarga del enrutamiento de los paquetes desde un host a otro.

Lo que sucede al interno es totalmente invisible al externo que ve a esta red como una entidad en grado de entregar los paquetes. El enrutamiento al interno de una subred es trivial, por lo que dicha subred coincide con una red física que, por si misma, garantiza la accesibilidad directa de las estaciones a ella conectadas. El único problema que se puede encontrar es el de realizar un mapa de las direcciones IP con sus correspondientes direcciones del nivel físico (MAC). En la actualidad este problema es resuelto en modo automático por los protocolos ARP y RARP.

En este párrafo nos focalizaremos en el enrutamiento Internet llamado también enrutamiento IP. Análogamente al enrutamiento al interno de una red física, el enrutamiento IP elige el camino por el cual enviar los paquetes, pero esta vez necesita de algoritmos específicos para que los datagramas puedan atravesar más redes físicas independientes.

El enrutamiento entre las subredes es dirigido por los ruteadores IP. Dichos aparatos de red no se limitan al simple reenvío de los paquetes IP, como sucede en el interno de las redes físicas, sino que también se encargan de: el reenvío, el enrutamiento y el control y la gestión del estado de la red.

Para entender mejor el enrutamiento IP es necesario tener presente la estructura de Internet, una gigantesca red compuesta de la interconexión de muchas redes físicas. Cada red física está conectada a una o más redes mediante un ruteador. Los datagramas atraviesan las redes pasando de ruteador en ruteador según las decisiones tomadas por ellos mismos, hasta que llegan al último ruteador del camino, que es el que está conectado físicamente a la subred que contiene el host de destino.

Para poder decidir los caminos por los cuales enviar los paquetes, los ruteadores tienen constantemente actualizadas las propias tablas de enrutamiento, monitoreando las conexiones con los ruteadores adyacentes e intercambiando información (tráfico sobre los enlaces, estado de las conexiones, etc.). Pensando en la totalidad de la red Internet, es comprensible como la topología de las conexiones pueden no ser estáticas, las conexiones pueden interrumpirse, obstruirse o pueden cambiar las políticas de servicio que regulan el uso. En un escenario parecido los ruteadores deben estar en grado de administrar todas sus funciones dinámicamente, teniendo en cuenta de los eventuales cambios topológicos que pueden ocurrir.

Esta situación necesita de algoritmos de enrutamiento particulares, debido a que los trayectos que los datos recorren entre la fuente y el destino pueden variar. La clasificación de los datos puede ser calculada según políticas específicas de servicio que proveen diferentes niveles de prioridad para los simples flujos de datos, puede ser importante, por ejemplo que las informaciones privilegiadas tengan a disposición caminos “preferenciales” que hagan más veloz la

transmisión o garanticen una mayor seguridad. Estas topologías de tráfico son llamadas Calidad de Servicio (QoS).

1.4.1 Tabla de enrutamiento

Los algoritmos de enrutamiento se sirven de tablas residentes en cada ruteador, llamadas tablas de enrutamiento o tablas IP, que contienen información sobre las posibles destinaciones y sobre como alcanzarlas.

Hipotéticamente sería útil que cada tabla de enrutamiento contenga información sobre cada posible destino, pero si así fuera, sería imposible tenerlas actualizadas, además, vista la enorme cantidad de hosts conectados en red, ningún ruteador tendría espacio físico suficiente para memorizar toda la información. En realidad, los algoritmos de enrutamiento, son más eficientes cuando menor es la cantidad de información memorizada en las tablas IP.

En los párrafos anteriores hemos visto como los ruteadores conectan las varias redes y subredes en manera jerárquica y como las decisiones de enrutamiento son tomadas a diferentes niveles.

Conceptualmente, el principio usado es aquel del information hiding (literalmente “ocultamiento de información”) según el cual un ruteador tiene en su propia tabla de enrutamiento información relativa a los otros ruteadores del mismo nivel jerárquico y de aquellos inferiores directamente conectados a él. Podemos citar un ejemplo, un ruteador de nivel regional tiene información relativa de otros ruteadores regionales pero no información relativa a la subred de las pequeñas redes metropolitanas.

La estructura de la dirección IP y el mecanismo de subnetting permiten realizar esto de manera simple y eficaz.

Hemos visto como todos los hosts conectados a una red física comparten un prefijo común en su dirección IP, este prefijo (el *net-id*) es la información que viene memorizada en las tablas de enrutamiento. Utilizar sólo el indicativo de la red en lugar de la dirección IP completa permite realizar un enrutamiento más eficiente y mantener pequeñas las tablas IP.

Veamos ahora con detalle como son realizadas las tablas IP y como se utilizan. Típicamente las tablas de enrutamiento contienen las relaciones (N,A) donde N es el indicativo de una red (con su máscara de red) y A es la dirección IP del ruteador siguiente que debe alcanzar para llegar al destino (comúnmente llamado "*next-hop*"). A estas relaciones se adjunta la información de la línea de salida por la cual enviar los paquetes para llegar al siguiente salto (*next-hop*) y eventualmente el "costo" del camino. El ruteador en este modo no debe conocer todos los caminos para llegar a todos los destinos, solo se limita a tener información sobre el paso siguiente (atravesando una solo red física), será el siguiente salto que decidirá por el nuevo trayecto a seguir.

A la llegada de cada datagrama IP, el ruteador extrae la dirección de destino y hace un AND con todas las máscaras de red memorizadas en la propia tabla de red. Si el resultado de una de estas operaciones coincide con la *net-id* a la cual es asociada la máscara de red, el ruteador ha identificado la red de destino y envía el paquete sobre la línea de salida indicada en la tabla. En el caso existan algunas líneas de salida para alcanzar el destino, los algoritmos de enrutamiento decidirán en base a políticas específicas de servicio (que se pueden basar sobre información de costo del camino, sobre eventuales prioridades del paquete, etc.).

En la figura 1.6 se muestra un ejemplo de cómo se realiza el enrutamiento utilizando las tablas IP. En dicho ejemplo la red 190.3 de clase B ha sido dividida en 256 subredes, cada una con 256 direcciones. En el ejemplo son

usadas 5 subredes: la 190.3.1.0, la 190.3.3.0, la 190.3.6.0, la 190.3.7.0 y la 190.3.9.0

De estas 5 subredes, dos pertenecen a redes físicas de tipo Ethernet (la 190.3.6.0 y la 190.3.1.0), una a una red FDDI (la 190.3.3.0) y dos a canales geográficos de tipo punto-punto (la 190.3.7.0 y la 190.3.9.0)

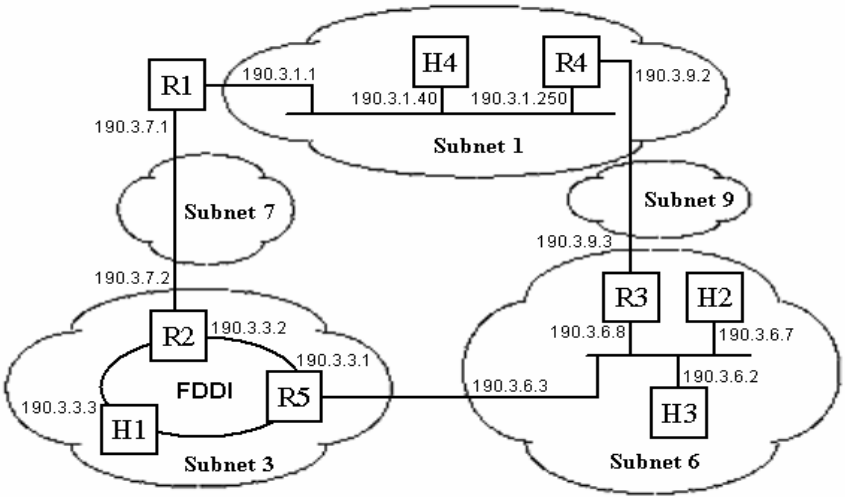


Fig. 1.6: Ejemplo de enrutamiento mediante el uso de tablas IP.

Los ruteadores tienen tantas direcciones como interfaces y por lo tanto las subredes que conectan. Por ejemplo, el ruteador R5 tiene dos direcciones, una asociada a la red FDDI (190.3.3.1) y la otra asociada a la red Ethernet (190.3.6.3)

Subred de destino	Máscara	Dirección del next-hop	Costo
190.3.6.0	255.255.255.0	Directa	0
190.3.3.0	255.255.255.0	Directa	0
190.3.1.0	255.255.255.0	190.3.3.2	1
190.3.7.0	255.255.255.0	190.3.3.2	3
190.3.9.0	255.255.255.0	190.3.6.8	1
default	*	190.3.7.1	9

Tab. III: Tabla de enrutamiento del ruteador R5

Considerando el ruteador R5, éste tendrá una tabla de enrutamiento que comprende una entrada para todas las subredes, de la misma red, con las cuales el ruteador no está directamente conectado (en este caso tres: la 190.3.1.0, la 190.3.7.0 y la 190.3.9.0) más las indicaciones de un ruteador por defecto (*default router*) por el cual enviar paquetes al exterior de la red. La tabla R5 esta mostrada en la tabla III.

Podemos ver que todas las direcciones de la tercera columna tienen que pertenecer a las redes en las cuales el ruteador esté directamente conectado. En el ejemplo, R5 está conectado a las subredes 3 y 6.

La tabla de enrutamiento puede ser creada manualmente o bien calculada por los algoritmos de enrutamiento descritos a continuación.

Cuando el ruteador R5 tiene que enviar un paquete, primero verifica si el destino del paquete pertenece a una red directamente conectada a él. Si este es el caso, la transmisión es realizada directamente. En caso contrario, envía el paquete al ruteador por defecto (en este caso al R2). R2 envía el mensaje al destino. Si durante la operación de enrutamiento R2 debe enviar el mensaje a la misma red de la cual lo ha recibido, por ejemplo porque el destino del mensaje es la subred 7 y lo envía a R1, entonces R2 envía un mensaje de enrutamiento redirigido (*routing redirect)* al host fuente.

El mensaje de enrutamiento redirigido es enviado usando el protocolo ICMP que se apoya su IP y sirve para administrar información de servicio.

Cuando H4 decide a que dirección IP enviar el mensaje, tiene que descubrir, si es que no lo sabe aún, cual es la dirección de nivel 2 (en el caso de las LAN la dirección MAC) del destino. Para hacer esto utiliza el protocolo ARP.

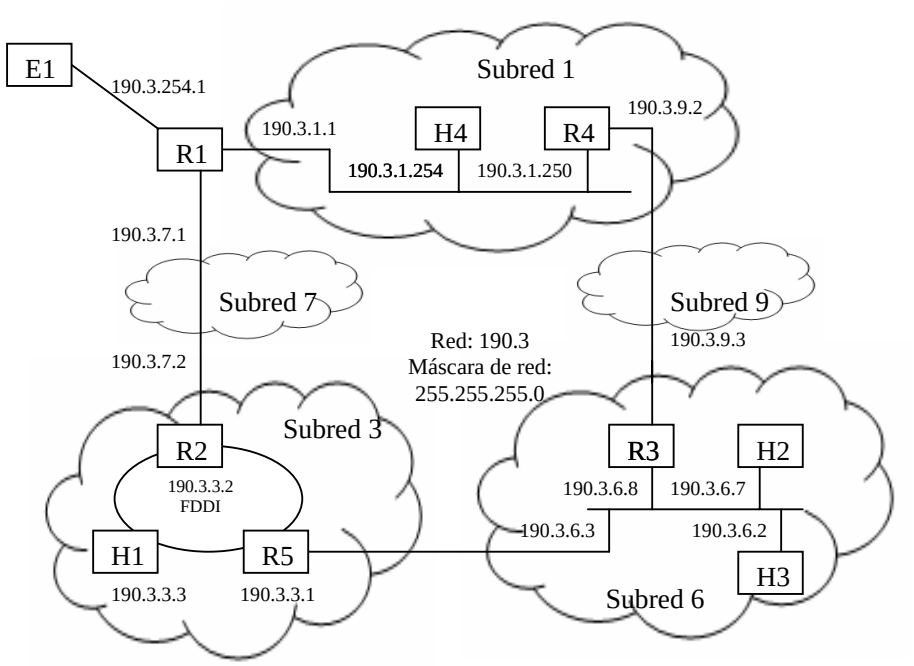


Fig. 1.7: Agregación de un ruteador en la tabla IP del ruteador E1

Consideremos ahora otro ruteador (E1) puesto en el exterior de estas subredes conectado al ruteador R1 como muestra la figura 1.7.

El único camino posible para los paquetes en tránsito desde E1 hacia las subredes vistas antes tiene como siguiente salto (*next-hop)* R1. Para evitar inserir en la tabla IP de E1 256 elementos (uno por cada uno de los 256 posibles destinos), E1 agrega los 256 elementos relativos en un único elemento que tiene una máscara de subred válida para todas las subredes. Mirando las direcciones IP de las subredes vistas en el ejemplo, se ve como todas tienen una raíz común: 190.3.0.0, una máscara en grado de extraer esta raíz común es por lo tanto en grado de identificar todas las subredes 190.3.x desde el punto de vista de E1. La tabla IP de E1 será como en Tab. IV

subred de destino	Máscara	dirección del *next-hop*	Costo
…	…	…	…
190.3.0.0	255.255.0.0	190.3.254.1	n

Tab. IV: Tabla IP del ruteador E1

1.4.2 ARP y RARP

Los protocolos Resolución de Direcciones (*Address Resolution Protocol*, ARP) [7] y Resolución de Direcciones Inverso (*Reverse Address Resolution Protocol*, RARP) [8] son utilizados para obtener de manera automática las correspondencias entre las direcciones de nivel 3 y las direcciones de nivel 2 y viceversa. Esto es importante en las LAN donde es necesario crear una relación entre las direcciones IP y las direcciones MAC.

El protocolo ARP es usado cada vez que un host tiene que enviar un mensaje a un nodo sobre la misma red LAN de la cual solo conoce la dirección IP. Para tal fin, el host en cuestión envía en broadcast un mensaje que contiene sus propias direcciones MAC e IP y la dirección IP del host destino, esperando como respuesta la correspondiente dirección MAC.

El protocolo RARP es utilizado por los hosts que no tienen memoria de masa (llamada *diskless*) para obtener la correspondiente dirección IP en fase de autoarranque.

1.4.3 ICMP

Hemos dicho como los ruteadores hablan entre sí y con los hosts con los cuales están directamente conectados para monitorear las conexiones. Esto es realizado

mediante un protocolo para el intercambio de mensajes en la capa de red: el Internet Control Message Protocol (ICMP) [5] que define el formato de algunos mensajes como la señalización de hosts inalcanzables, la caducidad del TTL de un datagrama o solicitud de echo por parte de un host.

1.5 Sistemas Autónomos

Hasta aquí el enrutamiento TCP/IP ha sido descrito como jerárquico en dos niveles: un primer nivel al interior de la subred; un segundo nivel entre subredes administrado por los ruteadores IP mediante las tablas de enrutamiento.
Las subredes derivan de la subdivisión de una red.

Las redes son reagrupadas en Sistemas Autónomos (*Autonomous System,* AS), es decir en grupo de redes controladas y administradas por una sola autoridad.

Los AS son especificados por un número entero, unívoco a nivel mundial, asignado por la misma autoridad que concede las direcciones Internet.

Los ruteadores que envían los mensajes en el AS son llamados ruteadores internos (*interior* router*),* mientras los que envían mensajes entre AS diferentes son llamados ruteadores externos (*exterior* router*)*. Un ejemplo de interconexión entre dos AS es mostrado en la figura 1.8.

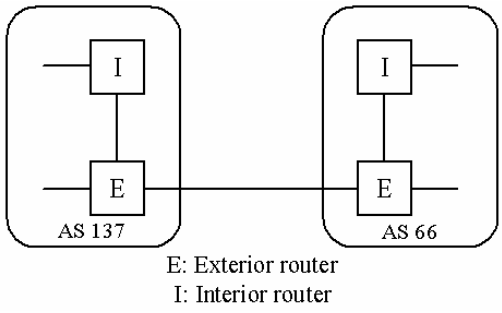


Fig. 1.8: Ejemplo de interconexión entre AS.

Los ruteadores internos pueden intercambiar información de enrutamiento mediante un *Interior Gateway Protocol* (IGP), mientras los ruteadores externos usan un *Exterior Gateway Protocol* (EGP). En el interior de un AS normalmente se usa el mismo IGP para todos los ruteadores.

En los párrafos siguientes serán ilustrados los principales protocolos de enrutamiento de tipo EGP e IGP.

1.6 Protocolos de enrutamiento

La arquitectura TCP/IP tiene una gran variedad de protocolos de enrutamiento. A continuación mostraremos aquellos que son de mayor interés para fines investigativos.

1.6.1 RIP

El Protocolo de Información de Enrutamiento (*Routing Information Protocol*, RIP) es un IGP originalmente desarrollado por la Xerox para su red XNS. Introducido en la arquitectura TCP/IP por la Universidad de Berkeley en 1982, definido en [9] en 1988 y actualizado en [10] en 1993.

RIP tuvo una gran difusión, sobretodo en la implementación de redes de computadoras, y es la base de otros protocolos de enrutamiento: Novell, 3Com, Banyan, etc.

Es un protocolo de tipo vector distancia (*distance vector)* donde cada ruteador envía su vector distancia (*distance vector)* a los ruteadores adyacentes, cada 30 segundos. Las tablas de enrutamiento memorizan solo un camino para cada destino.

La desventaja principal de RIP es que permite un número máximo de pasos (*hop)* igual a 15: cada destino más allá de 15 saltos es considerado inalcanzable.
Además RIP ignora la velocidad de las líneas, no permite definir costos u otras métricas, solo basa el enrutamiento en la minimización del número de pasos. En caso de modificaciones de la topología de la red, RIP es lento a converger.
Por estas razones RIP puede ser usado solo en redes de pequeñas dimensiones.

1.6.2 IGRP

El *Interior Gateway Routing Protocol* (IGRP) [11] es un IGP realizado por Cisco System Inc. en los años 80 para mejorar las limitaciones de RIP.

También es un protocolo de tipo vector distancia, pero con una métrica muy sofisticada. La elección del mejor camino es realizada por IGRP combinando vectores de métricas que contienen: retardo, banda, fiabilidad, longitud máxima del paquete y carga.

Además IGRP permite el enrutamiento de multicaminos (*multipath routing),* es decir la subdivisión del tráfico entre líneas paralelas. La carga es subdividida en función de las métricas asociadas a las líneas.

IGRP nació como protocolo propietario Cisco y hasta ahora es disponible solo en los ruteadores Cisco.

1.6.3 OSPF

El *Open Shortest Path First* (OSPF) es un IGP desarrollado para TCP/IP por el *Internet Engineering Task Force* (IETF). El grupo de trabajo fue constituido en 1988 con el fin de realizar un protocolo de tipo estado de línea (*link state)* para TCP/IP. Ha sido definido en [12] en 1991y redefinido en [13] en 1994.
OSPF se basa en el concepto de jerarquía. La raíz de la jerarquía es el AS que puede ser subdivido en áreas, cada una de las cuales contiene un grupo de redes contiguas.

El enrutamiento en el interior de un área es llamado intra-área, aquel entre áreas diferentes inter-área. Cada AS tiene un área llamada backbone identificada con 0.0.0.0 o con un 0.

El área backbone puede ser no contigua; en tal caso es necesario configurar los enlaces virtuales para garantizar la cohesión del backbone.

Los ruteadores OSPF son clasificados en 4 categorías no mutuamente excluyentes:

- *Ruteador Interno (Internal Router).* Un ruteador en el cual todas las redes directamente conectadas pertenecen a la misma área. Estos ruteadores utilizan una sola pareja del algoritmo OSPF. Los ruteadores que tienen solo interfaces en el backbone pertenecen a esta categoría.

- *Ruteador de Borde de Area (Area Border Router)*. Un ruteador que conecta muchas áreas. Estos ruteadores utilizan diferentes parejas del algoritmo OSPF: una pareja para cada área directamente conectada y una pareja para el backbone. Los ruteadores de borde de área reagrupan información de las áreas a ellas conectadas y las redistribuyen en el backbone. El backbone redistribuye a su vez esta información a otras áreas.
- *Ruteador Backbone*. Un ruteador que tienen una interfaz sobre el backbone. Esto incluye todos los ruteadores que se conectan a más de un área (*area border router*). Los ruteadores backbone que tienen todas las interfaces en el backbone son considerados ruteadores internos.
- *Ruteador Frontera de AS (AS boundary router)*. Un router que intercambia información de enrutamiento con otros ruteadores que pertenecen a otros AS. Esta clasificación es ortogonal a las otras precedentes: un ruteador de frontera de AS puede ser un ruteador interno o un ruteador de Borde de área.

La figura 1.9 muestra un ejemplo de AS TCP/IP subdividido en tres áreas OSPF y conectado a otro AS.

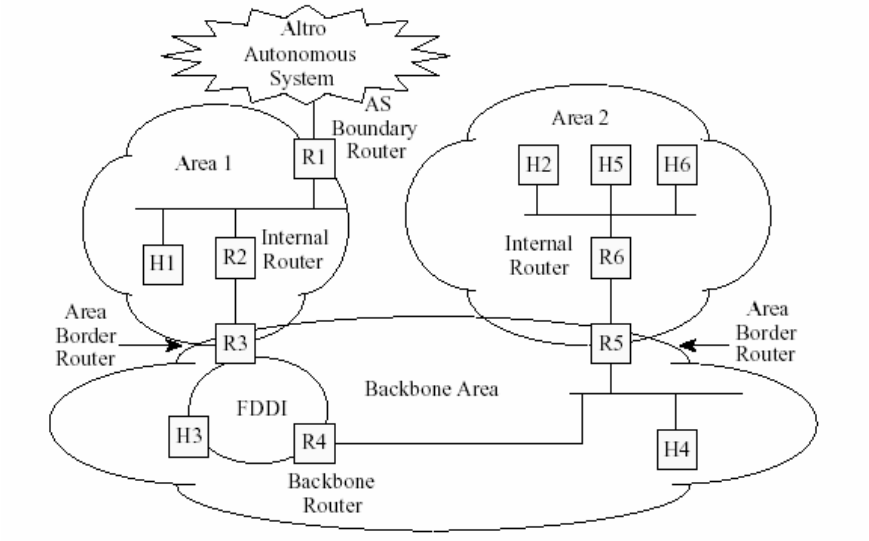


Fig. 1.9: Estructura interna de un AS.

OSPF es el protocolo que más promete para el enrutamiento de TCP/IP. Está disponible en ruteadores de todas las casas constructoras, puede administrar redes de grandes dimensiones y utiliza la tecnología de paquete de estado del enlace (*link state packet*).

1.6.4 EGP

El *Exterior Gateway Protocol* (EGP) es el primer protocolo de enrutamiento exterior, que fue ampliamente utilizado al interior de las redes Internet. Definido en [14] en 1984 y hoy es ampliamente disponible en todos los ruteadores, aunque ahora es considerado un protocolo obsoleto e Internet lo está sustituyendo con el BGP.

EGP es similar a un algoritmo vector distancia, que en lugar del concepto de costo especifica solo si el destino es alcanzable o no. Esto impide el funcionamiento en topologías malladas.

Existe el concepto de un sistema de núcleo (*core system)* formado de una interconexión de ruteador de núcleo (*core router)* (figura 1.10).

EGP genera paquetes de enrutamiento update que contienen información de redes alcanzables, es decir anuncia que ciertas redes pueden ser alcanzadas mediante determinados ruteadores. Los paquetes de enrutamiento update son enviados a los ruteadores vecinos en intervalos de tiempo regulares y alcanzan todos los ruteadores EGP. La información contenida es utilizada para construir las tablas de enrutamiento.

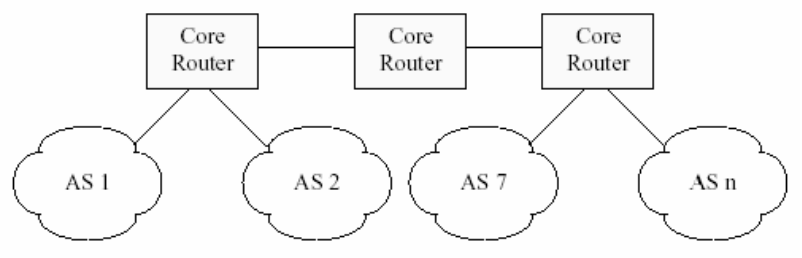


Fig. 1.10: Interconexión de un ruteador de núcleo.

Los límites de EGP son muchos y graves: EGP no tiene una métrica asociada a las líneas por lo tanto basa sus decisiones solo exclusivamente en alcanzar otras redes; EGP no admite la presencia de mallas en la topología, y todos los AS tienen que ser conectados en modo estrella a un sistema de núcleo; los paquetes de enrutamiento update pueden ser muy grandes.

1.6.5 BGP

El Protocolo de Frontera de Entrada (*Border Gateway Protocol*, BGP*)* es un protocolo exterior pensado para reemplazar al protocolo EGP en la actualidad obsoleto. El BGP ha sido definido en [15] en 1988, redefinido como BGP-2 [16] en 1990, como BGP-3 [17] en 1991 y como BGP-4 [18] en 1995.

Los ruteadores BGP se comunican utilizando un nivel de transporte fiable. El BGP es un algoritmo de tipo vector distancia, que transmite la secuencia de sistemas autónomos que tienen que atravesar para alcanzar el destino.

Cada ruteador calcula su enrutamiento favorito hacia un destino dado y lo comunica a los ruteadores BGP adyacentes mediante un vector distancia. La política con la cual se realiza el cálculo es configurable en cada ruteador BGP. Veremos con más detalles el funcionamiento de BGP en el capítulo 3.

Capítulo 2 : Arquitectura del Open Router

La arquitectura PC nace como una plataforma de propósito general y no es optimizada específicamente para las operaciones de redes.

Esta característica tiene un mayor impacto sobre las prestaciones del plano de datos, donde los aparatos ad hoc utilizan componentes ASIC, FPGA, Procesador de Red y buses internos específicos para administrar un nivel alto de paralelismo en la elaboración de los paquetes.

El criterio de selección de los componentes para una arquitectura de hardware en grado de soportar eficazmente las funcionalidades de Ruteador PC es generalmente complejo: los índices finales de prestación dependerán ya sea de la arquitectura de hardware utilizada, del software o de la integración de ambas.
Cuanto concierne a las arquitecturas de software, han sido descartados los Sistemas Operativos (SO) comerciales (como Windows o Unix) porque, al no permitir actuar sobre el código fuente y al no tener una documentación clara de su funcionamiento a bajo nivel, introduciría numerables variables sobre las cuales no se tendría control, haciendo imposible una configuración ad hoc y una optimización eficaz a nuestro objetivo.

El desarrollo y la optimización de arquitecturas de software de código abierto para el networking (establecimiento de una red) constituyen hoy en día un argumento de notable interés: de hecho, si por una parte la utilización de estas arquitecturas de software como servidores o firewall es en la actualidad consolidada, por la otra son siempre más comunes los casos en que las versiones de SO de código abierto (y en modo particular de Linux [19-20] y FreeBSD [21] que integran funciones de networking siempre más sofisticadas) vienen adaptadas y utilizadas como firmware para aparatos de red. Estos SO permiten actuar sobre el código fuente, modificarlo y adaptarlo a las propias necesidades.

En los próximos párrafos analizaremos, en lo específico, las arquitecturas hardware y software.

2.1 Arquitectura de Hardware

Como indicamos antes, la arquitectura Hardware PC-like en la actualidad no está en grado de facilitar mecanismos de I/O con un nivel de eficacia comparable a soluciones realizadas con hardware ad hoc. Utilizan únicamente componentes PC COTS, de hecho, dicha arquitectura permite realizar solo mecanismos de I/O de tipo centralizado, los cuales anticipan el transporte de los paquetes sobre el camino DMA desde las interfaces de red a la RAM (mediante el bus de I/O y el canal de memoria) y el sucesivo procesamiento sobre el(os) CPU, ocupando sea el canal de memoria que el bus de la parte delantera (*front side bus*, FSB). En figura 2.1 se ilustra un esquema del mecanismo arriba explicado.

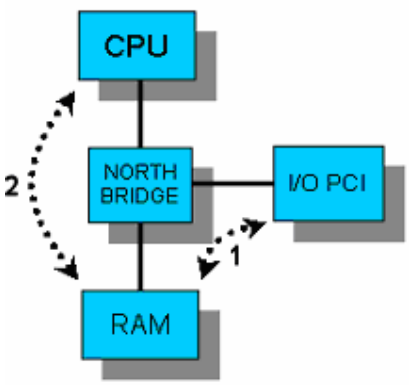


Fig. 2.1: Recorrido de un paquete en arquitectura PC

En detalle, estamos considerando una arquitectura en que las interfaces de red poseen una velocidad tal (1 Gbps *full duplex*) de hacer obsoleto el empleo del tradicional bus PCI a 32 bits de paralelismo y 33 MHz de frecuencia. Una solución a este problema viene dada de dos recientes evoluciones de este estándar, PCI-X y PCI-Express, las cuales garantizan una banda adecuada para soportar el tráfico proveniente desde las interfaces Gigabit Ethernet.

Con el fin de evitar eventuales cuellos de botella al interior de la arquitectura, es necesario dimensionar en relación a la velocidad máxima permitida por el bus de I/O, las velocidades de acceso a la memoria y de elaboración. Tales requisitos se traducen en la selección de procesadores con buenas prestaciones tanto de cálculo (por ejemplo frecuencia del reloj interno) como de I/O (frecuencia FSB) y de memoria con bajos tiempos de acceso y de transferencia.

En este contexto, nuestra selección recae sobre una arquitectura basada sobre el chipset ServerWorks GC-LE, cuya arquitectura es ilustrada en figura 2.2. El *chipset* está en grado de soportar un sistema basado sobre doble procesador Xeon con canal dual de memoria y bus PCI-C a 133 MHz por 64 bits de paralelismo.

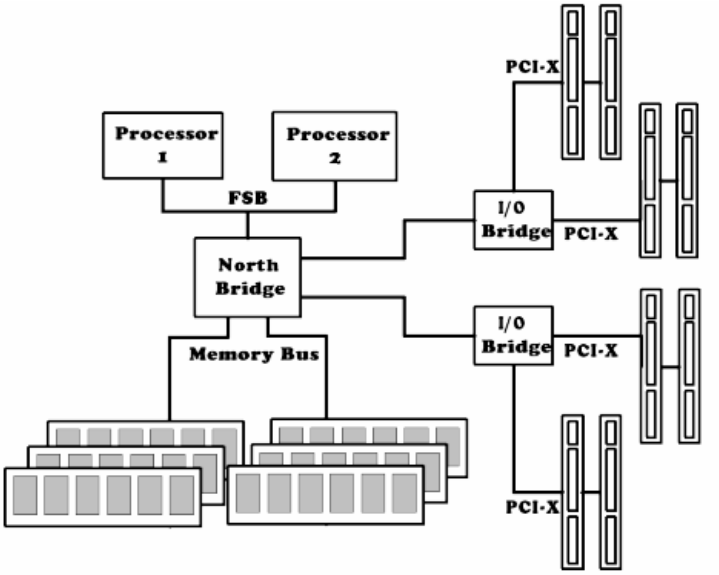


Fig. 2.2: Esquema de la mainboard Supermicro X5DL8-GC

El procesador Intel Xeon se basa sobre el núcleo del Pentium 4 con 603 pines que permite configuraciones multiprocesador con soporte Hyper-Threading. Deriva de la arquitectura NetBurst de Intel y es uno de los procesadores que mejor se prestan a arquitecturas server high-end gracias a la notable dimensión de las cache L2/L3. En particular en nuestras pruebas hemos usado dos procesadores Xeon con reloj de 2.4 GHz y 512 KB de cache.

El ancho de banda disponible para la memoria es la misma que el bus del sistema, está conectada a los procesadores mediante un canal de dos vías que dobla las prestaciones.

El bus entre el puente del norte (north bridge) y los puentes PCI tiene más banda (> 25 Gbps) que la suma de la banda de los 2 buses PCI-X conectados a cada I/O bridge.

2.1.1 Tarjetas de red

Las tarjetas de red son otro elemento crítico del sistema, en cuanto pueden condicionar el desempeño del OR. Como descrito en [22], las tarjetas presentes en el mercado tienen diferentes niveles de prestaciones máximas y un diverso nivel configurable. Después de haber realizado las debidas consideraciones nuestra selección recae sobre las tarjetas de red Gigabit Ethernet Intel Pro/1000 XT Server, basadas sobre el chipset Intel 8254x y dotadas de controladores PCI-X a 133 Mhz. Estas proporcionan, además, un amplio rango de configuración para varios parámetros como por ejemplo, la longitud de los buffer de recepción y transmisión, la unión de las interrupciones y otros aspectos importantes [23].

Además estas tarjetas son dotadas de un módulo de aceleración hardware de los datos de entrada, llamado Paquete de Prioridad Intel (Intel Priority *Packet*) [24], que soporta el tagging IEEE 802.1p y L3 IP ToS conforme a los estándares Differentiated Services CodePoint (DSCP) especificados en [25] y [26], y Legacy IP Precedence. A continuación veremos como esta última característica asume una gran importancia en el soporte efectivo del QdS.

En estas tarjetas, la unión de las interrupciones es implementada mediante parámetros pasados por los drivers, de manera que se pueda configurar la capacidad de la tarjeta según el tipo de tráfico, operación que se efectúa con mucha atención cuando el paso de parámetros errados puede influir sobre las prestaciones finales de la tarjeta: una unión de interrupciones muy alta (tasa de interrupción muy baja) causará una inundación del buffer, mientras una tasa de interrupción alta cargará excesivamente al sistema llevando a una inevitable pérdida de datos causada de la escasa disponibilidad de la CPU para las operaciones de cálculo.

Los controladores Ethernet Intel implementan tres métodos diferentes para moderar la tasa de interrupción:

- Un reloj absoluto, que inicia al recibir el primer paquete. Una interrupción se genera cuando finaliza el reloj. El objetivo es asegurar la moderación de las interrupciones en condiciones de tráfico pesado.
- Un reloj de paquete, que inicia al recibir cada paquete. Una interrupción se genera solo cuando finaliza este reloj. Visto que el reloj es reseteado al paso de cada paquete, no se garantiza la generación de una interrupción si el tráfico es alto. El objetivo en este caso, es de bajar la latencia en situaciones de baja carga.
- Un mecanismo de regulación de las interrupciones que pone un límite superior a la tasa de generación de las interrupciones por parte del controlador.

Mientras los dos primeros relojes trabajan independientemente para la recepción y transmisión de las tramas, la regulación puede ser usada para asegurarse que la tasa de las interrupciones no supere un valor dado, asimismo en caso de condiciones de tráfico fuertemente variable.

2.2 Arquitectura de Software

Entre las innumerables opciones se decidió utilizar el ambiente Linux. Esta selección ha sido mayormente sugerida por el amplio y fiable soporte que ofrece a las aplicaciones de red (server, firewall, etc.). Más detalladamente, hemos elegido la distribución Debian que, como se explica a continuación, permite obtener de manera simple y directa configuraciones avanzadas del sistema.

2.2.1 Kernel Linux

El kernel, en general, constituye el núcleo fundamental de un SO, en cuanto es el elemento que, trabaja directamente con el hardware, administra los recursos del sistema proporcionando a los procesos y a los servicios software una interfaz de alto perfil, que no debe por lo tanto, tener en consideración las peculiaridades de la arquitectura de hardware. En este sentido el kernel, proporcionando todas las utilidades base del sistema, resulta ser el elemento fundamental de la arquitectura de software caracterizando en parte del mismo modo a todas las aplicaciones que se apoyan en él.

En particular un kernel debe necesariamente cumplir dos tareas:

- Interactuar con los componentes hardware, administrando todos los elementos programables a bajo nivel incluidos en la arquitectura.
- Suministrar un ambiente de ejecución a las aplicaciones activas en el sistema.

Los kernel de los SO de nueva generación esconden a las aplicaciones activas de los usuarios todos los detalles de bajo nivel de la arquitectura hardware del PC, en donde los recursos pueden ser asignados solo sobre petición de las aplicaciones y mediante mediación del kernel mismo.

Linux en particular pertenece a la familia de SO Unix-like como BSD, desarrollado por la universidad Californiana de Berkeley [21], Solaris de Sun Microsystems [27], Mac OS X de Apple Computer [28] y otros. Ha sido desarrollado por Linus Torvalds en 1991 como SO para PC basado en el microprocesador 80386 de Intel. Torvalds, en el transcurso de estos años, ha trabajado activamente en la evolución de Linux, manteniéndolo actualizado para

las nuevas tecnologías emergentes y en continua evolución y coordinando el trabajo de muchos desarrolladores de software en todo el mundo. Hoy en día es un SO que ofrece soporte a casi todas las arquitecturas de hardware.

El Kernel posee una estructura monolítica, en cuanto todo el código fuente es compilado en un único programa independiente compuesto por diferentes componentes lógicos. A la época de su creación, el kernel monolítico era ya considerado obsoleto a favor de la emergente estructura microkernel donde el SO es dividido en varios componentes completamente separados, y el kernel administra un poco más de lo indispensable para la comunicación entre las varias aplicaciones. Este continuo intercambio de mensajes entre procesos hace al SO más lento pero el mundo académico tiende a apoyar la tecnología microkernel para los beneficios derivados de la modularidad de su estructura.

Estos SO son desarrollados sobre niveles independientes y esto permite una simple portabilidad sobre cada arquitectura de hardware, además tienden a tener una mejor gestión de la RAM en cuanto a los procesos de sistema no necesariamente pueden ser removidos de la memoria.

Para garantizar algunas de las ventajas teóricas de los sistemas microkernel, sin heredar también las desventajas, el kernel Linux ha introducido el uso de los "módulos". Los módulos son objetos en el que el código puede ser conectado (o desconectado) al kernel en runtime y son usados para implementar drivers de periferia o funcionalidades de los niveles superiores del kernel.
Una de las mayores ventajas de Linux es el hecho que no es un SO comercial: su código fuente, distribuido sobre licencia GNU[1], es "abierto" y disponible a cualquiera para posibles desarrollos.

[1] El proyecto GNU es coordinado por la *Free Software Foundation Inc.* [http://www.gnu.org]; su objetivo es aquel de implementar un SO libremente utilizable por cualquier persona. La disponibilidad del compilador GNU C ha sido fundamentalmente por el éxito de Linux.

Otra característica importante de Linux es que es un sistema multiusuario: un sistema en grado de administrar concurrente e independientemente varios procesos pertenecientes a diversos usuarios. "Concurrentemente" por lo que las aplicaciones son activas contemporáneamente y se disputan los recursos del sistema, mientras que "independientemente" se entiende que cada aplicación lleva adelante la propia tarea sin interferir o interactuar en algún modo (a menos que sea estrictamente necesario) con las otras aplicaciones.

La gran acogida a la filosofía de código abierto pone al kernel Linux en continua evolución, las nuevas actualizaciones se encuentran en la web oficial [29], mientras miles de aplicaciones de cada género son disponibles en la red. La dimensión del código fuente del kernel crece de manera exponencial, incorporando nuevos módulos, nuevo soporte de hardware y nuevas funcionalidades.

Para asegurar la disponibilidad de un kernel estable, y al mismo tiempo, dar la posibilidad de probar las innovaciones más recientes, Linux distingue entre kernel estable y kernel experimental en desarrollo mediante un simple sistema de numeración. Cada versión es caracterizada de un número de tres cifras (figura 2.3), las primeras dos cifras indican el número principal de la versión, si la segunda cifra es par indica que es una versión estable (llamada "*stable*") si es impar la versión esta en fase de prueba (llamada "*unstable*"), la última cifra indica el número del release al interior de las cuales son introducidas pequeñas actualizaciones y correcciones de pequeños bug.

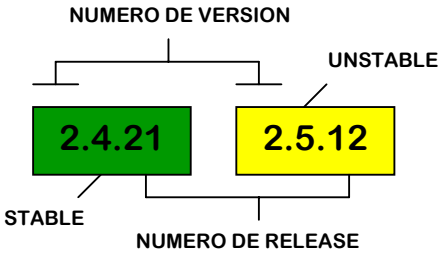


Fig. 2.3: Numeración de las versiones del kernel Linux

La flexibilidad de este kernel lo hace adecuado a todas las tecnologías emergentes hasta ser incorporado en una gran variedad de aparatos no solo de genero informático (electrodomésticos, video registradores digitales, teléfonos celulares, etc.)

Linux no se limita al kernel, está dotado de una gran variedad de aplicaciones y librerías, todo esto conforma la filosofía de código abierto. Como dijimos antes, tanto el kernel como la mayor parte de las aplicaciones, se pueden encontrar fácilmente en Internet, sin embargo en el curso de su historia Linux ha visto nacer y desarrollar varias "distribuciones", algunas de las cuales con soporte comercial, creadas inicialmente por grupos de investigadores con el objetivo de crear instrumentos de instalación y configuración del kernel y de los varios componentes. Las distribuciones más difundidas y conocidas son SuSE, Red Hat, Mandrake y Debian que se diferencian por la integración de los componentes, accesorios y por las herramientas de instalación y administración (rpm, apt…).

2.2.2 Debian GNU/Linux

Aunque todas las distribuciones se basan sobre el mismo kernel, cada distribución tiene su propia orientación específica y los instrumentos de gestión puestos a disposición pueden ser muy diferentes. Mandrake, por ejemplo, es una distribución orientada a proveer un SO con interfaz gráfica, la herramienta de instalación es simple y veloz pero deja pocas posibilidades de configurar el sistema según necesidades especificas.

La distribución mayormente utilizada en el ámbito académico para el desarrollo ad hoc es Debian en cuanto a la herramienta de instalación, aunque es más compleja respecto a las otras distribuciones, permite configurar cada mínima parte del sistema y escoger los componentes a instalar; también es importante el instrumento de instalación de los componentes adjuntos, apt, que permite instalar el SO desde la red, permitiendo tener siempre a disposición las versiones más actualizadas de los varios instrumentos de software.

El proyecto Debian [30] ha sido fundado oficialmente por Ian Murdock en 1993. En ese periodo, el concepto de “distribución” Linux era nuevo. La idea era de crear una distribución abierta. El proyecto Debian inicialmente estaba compuesto de un reducido grupo de desarrolladores de software y ha crecido de manera gradual hasta convertirse en una grande y bien organizada comunidad de desarrolladores y usuarios.

Para conseguir y mantener altos estándares de calidad, Debian tiene adoptado numerosas políticas y procedimientos para el empaquetamiento y la distribución del software.

Actualmente el proyecto Debian involucra a miles de usuarios, en constante contacto entre ellos mediante grupos de información e innumerables forum

dedicados en Internet, dando así un soporte completo y puntual de cada característica.

Por este motivo resulta una distribución óptima para construir un sistema, basado sobre el kernel Linux, que tenga características especificas y de hecho es el punto de partida para arquitecturas server y networking.

2.3 Ruteador Linux

Para poder entender plenamente la estructura nodal del sistema es necesario entender primero como la arquitectura Linux integra las funcionalidades requeridas a un ruteador IP. Hay dos conjuntos de funcionalidades a considerar: las funciones de soporte al reenvío de paquetes (operación de conmutación), y las funciones de soporte del plano de control (los protocolos de señalización como, por ejemplo, los protocolos de enrutamiento, los protocolos de control, etc).

En cuanto concierne al plano de control, al no tener alguna influencia sobre el perfomance del reenvío de paquetes, su selección no es crítica para nuestros fines. En tal caso, pueden usarse cualquiera de los instrumentos de código abierto que trabajan como aplicaciones o demonios en el espacio de usuario (*user space)* como Zebra [31], Quagga [32] y XORP [33-34].

El elemento crucial en la operación de enrutamiento de paquetes IP es el kernel, donde son realizadas todas las operaciones de los niveles de línea, red y transporte (figura 2.4). Para este motivo, con el fin de explicar los procedimientos de selección y optimización y para mejor comprensión de las consideraciones sobre las prestaciones del sistema, presentadas al final de este

trabajo, serán descritas en el detalle las varias operaciones del kernel en la cadena de reenvío.

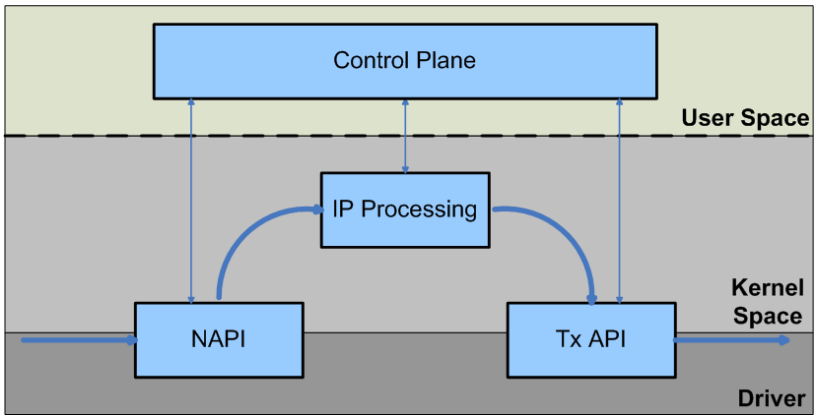


Fig. 2.4: Diagrama de bloques de la estructura software de un ruteador PC Linux.

En el curso de los últimos años, el soporte al networking integrado en el kernel Linux ha visto el acontecer de numerosos cambios estructurales y operaciones de perfeccionamiento, principalmente a lo que concierne el mecanismo de recepción de paquetes: partiendo de una simple arquitectura de interrupciones (usada hasta la versión 2.2 del kernel) se pasó a un mecanismo de interrupciones de software (llamado Softnet, usado hasta la versión 2.4.21) para llegar a un mecanismo de moderación de interrupciones (llamado NAPI, New API, adoptado a partir del kernel 2.4.22).

La arquitectura Sofnet, aunque mantiene una estructura basada sobre las interrupciones, mejora las prestaciones porque logra disminuir la sobrecarga computacional de la conmutación de contexto (*context switching)* retrasando la elaboración de los paquetes recibidos.

A pesar de estos mejoramientos, esta arquitectura resultó ser inadecuada a tráfico medio/alto. De hecho, en situaciones de tráfico alto en la entrada, el fenómeno de “interrupt livelock” (descrito en [35]) causa un gran deterioro de prestaciones. La arquitectura NAPI ha sido intencionalmente proyectada para aumentar la escalabilidad del sistema, de hecho puede administrar los requerimientos de las tarjetas de red con un mecanismo de moderación de las interrupciones que permite pasar de una gestión clásica a interrupciones a un sondeo (*polling)* de las interfaces.

Por este motivo, hemos decidido utilizar la ultima generación del kernel Linux, en lo especifico la versión 2.6 que, además del soporte del NAPI, posee nuevas dotaciones (por ejemplo la kernel preemptivness y un programador de complejidad O(1)).

El mecanismo de reenvío de todos los kernel Linux está constituido de una cadena de tres diferentes módulos: una “API de recepción” que administra la recepción de los paquetes (NAPI), un módulo que extrae la elaboración del nivel IP y, una “API de transmisión” que se ocupa de las operaciones de reenvío hacia las interfaces de salida.

Iniciando el análisis de la estructura del código de networking del kernel 2.6, tenemos que precisar que todo el código del kernel es estructurado sobre una condición de “zero-copy” [36]: para evitar inútiles y pesadas operaciones de transferencia en memoria de los paquetes, estos son dejados en las localizaciones de memoria usadas por los DMA-engine de las tarjetas de red en la entrada, mientras todas las operaciones sucesivas se realizan utilizando indicadores a los paquetes y a sus campos claves, llamados *sk_buff*.

Estos descriptores son compuestos sustancialmente de los indicadores a los diferentes campos de los encabezados de los paquetes asociados. Un descriptor

es colocado inmediatamente a la recepción de cada paquete y asociado en el subnivel IP, sucesivamente usado en cada operación de networking, y al final retirado cuando la tarjeta de red señala la transmisión realizada.

En las configuraciones con soporte SMP (Simmetric Multi-Processor), con kernel NAPI, para reducir el deterioramiento de las prestaciones consecuentes a la concurrencia de las CPU, la gestión de cada interfaz de red debe ser asignada estáticamente a una sola CPU para las operaciones de recepción y transmisión. Sin embargo, la asignación CPU-interfaz puede ser modificada dinámicamente: el módulo *cpu_affinity* por ejemplo, busca distribuir adecuadamente el tráfico a todas las CPU del sistema.

En breve, las operaciones estándar de reenvío se disponen a transferir los paquetes recibidos mediante el DMA-engine de la interfaz de red, de manera completamente transparente al sistema, y a depositar en áreas de memoria reservada del kernel, donde los paquetes esperan para la elaboración a nivel kernel. Mientras el kernel administra los paquetes, los asocia a los descriptores y, enseguida, los elabora uno a la vez. Durante la elaboración del nivel IP es elegida la interfaz de red de salida para cada paquete y su descriptor es transferido a una cola, llamada *qdics* de salida, en espera de la transmisión.

En los próximos párrafos sigue una descripción más detallada de los tres módulos fundamentales de la parte de código del kernel 2.6 que se ocupa del networking: el NAPI, el procesamiento IP y el API de transmisión (Tx API).

2.3.1 NAPI

Como dijimos antes, el NAPI es la nueva API de recepción de Linux, y ha sido desarrollado para obtener el nivel de escalabilidad necesario a soportar por las

tarjetas de red Gigabit Ethernet. Implementa un mecanismo adaptable que, mediante mitigaciones de las interrupciones, se comporta como el clásico Sofnet para tasas bajas de ingreso, mientras para tasas altas trabaja como un mecanismo de sondeo. Vamos ahora a analizar con mayor profundidad la estructura del NAPI.

Después de haber depositado en la cola circular (*ring buffer)* el primer paquete recibido, la tarjeta de red genera una interrupción, cuando una CPU lo recibe, llama a la rutina *next_rx_action*, la cual llama a su vez *netif_rx_schedule* y *__netif_rx_schedule*. Esta última en orden:

- Registra la identificación de la interfaz que ha generado la interrupción en una cola llamada *poll_list.*
- Programa una interrupción SW (*NET_RX_SOFTIRQ*) en lugar de servir enseguida la tarjeta solicitante, para retardar la conmutación de contexto.
- Deshabilita la generación de las interrupciones de hardware.

Todos los paquetes sucesivos, que son recibidos antes que el programador del kernel sirva la interrupción software, no causan la generación de interrupciones al SO, solo son movidas a una área de memoria reservada por el DMA-engine. Para conocer la dirección del área de memoria reservada, el DMA-engine usa una lista de descriptores disponibles.

Esta lista de descriptores es organizada como una cola circular, llamada *rx_ring*, en la memoria reservada del kernel. Si no hay suficiente espacio o descriptores a disposición, los paquetes son descartados.

El programador despierta al gestor de los *softirq (net_rx_action)* que mide la *poll-list* y por cada interfaz registrada:

- Es llamada la función *dev->poll* que procesa, utilizando algunas funciones del driver, un número de paquetes igual al valor mínimo entre el número de paquetes en el *rx_ring* y el parámetro *quota*.
- Por cada paquete recibido es solicitada la función *netif_receive_skb*
- Si el *rx_ring* de la tarjeta actualmente servida se libera, la respectiva identificación es removida de la *poll_list* y se habilita la generación de las interrupciones para dicha tarjeta.
- Si el *rx_ring* de la tarjeta servida no se libera, la *poll_list* se deja invariable, el procedimiento termina cuando no hay más identificaciones registradas en la *poll_list* o cuando se procesa un número de paquetes igual al parámetro *quota*. Es importante saber que el parámetro *quota* es el número máximo de paquetes que pueden ser procesados antes de realizar otras tareas.

2.3.2 Procesamiento IP

El código del nivel de Red IP está generalmente compuesto de cuatro segmentos importantes, que se ocupan respectivamente del proceso de recepción, enrutamiento, reenvío y de transmisión de los datagramas.

En particular, el proceso de recepción es realizado por dos funciones: *ip_rcv* que controla la consistencia del encabezado de los datagramas *(checksum),* e *ip_rcv_finish* que decide si el destino IP es válido y a donde enviar el datagrama (localmente o a la cadena de reenvío). Si el datagrama no es entregado localmente, es elaborado por el módulo de enrutamiento que está estructurado sobre un lookup de la dirección a dos niveles: primero se realiza un lookup rápido desde la función *ip_route_input* usando una cache de las direcciones de

destino más frecuentemente usadas y, en caso de pérdida en la *cache*, el lookup procede mediante la rutina *ip_route_input_slow* que provee a un control en toda la tabla IP.

A continuación, *ip_forward* disminuye en uno el TTL (*time to live*) en el encabezado del datagrama. La operación de reenvío es completada por la función *ip_forwarding_finish* que, en el caso hayan opciones IP, llama a la función *ip_forwarding_options* y luego pasa el *sk_buff* directamente al módulo de transmisión IP.

La primera función de este módulo (*ip_output*) administra la fragmentación del datagrama, si es necesario, por lo tanto llama la función *ip_output_finish* que concluye la elaboración del nivel IP.

2.3.3 Tx API

Otra parte importante del código de networking del kernel es el API de transmisión. Es muy simple y, a diferencia del API de recepción, no ha sufrido revisiones particularmente significativas en los últimos años. La Tx API, se basa sobre la funciones del driver que señalan a la interfaz de red la presencia de paquetes a transmitir o que reciben mensajes de elaboración realizada con éxito.

En particular, la elaboración del API de transmisión inicia llamando, para cada paquete de llegada desde el nivel IP, *dev_queue_xmit*: esta función encola el descriptor en una cola asociada a la tarjeta de red de salida, llamado *qdisc*.

Por lo tanto, cuando hay suficiente espacio en el *tx_ring* de la interfaz, los descriptores de la *qdisc* de salida son servidos por las funciones *qdisc_restart*. Para cada descriptor servido, se llama el método virtual *hard_start_xmit* que es

luego implementado por el driver de la periferia (por ejemplo, para las tarjetas Intel Gigabit tenemos *e1000_start_xmit*). Esta función añade el descriptor en el *tx_ring* y señala a la tarjeta de red que hay datos para transferir. A continuación, el *DMA-engine* de la interfaz de salida transfiere los paquetes de la aérea de memoria del kernel al buffer circular hardware de la tarjeta y los envía.

Cuando la transmisión del paquete es culminada con éxito, la tarjeta genera una interrupción hardware, en donde las rutinas programan una interrupción de software mediante la bandera *NEXT_TX_ACTION*. En fin, el programador despierta el *handler net_tx_action* que mueve los descriptores de los paquetes transmitidos en una cola, llamada *completition_queue* donde los descriptores son encolados periódicamente.

2.3.4 Configuración y optimización del Open Router

Como hemos descrito anteriormente y esquematizado en el anexo A, la estructura de networking del kernel Linux es bastante compleja y presenta muchos aspectos y parámetros que pueden ser finamente regulados para la optimización del sistema.

Para algunos parámetros, la identificación de valores óptimos puede ser derivada de consideraciones lógicas, pero para la mayor parte tales valores son encontrados empíricamente, porque no es posible obtenerlos desde la estructura del software y a menudo depende estrictamente de los componentes de hardware instalados. Por lo tanto, nuestras regulaciones han sido realizadas inicialmente identificando los parámetros cruciales sobre los cuales operar, luego encontrando valores óptimos según consideraciones lógicas y experimentales.

La arquitectura del kernel incluye solamente tres puntos en donde se ingresa los paquetes a una cola (ver Anexo A): las colas circulares de recepción, las *qdisc* de salida y las *completition queue*. Mientras esta última cola ha demostrado no ser un parámetro crítico para la optimización (contiene solamente los descriptores de los paquetes ya enviados), la óptima regulación de las otras dos colas es muy importante para maximizar las prestaciones.

De hecho, para evitar desperdicios de CPU, es preferible minimizar las pérdidas en la *qdisc* de salida, donde los paquetes ya han sido procesados por el nivel IP, y mantener estas pérdidas antes de la elaboración, en las colas de recepción. Esto se puede obtener fácilmente, por ejemplo, aumentando las dimensiones de las *qdisc* a un valor alrededor de los 20000 descriptores.

Muchos drivers de tarjetas de red, por ejemplo, permiten dimensionar las colas de recepción y la tasa máxima de generación de las interrupciones. Ambos parámetros tienen notable influencia en las prestaciones del NAPI. De hecho, si consideramos condiciones de tráfico medio/alto, el NAPI prácticamente trabaja procediendo al sondeo de las tarjetas de red, que se registran en la *poll_list* con la interrupción generada a la recepción del primer paquete; todos los paquetes, recibidos antes que las tarjetas de red entren en la secuencia de sondeo, son ingresados en las colas circulares de hardware.

Además, si bien es claramente deseable no limitar la tasa de las interrupciones de las tarjetas de red en los kernel NAPI, una amplia cola circular (*ring buffer)* permite reducir la pérdida de paquetes en presencia de tráfico a altas velocidades. Otro parámetro interesante de regular es el valor de la *quota*, que fija el número de paquetes que cada tarjeta de red puede elaborar con cada ciclo de sondeo.

La racionalización de la gestión de la memoria es otro aspecto importante, una parte considerable de los recursos disponibles es ocupada en el proceso de

asignación y reasignación de los descriptores de los paquetes. [37] propone una patch del kernel que permite reciclar los descriptores de los paquetes enviados con éxito: la idea es de ahorrar recursos del CPU durante las operaciones de recepción del NAPI reutilizando los descriptores de los paquetes. El uso de esta patch permite aumentar las prestaciones.

Resumiendo, nuestra versión optimizada de la imagen del kernel incluye el patch para el reciclaje de los descriptores, los drivers e1000 versión 5.2.39-k2 (para las tarjetas de red) han sido configurados con los siguientes parámetros de optimización:

- La dimensión de las colas circulares de recepción y transmisión ha sido planteada al máximo valor: 4096 descriptores;
- La generación de las interrupciones en recepción no ha sido limitada;
- La dimensión de la **qdisc** para todas las tarjetas de red ha sido planteada a 20000 descriptores;
- El parámetro *quota* del NAPI ha sido establecido a 23 descriptores;
- La frecuencia de reloj del programador ha sido fijada a 100 Hz.

Capítulo 3:

Protocolos de enrutamiento

Para transferir los paquetes entre una pareja de hosts, la capa de red tiene el deber de determinar el camino fuente-destino (path), sea que provea un servicio de tipo datagrama (en este caso diferentes paquetes entre una pareja de hosts pueden seguir caminos diferentes) o de tipo circuito virtual (todos los paquetes entre una determinada fuente y un determinado destino seguirán el mismo camino). Este es el deber del protocolo de enrutamiento de la capa de red.

Típicamente cada host terminal conectado a la red Internet utiliza un ruteador de referencia, llamado ruteador por defecto (también conocido como first hop router), que tiene el deber de recoger el tráfico generado por los hosts a él conectados y enviarlo oportunamente hacia otros ruteadores IP. El problema de enrutar un paquete desde la terminal fuente hacia la terminal destino claramente se reduce al problema de enrutar el paquete desde el ruteador fuente hacia el ruteador destino.

El núcleo de cada uno de los protocolos de enrutamiento es el algoritmo que tiene la tarea de determinar el camino de los paquetes en la red.

El objetivo de un algoritmo de enrutamiento es simple: dado un grupo de ruteadores, conectados mediante enlaces, un algoritmo de enrutamiento encuentra un "buen" camino (a "costo mínimo") desde la fuente hacia el destino.

Los algoritmos de enrutamiento pueden ser divididos en tres categorías:

- *Algoritmos aislados*, en el que cada ruteador calcula de manera independiente las propias tablas de enrutamiento sin intercambiar información con otros ruteadores; puede ser realizado de manera sencilla pero ineficiente, por ejemplo cuando un ruteador recibe un paquete desde una interfaz lo envía sobre una libre.
- *Algoritmos centralizados*, contrariamente a la precedente categoría, en este caso existe un nodo central que recibe información de todos los ruteadores y conoce de esta manera la topología de la red; calcula las tablas de enrutamiento que distribuye a los ruteadores correspondientes;
- *Algoritmos distribuidos*, representan una selección de compromiso entre las dos categorías precedentes; las tablas de enrutamiento son calculadas por cada ruteador.

El término distribuido es el más usado en la redes datagrama, y es realizado en dos variantes:

- Protocolos Vector Distancia
    - Tradicionales: utilizan el algoritmo de Bellman-Ford
    - Avanzados: utilizan el algoritmo Distributed Update Algorithm (DUAL)
- Protocolos Estado de Enlace
    - Utilizan el algoritmo de Dijkstra

	Algoritmo	Protocolo
Link State	Dijkstra SPF	OSPF
Distance Vector	Bellman-Ford, DUAL	EGP, RIP, IGRP, BGP(Path Vector) EIGRP

OSPF : Open Shortest Path First

3.1 Principio de funcionamiento

El escenario en el que OSPF actúa es el de Sistemas Autónomos (AS) en tecnología TCP/IP. El AS puede estar compuesto por diferentes redes.

Las redes que forman parte de Internet pueden diferir en arquitectura, servicios o en prestaciones. Por este motivo el internetworking prevé un enfoque de mayor esfuerzo. Consideremos como primera aproximación que Internet está constituida por un conjunto de redes realizadas en diferentes tecnologías entre ellas conectadas mediante un ruteador.

La tarea principal del ruteador es de efectuar el correcto enrutamiento salto por salto (hop-by-hop) de los datagramas IP en la red, hasta su destino final. Cada paquete IP es enviado separadamente de los otros, según una tabla de reenvío que a cada red del AS asocia una interfaz sobre el cual el paquete debe ser retransmitido para alcanzarla.

Para este fin los ruteadores se comunican entre ellos respetando las especificaciones del protocolo de enrutamiento, globalmente establecido a nivel del AS, con el objetivo de obtener información útil acerca de la topología del AS,

es decir, de la red y de los ruteadores que la componen y el modo en el que se interconectan entre ellos. Esta información es utilizada por cada uno de los ruteadores que, individualmente e independientemente de los otros calcula la propia tabla de enrutamiento; ésta no hace otra cosa que asociar a cada posible destino del AS un siguiente paso.

A partir de la tabla de enrutamiento, es elaborada la tabla de reenvío, que es la que utiliza el ruteador para el envío de paquetes: la tabla de reenvío, de hecho, asocia al destino del paquete la interfaz y una eventual dirección de nivel 2 por el cual el paquete IP debe ser efectivamente transmitido.

Dada la topología del AS, constituida por las redes y los ruteadores, es necesario fijar una métrica, es decir un "costo" para atravesar una red. La métrica es fijada convencionalmente según criterios arbitrarios elegidos al interno del AS que representa una medida de la "preferencia" de la conexión: a un menor costo corresponde un enlace que es preferible atravesar con respecto a otro de mayor costo.

La tabla de enrutamiento debe ser compilada no solo en modo de que cada paquete IP alcance el propio destino, sino que también lo alcance mediante el camino de costo mínimo.

El protocolo de enrutamiento OSPF regula por lo tanto la comunicación entre los ruteadores y de manera más general las modalidades con las que logra hacer llegar a todos, la información necesaria para la compilación de las tablas de enrutamiento y reenvío, que permiten el enrutamiento de los paquetes a lo largo del camino de costo mínimo.

El protocolo de enrutamiento OSPF, que debe ser instalado sobre todos los ruteadores de la red, actúa en modo automático y dinámico, es decir requiere lo

menos posible la intervención por parte de los operadores, que deberán limitarse a la configuración inicial de cada ruteador, los cuales, una vez configurados saben reaccionar a cambios eventuales de la topología de red y construir de manera rápida y veloz una nueva y correcta tabla de enrutamiento.

Para hacer esto OSPF se basa en el algoritmo estado de enlace (*link*-state) en colaboración con algoritmos de tipo SPF (*Shortest Path First*).

El primero permite la distribución entre ruteadores de la información necesaria a la determinación de la topología de red, el segundo hace posible a partir de la información recogida la determinación del camino más corto.

El algoritmo estado del enlace está basado sobre algunos principios fundamentales:

- Cada nodo de la red que está implicado en el enrutamiento (ruteador) sabe reconocer cuales son sus conexiones directas, sean estos otros ruteadores o redes, y la métrica de la interfaz que los conecta. Cada ruteador establece una adyacencia con otros, a los cuales se conecta directamente (vecinos) mediante una red.

- Cada ruteador estructura la información del estado de sus propias adyacencias y de las redes a las cuales se conecta directamente al interno de particulares paquetes, los LSP (*Link State Packet*) los cuales alcanzan mediante la red, todos los otros ruteadores, que deben memorizar. Estos paquetes tienen que ser retransmitidos y actualizados periódicamente, o cada vez que haya un cambio en la topología de red. El proceso con el cual los LSP son distribuidos en la red es un proceso muy delicado que se llama Realible Flooding.

- El conjunto de todos los LSP transmitidos por todos los ruteadores constituye la Base de Datos del Estado de Enlace (LS Database). Está presente en todos los ruteadores del AS, y debe ser el mismo en cada uno, en otras palabras la base de datos (*database)* deber ser sincronizado, porque consiente a cada ruteador no solo de conocer las redes y los ruteadores de la red con los cuales no está directamente conectado, sino también de reconstruir completamente la topología de la red.

A partir de la base de datos, cada ruteador de la red está en grado de determinar, autónomamente, mediante un algoritmo SPF, el camino de costo mínimo hacia todas las redes del AS, y de construir oportunamente la tabla de enrutamiento y la tabla de reenvío, esto permitirá a los paquetes IP alcanzar su destino mediante el camino de costo mínimo.

3.2 Paquete OSPF

Los ruteadores OSPF se comunican entre ellos para cumplir una doble función:

- Destacar la presencia de ruteadores adyacentes, con el objetivo de compilar el propio estado del enlace.

- Distribuir a los ruteadores del AS la información relativa al propio estado del enlace.

Todas las comunicaciones entre ruteadores se realizan mediante el intercambio de paquetes OSPF, que son transmitidos en el Internet junto al tráfico IP de usuario. Los paquetes OSPF son encapsulados directamente en el payload de los

paquetes IP, sin la ayuda de protocolos de nivel 4 como TCP o UDP. Porque OSPF provee por sí mismo el control de la transmisión mediante un mecanismo de reconocimiento (acknowledgment), y por lo tanto no requiere el soporte de un protocolo de transporte.

Un paquete OSPF es intercambiado solamente entre ruteadores adyacentes. Por este motivo el TTL de un paquete IP que transporta un mensaje OSPF es siempre 1.

El campo *Protocol* (Protocolo) del paquete IP siempre es 89. El campo *Source Address* (Dirección fuente) contiene la dirección IP de la interfaz del ruteador que trasmite el paquete.

El campo *Destination Address* (Dirección destino) puede contener la dirección IP unicast de la interfaz del ruteador al cual el paquete es destinado o, a redes que soportan el broadcast, una dirección IP Multicast si el paquete es destinado a más ruteadores.

Todos los paquetes OSPF inician con un encabezado de 24 bytes, representado en figura 3.1 y del cual describiremos las funciones de cada campo informativo:

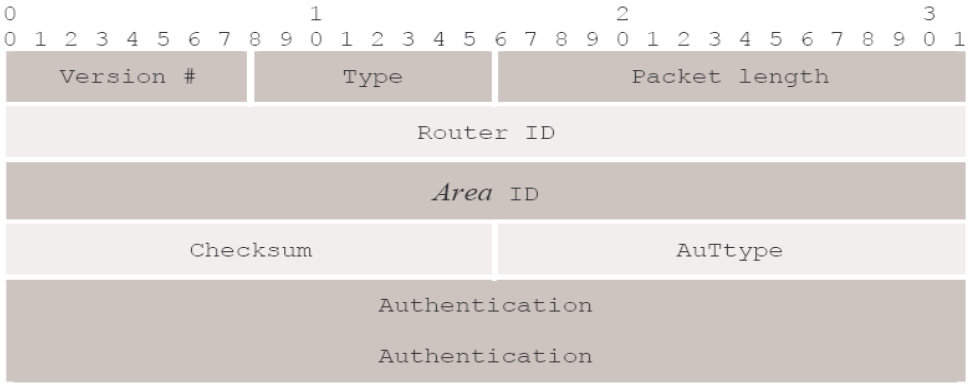


Fig. 3.1: Estructura del encabezado del paquete OSPF

Version: Hace referencia a la versión OPSF. En este documento se analiza OSPF versión 2.

Packet type: OSPF versión 2 prevé 5 tipos de mensajes que serán descritos a continuación

1. Hello
2. Descripción de Base de Datos
3. Requerimiento de Estado del Enlace
4. Actualización de Estado del Enlace
5. Reconocimiento de Estado del Enlace

Packet lenght: Longitud del paquete en bytes, incluyendo el encabezado.

Router ID: Identificación de 32 bits que individua el ruteador que ha transmitido el paquete.

Area ID: El área a la que el paquete OSPF hace referencia. Como el paquete OSPF es transmitido solo a ruteadores adyacentes, el contenido de este campo indica el área de pertenencia de la red que transporta el paquete.

Checksum: Es el resultado del algoritmo de control de tipo IP checksum que incluye en la cómputo todo el contenido del paquete OSPF, incluido el encabezado, y exceptuando el contenido del campo *Authentication Data*.

Los campos Aut Type y Authentication hacen referencia a los mecanismos de autenticación que hemos indicado. El mecanismo de autenticación es planteado a nivel de red. Todas las interfaces del ruteador que se conectan a la misma red deben evidentemente ajustarse al mismo método. El ruteador que reciba un paquete OSPF del cual la autenticación es errada, debe desecharlo sin elaborarlo.

El campo Aut Type hace referencia al tipo de autenticación utilizada. En OPSFv2 hay tres posibles modalidades de autenticación:

1. Null_authentication: no es necesario que los paquetes en la interfaz sean verificados. El campo Authentication debe ser completado con una máscara de 0. No ofrece ninguna garantía.

2. Simple Password: el campo Authentication debe contener una contraseña clara, igual en cada paquete, que es planteada a nivel de red. Protege esencialmente contra la activación de ruteadores que se conectan a la red sin autorización.

3. Crypthographic Authentication: el campo Authentication contiene un mensaje cifrado que cambia en cada paquete según una llave secreta programada únicamente sobre los ruteadores autorizados. Es la más eficaz porque protege tentativos de sabotaje o de piratería informática.

3.2.1 Anuncio de Estado del Enlace

Antes de proceder a la descripción detallada de las cincos topologías de paquete OSPF, es necesario introducir uno de los elementos fundamentales en la comunicación y en la estructuración de la información prevista del protocolo OSPF, el LSA (*Link State Advertisement*).

Hasta ahora hemos hablado de los LSP, o bien de los paquetes OSPF que contienen información sobre la topología de red. En OSPF la información de *link*-state, es decir el estado de las adyacencias de cada ruteador, son estructuras

en los LSA, que permiten la memorización en la Base de Datos LS, como la transmisión de paquetes OSPF.

Los LSA (figura 3.2) son generados por cada ruteador para describir el propio estado del enlace (link state, LS) y son distribuidos a los otros ruteadores para que sean memorizados con el objetivo de constituir la base de datos LS, del cual mediante los algoritmos SPF es posible reconstruir la topología de red. Se puede decir genéricamente que los LSA son los "ladrillos" que constituyen la base de datos LS, y por esto, mientras el paquete OSPF es transmitido solamente al ruteador adyacente, los LSA son distribuidos a más de un ruteador del AS mediante el mecanismo de *Realible Flooding*.

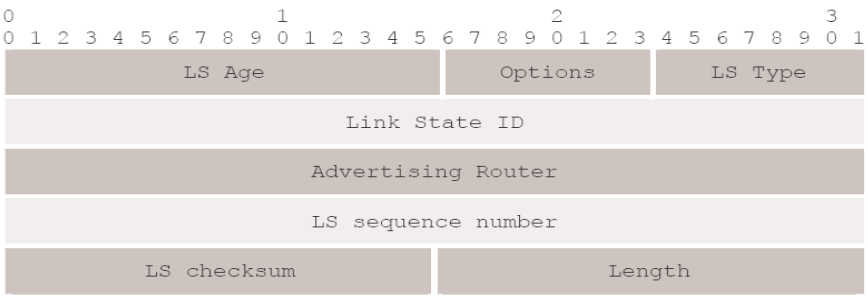


Fig. 3.2: Estructura del encabezado LSA

Los LSA son transmitidos al interno de algunas de los cinco tipos de paquete OSPF, cada uno de los cuales puede contener más de un LSA. Cada ruteador genera al menos un tipo de LSA para describir el propio LS, pero algunos pueden generar más de uno. El LSA es un paquete en el "paquete", en cuanto cada LSA está constituido por un encabezado, común en todos los LSA, y por otros campos informativos adjuntos que difieren según el tipo del LSA.
Existen cinco tipos de LSA:

1. Router LSA
2. Network *LSA*

3. Summary LSA
4. AS boundary *Router* LSA
5. AS external LSA

De estos, los dos últimos contienen información relativa al alcance de LIS (*Logical IP subnet*, redes en OSPF) externas al AS, y por este motivo no serán considerados en los siguientes puntos.

- **Router LSA**

  Cada ruteador genera uno o más Router LSA, en donde está descrita la información de estado de enlace bajo la forma de una lista de conexiones. Cada conexión puede hacer referencia al enlace de una red stub, o bien a una red de tránsito junto a la instauración de una adyacencia con otro ruteador.

  Los ruteadores internos, que se conectan a una sola área, generan un único Router LSA que transmiten a todas las interfaces propias. Los ABR, en cambio generan un Router LSA por cada área a la que se conectan, cada uno contiene la lista de las conexiones a aquella área; cada uno es distribuido solo sobre las interfaces que pertenecen al área al que hace referencia. Como todos los LSA, el Router LSA está constituido por el encabezado, donde el campo LS type está planteado a 1 y el LS ID y Advertising Router contienen ambos la identificación del ruteador que lo ha generado.

- **Network LSA**

  Los Network LSA son generados exclusivamente por los Ruteadores Designados (*Designated Router, DR)* de redes broadcast. Un ruteador que cubra el papel de DR relativamente a más redes, genera un Network LSA por

cada una de ellas. Naturalmente un ruteador que cubra el rol de DR genera también el propio router LSA. El DR es el único que anuncia la adyacencia con la LIS. Los Network LSA desarrollan exactamente este papel, anunciando el alcance por parte del DR sea de la LIS sea de todos los ruteadores que a ella se conectan.

Al igual que los Router LSA, los Network LSA deben ser distribuidos solo al interno del área a la que la LIS pertenece.

- **Summary LSA**

  Los Summary LSA son generados exclusivamente por los ABR con el objetivo de señalar al interno de una área el alcance de destinos que son externos al área misma. Los LSA de tipo 3 en particular anuncian el avance de LIS. Por cada área a la que el ABR se conecta, genera un Summary LSA por cada LIS que es capaz de alcanzar y que es externa al área en cuestión. Los Summary LSA también son distribuidos entre ruteadores de una sola área.

3.2.2 Paquete Hello

Los Paquetes Hello (*Hello Packet*, HP) son paquetes OSPF de tipo 1. Son transmitidos periódicamente por los ruteadores (con un periodo en el orden de segundos) a las propias interfaces y sirven para individuar eventuales ruteadores adyacentes y establecer una comunicación bidireccional, con el objetivo de instaurar una adyacencia (párrafo. 3.3.1 Protocolo Paquete Hello). Son el primer instrumento con el cual los ruteadores establecen una conexión con otro ruteador.

Además del encabezado típico de cada paquete OSPF, los HP contienen también otros campos informativos estructurados como en figura 3.3:

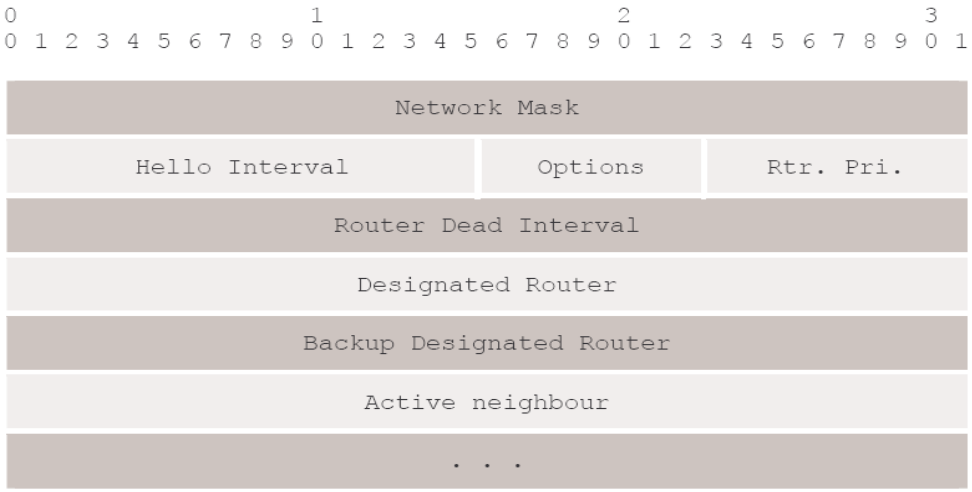


Fig. 3.3: Cuerpo del paquete Hello

Network mask: Como ya hemos visto, OSPFv2 soporta el mecanismo de subnetting. Con el objetivo de establecer una adyacencia dos ruteadores deben concordar sobre la máscara de red relativa a la LIS mediante la cual se conectan. Junto a la dirección IP de la interfaz que transmite, este campo permite la identificación completa de la LIS a la cual el ruteador se conecta, mediante el binomio Dirección IP/máscara de red.

Hello Interval: indica el periodo de tiempo, en segundos, con el cual el ruteador envía los propios HP sobre esa interfaz. Valores típicos están en el orden de pocas decenas de segundos.

Rt prt: Prioridad de Ruteador. Contiene el valor de prioridad utilizado en la elección del DR. Un valor igual a 0 indica que el ruteador es elegible.

Router Dead Interval: Es el número de segundos desde la recepción del último HP después de los cuales un ruteador se puede considerar no en función. Debe ser mayor al valor del campo Hello Interval (típicamente 3 o 4 veces mayor).

Designated Router: contiene la identificación del que actualmente es considerado el DR. En este campo el DR es identificado con la dirección IP de la interfaz con la que se conecta a la LIS. La identificación 0.0.0.0 indica que no hay un DR.

Backup DR: como el campo precedente pero identifica el DR de respaldo (BDR).

Neighbours: es la lista de las identificaciones de todos los ruteadores que están actualmente conectados a la LIS, es decir contiene las identificaciones de todos los ruteadores de los que se ha recibido correctamente los HP dentro del tiempo del Router Dead Interval.

3.2.3 Paquete de Descripción de Base de Datos

Los DDP (Database Description Packet) son paquetes OSPF de tipo 2. Estos paquetes son utilizados entre dos ruteadores que han establecido una comunicación bidireccional con el objetivo de efectuar la sincronización de las propias Base de Datos LS (3.3.3 Sincronización de la Base de Datos del Estado del Enlace).

Este mecanismo junto al Realible Flooding garantiza la sincronización de las Base de Datos LS a nivel global. OSPF prevé la instauración de una relación amo/esclavo entre dos ruteadores que se comunican, con el objetivo de garantizar una transferencia fiable de la información.

Analicemos ahora el contenido y la estructura (figura 3.4) del DDP:

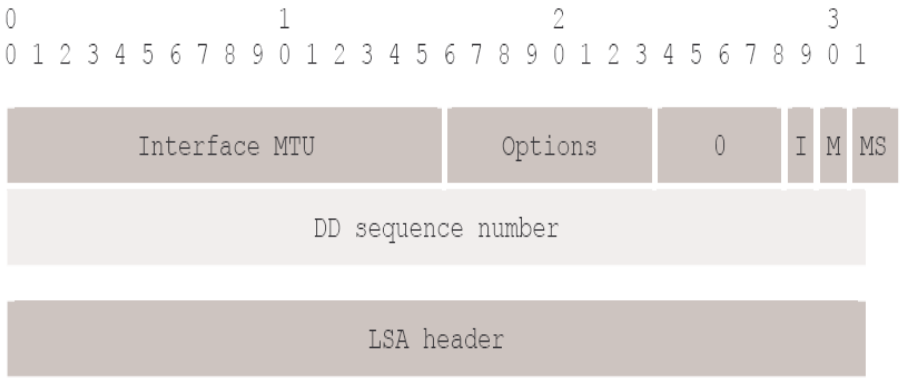


Fig. 3.4: Cuerpo del paquete de descripción de base de datos

MTU: contiene la MTU (Maximun Transfer Unit) relativa a la red mediante la cual los ruteadores se están comunicando, es decir la dimensión máxima (en byte) del datagrama IP que puede ser transmitido sobre la red sin ser fragmentado. Si la MTU es tal que un solo DDP no pueda contener toda la información de LIS es necesario transmitir más de uno.

I-bit: initial bit. Puesto a 1 en el primer DDP a ser transmitido.
M-bit: more bit. Puesto a 1 si deben ser transmitidos DDP adicionales, es decir que el ruteador no ha terminado de transferir todo el propio LS Database.
MS-bit: master-bit. Puesto a 1 si el ruteador que ha transmitido el DDP tiene el papel de maestro durante la comunicación.
DD Sequence Number: es utilizado para la numeración secuencial de los DDP transmitidos.

A estos campos informativos sigue una secuencia de encabezado LSA elegidos entre aquellos presentes en la Base de Datos LS del ruteador que está transmitiendo. Durante la sincronización de las Bases de Datos LS cada ruteador debe enviar al otro ruteador un "índice" de todos los LSA que tiene en posesión. Una vez aprendido el contenido de la Base de Datos LS cada uno puede requerir

(mediante la transmisión de un Paquete de Requerimiento de Estado del Enlace) las copias completas de los LSA de cual no se tiene posesión.

3.2.4 Paquete de Requerimiento de Estado del Enlace

Los LSRP (*Link State Request Packet*) son paquetes OSPF de tipo 3. En fase de sincronización de las Bases de Datos LS, o en momentos sucesivos, un ruteador puede necesitar una copia de un LSA para insertar en la propia Base de Datos LS.

Esto puede suceder porque el ruteador ha aprendido la existencia del LSRP en la fase de sincronización con otro ruteador, pero también porque ha encontrado al interno de la propia Base de Datos LS uno o más LSA caducados, es decir generados en un tiempo mayor del Max Age. Esto es fácilmente verificable consultando el campo LS Age de cada LSA. Los LSA caducados pueden llevar información obsoleta, es decir que se refieren a un estado pasado de la red que no corresponde más a la actual.

Es necesario por lo tanto por parte del ruteador renovar la propia Base de Datos LS sin atender el próximo Flooding, para evitar errores en la compilación de la tabla de enrutamiento.

En todas estas situaciones el ruteador utiliza un LSRP para requerir a uno de los ruteadores adyacentes una copia del LSA.

Por cada LSA que el ruteador necesita una copia, es insertado en el LSRP los campos LS type, LS ID y Advertising Router. Estos identifican unívocamente el LSA que es requerido. El ruteador que recibe la solicitud responde con un paquete de actualización de estado del enlace (*Link State Update Packet)* incluyendo una copia de los LSA requeridos.

3.2.5 Paquete de Actualización de Estado del Enlace

Los LSUP (*Link State Update Packet*) son paquetes OSPF de tipo 4. Son paquetes OSPF destinados al transporte de los LSA, sea en el ámbito del proceso de *Realible Flooding*, pero también en el ámbito de la sincronización de las Bases de Datos LS entre ruteadores adyacentes, en cuanto pueden servir a la transferencia de LSA requeridos mediante un LSRP.

Es preciso recordar que los LSUP, como todos los paquetes OSPF, son transmitidos solamente a ruteadores adyacentes.

La transmisión de un LSUP prevé siempre la retransmisión de un paquete de reconocimiento de estado del enlace (*Link State Acknowledgment Packet)* por parte del ruteador receptor.

La estructura de un LSUP es sencilla y es mostrada en figura 3.5.

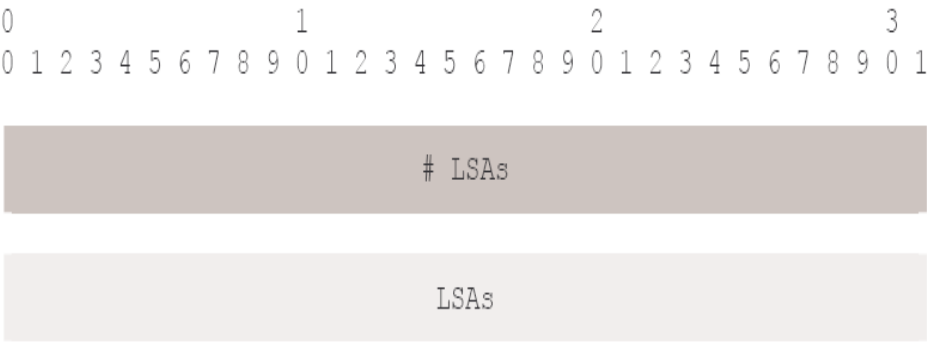


Fig. 3.5: Cuerpo del paquete de actualización de estado del enlace

El campo LSA# contiene el número de LSA que el paquete trasporta.

3.2.6 Paquete de Reconocimiento de Estado del Enlace

Los LSAP (*Link State Acknowledgment Packet*) son paquetes OSPF de tipo 5. Como dijimos antes OSPF prevé el envío de un reconocimiento por la correcta recepción de cada LSA. Los LSA son utilizados para confirmar la recepción de los LSA transmitidos a un ruteador adyacente mediante LSUP.

Un solo LSA puede contener en realidad el reconocimiento relativo a más de un LSA; cada LSA es confirmado simplemente insertando el encabezado al interno del LSAP.

3.3 Adyacencias

Hemos descrito como el protocolo OSPF anuncia que dos ruteadores, conectados directamente por una red, deben estar en grado de comunicarse, sea para establecer una adyacencia que debe ser incluida en los LSA de cada uno, sea para garantizar el proceso de distribución global de los LSA, en el ámbito del Realible Flooding.

El evolucionar de dichas comunicaciones implica diferentes pasos, desde el interés de la adyacencia a nivel físico, hasta la designación de los DR en las redes broadcast, pasando desde la sincronización de las Base de Datos LS, hasta la completa instauración de la adyacencia, que puede a este punto ser incluida en el LS de cada uno de los dos ruteadores a ser utilizada para la distribución de los LSA.

OSPF regula las diferentes fases, definiendo diferentes estados, según una estructura que la RFC 2328 llama Neighbours Data Structure (NDS). La NDS es

señalada por diferentes estados, que miden la evolución de la comunicación entre dos ruteadores directamente conectados, desde el momento en que los ruteadores se conectan físicamente, hasta que la adyacencia se ha establecido, y dicha conexión entra a formar parte de la Base de Datos LS, con el objetivo de ser utilizado para el enrutamiento de los datagramas IP. La figura 3.6 ejemplifica la evolución de la NDS.

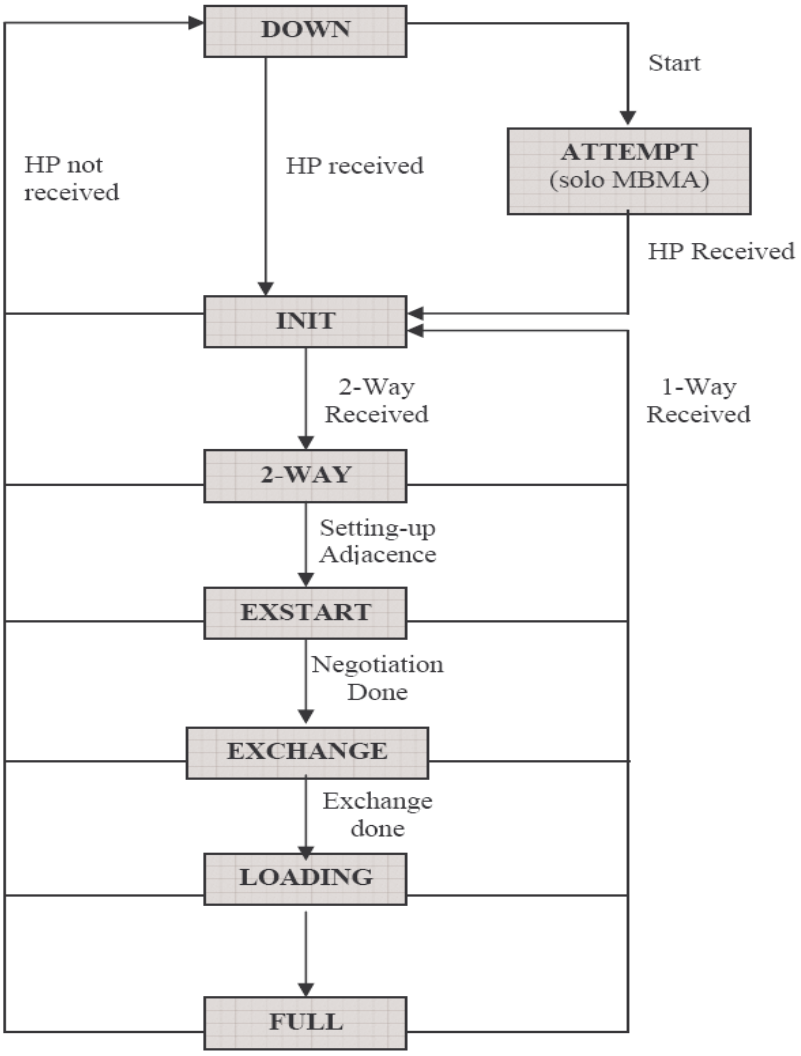


Fig. 3.6: Los estados de la NDS

3.3.1 Protocolo Paquete Hello

El primer instrumento que permite la comunicación entre ruteadores adyacentes son los paquetes hello (HP). Los HP son, como ya vimos, paquetes OSPF de tipo 1.

Cada ruteador transmite un HP periódicamente, en intervalos de pocos segundos, sobre las propias interfaces activas, para señalar a los ruteadores adyacentes su presencia. El estado Down es el estado inicial, en que el ruteador no ha recibido todavía algún HP del ruteador vecino.

El estado Attempt es un estado válido solo si los ruteadores se conectan en modalidad NBMA. En este caso cada uno posee una lista preconfigurada de todos los posibles ruteadores adyacentes e intenta ponerse en contacto enviando HP sobre las propias interfaces; en este estado cada ruteador envía periódicamente HP al ruteador vecino, pero no recibe respuesta.

Los HP contienen además de la identificación de 32 bits del ruteador que lo ha generado y el área a la que hace referencia, otra información importante para la instauración de una comunicación correcta entre los ruteadores.

Los campos Hello Interval y Dead Time Interval permiten a los dos ruteadores que formarán una adyacencia, sincronizar los relojes que regulan la transmisión de los HP. Entre otra información los HP contienen también una lista de los ruteadores reconocidos como vecinos. Esto permite a cada ruteador verificar la correcta instauración de una comunicación bidireccional.

Cuando un ruteador recibe sobre una de las propias interfaces activas un HP generado por otro ruteador la NDS entra en el estado Init: el ruteador ha recibido un HP generado por el vecino pero la comunicación bidireccional todavía no está

establecida, por lo que no aparece el Router ID del receptor en el campo *neighbours router*. El ruteador incluye en el campo *neighbour router* de los propios HP el ID del ruteador del que apenas ha conocido su existencia.

Cuando ambos ruteadores incluyen sus respectivos identificadores en el campo *neighbour router* de sus propios HP, la NDS entra en el estado 2-vías (2-way), en el que los ruteadores pueden comunicarse entre ellos, estableciendo una comunicación bidireccional.

Una vez establecida una comunicación en redes broadcast y NBMA se procede a la elección del DR. Los HP sirven también a la designación del DR. Los campos *Router Priority* y *Designated Router* transportan toda la información necesaria al algoritmo de elección, que será explicado a continuación. Una vez alcanzado el estado 2-vías, y elegido el DR, no está dicho que la instauración de la adyacencia se complete y que la NDS proceda a otro estado.

En las redes broadcast o NBMA la adyacencia se establece solo entre un ruteador y DR; si uno de los ruteadores comunicantes no es uno de los DR la NDS permanece en el estado 2-vías.

En todos los estados sucesivos a 2-vías los ruteadores continúan a intercambiarse HP para verificar el correcto funcionamiento de la comunicación y del enlace. En cualquier estado que se encuentre la NDS, si falta la recepción de un HP en un tiempo igual a *Router Dead Interval* después de la llegada del último, la NDS pasa al estado Down, por lo que el ruteador podría estar desactivado; si al recibir un HP no se incluye en el campo *neighbour Router* de éste el ID del receptor la NDS pasa al estado Init, por lo que el otro ruteador no está en grado de mantener la comunicación bidireccional activa.

3.3.2 Elección del Ruteador Designado

Una vez que la NSS (*Neighbours State Structure)* alcanza el estado 2-vías en redes de tipo broadcast y redes NBMA se debe proceder a la elección del DR y del BDR (*Backup Designated Router).*

En la fase de elección entra en juego solo paquetes OSPF de tipo 1, los paquetes Hello, los cuales contienen toda la información útil para el algoritmo de designación: los campos *Designated Router*, *Backup Designated Router*, y *Router* Priority.

Los dos primeros indican la identidad de los DR según el ruteador que ha generado el HP, mientras el tercer campo indica la prioridad de elección del ruteador que genera el HP. Si los primeros campos están llenos con una máscara de 0 significa que el ruteador que ha generado el HP no tiene un DR. Si la prioridad de elección es igual a 0 significa que el ruteador no puede ser elegido como DR.

La elección del DR tiene lugar solo la primera vez que la red se activa o cuando uno de los DR deja de conectarse a la red. Si un ruteador se conecta a una red que ya eligió DR debe adaptarse. La elección designa al ruteador con mayor prioridad como el nuevo DR; en el caso en que más de un ruteador tenga la misma prioridad se elige aquel con Router ID mayor. El algoritmo de elección es un algoritmo distribuido, es decir que es seguido por cada ruteador que participa de la elección, llevando a todos a converger al mismo DR. Cada ruteador sigue un nuevo algoritmo de elección en dos casos:

- Cuando se conecta nuevamente a la red. Los ruteadores que se conectan a la red, antes de proceder a la elección de los DR, deben completar los campos DR y BDR de los propios HP con una máscara de 0.

- Cuando al menos uno de los DR elegidos en precedencia se desconecta de la red, o sea cuando la NDS pasa a un estado inferior a 2-vías.

El algoritmo de elección es ilustrado en figura 3.7. La primera instrucción prevé la compilación de una lista de posibles ruteadores elegibles: en esta lista cada ruteador incluye todos los ruteadores con el cual la NDS ha alcanzado al menos el estado 2-vías, excluyendo los ruteadores que tienen una prioridad igual a 0 (y que por lo tanto no pueden ser elegidos), e incluyéndose a sí mismo.

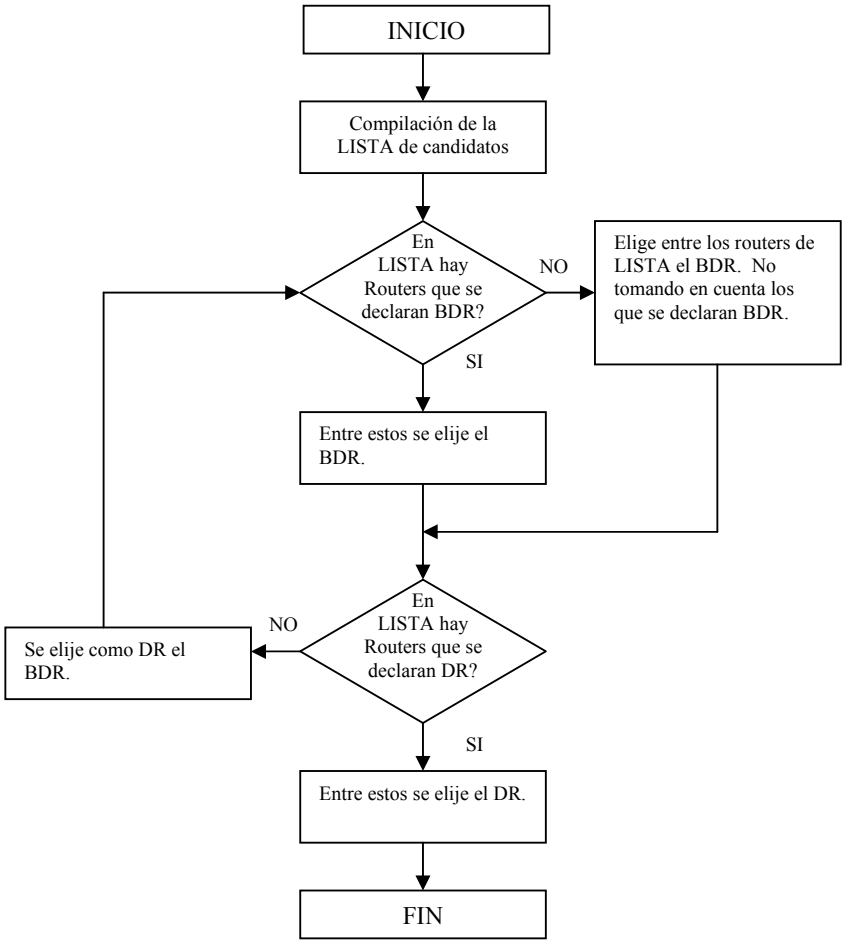


Fig. 3.7: Algoritmo de elección del Ruteador Designado

Las dos instrucciones de decisión prevén dos verificas que consideran a todos los ruteadores presentes en la lista de los elegibles:

- Cada ruteador examina los HP que está recibiendo de los ruteadores vecinos y verifica si en los campos DR o BDR está la identificación de los ruteadores que los han generado.

- El mismo ruteador que está siguiendo el algoritmo debe también tomarse en cuenta, es decir debe verificar si antes ha sido designado como DR o DBR, en éste caso debe inserir la propia identificación en el campo apropiado de sus HP.

La elaboración de las instrucciones de elaboración prevé que el ruteador no solo memorice el nuevo DR sino que también incluya su identificación en el campo apropiado de los propios HP.

El algoritmo es funcional en cada situación y lleva a resultados en la elección de los DR que son consistentes con los principios expuestos en precedencia:

- Un ruteador que acceda a una red que ya ha elegido los DR debe adaptarse.
- En una red que se activa por primera vez, el algoritmo primero designa los DR y después los BDR según los criterios de prioridad ya mostrados.
- En una red en que el DR se desconecta, el algoritmo hace que el BDR tome el puesto del DR, y que proceda a la elección de un nuevo BDR.
- En una red en que el BDR se desconecta, el algoritmo busca la manera de conservar el viejo DR y procede a la elección de un nuevo BDR.

La elección de un nuevo DR en una red broadcast o NBMA trae como consecuencia una variación en los estados de la NDS entre ruteadores que se

conectan entre ellos. Se recuerda que en estos tipos de red las únicas NDS que deben alcanzar el estado full son aquellas en que el DR está involucrado, mientras las NDS que se refieren a otros ruteadores deben detenerse en el estado 2-way. Cada vez que la red elige un nuevo DR las NDS se deben conformar adecuadamente.

3.3.3 Sincronización de la Base de datos del Estado del Enlace

Una vez alcanzado el estado 2-vías los ruteadores entran en la fase de sincronización de las Bases de Datos.

Como dijimos anteriormente un requisito necesario para que los algoritmos de estado del enlace y SPF den lugar a la determinación de una correcta tabla de enrutamiento es que las Bases de Datos LS presentes en cada ruteador estén sincronizadas; esto quiero decir que todos los ruteadores que participan deben tener memorizada la misma información de LS recibida de los ruteadores vecinos; pero no los mismos LSA, en cuanto los ruteadores que hacen referencia a áreas diferentes tienen una versión diferente de la red, sin embargo tienen información correcta y coherente acerca del alcance de las LIS y la métrica de los caminos al interior del AS. En caso contrario ruteadores diferentes podrían determinar a partir de Bases de Datos LS diferentes e incorrectas, una reconstrucción errada de la topología de red con resultados que pueden ser desastrosos para el reenvío de los datagramas IP.

El mecanismo de Realible Flooding permite mantener las Bases de Datos LS sincronizadas cuando la red está cercana a alcanzar el equilibrio. Sin embargo en las situaciones de fuerte desalineación el Realible Flooding puede no bastar para una convergencia rápida del algoritmo.

Para proveer una convergencia veloz en la sincronización de las bases de datos de las cuales se basa la compilación de las tablas de enrutamiento, OSPF prevé un mecanismo de sincronización de la Base de Datos LS cada vez que dos ruteadores establezcan entre ellos una nueva adyacencia. Ahora describiremos con detalle el procedimiento de sincronización de las Bases de Datos LS:

1. Después que se ha establecido la comunicación bidireccional mediante el intercambio de paquetes Hello (2-vías), sigue una primera fase (Exstart) en que los ruteadores deben establecer una relación de comunicación de tipo maestro/esclavo (master/slave) y deben por lo tanto establecer recíprocamente los propios papeles. La funcionalidad maestro asigna al ruteador el deber de administrar la transferencia de datos de manera confiable. Las especificaciones prevén que el papel de maestro corresponda al ruteador señalado con el ID mayor. En esta fase los ruteadores tienen posesión de las identificaciones que están contenidas en los paquetes Hello intercambiados, y están en grado de establecer sus correspondientes papeles, pero tienen la necesidad de reconocerlos recíprocamente intercambiando algunos mensajes:

    - El maestro “candidato” transmite un DDP en el que los tres bits del campo opciones (inicial, more, master) están con 1.

    - El esclavo “candidato” responde con un DDP para confirmar en que los bits de campo opciones inicial y master tienen 0.

    - Una vez que el maestro recibe el mensaje de confirmación, la comunicación maestro/esclavo se considera establecida y puede iniciar el intercambio de información.

2. En esta segunda fase (Exchange) los dos ruteadores verifican si uno de los dos tiene posesión de LSA recientes o si les falta la Base de Datos LS del otro. En esta fase, comunicación de tipo maestro/esclavo, la transferencia de la Base de Datos LS debe ser recíproca. Dicha comunicación tiene el objetivo de regular la transferencia de los índices de manera confiable, mediante un mecanismo de reconocimiento. Los estados Exstart y Exchange, pueden ser considerados generalmente como un único proceso de Intercambio de Base de Datos (Database Exchange), en cuanto la fase de Exstart, en la cual la comunicación maestro/esclavo es establecida, es solamente una fase preliminar al proceso de Intercambio de Base de Datos.

En esta fase las especificaciones del protocolo imponen que los DDP intercambiados no sean puestos a fragmentación a nivel IP; en caso que los datos a transferir excedan la MTU es necesario subdividirlos en más DDP.

- El maestro regula la comunicación y siempre es el primero en transmitir el propio DDP.

- A cada nueva transmisión el campo DD Sequence number es incrementado. El MS-bit tiene 1, el I-bit tiene 0, mientras el M-bit tiene 1 si se necesita enviar otros DDP sucesivamente.

- El esclavo espera la transmisión del maestro, si recibe el paquete correctamente y con DD Sequence number correcto está autorizado a transmitir el propio DDP. El MS-bit tiene 0, I bit tiene 1, M-bit tiene 0 si el esclavo necesita enviar otros DDP sucesivamente. El DD Sequence number debe ser el mismo que el DDP recibido del maestro.

- Si uno de los dos ruteadores recibe un DDP con errores simplemente se pone en espera. El ruteador que ha apenas transmitido, si no recibe ninguna confirmación después de un tiempo fijado, retransmite el DDP enviado.

- Si uno de los dos ruteadores termina de transmitir sus DDP, pero no el otro, en cuanto M.bit todavía tiene 1, el primero transmite los DDP de confirmación que no contengan algún LSA *HEADER* pero que sirven solamente como reconocimiento.

- Cuando ambos ruteadores han terminado de transmitir sus propios DDP (M-bit de todos los paquetes igual a 0) la última confirmación corresponde al esclavo. El esclavo confirma el último paquete del maestro. Después de que, espera un tiempo mayor a *Router Dead Interval.* Si el maestro o el esclavo dejan de transmitir significa que la fase de Exchange ha terminado y se pude pasar a la sucesiva. Ambos ruteadores han transferido correctamente el propio LS Database.

3.3.4 Carga de la Base de Datos

En la última fase (Loading) cada ruteador aprende el contenido de la Base de Datos LS del otro y aún cuando éste contenga LSA recientes o ausentes, el ruteador los solicita mediante el envío de un LSRP. El otro ruteador debe responder enviando los LSUP que contengan los LSA apenas requeridos. Si los LSUP son recibidos correctamente deben ser confirmados mediante un LSAP. En este estado la conexión entre los dos ruteadores debe entrar a formar parte del procedimiento de *Flooding*.

Una vez que la fase de Carga (Loading) está completa, la adyacencia se instaura, sin embargo es necesario que todos los LSA que son modificados por la creación de la nueva adyacencia sean renovados y distribuidos en todo el AS con un procedimiento de inundación (*flooding)*; tales son seguramente los Routers LSA respectivos a los dos ruteadores que han establecido una adyacencia, a los cuales se adjunta la referencia al nuevo enlace, si los dos ruteadores se conectan mediante una red broadcast se debe renovar también el Network LSA.

Todo esto se hace con el objetivo de distribuir velozmente a todos los ruteadores del AS la información de la nueva topología de la red, y eventualmente recalcular las tablas de enrutamiento y reenvío.

Una vez completa la sincronización de las Base de Datos LS el NDS se pone en el estado full. En este estado la adyacencia se establece y debe ser incluida en la generación de los LSA, además de tomar parte del procedimiento de Flooding.

Las especificaciones, en realidad, no prevén que la comunicación deba desarrollarse rígidamente según éste esquema, pero es posible que más operaciones previstas por estados diferentes sean realizadas contemporáneamente: por ejemplo es posible que la fase de Carga de la Base de Datos (*Database Loading*) tenga inicio apenas un encabezado LS sea transmitido en un DDP, y que por lo tanto la transferencia de LSA completos sea más larga cuando todavía la fase de Intercambio no ha terminado.

3.4 Generación y distribución de los LSA

Uno de los aspectos más importantes en OSPF es la generación y la distribución de la información de LS que cada ruteador debe generar para luego distribuirlos a

todos los ruteadores que los necesitan para la compilación de las propias tablas de enrutamiento y reenvío. En OSPF la información de LS es insertada en los LSA, que constituyen la Base de Datos LS.

Las tablas de enrutamiento y reenvío son compiladas en modo independiente por cada router en base a la información contenida en la propia Base de Datos LS, y para garantizar que las tablas calculadas separadamente por cada ruteador sean coherentes es necesario que las Base de Datos LS de todos los ruteadores estén sincronizados entre ellos.

Sin embargo, esto no quiere decir que todos los ruteadores del AS deban memorizar en la propia Base de Datos LS los mismo LSA; el enrutamiento jerárquico y la partición en áreas da a cada nodo de la red una visión parcial de ésta; es necesario que cada ruteador tenga una descripción consistente y exacta de la topología de la red.

Hemos dicho que los ruteadores renuevan periódicamente los propios LSA, y por lo tanto es deber fundamental de OSPF garantizar el envío confiable de los LSA. Es decir, es necesario que cada ruteador tenga en posesión en el menor tiempo posible los LSA recientes que sean necesarios.

El primer aspecto por analizar es aquel de la generación de los LSA, es decir entender los LSA que cada ruteador debe generar y cuando generarlos, para luego distribuirlos.

El segundo aspecto es la determinación de un algoritmo que permita realizar el Realible Flooding de los LSA, minimizando en el mismo tiempo la utilización de los recursos de la red. El Realible Flooding no solo debe distribuir entre los ruteadores los LSA generados, también debe remover de la red la información de LS vieja que podría resultar obsoleta.

a) Generación de los LSA

Primero analizamos cuando y como un ruteador debe generar los propios LSA. Un ruteador que genera una nueva instancia de LSA debe memorizarla en la propia Base de Datos LS y difundirla mediante el procedimiento Realible Flooding. Recordemos que en los propios LSA cada ruteador debe hacer referencia de las adyacencias con otros ruteadores solo si la NDS ha alcanzado el estado full. Cada ruteador genera al menos un Router LSA; si un ruteador hace el papel de DR en una de las redes al cual está conectado, entonces debe generar el relativo Network LSA. Los ABR también generan más Summary LSA. A continuación citaremos las situaciones en que un ruteador debe dar lugar a una nueva instancia de LSA:

- Cuando uno de los LSA ha sido generado en un tiempo mayor a LSRfreshTime. El que equivale a decir que cada ruteador genera los propios LSA periódicamente. La RFC 2328 requiere que LSRfreshTime tenga un valor entre 5 segundos y 30 minutos. Cada vez que un ruteador genera una nueva instancia del mismo LSA debe incrementar el campo LS Sequence number.

- En caso de que existan cambios en el LS del ruteador. Para todos los ruteadores se contemplan los siguientes casos:

    - En caso de que cambie el DR de una de las redes a las que el ruteador está conectado. Los ruteadores que se conectan a una red broadcast deben declarar en los propios routers LSA la conexión con el DR.

- En caso de que una de las NDS respectiva a una conexión con un ruteador adyacente salga del estado full: un nuevo router LSA debe generarse excluyendo la conexión con dicho ruteador.

- En caso de que una de las NDS respectiva a una conexión con un ruteador adyacente alcance el estado full. En tal caso un nuevo Router LSA debe ser generado incluyendo la nueva conexión.

Para los ABR se contemplan las siguientes eventualidades:

- En caso de que sucedan cambios en la topología de una de las áreas a la que el ruteador se conecta. Esto lleva a la generación de nuevas instancias de Summary LSA respectivamente a todos las otras áreas.

- En caso de que sucedan cambios en la topología de una de las áreas a la que el ABR no se conecta directamente, pero que son señalados por los Summary LSA que recibe de el área Backbone. Esto lleva a la generación de una nueva instancia de Summary LSA respectivamente a todas las áreas a las que el ABR se conecta.

- El ABR se conecta a una nueva área y debe generar una nueva instancia de Summary LSA en todas las áreas a las que se conecta, excepto el área Backbone.

b) Distribución de los LSA: Realible Flooding

Los LSA que deben ser distribuidos son transmitidos al interno de los LSUP. Los LSA son intercambiados en fase de sincronización de las bases de datos durante la instauración de las adyacencias, pero éstos últimos son

intercambiados solo entre ruteadores adyacentes y no deben participar al proceso de Realible Flooding.

- El procedimiento inicia cuando en una cierta interfaz el ruteador recibe un LSUP. Desde el campo Area ID al interno del encabezado OSPF es determinada el área a la cual los LSA contenidos del LSUP hacen referencia.

Para cada LSA contenido en el LSUP:

- Es controlado el checksum del LSA. Si es errado el LSA es descartado y se examina el sucesivo.

- Si el campo LSA age en el encabezado es igual a MaxAge se agrega el LSA en la lista de aquellos por confirmar y luego se lo descarta, pasando al análisis del sucesivo.

- Si no hay en el interior de la Base de Datos LS copias del LSA o bien si el que se recibe es reciente, es decir su LS Sequence number es menor, se realizan las siguientes acciones:

    - Sustituye el LSA precedente
    - Agrega el LSA a la lista de aquellos por distribuir
    - Agrega el LSA a la alista de aquellos por confirmar.

- Si el LSA recibido no es reciente de aquellos ya en posesión por el ruteador, se agrega a la lista de aquellos por confirmar y se pasa al análisis sucesivo.

Cuando el ruteador termina de procesar todos los LSA contenidos en el LSUP debe realizar dos acciones más:

- Transmitir el LSAP para confirmar la correcta recepción de todos los LSA insertados en la lista de aquellos por confirmar.

- Proceder a la distribución de los LSA insertados en la lista de aquellos por retransmitir. Estos deben ser insertados en un nuevo LSUP el cual es transmitido a todos los ruteadores adyacentes que forman parte del área a la cual el primer LSUP hacia referencia y con la cual la NDS ha alcanzado al menos el estado Exchange.

Este algoritmo respeta los principios sobre los cuales se basa el proceso de realible Flooding:

- Eliminación de la red de LSA obsoletos mediante la temporización suministrada por los campos LS Age y el ordenamiento provisto por el campo LS Sequence number.

- Transmisión confiable hop-by-hop de los LSA, mediante la retransmisión de los LSAP.

- Correcta distribución a todos los ruteadores que lo necesitan, de los LSA generados recientemente, mediante la transmisión de LSUP cada vez que un ruteador recibe una copia de LSA actualizados.

BGP: Border Gateway Protocol

3.5 Características del Protocolo

El BGP versión 4, definido en [18], es el protocolo estándar para el enrutamiento inter-dominio en el Internet actual, conocido como BGP4 o simplemente como BGP.

Es un protocolo de red usado para conectar entre ellos más ruteadores que pertenecen a sistemas autónomos distintos, y que son llamados gateway.

BGP es caracterizado como un protocolo vector de caminos (*path vector)*, porque los ruteadores BGP contiguos, llamados par BGP (*BGP peer)*, se intercambian información detallada de los caminos (como la lista de los AS en el camino hacia un destino dado) que información sobre los costos. Como todos los protocolos basados en el algoritmo vector distancia, BGP es un protocolo distribuido, en cuanto los ruteadores BGP calculan independientemente sus tablas, y se comunican solo con los ruteadores BGP directamente conectados.

La información global sobre los caminos hacia destinos remotos se propaga de AS a AS mediante el intercambio de información de enrutamiento BGP entre parejas de ruteadores BGP directamente conectadas. Notamos que BGP enruta hacia redes de destino (en el sentido de direccionamiento), que a hosts o ruteadores específicos. Una vez que un datagrama alcanza la red de destino, para enrutar el datagrama hacia el destino final se utilizará el sistema de enrutamiento intra-AS de esa red. En BGP, un sistema autónomo es identificado por un número particular (llamado ASN, *Autonomous System Number*) único en el Internet [38].

Técnicamente no todos los AS tienen un ASN. En particular, un llamado *AS stub*, que transporta solo tráfico por lo cual es fuente o destino (y, por lo tanto, no es atravesado de tráfico de otros AS) no tendrá un ASN. Los ASN, como las direcciones IP, son asignados por registros regionales ICANN [39].

BGP soporta el CIDR, y usa un mecanismo de agregación de caminos de enrutamiento para disminuir la dimensión de las relativas tablas.

El protocolo BGP ha sido creado para sustituir el protocolo de enrutamiento EGP y consentir un enrutamiento completamente descentralizado.

Los ruteadores adyacentes se comunican mediante una conexión de nivel de transporte TCP, que garantiza la confiabilidad. El uso de TCP evita considerar los problemas de retransmisión.

Es necesario prestar atención al hecho de que el ruteador esté en grado de privilegiar eventualmente los paquetes BGP que los TCP, en caso de congestión, reduce el tasa de bit (bit rate). Esto lleva a disminuciones en la velocidad de propagación de la información de enrutamiento.

Otras características de BGP pueden ser sintetizadas en:

- Protocolo *Path Vector* (variante del protocolo *Distance Vector*)
- Para cada destino IP es suministrada la secuencia de AS para atravesar. La descripción del camino hacia un destino es efectuada mediante el uso de campos llamados atributos.
- Los destinos IP son expresados en términos de prefijos de dirección.

- Los ruteadores pueden agregar información de enrutamiento recibida antes de propagarla. Esto permite un beneficio en términos de disminución del tráfico de enrutamiento, y de disminución de las dimensiones de las bases de datos en los ruteadores.
- Cada ruteador tiene un algoritmo (Proceso de Decisión) para clasificar los caminos alternativos.

3.6 Mensajes BGP

El protocolo BGP define cuatro tipos de mensajes: *Open*, *Update*, *Keepalive* y *Notification*. El encabezado de los mensajes BGP (figura 3.8) es de 19 bytes y contiene:

- Marcador (*Marker*)
- Longitud del mensaje (*Lenght*)
- Tipo de mensaje (*Type*)
    - *type* = 1 → *open* (apertura)
    - *type* = 2 → *update* (actualización)
    - *type* = 3 → *notification* (notificación)
    - *type* = 4 → *keepalive*

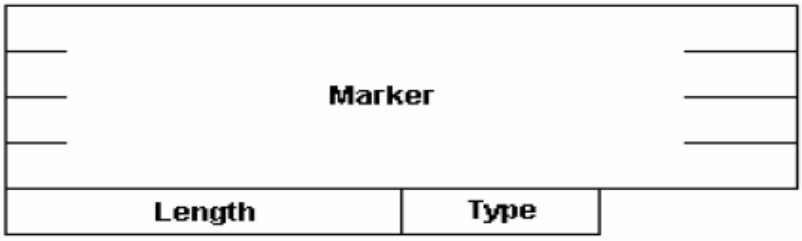


Fig. 3.8: Encabezado de los mensajes BGP

- **OPEN**

Cuando un ruteador BGP desea establecer un contacto por primera vez con un par BGP (por ejemplo, después de que el ruteador mismo o un enlace que lo conecta es reiniciado), se envía un mensaje *Open* al par. El mensaje *Open* permite al ruteador BGP identificarse y autenticarse. Si el mensaje *Open* es aceptado por el par, éste responderá con un mensaje *Keepalive*.

- **UPDATE**

Un ruteador BGP usa el mensaje *Update* para anunciar el camino hacia un destino dado al par BGP. El mensaje *Update* es usado también para eliminar un camino que ha sido anunciado (es decir, informa al par que el camino previamente anunciado ya no es válido). Un camino BGP es considerado válido hasta que no sea explícitamente eliminado.

- **KEEPALIVE**

Este mensaje BGP es usado para hacer conocer a un par que el remitente está activo pero no tiene información adicional para enviar. Sirve también como comprobación de un mensaje Open recibido.

- **NOTIFICATION**

Este mensaje BGP es usado para informar a un par que se ha producido un error y que se cerrará la sesión BGP.

3.7 Máquina de Estados Finitos

Los pares BGP basan la toma de decisiones para la interacción con otros peer BGP en una máquina de estados finitos (figura 3.9) compuesta de 6 estados.

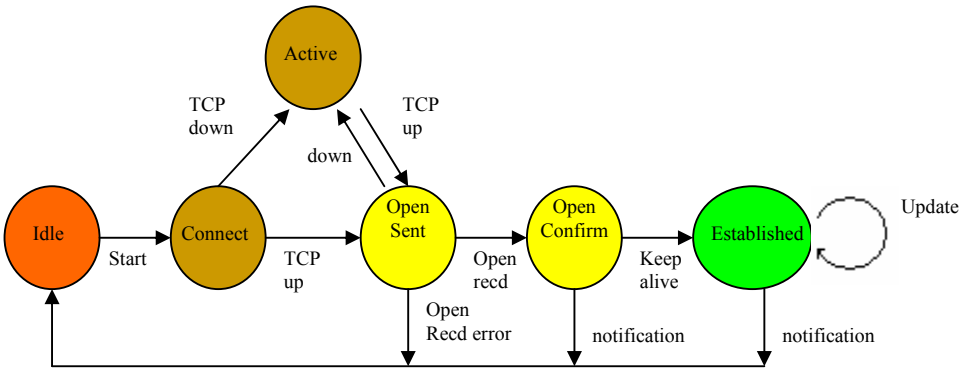


Fig. 3.9: Estados de la conexión BGP

- **Estado Idle:**

En este estado BGP rechaza todas las conexiones entrantes. En respuesta a un evento de Start (generado por el sistema o por un operador) el sistema local inicializa todos los recursos BGP, inicializa el reloj *ConnectRetry*, inicia una conexión de transporte para otro par BGP, mientras escucha una conexión que puede ser iniciada por el BGP remoto y cambia su estado a Connect. Si el par BGP se da cuenta de un error, cierra la conexión y cambia su estado a Idle.

Si hay un error pendiente y se genera automáticamente el evento de Start, se corre el riesgo de un bucle entre el estado Idle y el estado Connect. Para evitar esto, el evento Start no debe ser generado inmediatamente y que el tiempo entre generaciones consecutivas del evento Start tenga un crecimiento exponencial. Cualquier otro evento en este estado es ignorado.

- **Estado Connect:**

En este estado, BGP espera que la conexión por parte del protocolo de transporte implícito se complete. Si la conexión TCP ha sido establecida, el sistema anula el reloj *ConnectRetry*, transmite un mensaje de *Open* y cambia su estado en OpenSent. Si la conexión TCP falla, el sistema local reinicia el reloj *ConnectRetry*, mientras escucha una nueva conexión del BGP remoto, y cambia su estado en Active.

Si el reloj *ConnectRetry* expira, el sistema local reinicia el reloj *ConnectRetry*, inicia una conexión para otro par BGP, escucha una conexión que puede ser iniciada por el par BGP remoto, y permanece en el estado Connect. En respuesta a cualquier evento, con excepción de Start que es ignorado, el sistema libera los recursos y cambia su estado a Idle.

- **Estado Active:**

En este estado un par BGP intenta "adquirir un par" iniciando una conexión TCP. Si la conexión se logra, el sistema elimina el reloj *ConnectRetry*, completa la inicialización, transmite un mensaje *Open* a su par, pone el Hold Timer con un valor grande predefinido, y pasa al estado OpenSent.

Si el reloj *ConnectRetry* expira, el sistema reinicia el reloj *ConnectRetry*, inicia una conexión para otro par BGP, escucha por una conexión que puede ser iniciada por el par BGP remoto, y cambia su estado a Connect

Si el sistema detecta que el par remoto está intentando establecer una conexión con él, y la dirección IP de este último no es la que se esperaba, el sistema local

reinicia el reloj *ConnectRetry*, rechaza el tentativo de conexión, escucha por una conexión que puede ser iniciada por el par BGP remoto, y permanece en el estado Active.

Análogamente al estado Connect, en respuesta a cualquier evento, con excepción Start que es ignorado, el sistema libera los recursos asociados a la conexión y cambia su estado a Idle.

- **Estado OpenSent:**

En este estado el sistema está esperando un mensaje *Open* por parte del par BGP. Cuando recibe el mensaje, es controlada la exactitud de los campos. Si hay un error en el encabezado o en el mensaje *Open*, el sistema local envía un mensaje *Notification* y cambia su estado a Idle.

Si no hay errores, BGP transmite un mensaje *Keepalive* y pone un reloj KeepAlive. El Hold Timer, puesto anteriormente, es remplazado con el valor Hold Time negociado.

Si este último es cero, el Hold Timer y el reloj KeepAlive no parten. Si el valor del AS es el mismo del AS local, la conexión es "interna", de otra manera es externa. Finalmente, el sistema cambia su estado a OpenConfirm.

Si hay una señal de desconexión por parte del protocolo de transporte implícito, el sistema cierra la conexión BGP, reinicia el reloj ConnectRetry y pasa al estado Active.

Si el Hold Timer expira, el sistema local envía un mensaje *Notification* con error code Hold Timer Expired y cambia su estado a Idle.

En respuesta a un evento de Stop, el sistema envía un mensaje *Notification* con *Error Code Cease* y cambia su estado a Idle. El evento Start es ignorado, y para cualquier otro evento el sistema local envía un mensaje *Notification* con Error Code Finite State Machine Error y cambia su estado a Idle.

- **Estado OpenConfirm:**

En este estado el BGP espera por un mensaje *Keepalive* o *Notification*. Si se recibe un mensaje *Keepalive*, cambia su estado a Established. Si el Hold Timer expira antes de recibir un mensaje *Keepalive*, el sistema local envía un mensaje *Notification* con *error code Hold Timer Expired* y cambia su estado a Idle.

Si el sistema recibe un mensaje *Notification*, pasa al estado Idle. Si el reloj *Keepalive* expira, es enviado un mensaje *Keepalive* y se reinicia el reloj correspondiente. Si existe la señal de una desconexión por parte del protocolo de transporte implícito, el sistema cambia su estado a Idle.

En respuesta a un evento Stop, el sistema local envía un mensaje *Notification* con *Error Code Cease* y cambia su estado a Idle. El evento Start es ignorado, y con cualquier otro evento el sistema local envía un mensaje *Notification* con *Error Code Finite State Machine Error* y cambia su estado a Idle.

- **Estado Established:**

En este estado el BGP puede intercambiar mensajes *Update*, *Keepalive* o *Notification* con su par. Si el sistema local recibe un mensaje *Update* o

Keepalive, reinicia su Hold Timer, si el valor negociado de este último es diferente de cero.

Si recibe un mensaje *Notification*, cambia su estado a Idle. Si el sistema recibe un mensaje *Update*, se controla el encabezado y el mensaje mismo, si se detecta un error transmite un mensaje *Notification* y cambia su estado a Idle.

Si recibe una notificación de desconexión por parte del protocolo de transporte implícito, el sistema cambia su estado a Idle. Si el Hold Timer expira, se envía un mensaje *Notification* análogamente como ocurre en los otros estados.

Si expira el reloj Keepalive, el sistema se comporta de manera análoga al estado OpenConfirm. Cada vez que el sistema transmite un mensaje *Keepalive* o *Update*, el reloj Keepalive se reinicia. Para cualquier otro evento recibido el sistema se comporta como en el estado OpenConfirm.

3.8 Funcionamiento del protocolo

El funcionamiento de BGP gira alrededor a tres actividades, todas ligadas a los anuncios sobre caminos:

- *Recepción y filtraje de anuncios en los caminos por parte de vecinos directamente conectados*. Un ruteador BGP recibe anuncios de un par BGP. Un par BGP que anuncia un camino hacia un AS de destino promete que si un AS contiguo le manda un datagrama al AS de destino, él estará en grado de enviarle con éxito el datagrama. Un ruteador BGP puede filtrar los anuncios recibidos, por ejemplo, un ruteador BGP ignorará los anuncios que

contengan el propio número de AS en el AS-PATH, debido a que puede provocar un bucle de enrutamiento.

- *Selección del camino*. Un ruteador BGP puede recibir diferentes anuncios dirigidos al mismo AS de destino, y debe elegir cual camino usar de entre todos los anuncios. El AS de destino y el siguiente paso (*next-hop)* para el camino elegido deben, por lo tanto, ser insertados en la tabla de enrutamiento del ruteador. Un ruteador BGP puede conocer diferentes caminos hacia un mismo destino, pero instalará un solo ingreso (un solo *next-hop*) para aquel destino en la tabla de enrutamiento. ¿Como hace un ruteador BGP para elegir un camino entre tantos? Hace una distinción entre mecanismo de enrutamiento y política de enrutamiento. En particular, BGP no especifica como un AS debe elegir un camino, ésta es una decisión política que concierne al administrador de red del AS.

- *Envío de anuncios a los vecinos.* Así como un ruteador BGP recibe anuncios de sus vecinos, él también puede anunciar caminos a sus vecinos. BGP provee un mecanismo, y no una política, para estos anuncios.

  Para entender mejor el funcionamiento de BGP debemos conocer de manera más detallada la recepción y propagación de los paquetes que anuncian cambios topológicos y el proceso a seguir para la selección de la mejor ruta:

- *Recepción y Propagación*

  Antes de ver los aspectos legados a la recepción y a la propagación de paquetes, es necesario introducir la definición de RIB (*Routing* Information Base). Existen 3 tipos de RIB:

  - **Adj-RIBs-In**: Información aprendida de los anuncios recibidos.

- **Loc-RIB**: Información utilizada para el enrutamiento y seleccionada mediante el proceso de decisión.
- **Adj-RIBs-Out**: Información a propagar y seleccionada mediante el proceso de decisión.

Cuando se recibe un anuncio:

- Si el destino se encuentra en **Adj-RIBs-In**, la nueva ruta remplaza a la vieja; se realiza el proceso de decisión.
- Si la ruta es más específica que otra:
  - Con atributos diferentes, se realiza el proceso de distribución y la parte de la ruta menos específica se considera no válida.
  - Con los mimos atributos, la nueva ruta es ignorada.
- Si el destino no está presente en **Adj-RIBs-In** se inserta la nueva ruta y se realiza el proceso de decisión.

Si la nueva ruta es menos específica de una existente se realiza el proceso de decisión que concierne solo los destinos descritos por la nueva ruta.

- *Proceso de decisión*

  El proceso de decisión selecciona rutas para sucesivos anuncios aplicando políticas en el *Local Policy Information Base* (PIB) a las rutas memorizadas en su Adj-RIB-In. La salida del proceso de decisión es el conjunto de rutas que serán anunciadas a todos los par; las rutas seleccionadas serán memorizadas en el Adj-RIB-Out del speaker local.

  El proceso de decisión se hace oficial definiendo una función que toma el atributo de una ruta dada como un argumento y retorna un entero positivo indicando el grado de preferencia para la ruta.

La función que calcula el grado de preferencia para una ruta dada no usará como entrada ninguna de las siguientes: la existencia de otras rutas, la inexistencia de otras rutas, o bien los atributos de camino de otras rutas. La selección de rutas consiste en la simple aplicación del grado de la función de preferencia hacia cada ruta alcanzable, seguida de la selección de aquella con el grado más alto de preferencia.

El proceso de decisión opera sobre las rutas contenidas en cada Adj-RIB-In, y es responsable de la:

- Selección de rutas para ser anunciadas a los BGP speaker situados en el sistema autónomo del speaker local.
- Selección de rutas para ser anunciadas a los BGP speaker situados en los sistemas autónomos vecinos.
- Agregación de las rutas y la reducción de información de las rutas.

El proceso de decisión se realiza en tres fases:

a) La fase 1 es responsable del cálculo del grado de preferencia para cada ruta recibida por un BGP speaker situado en un sistema autónomo vecino, y del anuncio a los otros BGP speaker en el sistema autónomo local las rutas que tienen el grado más alto de preferencia para cada situación.

b) La fase 2 es invocada al final de la fase 1. Es responsable de la selección de la mejor ruta entre aquellas disponibles para cada destino diferente, y de la instalación de cada ruta elegida dentro del Loc-RIB.

c) La fase 3 es invocada después que el Loc-RIB ha sido modificado. Es responsable de la diseminación de las rutas en el Loc-RIB a cada par situado en el sistema autónomo vecino, según las políticas contenidas en el PIB. La agregación de las rutas y la reducción de la información pueden ser facultativamente desarrolladas al interno de esta fase.

La figura 3.10 resume una parte de los conceptos hasta el momento expresados, en particular para lo que concierne las políticas, recepción y propagación, y proceso de decisión:

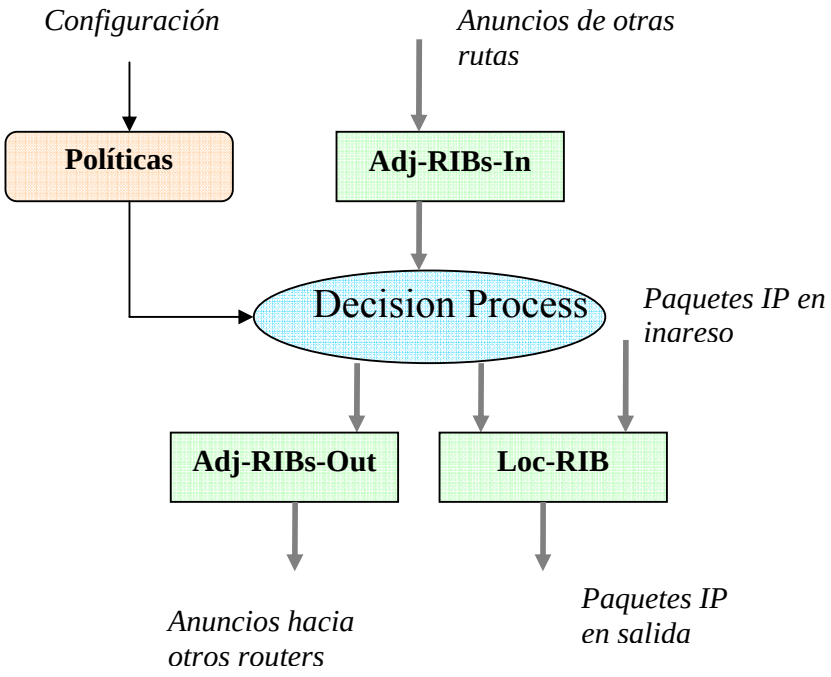


Fig. 3.10: Funcionamiento del protocolo BGP

Capítulo 4: Xorp-Plataforma extensible de Código Abierto

4.1 Introducción

El mercado del software de enrutamiento es cerrado, o bien cualquier ruteador utilizará solamente el software propietario de la casa productora a la que pertenece.

Este vínculo hace imposible, para los investigadores, la experimentación en redes reales y por lo tanto el desarrollo de pruebas que podrían convencer a los operadores de red de la presencia de alternativas a las situaciones actuales.

Otro obstáculo a la investigación, está representado en las grandes inversiones que un ruteador con altas prestaciones representaría.

La solución parecería simple: el software de los ruteadores debería tener una API de tipo abierto.

Desafortunadamente la extensibilidad puede ocasionar conflictos con otros requisitos fundamentales como la robustez, perfomance y con la complejidad de algunos algoritmos de enrutamiento como BGP.

Relativamente pocos sistemas de software tienen objetivos exigentes de robustez y seguridad como en los ruteadores, donde inestabilidad localizada y definiciones erradas de la configuración pueden propagarse a través de la red, esto ha sucedido en cuanto a los desarrolladores del software que han adoptado como vínculo primario la escalabilidad, poniendo en segundo plano elementos como la estabilidad y la latencia.

Por lo tanto podemos decir que muchos problemas con la infraestructura de enrutamiento, al interno del Internet actual, no son arquitecturas o problemas de protocolo, sino problemas realizados de software.

Hemos visto la necesidad de una nueva *suite* de ruteador software que tenga como objetivo primario la extensibilidad, con el fin de permitir el desarrollo de un protocolo experimental con riesgo mínimo respecto a los servicios actualmente soportados.

Los investigadores de Internet que tuvieran acceso al ruteador de software compartirían una plataforma común para la experimentación que les permitiera desarrollarlo; Xorp representa una respuesta a la necesidad apenas explicada.

Existen además de los protocolos de enrutamiento apenas mencionados, otras alternativas que mencionaremos antes de pasar al análisis detallado de Xorp; estas alternativas son Quagga y BIRD (BIRD Internet Routing Daemon).

La suite protocolar Quagga provee implementaciones de OSPFv2, OSPFv3, RIPv1, v2 y v3, y BGPv4, además soporta IPv4 e IPv6 y gira sobre diferentes plataformas Linux. Quagga deriva de Zebra y utiliza el lenguaje de programación C; un aspecto importante de éste proyecto es el hecho que será el primero en implementar el soporte MPLS.

La configuración Quagga, resulta no ser excesivamente intuitiva, por lo que cada demon necesita de un propio archivo de configuración.

BIRD soporta IPv4 y IPv6, BGP, RIP, OSPF (solo IPv4) y un enrutamiento estático e incluye en sus futuras implementaciones OSPF para IPv6 y OSPF NSSA y la agregación de rutas.

Xorp es un software de enrutamiento que garantiza a los aspectos de estabilidad y latencia la atención necesaria.

Xorp es dirigido por eventos (*event-driven)* y se propone de responder a los cambios de enrutamiento con el mínimo retraso; esto representa un requisito crucial para las crecientes expectativas en términos de seriedad y tiempos de convergencia.

El diseño de Xorp consiste de una estructura compuesta de procesos de enrutamiento, cada uno compuesto de bloques modulares mediante el cual fluyen las rutas.

Xorp, por lo tanto, deriva de estrategias utilizadas para subdividir el plano de control y los protocolos de enrutamiento individuales, en componentes que facilitan sea la extensión que el mejoramiento del perfomance.

Hemos dicho, que la modularidad del software y su facilidad de configuración, garantizan en términos de extensibilidad del mismo, una atrayente característica. De hecho, se pude decir, que el interés de los operadores de red en el proyecto Xorp, es en gran parte, a la interesante prospectiva de fácil extensibilidad del proyecto a nuevos protocolos [40].

Un mecanismo IPC flexible consiente a los módulos de comunicarse el uno con el otro independientemente del hecho que estos sean parte del mismo proceso o de la misma máquina.

Los procesos de Xorp, siendo dirigidos por eventos, no sufren de grandes variaciones de retraso, característica que une las arquitecturas a base de temporizador (*timer-based)*.

Si bien Xorp está en plena fase de desarrollo, las selecciones de diseño han señalado de ser suficientemente estables y demuestran que los objetivos de extensibilidad, escalabilidad y robustez del ruteador de software son alcanzables.

Citamos también, la fundamental característica de ahorro de capital, que representaría la realización y el uso de un ruteador software con altas prestaciones. En fin no podemos descuidar, o considerar en segundo plano, la implícita facilidad de actualizaciones que un ruteador software representa, con respecto a las correspondientes realizaciones hardware, actualmente utilizadas.

4.2 Arquitectura modular de Xorp

4.2.1 Procesos de gestión y de enrutamiento

Gran parte del software en un ruteador está constituido de aplicativos para el soporte de funcionalidades para el plano de control.
En este sentido podemos, citar por ejemplo, funcionalidades para la gestión de los protocolos de enrutamiento [41], el RIB, la gestión de las funciones de firewall, la interfaz usuario y en fin la gestión de la red.

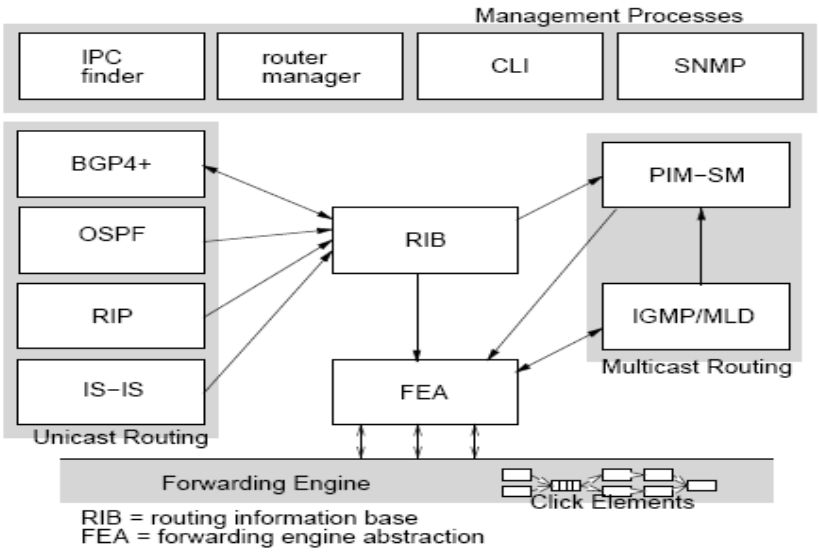


Fig 4.1: Proceso de Administración[1]

La figura 4.1 muestra la estructura de software general del plano de control de un ruteador en lo que se refiere al enrutamiento, poniendo en evidencia los principales procesos involucrados y las relaciones más importantes entre ellos. En lo que concierne los procesos de enrutamiento (unicast y multicast) los principales son:

- **RIB**: El RIB representa la conexión entre los diversos protocolos de enrutamiento, visto que los protocolos pueden declarar diferentes rutas hacia la misma subred de destino, el RIB deberá decidir cual utilizar entre las diferentes alternativas. Este módulo, instrumento clave de la disciplina de enrutamiento, realiza el proceso de redistribución de las rutas; mediante esto las rutas, de un protocolo de enrutamiento, que corresponden a una determinada política vienen redistribuidas a otro protocolo de enrutamiento para avisar también a otros ruteadores.

[1] Gráfico obtenido del artículo “XORP: An Open Platform for Network Research”; con libre autorización de realizar copias digitales de cualquier parte del mismo.

RIB, como parte del sistema que ve las rutas de cualquiera, desarrolla un rol central en este proceso.

RIB resulta, por lo tanto, crucial en el correcto funcionamiento de un ruteador y debería ser extenso con mucha precaución, porque los diferentes protocolos de enrutamiento pueden ser añadidos o sustraídos.

- **BGP**: BGP tiene una relación más compleja con RIB, las rutas BGP entrantes normalmente individúan más bien un ruteador de siguiente paso hacia el destino que un vecino.

- **FEA**: El FEA garantiza una API estable para la comunicación con uno o más motores de reenvío (*forwarding engine)*.

  En el caso de Xorp, el motor de reenvío podría ser el del kernel Linux o FreeBSD, o podría corresponder a un hardware para el reenvío, el motor de reenvío consigue que Xorp tenga la capacidad de habilitar y configurar diferentes funcionalidades de reenvío.

  El término “abstracción” correspondiente a la última letra del acrónimo FEA se refiere a la configuración de las interfaces de alto nivel operadas por Xorp, que se podría comparar a un caso en el que el motor de reenvío, en un kernel activo en un sistema, fuera realizado vía software o mediante un hardware externo.

  Si las rutas de OSPF resultan ser las “mejores rutas” entre las presentadas por todos los protocolos, entonces el RIB las envía a la FEA; este último añade o elimina las rutas, arriba mencionadas, del kernel.

En un ruteador Xorp el FEA debe ser explícitamente habilitado o no serán enviados los paquetes; el forwarding puede ser separadamente habilitado para ser unicast o multicast, IPv4 o IPv6.

Además, las interfaces/vifs multicast deberán ser explícitamente e individualmente habilitadas junto a funcionalidades propias del enrutamiento multicast.

- **PIM-SM e IGMP**: PIM-SM e IGMP proveen las funcionalidades de enrutamiento multicast, donde PIM realiza el reenvío e IGMP informa al PIM de la existencia de receptores locales.

- **IS-IS**: este módulo, todavia no está implementado.

- **IPC**: garantiza la comunicación entre los procesos de Xorp y las aplicaciones de enrutamiento construidas.

Pasando a ocuparnos de los procesos de gestión citamos:

- **Router Manager**: basándose en los archivo de configuración del ruteador da inicio, configura y bloquea el protocolo de enrutamiento y otras funcionalidades del ruteador.

  Esconde la estructura interna del ruteador al usuario, ofreciendo al operador una interfaz de gestión única para la reconfiguración y el control.

- **SNMP**: El Router Manager está configurado para dirigir el agente SNMP con el fin de supervisar la administración de los aparatos conectados a la red.

El objetivo de toda esta estructura es un plano de control del ruteador que ponga a disposición todas las funcionalidades ilustradas hasta el momento, comprendiendo todos los protocolos de enrutamiento más utilizados.

4.2.2 Comunicación entre procesos

El plano de control de Xorp precedentemente ilustrado implementa las funcionalidades en figura 4.1 mediante un conjunto de procesos comunicantes. Cada protocolo de enrutamiento y cada funcionalidad de administración es administrada mediante un proceso separado.

Los procesos se comunican entre ellos utilizando un mecanismo IPC extenso llamado XRLs. Este mecanismo es soportado por cada proceso; cuando un proceso desea comunicarse con otro escribe un XRL, diseccionado al proceso genérico (BGP, OSPF, RIP, etc) y lo envía.

El llamado Finder interpreta “resuelve” este XRL genérico, en una forma que especifica definitivamente cual forma de comunicación debe producirse; el resultado consiste en la individuación del protocolo de transporte que será utilizado para la comunicación, como por ejemplo TCP, y cada parámetro de la misma, así como nombre de host y puerta.

El código de Xorp ha sido escrito en C++, gracias a las características de ser un lenguaje orientado a objetos y garante de buen perfomance; gracias al gran uso de templates C++, se ha podido obtener un código fuente para IPv4 e IPv6, que mediante su compilación ha dado origen a una implementación eficiente para ambos.

Cada proceso de Xorp, adopta un modelo de programación, como dijimos antes, dirigido por eventos y *single-threaded*.

Al interno del corazón del modelo de Xorp, los procesos son generados por un reloj y descriptores de archivo; los callbacks son enviados cada vez que un evento se produce.

Calcular las dependencias arriba mencionadas, veloz y eficazmente, es difícil e introduce una fuerte presión al mecanismo periódico de scanner de las rutas.

Desafortunadamente este scanning periódico introduce una latencia variable y genera una carga a bursts creciente que podría influenciar el perfomance de envío.

Completar en un solo paso la elaboración de un proceso no es una operación que lleva siempre a un buen fin en un ruteador de tipo dirigido por eventos.

Xorp soporta además tareas de instrucciones, implementado mediante un gestor de relojes, que resultan ser activas solo cuando ningún evento está en elaboración.

Estas tareas, son esencialmente threads cooperativas que dividen una sola elaboración en varias partes, y que, al termine de las ejecuciones, regresan al proceso principal, hasta que estas terminen.

4.3 Instalación y modalidad operativa de Xorp

Como es posible conocer directamente del sitio oficial www.xorp.org, las plataformas de desarrollo de Xorp son FreeBSD y Linux.

Se puede afirmar que Xorp no presenta problemas, con todas las versiones recientes FreeBSD y con la mayor parte de las versiones de Linux con un kernel 2.4.x.

El primer paso de la instalación del software sobre el aparato de experimentación ha sido descargar el archivo Xorp-1.3.tar.gz desde la sección Download Code del sitio principal antes indicado, también se puede obtener el archivo mediante CVS (Concurrent Version System) [42].

En nuestro caso el uso de la versión CVS del código se lo realizó en las fases de actualización del software.

El uso de CVS permite una rápida evolución del código a partir de un código fuente común; los desarrolladores acceden, de hecho, a un depósito, donde están contenidos los archivos fuente, y de donde cada uno obtiene una copia en donde puede aplicar modificaciones.

Una vez completada la versión modificada de código esta es enviada de nuevo al servidor CVS que entre las varias tareas que tiene, junta las modificaciones. Una vez descomprimido e instalado el archivo creamos una simple configuración del ruteador, mostrada en figura 4.2, de modo tal que cuando se lo inicie tenga un archivo de autoarranque válido.

El archivo ha sido grabado como config.boot, que al inicio será cargado como archivo por defecto por xorp_rtmgr.

Una vez activo, Xorp pone a disposición dos modalidades operativas de interfaz, llamadas Modo Operacional (*Operational Mode*) y Modo de Configuración (*Configuration Mode).*

La primera permite monitorear las operaciones y el estado del ruteador, la segunda permite al usuario ver, modificar y actualizar en tiempo real la configuración del ruteador.

Generalmente la modalidad Operacional no permite algún acceso privilegiado, o bien, nada de lo que se digite influenciará las características operativas del ruteador.

El Modo de Configuración permite modificar la configuración del ruteador por lo tanto, con el fin de proteger las características del aparato mismo de los usuarios no habilitados, el acceso al modo de Configuración es consentido solo a los usuarios pertenecientes al grupo “Xorp” del sistema operativo.

Los principales comandos en modalidad operativa son:

- Configure: permite acceder a la modalidad “operativa”
- Help: pone a disposición una guía de ayuda
- Quit: termina la ejecución de la shell de Xorp
- Show: visualiza los parámetros que describen el estado de actividad del ruteador.

Analizando los comandos disponibles, utilizados con mayor frecuencia, en la modalidad “configuración” podemos citar:

- Create: permite crear un nuevo módulo del ruteador.
- Set: permite definir el valor de parámetros de los módulos ya existentes.
- Disable: deshabilita interfaces físicas y lógicas, direcciones asociadas a interfaces o protocolos de enrutamiento mediante la simple asignación a la variable *disable* del valor verdadero (*true*).
- Show: visualiza las características actualmente activas en el ruteador.

La configuración del ruteador puede realizarse de manera simple e intuitiva gracias a su estructura de árbol, similar a la estructura de directorio en un archivo de sistema Unix.

En Modo de Configuración el prompt de los comandos es típicamente precedido de una línea que indica en que parte del árbol de configuración nos encontramos.

Una vez creada o modificada la configuración, es importante, hacer efectivas todas las variaciones, efectuando el comando “commit”.

A este punto es posible grabar la configuración, con el comando guardar (save), un resultado idéntico hubiéramos obtenido sin la ayuda de la modalidad de Configuración mediante la escritura, de todas la nuevas directivas, en un simple archivo de texto.

Con el fin de la correcta activación de xorp_rtmgr, la configuración deberá contener la declaración de al menos una interfaz del ruteador y el planteamiento del FEA.

Debemos tomar en cuenta que el ruteador utilizará solo las interfaces que configuremos; inclusive los protocolos agnósticos hacia las interfaces, como BGP, no aceptarán rutas que tengan como ruteador de siguiente paso una interfaz no configurada.

En lo que concierne al FEA, en cambio, es necesario evidenciar que dicho módulo debe ser explícita y separadamente habilitado para el reenvío de tráfico IPv4 e IPv6.

En la configuración de Xorp se debe tener presente, que cada interfaz lógica, presente en el sistema debe ser considerada al par de una interfaz física.

También desde el punto de vista del reenvío, las interfaces físicas que incluyen unidades lógicas se comportarán del mismo modo de puertas múltiples.

En particular el ruteador podría enviar paquetes entre diversos "canales" asociados a la misma interfaz. Por esta razón los desarrolladores de Xorp, con el fin de permitir un proceso de configuración intuitivo, han decidido distinguir entre interfaces físicas indicada como "*interfaces*", e interfaces lógicas, correspondientes a las interfaces virtuales e indicadas con el acrónimo "vifs".

En figura 4.2 se muestra un simple ejemplo de un archivo de configuración de Xorp, y como expondremos a continuación la sintaxis de este archivo es muy simple a la utilizada por los ruteadores Juniper.

En el archivo de ejemplo podemos notar como la declaración de una interfaz a la que es asociada una interfaz virtual y asignada una dirección IPv4, es además activado el FEA.

Ahora analizaremos con detalles la sintaxis de configuración de las interfaces de red.

N.B: para cada interfaz, dirección, protocolo y de manera general para cada macro bloque son configurables una serie de parámetros de los cuales nos limitaremos a nominar los de interés propio; para un completo conocimiento de los mismo ver [43].

```
interfaces {
interface eth1 {
        description:"data interface"
        disable:false
        vif eth1 {
                disable:false
                address 192.168.6.4 {
                        prefix-length: 24
                        broadcast: 192.168.7.255
                        disable:false
                        }
                }
        }
}
fea {
        targetname: "fea"
        unicast-forwarding4 {
                disable: false
                }
        }
protocols {
         ospf4 {
                router-id: 192.168.11.3|
                area 0.0.0.0 {
                        area-type: "normal"
                        interface eth1 {
                                link-type: "broadcast"
                                vif eth1 {
                                        address 192.168.6.4
                                }
                        }
                }
        }
}
```

Fig 4.2: Ejemplo de un archivo config.boot

Interfaces: contiene todas las características referentes a la configuración de las interfaces al interno del archivo de configuración de Xorp.

Interface: contiene la configuración de una particular interfaz; el parámetro es el nombre de la interfaz misma.

Description: es una pequeña descripción de la interfaz, utilizada por los operadores, es facultativa.

Disable: esta bandera habilita o deshabilita la interfaz para el enrutamiento y el reenvío; asume valores falso (*false*) o verdadero (*true*) con el efecto que al deshabilitar la interfaz, ésta es considerada come si no existiera en el archivo de configuración.

Vif: configura una interfaz virtual sobre la interfaz física correspondiente; el parámetro es el nombre de la vif y es el mismo que la FEA conoce.

Disable: corresponde a las funcionalidades descritas antes.

Address: especifica la dirección IP para esta vif. Una sola vif puede tener más direcciones IP, tanto Ipv4 como Ipv6.

prefix-lenght: define la longitud de la máscara de red de la red conectada a esta interfaz.

Broadcast: representa la dirección broadcast de la subred correspondiente a la dirección de la vif; es obligatorio.

Disable: esta bandera habilita o deshabilita la dirección IP asignada a esta vif.

Fea: contiene las directivas de la configuración de las funcionalidades del forwarding engine.

Unicast-forwarding4: directiva utilizada para configurar el reenvío Ipv4.

Disable: asume el valor "*true*" o "*false*" deshabilitando o habilitando un reenvío unicast Ipv4 sobre el ruteador.

Protocols: plantea la configuración de los protocolos de enrutamiento del ruteador Xorp.

Ospf4: contiene las directivas del protocolo OSPF.

Router-id: el parámetro es una dirección IPv4 o IPv6 asociado al ruteador de software como su identificación. A un mismo ruteador-*id* corresponderán varias vifs activas, en donde sus direcciones serán del todo independientes a la ligada al ruteador-*id*.

Area: Este campo define el área de pertenencia del ruteador OSPF. Define la configuración de las interfaces activas al interno de el área determinada.

Area-type: Define el tipo del área apenas mencionada. El parámetro puede asumir los siguientes valores "*normal*", "*stub*" o bien "nssa" (*not-so-stubby*).

Interface: Permite el cumplimiento de la descripción de una interfaz, declarada en precedencia, con elementos útiles al protocolo.

Link-type: provee una indicación sobre el tipo del enlace a instaurar con los vecinos al interno del área. El parámetro puede asumir los valores "*broadcast*", "p2m" (punto-multipunto) y "p2p" (punto-punto).

Los campos vif y address explicamos previamente.

4.4 Distancias administrativas

Los ruteadores IP adoptan las decisiones de enrutamiento en base a un mecanismo de correspondencia del prefijo de mayor longitud; esto significa que un ruteador, teniendo varias rutas para una misma dirección, utiliza la que tiene el prefijo mayor.

Puede suceder que un ruteador, en el cual están activos más de un protocolo de enrutamiento, conozca una misma ruta para más de un protocolo.

En este caso la regla de “longest prefix matching” no nos ayuda porque todos los prefijos tendrán la misma longitud, por tanto Xorp deberá adoptar otra política de selección.

Con este propósito, los ruteadores Xorp usan (como la gran mayoría de los ruteadores comerciales) el concepto de distancia administrativa: simplemente cada protocolo de enrutamiento tiene una “distancia” asignada y, en el caso en que la misma ruta sea anunciada por más de un protocolo, ganará la del protocolo con distancia mínima.

La tabla V muestra los valores por defecto de las distancias administrativas utilizadas por Xorp:

Subredes directamente conectadas	0
Rutas estáticas	1
BGP (para externos)	20
OSPF	110
IS-IS	115
RIP	120
BGP (para internos)	200
FIB2MRIB	254

Tab V: Tabla de las distancias administrativas de Xorp

4.5 El proceso de recálculo de las rutas

Nuestra atención se focaliza en el algoritmo de recálculo de las rutas, que tiene inicio en el momento en que se realiza un cambio en la conformación de las redes (cambio de métrica, falla de enlace, etc).

Consecuentemente al evento arriba citado y a la recepción de un LSU que anuncia dicho cambio, se realiza un recálculo, mediante el algoritmo de Dijkstra, del mejor camino hacia cada destino.

El proceso de recálculo de las tablas de enrutamiento es, obviamente, un elemento crítico para el análisis de las prestaciones de un ruteador OSPF, en cuanto establece la velocidad de reacción a cambios topológicos.

Resulta, por lo tanto, de gran interés el estudio de las operaciones relacionadas a éste proceso y sus realizaciones en Xorp. En este sentido, hemos intentado analizar en el detalle el código fuente de las operaciones involucradas en dicho proceso.

El diagrama de la figura 4.3 muestra el desarrollo general del recálculo de los caminos óptimos hacia un destino, desde que recibe el LSA hasta que se actualiza la tabla de enrutamiento según los cálculos realizados por el Dijkstra.

En el momento del cambio topológico verificado al interno de una área de nuestra red, el ruteador directamente conectado al enlace que ha sufrido el cambio de métrica genera un LSUP, y lo envía a todos los ruteadores que forman parte del área al cual el LSU se refería.

Cuando el LSU es recibido en el ruteador Xorp se realizan las siguientes operaciones:

- Se extrae el paquete LSA que anuncia el cambio y se realiza el control del error (aquí finaliza el primer bloque de elaboración que hemos llamado “Recepción y primeras elaboraciones de un LSU”).
- Después de haber controlado que la actualización recibida no sea datada, y que no esté presente en el LSA Database, la elaboración entra en el segundo bloque, llamado “inicio de la elaboración de un nuevo LSA”.
- En el bloque, apenas mencionado el nuevo LSA es insertado en la base de datos y sucesivamente enviado a los vecinos mediante las interfaces del ruteador.
- Cuando se realiza ésta ultima operación, se programa el proceso de cálculo del camino más corto que en el caso de Xorp se realiza con un retardo, encerado por los realizadores a los 5 segundos.

En fin, paralelamente a un mensaje de reconocimiento respectivo a la correcta recepción y difusión de la información contenida en el LSA, inicia el cálculo efectivo de los caminos de menor costo, y la coherente actualización de la tabla de enrutamiento.

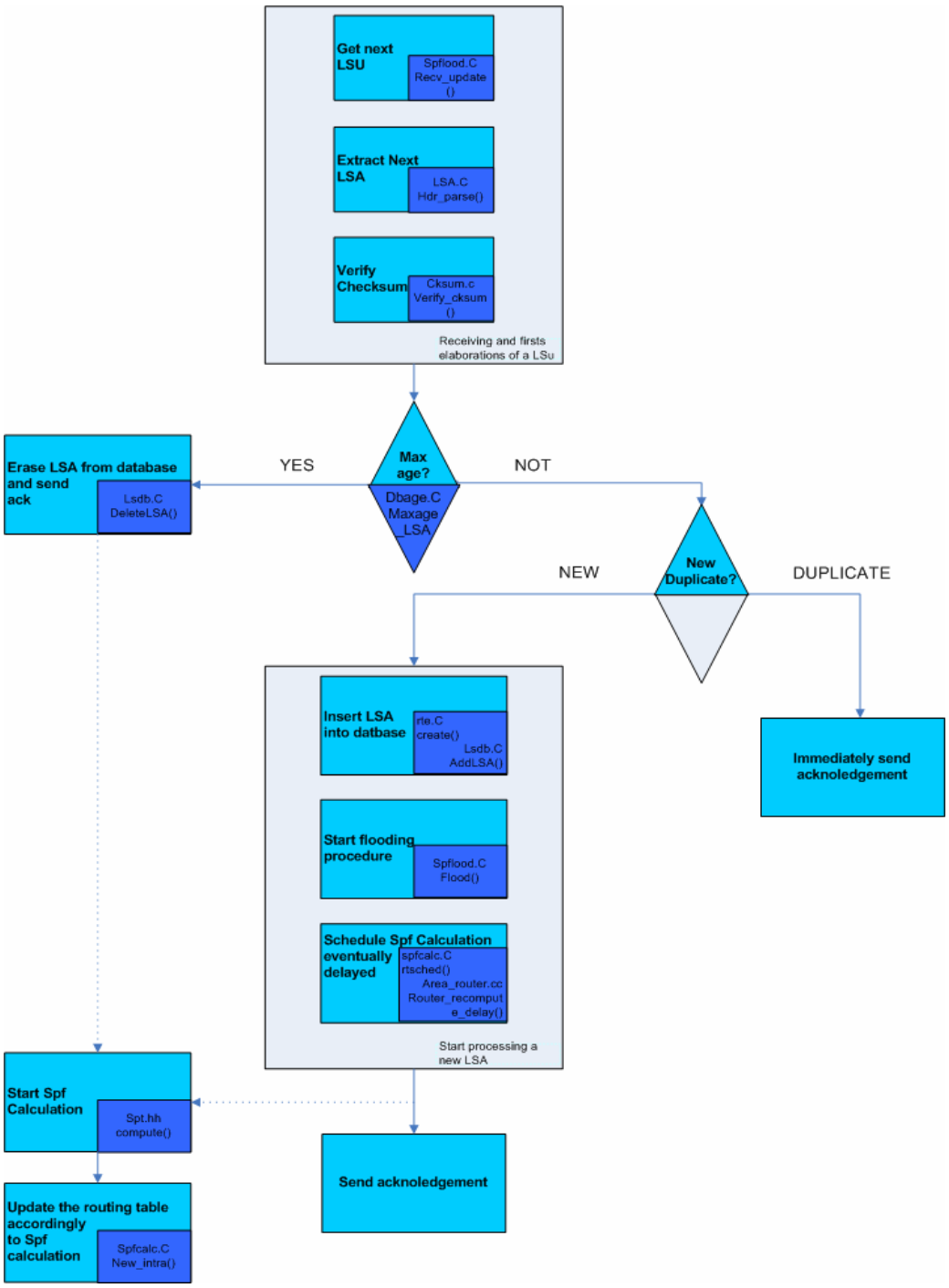


Fig 4.3: Diagrama de las operaciones realizadas sucesivamente a la recepción de un LSA

Una vez estudiado el funcionamiento general del algoritmo, pasamos al análisis de las funciones llamadas al interno de cada uno de los procedimientos presentados anteriormente, poniendo particular atención a su posición al interno del código, a las estructuras de datos utilizadas y a las sub-funciones llamadas.

En los gráficos que siguen, las flechas verticales indican la posición recíproca de las funciones al interno del algoritmo, mientras que las horizontales llevan a la descripción de pequeñas funciones involucradas. En las referencias a los archivos fuentes, descritos a continuación, nos referiremos a la carpeta indicada por el camino xorp/conrtib/ospfd/src/.

El diagrama de flujo del primer bloque analizado ha sido descrito en figura 4.4; funciones correspondientes a la macro acción "*get next* LSU" están contenidas al interno del archivo spflood.c y reagrupadas en una función llamada recv_update().

Al interno de ésta última es comparada la identidad del vecino, del remitente del paquete, y se efectúan las primeras elaboraciones del paquete, como en sección 13 de las especificaciones OSPF [44]. Estas funciones analizan todos los LSA internos al paquete LSU, el reconocimiento del paquete mismo, el control de eventuales copias ya presentes en la base de datos, y otras funciones que veremos en el detalle a continuación.

Al interno de LSA.C encontramos la función "extract next LSA" o bien hdr_parse(), mediante el cual el paquete recibido es convertido en el formato adecuado.

En fin, la última de las primeras elaboraciones efectuadas sobre el LSU recibido, es el cálculo del *checksum* en el encabezado, realizado por *verify_cksum*() definida al interno de cksum.C

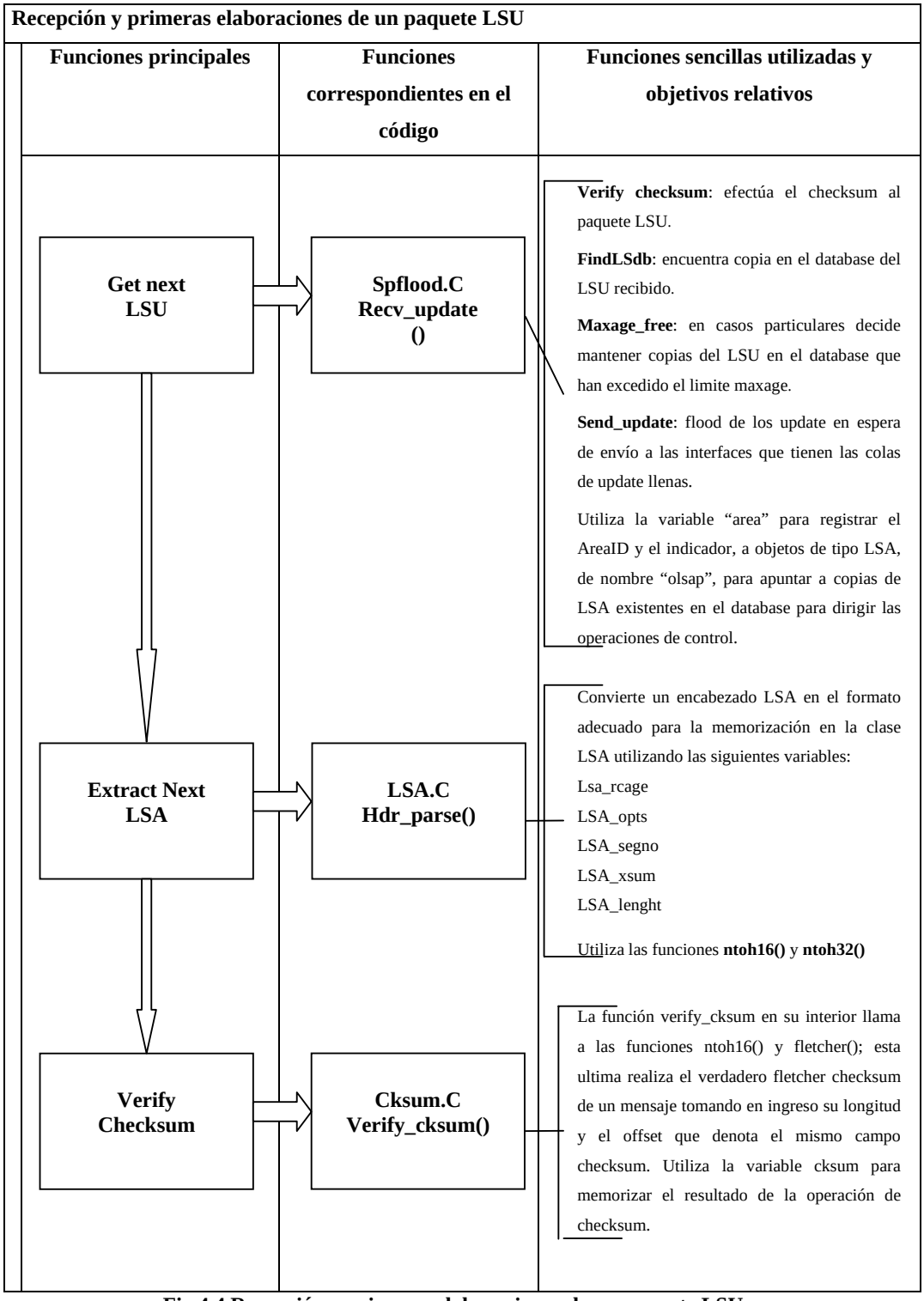


Fig 4.4 Recepción y primeras elaboraciones de un paquete LSU

El bloque de elaboraciones del nuevo LSA, analizado en figura 4.5, inicia con la inserción de la información contenida en el paquete LSA al interno de la base de datos, anticipado por la creación del ingreso mismo realizado por las funciones create() y AddLSA() contenidas respectivamente en los archivo de código rte.C y lsdb.C

Una vez realizada la actualización del LSA databas, el LSA mismo es propagado a los nodos adyacentes por las funciones flood() contenidas en el archivo spflood.C

Después de haber recibido, controlado, insertado en las base de datos y propagado el *update*, es el momento de programar el proceso de cálculo del camino óptimo hacia cada subred de destino, mediante la función rtsched() contenida en spfcalc.C

La programación del proceso de recálculo de las rutas es realizado por un retardo originado por la función router_recompute_delay() con el valor de 5 segundos (que será menor en las versiones sucesivas de Xorp); el archivo que contiene esta función, Area_router.cc se encuentra en la carpeta xorp/ospf/.

Siendo 5 segundos un intervalo de tiempo, a nuestro parecer, muy alto hemos decido recompilar el código cambiando a “0” este retardo de modo que podamos obtener una reacción inmediata al cambio de topología.

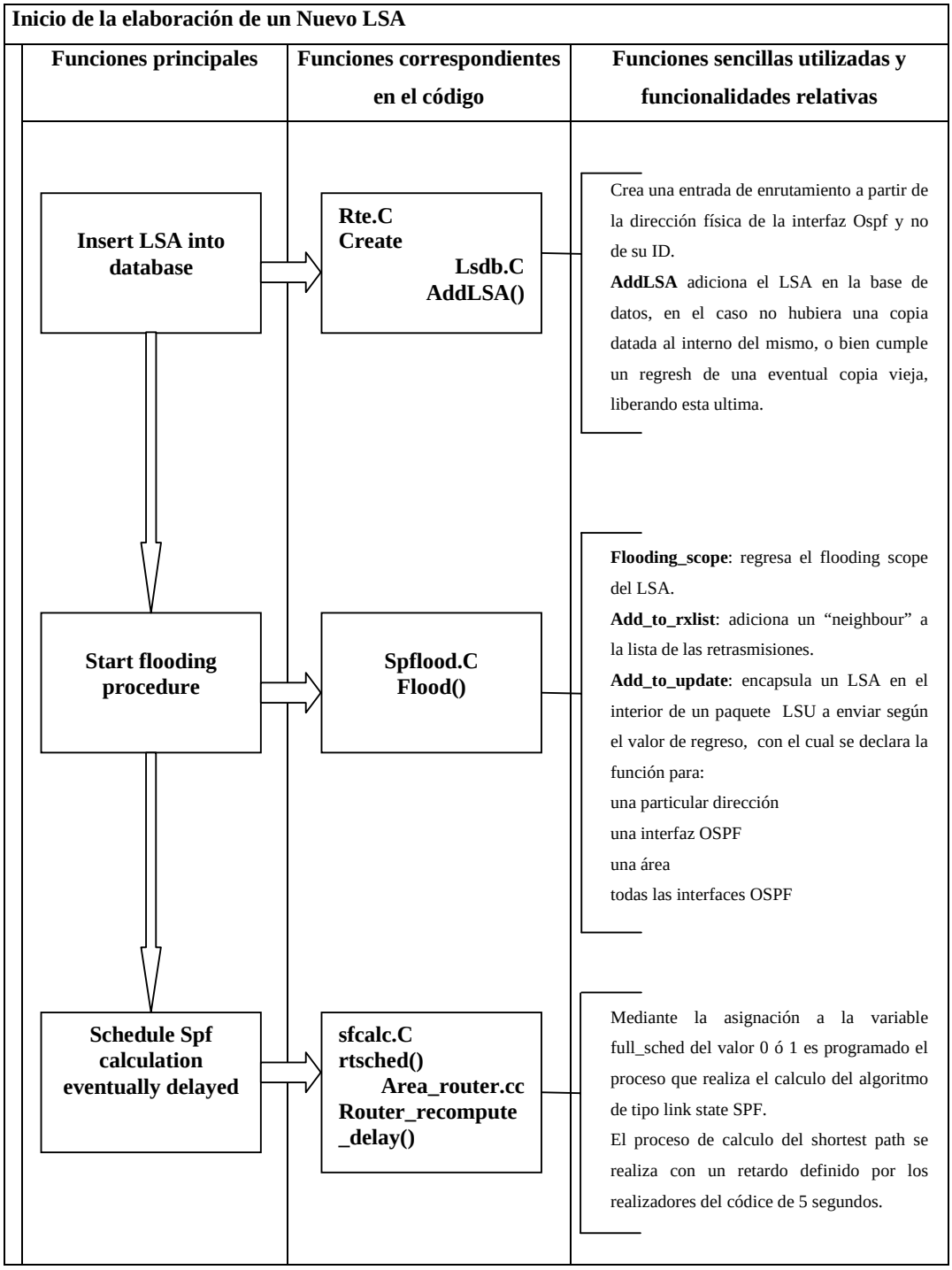


Fig 4.5: Inicio de la elaboración de un nuevo LSA

Las funciones ilustradas han sido recogidas e ilustradas en figura 4.6.

Delete_LSA() contenido en el interior de lsdb.C efectúa la eliminación de un LSA de la base de datos, en caso de que un control sobre su edad dé como resultado un valor superior a la edad máxima definida en *Maxage*; en el caso apenas introducido además la función se preocupa de restablecer las conexione de árbol, donde la estructura es memorizada mediante la clase *AVLtree*.

El archivo spt.hh, donde está la función compute(), se encuentra en la carpeta xorp/libproto/ e implementa el algoritmo de Dijkstra, construyendo, adicionando o sustrayendo nodos del árbol, coherentemente a la información sobre la topología global de la red, obtenida mediante la recepción de los LSA de los nodos vecinos.

Sucesivamente al cálculo de los caminos óptimos, la tabla de enrutamiento deberá actualizarse correctamente, según la información obtenida de los cálculos efectuados.

Una vez actualizada la tabla de enrutamiento estaremos listos para enviar los paquetes de tráfico según el camino actual de menor costo hacia cualquier destino.

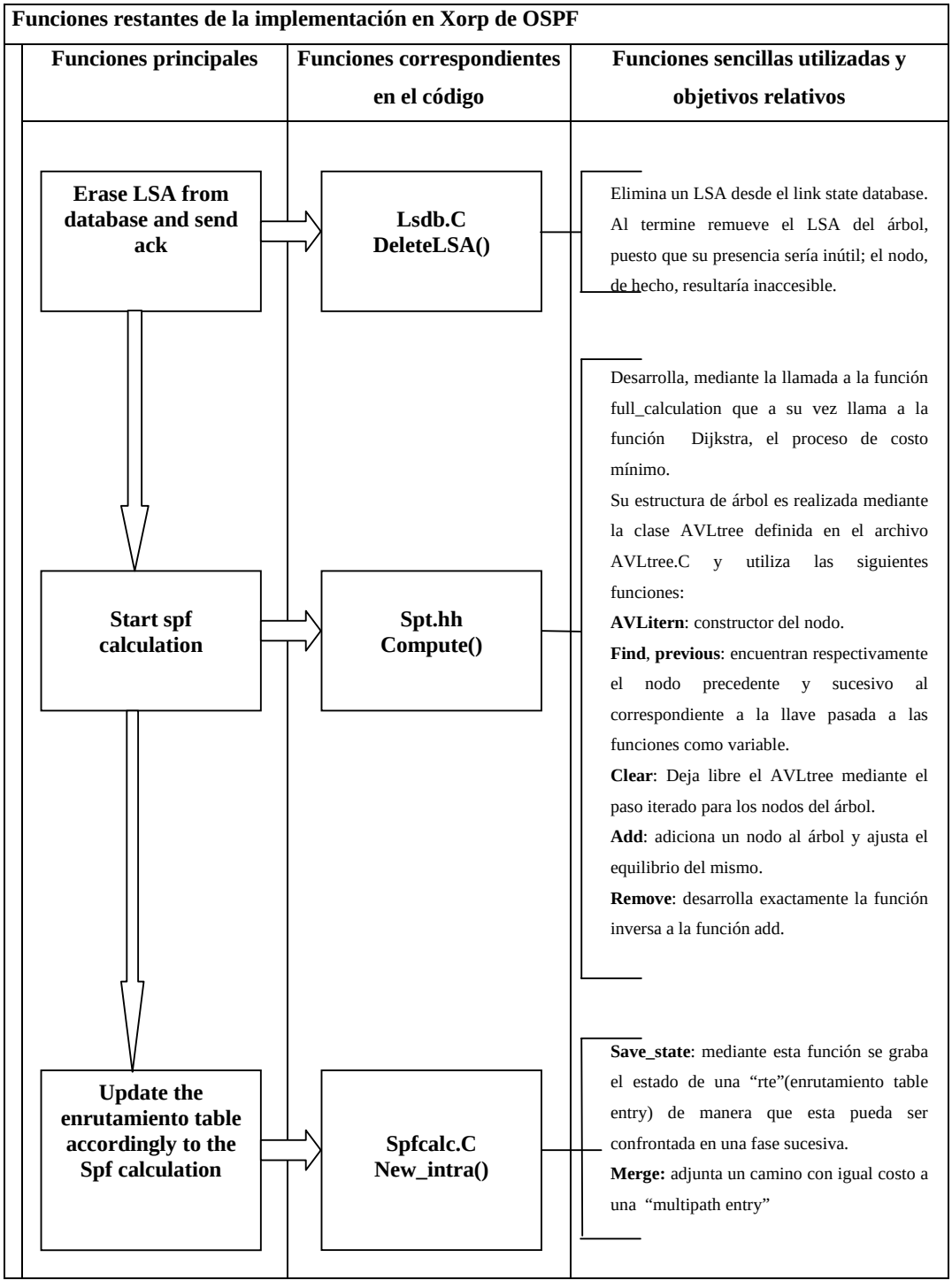


Fig 4.6: Funciones restantes de la implementación en Xorp de OSPF.

Capítulo 5: Resultados

5.1 Introducción

En este capítulo presentaremos los resultados obtenidos en el curso del desarrollo de esta tesis. Ilustraremos las pruebas experimentales efectuadas, describiremos las diferentes configuraciones de Xorp para los protocolos de enrutamiento BGP y OSPF, describiendo con detalles los resultados obtenidos.

En las pruebas referidas en esta tesis, el OR está constituido de la arquitectura de hardware descrita en el capítulo 2, mientras el soporte a los protocolos de enrutamiento es realizado mediante la suite software Xorp.

Los resultados de los experimentos han sido obtenidos mediante el uso del Router *Tester* 900 de *Agilent Technologies*, equipo profesional desarrollado específicamente para las pruebas de referencia (*benchmarking)* de Ruteadores IP.

El objetivo principal de las pruebas ejecutadas, es analizar, estudiar y validar las funcionalidades de control base BGP y OSPF de un Ruteador IP estándar, realizadas mediante Xorp.

Con este fin, una vez que instalamos Xorp sobre el OR y realizamos los primeros intentos de configuración, iniciamos a verificar funcionalidades de base, como la instauración y mantenimiento de la adyacencia, la actualización de la tabla de

enrutamiento y el intercambio de las bases de datos en el caso de OSPF, o de los mensajes de *update* en el caso de BGP.

Después realizamos el análisis de algunos índices de prestación importantes, como por ejemplo el tiempo de convergencia.
Todos los equipos utilizados para tal objetivo, serán descritos a continuación.

5.2 Instrumentos hardware

5.2.1 Router Tester

Para ejecutar pruebas sobre el plano de control, hemos utilizado el Router Tester 900 de la *Agilent Technologies* (ART) [45].

Dicho aparato representa un punto de referencia en las pruebas de las capacidades de generación de tráfico y de emulación de los protocolos para dispositivos de red.

Fig 5.1: Módulos que componen el Chassis del Router Tester 900

5.2.1.1 Hardware

Al interior de las ranuras del sistema de control pueden añadirse las *Test Card* o las tarjetas de emulación de protocolos, disponibles en soportes diferentes para la conectividad Ethernet (10/100, 1 Gb e 10 Gb).

El ART está constituido de dos módulos, cada uno en grado de administrar algunas interfaces del mismo tipo:

- Un módulo con 16 interfaces Fast-Ethernet. Las interfaces son indicadas con las siglas 101/1 y 101/2
- Un módulo con 4 interfaces Gigabit Ethernet (1000BaseLX), indicadas con las siglas 1A, 1B, 1C e 1D.

Los módulos se conectan mediante una LAN de tipo Fast Ethernet (100BaseTX) a un PC Windows que funciona como Controlador de Sistema, y sobre el cual opera el software que administra la interfaz de usuario.

5.2.1.2 Funcionalidades

Cada una de las puertas activas del Tester es conectada físicamente a una de las puertas del ruteador que van a ser examinadas, y para esto se asumen los acrónimos de *System Under Test* (SUT) o *Device Under Test* (DUT).
El ART está en grado de proveer las siguientes funcionalidades:

- *Generación y análisis de tráfico*

  Cada una de las puertas del ART puede generar y transmitir al SUT uno o más flujos de paquetes IP con diferentes características dinámicas y de contenido (dimensión de paquetes, contenido de los diversos campos de

los encabezados de los protocolos de nivel 2 y 3). Los SUT tramitan cada paquete recibido, según el destino final y el contenido de la propia tabla de reenvío, hasta una de las puertas del Tester, que de esta manera puede evaluar los diferentes parámetros de prestación como el rendimiento de procesamiento (*throughput)* y la latencia.

- *Simulación de topologías de redes virtuales*

  El ART provee el soporte para la emulación de los diferentes protocolos de enrutamiento, mediante la realización de uno o más ruteadores virtuales introducidos a sus puertas físicas. Los ruteadores virtuales pueden ser conectados entre ellos, creando de esta manera topologías "virtuales". Es posible crear topologías de red mucho más complejas (centenas de ruteadores/enlaces).

- *Análisis de paquetes*

  El ART está en grado de realizar la captura de paquetes (*packet sniffing*) recibidos sobre una o más puertas, y de elaborar el contenido permitiendo así conocer y analizar los mensajes intercambiados entre el SUT y el Tester relativos a una sesión de protocolo de enrutamiento.

5.3 Configuración de BGP con Xorp

Las primeras actividades experimentales, desarrolladas en esta tesis, han considerado la configuración y verifica del correcto funcionamiento de Xorp, en presencia de simples topologías, con los protocolos OSPF y BGP.

Hemos, por lo tanto, iniciado con algunas pruebas que conciernen el proceso de instauración de la adyacencia entre 2 ruteadores BGP que pertenecen a sistemas autónomos diferentes (AS *autonomous system*).

```
interfaces {
     restore-original-config-on-shutdown: false
    interface eth1 {
            description: "data interface"
            disable: false
            vif eth1 {
                    disable: false
                    address 10.0.3.1 {
                    prefix-length: 24
                    broadcast: 10.0.3.255
                    disable: false
                    }
            }
}
    interface eth2 {
            description: "data interface"
            disable: false
            vif eth2 {
                    disable: false
                    address 10.0.2.1 {
                    prefix-length: 24
                    broadcast: 10.0.2.255
                    disable: false
                    }
            }
            }
    }

    fea {
            targetname: "fea"
            unicast-forwarding4 {
            disable: false
            }

    }

    protocols {
            bgp {
            bgp-id: 10.0.2.1
            local-as: 65001
            peer 10.0.2.2 {
                    local-ip: 10.0.2.1
                    as: 65002
                    next-hop: 10.0.2.1
                    local-port:179
                    peer-port: 179
                    holdtime: 120
                    }
            }
    }
```

Fig 5.2: Ejemplo de un archivo configbgp.boot

La figura 5.2 representa un archivo de configuración BGP, en donde los campos más importantes son descritos a continuación:

Interfaces: contiene todas las impostaciones relativas a la configuración de las interfaces en el interior del archivo de configuración de Xorp.

Interface: contiene la configuración de una interfaz específica; el parámetro y el nombre de la interfaz misma.

Description: es una pequeña descripción de la interfaz, utilizada por los operadores, es facultativa.

Disable: esta bandera habilita o deshabilita la interfaz para el enrutamiento y el reenvío; asume valores "*false*" o "*true*" con el efecto que al deshabilitar la interfaz, ésta es considerada come si no existiera en el archivo de configuración.

Vif: configura una interfaz virtual sobre la interfaz física correspondiente; el parámetro es el nombre de la vif y es el mismo que la FEA conoce.

Disable: corresponde a las funcionalidades descritas antes.

Address: especifica la dirección IP para esta vif. Una sola vif puede tener más direcciones IP, tanto Ipv4 como Ipv6.

prefix-lenght: define la longitud de la máscara de red de la red conectada a esta interfaz.

Broadcast: representa la dirección broadcast de la subred correspondiente a la dirección de la vif; es obligatorio.

Disable: esta bandera habilita o deshabilita la dirección IP asignada a esta vif.

Fea: contiene las directivas de la configuración de las funcionalidades del motor de reenvío.

Unicast-forwarding4: directiva utilizada para configurar el reenvío Ipv4.

Disable: asume el valor "*true*" o "*false*" deshabilitando o habilitando un *forwarding unicast* IPv4 sobre el ruteador.

Protocols: plantea la configuración de los protocolos de enrutamiento del ruteador Xorp.

Bgp: contiene las directivas de configuración del protocolo BGP.

Bgp-id: el parámetro es una dirección IPv4 o IPv6 que identifica al ruteador BGP.

Local-as: Este campo contiene el número del sistema autónomo (AS) al cual éste ruteador pertenece. Es un número entero de 16 bits.

Peer: Establece una conexión BGP con otro ruteador. Se usa la dirección IP del ruteador con el cual se desea conectar como identificación del par.

Para conexiones IBGP se usa la dirección IP del loopback

Para conexiones EBGP se usa la dirección IP del ruteador con el cual se desea conectar.

local-ip: dirección IP que permite conexiones BGP con este par. Campo obligatorio. Debe ser igual a la dirección IP del par para esta conexión.

As: número del AS de este par, es un número entero de 16 bits.

Next-hop: dirección IP del ruteador vecino. Específico para conexiones EBGP.

Local-port: BGP establece sus conexiones TCP mediante la puerta 179.

Peer-port: igual al *local-port*.

Holdtime: Tiempo que BGP emplea para establecer una conexión con este par.

Dos ruteadores que usan BGP, inician una conexión utilizando TCP (puerta 179) y se definen "pares BGP". Una sesión BGP entre dos pares BGP de sistemas

autónomos diferentes (figura 5.3) es llamada Sesión BGP Externa (EBGP), y una sesión BGP entre pares BGP que se encuentran en el mismo AS (figura 5.4) es llamada Sesión BGP Interna (iBGP).

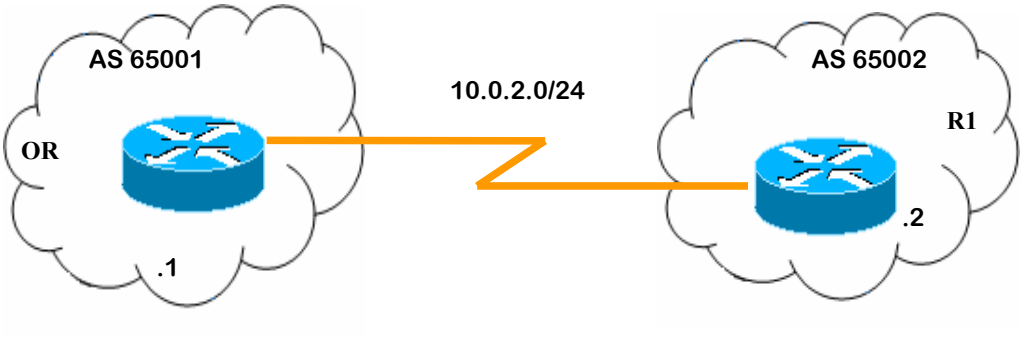


Fig 5.3: Conexión BGP entre ruteadores de AS diferentes.

Una ruta es considerada como la unidad de información que une la fuente con su destino. La ruta tiene la capacidad de aprender las propiedades y características de las conexiones BGP; de tal manera, cada BGP ruteador puede establecer la mejor ruta hacia el destino deseado, aunque si existen algunos caminos.

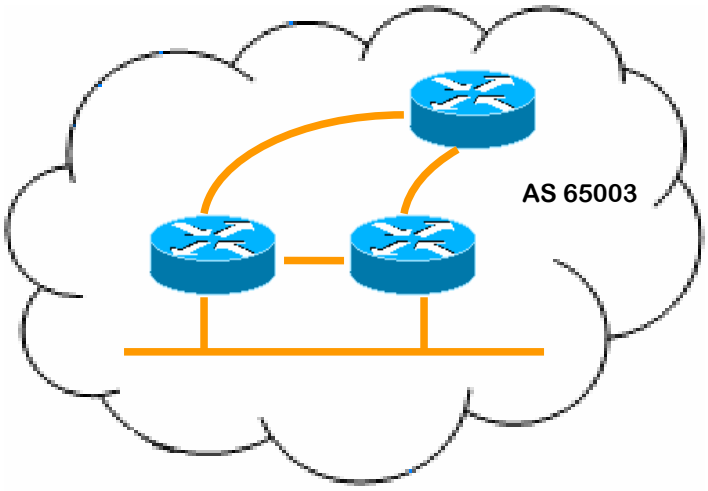


Fig 5.4: Conexión BGP entre ruteadores en el mismo AS.

5.3.1 Monitoreando BGP

Como arriba explicamos, figura 5.3 representa una conexión entre ruteadores de AS diferentes, en nuestro caso el OR se encuentra en el AS 65001 y un ruteador virtual R1 en el AS 65002. Hemos conectado una de las interfaces del OR a una de las puertas Gigabit del Router Tester.

En la primera prueba, conexión BGP 1 ruta, hemos simulado un ruteador virtual (R1) detrás de la puerta del router Tester (figura 5.5); y en la prueba conexión BGP 10 rutas, detrás de la misma puerta del Router Tester, hemos simulado 10 rutas diferentes para alcanzar R1 (figura 5.6).

Con la ayuda de la shell de Xorp hemos podido monitorear estas conexiones, verificar el estado de las conexiones BGP y de la tabla de enrutamiento.

Para entender mejor una conexión BGP hemos realizado pruebas de la instauración de una sesión BGP entre dos ruteadores cuando existe solo un camino para llegar al destino (figura 5.3) y cuando existen múltiples caminos, en nuestro caso hicimos una prueba con 10 rutas (figura 5.5). La configuración Xorp usada para esta prueba es la mostrada en figura 5.2.

5.3.1.1 Conexión BGP con 1 ruta

Con la configuración efectuada y el Router Tester activo, hemos verificado la correcta adquisición de la información BGP por parte de Xorp.

Para tal propósito, hemos usado la shell de Xorp para ver la lista de pares conectados, sus estadísticas y la tabla de las rutas BGP resultantes.

En particular, el siguiente comando (figura 5.5) nos permite visualizar la lista de pares declarada en el archivo de configuración:

```
root@openrouter> show bgp peers
        Peer 1: local 10.0.2.1/179 remote 10.0.2.2/179
```

Fig 5.5: Lista de pares declarada en el archivo de configuración Xorp

Para visualizar los detalles de los pares conectados (figura 5.6), se utiliza:

```
root@openrouter> show bgp peers detail
        Peer 1: local 10.0.2.1/179 remote 10.0.2.2/179
        Peer ID: 10.0.2.2
        Peer State: ESTABLISHED
        Admin State: START
        Negotiated BGP Version: 4
        Peer AS Number: 65002
        Updates Received: 2,  Updates Sent: 0
        Messages Received: 6,  Messages Sent: 4
        Time since last received update: 63 seconds
        Number of transitions to ESTABLISHED: 1
        Time since last entering ESTABLISHED state: 63 seconds
        Retry Interval: 120 seconds
        Hold Time: 90 seconds,  Keep Alive Time: 30 seconds
        Configured Hold Time: 120 seconds,  Configured Keep Alive Time:
        40 seconds
        Minimum AS Origination Interval: 0 seconds
        Minimum Route Advertisement Interval: 0 seconds
```

Fig 5.6: Detalle de los pares de una conexión BGP

Analizando la estructura de una conexión, podemos obtener la siguiente información:

Peer1:

local: IP del par ruteador

remote: IP del ruteador con el cual se desea conectar.

PeerID: Dirección IP del ruteador remoto

Peer State: Estado en el que se encuentra el par.

Negotiated BGP Version: Versión BGP a usar en la conexión.

Peer AS Number: Número del AS del par remoto.

Updates received: Número de *updates* que anuncian al par.

Retry Interval: Número de segundos dentro los cuales una conexión BGP debe ser establecida.

Hold Time: El Hold Time indica el tiempo máximo que puede transcurrir entre la recepción de mensajes "*keepalive*" y/o "*update*" sucesivos. Al término del Hold es posible rechazar la conexión.

Keep Alive Time: Número de segundos transcurridos en los cuales el remitente envía un mensaje "*keepalive*" y reinicia el reloj. Este reloj es utilizado para mantener los pares activos. Sin embargo, no es usado si el valor negociado de este último es igual a cero.

Podemos ver la tabla de enrutamiento BGP (figura 5.7) mediante el comando:

```
root@openrouter> show bgp routes
        Status Codes: * valid route, > best route
        Origin Codes: i IGP, e EGP, ? incomplete

          Prefix          Nexthop          Peer        AS Path
          ------          -------          ----        -------
        *> 10.0.2.0/24      10.0.2.2         10.0.2.2    65002 e
```

Fig 5.7: Tabla de enrutamiento de una conexión BGP

Analizando la estructura de una ruta, podamos notar como ésta está compuesta de 6 campos, con la siguiente semántica:

Status Code: Indica si la ruta es válida y si es considerada como la mejor ruta.

Prefix: Indica la dirección base de la red (dirección 10.0.2.0) y la longitud de prefijo (24 bits).

Nexthop: La dirección IP del ruteador vecino.

Peer: La dirección IP del ruteador BGP que ha enviado esta ruta.

AS Path: El número del AS atravesado para alcanzar este ruteador.

Route's origin: Indica si el origen de la ruta es IGP o EGP.

5.3.1.2 Conexión BGP con 10 rutas

En esta prueba, representado en figura 5.8, existen algunas rutas para llegar al mismo destino. En este caso el OR deberá elegir una ruta para enviar los paquetes a R1. BGP permite especificar las políticas de enrutamiento y todos los parámetros necesarios para seleccionar el mejor camino.

Las políticas de enrutamiento BGP son dependientes de factores de origen económico y de gestión, y además toman en cuenta aspectos ligados a la seguridad.

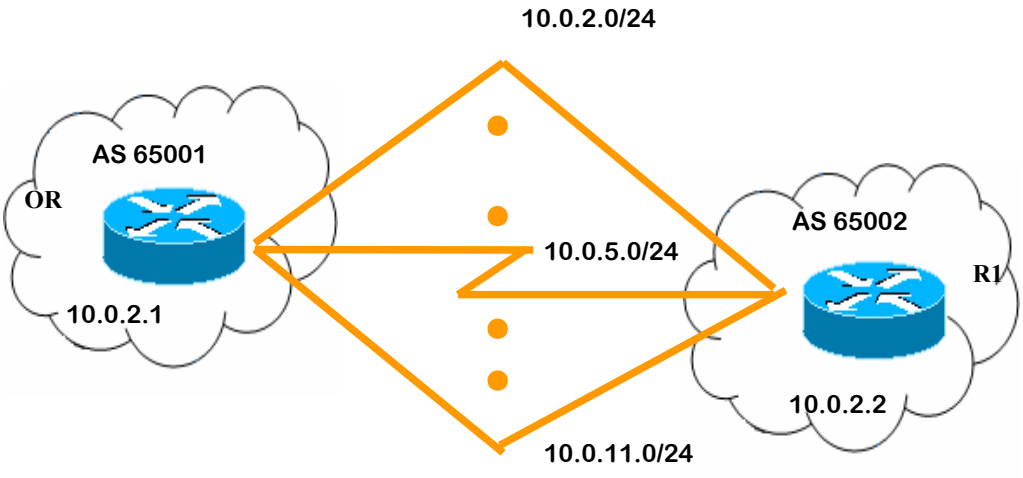


Fig 5.8: Conexión BGP entre ruteadores de AS diferentes con varios caminos.

También en este caso, hemos usado la consola de Xorp para verificar el correcto mantenimiento de las comunicaciones BGP.

Por lo tanto, debido a que tenemos a disposición varios caminos para llegar al mismo par R1, los detalles del par son análogos al precedente (figura 5.9), con la excepción de un alto número de *updates* debido al número de ruteadores sobre los diferentes caminos que anuncian al mismo par.

```
root@openrouter> show bgp peers detail
        Peer 1: local 10.0.2.1/179 remote 10.0.2.2/179
        Peer ID: 10.0.2.2
        Peer State: ESTABLISHED
        Admin State: START
        Negotiated BGP Version: 4
        Peer AS Number: 65002
        Updates Received: 11,  Updates Sent: 0
        Messages Received: 13,  Messages Sent: 4
        Time since last received update: 17 seconds
        Number of transitions to ESTABLISHED: 2
        Time since last entering ESTABLISHED state: 26 seconds
        Retry Interval: 120 seconds
        Hold Time: 90 seconds,  Keep Alive Time: 30 seconds
        Configured Hold Time: 120 seconds,  Configured Keep Alive Time: 40
        seconds
        Minimum AS Origination Interval: 0 seconds
        Minimum Route Advertisement Interval: 0 seconds
```

Fig 5.9: Detalle de los pares de una conexión BGP con múltiples caminos

Detallamos la tabla de enrutamiento (figura 5.10).

```
root@openrouter> show bgp routes
        Status Codes: * valid route, > best route
        Origin Codes: i IGP, e EGP, ? incomplete

         Prefix          Nexthop          Peer        AS Path
         ------          -------          ----        -------
        *> 10.0.2.0/24     10.0.2.2          10.0.2.2     65002 e
        *> 10.0.3.0/24     10.0.2.2          10.0.2.2     65002 e
        *> 10.0.4.0/24     10.0.2.2          10.0.2.2     65002 e
        *> 10.0.5.0/24     10.0.2.2          10.0.2.2     65002 e
        *> 10.0.6.0/24     10.0.2.2          10.0.2.2     65002 e
        *> 10.0.7.0/24     10.0.2.2          10.0.2.2     65002 e
        *> 10.0.8.0/24     10.0.2.2          10.0.2.2     65002 e
        *> 10.0.9.0/24     10.0.2.2          10.0.2.2     65002 e
        *> 10.0.10.0/24    10.0.2.2          10.0.2.2     65002 e
        *> 10.0.11.0/24    10.0.2.2          10.0.2.2     65002 e
```

Fig 5.10: Tabla de enrutamiento de una conexión BGP con múltiples caminos

5.3.2 Intercambio de mensajes BGP

Una vez establecida una conexión BGP entre pares de AS diferentes, y pudiendo monitorear el estado de esta conexión mediante la *shell* de Xorp, pasamos a analizar las modalidades de comunicación entre una pareja de pares conectados, y el proceso que siguen para establecer una sesión BGP.

Como vimos, BGP usa TCP como protocolo de transporte para el tráfico de señalización. Los pares desarrollan tres funciones principales:

- Establecen la conexión y concuerdan los parámetros de comunicación;
- Se intercambian información de alcance;
- Monitorean periódicamente el estado de los otros pares.

Establecida la sesión BGP, los pares se intercambian periódicamente mensajes "*keepalive*" para mantener la garantía de conectividad BGP.

El mecanismo de apertura de una sesión BGP está compuesto de seis diferentes estados (figura 5.11). Estos estados son: Idle (Inactivo), Connect (Conectado), Active (Activo), OpenSent (Apertura Enviada), OpenConfirm (Apertura Confirmada) y Established (Establecido). Cada par BGP atraviesa los estados descritos cuando intenta establecer y mantener viva una sesión con otro par.

El par de la sesión BGP (*speaker*) envía periódicamente (generalmente cada 60 segundos) mensajes *keepalive* de 19 bytes para mantener activa la conexión y evitar la culminación del Hold Timer (reloj relacionado al mantenimiento del estado activo de una sesión).

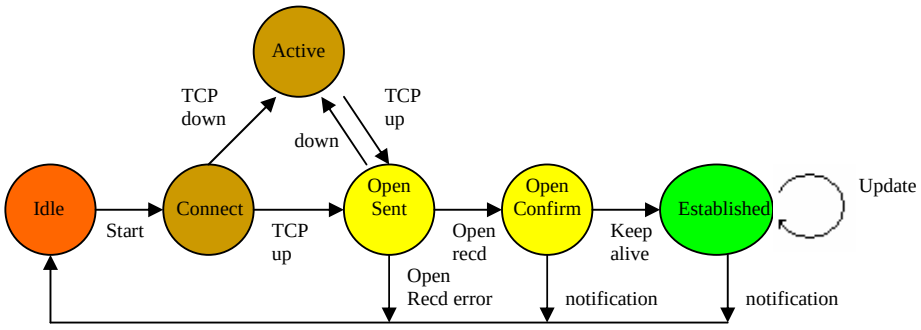


Fig 5.11: Estados de la conexión BGP

Ahora consideremos la situación ilustrada en figura 5.12, que muestra una interconexión BGP entre 3 AS diferentes.

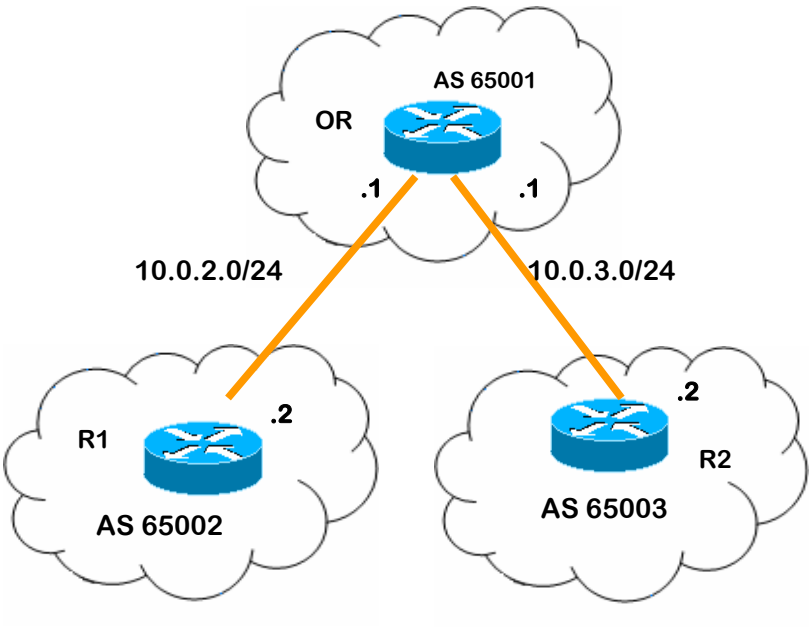


Fig 5.12: Conexión BGP entre ruteadores de 3 AS diferentes.

El OR se encuentra en el AS 65001 desea establecer una comunicación BGP con R1 y R2 que se encuentran en los AS 65002 y 65003 respectivamente. Cambiando prueba, debemos declarar el nuevo par en el archivo de configuración.

El nuevo archivo de configuración (figura 5.13) es referido a continuación (mostramos sólo las directivas BGP para brevedad):

```
protocols {
      bgp {
            bgp-id: 10.0.2.1
            local-as: 65001
            peer 10.0.2.2 {
                  local-ip: 10.0.2.1
                  as: 65002
                  next-hop: 10.0.2.1
                  holdtime: 120
                  }
            peer 10.0.3.2 {
                  local-ip: 10.0.3.1
                  as: 65003
                  next-hop: 10.0.3.1
                  holdtime: 120
                  }
            }
      }
```

Fig 5.13: Nuevo archivo de configuración configbgp.boot

BGP opera en términos de mensajes, que son enviados usando TCP. La RFC 1771 describe el formato y los campos de los mensajes BGP que dos ruteadores se intercambian para establecer una conexión entre ellos.

A continuación describiremos y analizaremos los mensajes BGP intercambiados entre el OR con los ruteadores R1 y R2 según nuestro archivo de configuración Xorp, y controlaremos la compatibilidad de los mensajes BGP obtenidos en nuestra prueba con los del estándar. La topología de los mensajes conocida por el estándar es:

- Open
- Keepalive
- Update

Cada mensaje BGP está precedido por un encabezado (figura 5.14) de 19 bytes. Según el tipo de mensaje, pueden existir o no datos después del encabezado. El tipo puede ser: Open (apertura), Keepalive, Update (actualización) y Notification (notificación). Antes de establecer una conexión entre un ruteador BGP con otro, es necesario definir los "vecinos BGP".

Una vez que se han establecido dos ruteadores como vecinos BGP, estos abren una conexión y se intercambian mensajes para abrir y confirmar la sesión BGP. En particular, el primer mensaje que el OR enviará a R1 y a R2 será el mensaje Open, usado para intercambiar información de configuración y negociar los primeros parámetros necesarios para la sesión BGP.

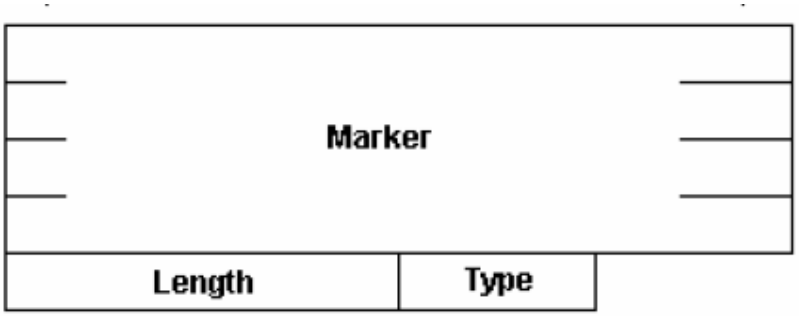


Fig 5.14: Encabezado de los mensajes BGP

El mensaje Open, enviado a ambos ruteadores está representado en figura 5.15 y contiene algunos parámetros, como la identificación BGP, el AS al que pertenece el ruteador y otros campos que serán explicados a continuación:

Message Lenght: La longitud del mensaje *open*, 37 bytes.

Message Type: La identificación del mensaje, 1 byte para el mensaje O*pen*.

Version: La versión BGP actual, versión 4

Autonomous System (AS) number of the sender: El número del AS al que pertenece el OR, 65001

Hold Time: Indica el número máximo de segundos que se deben esperar para recibir mensajes *Keepalive* y/o *Update* enviados por el OR.

BGP Identifier: La dirección IPv4 que identifica el ruteador BGP, 10.0.2.1

Optional Parameter Lenght: Indica la longitud del campo *Optional Parameters*.

Parameter Type: Especifica el tipo de parámetro

Parameter Length: Indica la longitud del campo *Parameter Value*.

```
Marker = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Message Length = 37
Message Type = Open (1)
Version = 4
Autonomous System (AS) Number Of The Sender = 65001
Hold Time (In Seconds) = 120
BGP Identifier = 10.0.2.1
Optional Parameter Length = 8
Parameter Type = Capabilities Advertisement (2)
Parameter Length = 6
Capability Code = Multiprotocol Extensions Capabilities (1)
Capability Length = 4
Address Family Identifier = IP (IP version 4) (0x1)
Subsequent Address Family Identifier = NLRI used for unicast forwarding
```

Fig 5.15: Mensaje Open.

Después que el OR ha enviado el mensaje Open a los ruteadores R1 y R2, las sesiones pasan al estado *OpenConfirm*. Si R1 y R2 aceptan la conexión BGP con

el OR, ambos enviarán un mensaje *Keepalive* (representado en figura 5.16). En el caso en el que ésta conexión sea rechazada se enviará un mensaje de error y la conexión TCP terminará. El mensaje *Keepalive* es intercambiado periódicamente entre los pares BGP, para que no finalice el Hold Timer; tal mensaje no debe ser transmitido con una frecuencia mayor a uno por segundo.

Fig 5.16: Mensaje keepalive.

Después que el mensaje *Keepalive* ha sido recibido por el ruteador remoto, las sesiones pasan al estado *Established* que es el único que permite a los pares anunciar sus propias rutas. En este punto el OR envía el primer mensaje de *update* a R1 (figura 5.17) y a R2 (figura 5.18) y recibe los update de R1 y R2. El mensaje *update* es usado para:

- Anunciar un solo camino válido y/o
- Remover varios caminos no válidos.

Después de la actualización de las tablas de enrutamiento, efectuado después de la recepción de los mensajes *update*, el mensaje de actualización es propagado a todos los ruteadores adyacentes con una excepción: una actualización recibida sobre un enlace interno al AS no es enviada a otros ruteadores del mismo AS.

```
Marker = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Message Length = 54
Message Type = Update (2)
Unfeasible Routes Length = 0
Total Path Attribute Length = 27
Attribute: Type = NEXT HOP (Length = 4)
        Flags = 0x40 - Well-known  Transitive      Complete  Length Field 1 Octet
            IP Address Of Border Router = 10.0.2.1
Attribute: Type = ORIGIN (Length = 1)
            Flags = 0x40 - Well-known  Transitive      Complete  Length Field 1 Octet
            Origin Of Path Info = EGP - Reachability Info Learned Via EGP (0x1)
Attribute: Type = AS_PATH (Length = 6)
                Flags = 0x40 - Well-known  Transitive      Complete  Length Field 1 Octet
                        Path Segment Type = AS_SEQUENCE: Ordered Set Of ASs (0x2)
                        Number Of ASs In Path Segment Value Field = 2
                        Autonomous System Number 0 = 65001
                        Autonomous System Number 1 = 65003
Attribute: Type = MULTI_EXIT_DISC (Length = 4)
                Flags = 0x80 - Optional    Non-transitive  Complete  Length Field 1 Octet
                Multiple Exit Point Descriminator = 42
Implicit Length For Network Layer Reachability Info = 4 Bytes
IP Address Prefix/Length = 10:0:3/24
```

Fig 5.17: Mensaje update que recibe R1.

```
Marker = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Message Length = 54
Message Type = Update (2)
Unfeasible Routes Length = 0
Total Path Attribute Length = 27
Attribute: Type = NEXT HOP (Length = 4)
            Flags = 0x40 - Well-known  Transitive      Complete  Length Field 1 Octet
            IP Address Of Border Router = 10.0.3.1
Attribute: Type = ORIGIN (Length = 1)
            Flags = 0x40 - Well-known  Transitive      Complete  Length Field 1 Octet
            Origin Of Path Info = EGP - Reachability Info Learned Via EGP (0x1)
Attribute: Type = AS_PATH (Length = 6)
            Flags = 0x40 - Well-known  Transitive      Complete  Length Field 1 Octet
            Path Segment Type = AS_SEQUENCE: Ordered Set Of ASs (0x2)
            Number Of ASs In Path Segment Value Field = 2
            Autonomous System Number 0 = 65001
            Autonomous System Number 1 = 65002
Attribute: Type = MULTI_EXIT_DISC (Length = 4)
            Flags = 0x80 - Optional    Non-transitive  Complete  Length Field 1 Octet
            Multiple Exit Point Descriminator = 42
Implicit Length For Network Layer Reachability Info = 4 Bytes
IP Address Prefix/Length = 10:0:2/24
```

Fig 5.18: Mensaje update que recibe R2.

El mensaje upda*t*e contiene varios campos, entre los cuales:

Message Lenght: La longitud del mensaje *update*, 54 bytes

Message Type: Identificación del mensaje, 4 bytes para el mensaje update.

Unfeasible Routes Length: Indica la longitud del campo *Withdrawn Routes.*

Total Path Attribute Length: Indica la longitud del campo *Path Attributes*. Si es cero, quiete decir que existen caminos por eliminar.

Attribute: Contiene información adjunta asociada a las rutas anunciadas.
Type: Consiste en un byte de bandera (figura 5.19) y un byte de código, del tipo de atributo.

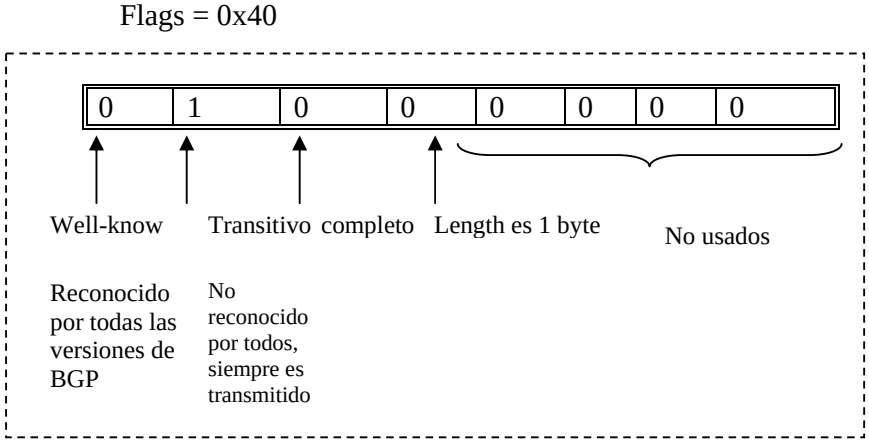


Fig 5.19: Byte de bandera del atributo tipo de una ruta BGP

Code: Campo de un byte, contiene el tipo de código del atributo.

NEXT HOP: Dirección IP del ruteador de siguiente salto a usar en el enrutamiento.

ORIGIN: Define el origen del camino de información.

AS_PATH: Los números de identificación de cada AS (2 byte) en la ruta hacia la red de destino.

AS_SEQUENCE: Lista de los AS que atraviesa para llegar al destino, es decir, para llegar desde R2 a R1, el *update* atraviesa el AS 65001 y después el AS 65003.

MULTI_EXIT_DISC: Este valor se puede usar para elegir entre tantas rutas hacia un AS (figura 5.20).

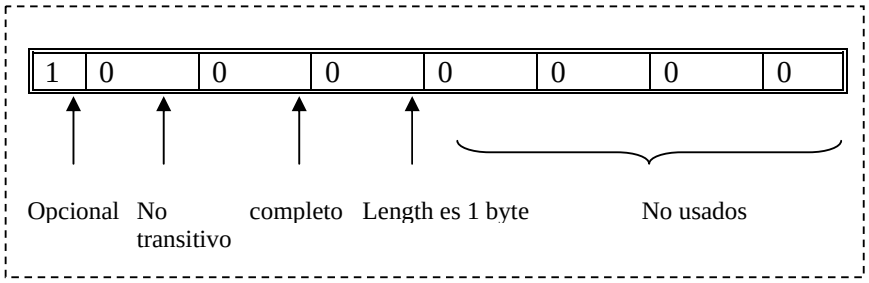


Fig 5.20: Byte de banderas del atributo discriminador de múltiples salidas

Implicit Length For Network Layer Reachability Info: Lista de prefijos de las direcciones IP de las rutas que deben ser añadidas a la tabla de enrutamiento.

5.3.3 Tiempo de convergencia

Desde este momento y hasta que la conexión se encuentre en el estado *Established*, los pares se intercambian información de enrutamiento propagando hacia otros pares eventuales cambios topológicos en la red.

Cuando un ruteador decide eliminar una o varias rutas, éste envía un mensaje update con campo Unfeasible Routes Length (Longitud de rutas irrealizables) con valor diferente de cero, indicando la presencia del Withdrawn Routes, que a su vez, contienen las rutas que han sido eliminadas. El ruteador que recibe éste

mensaje debe recalcular su tabla de enrutamiento BGP, eligiendo el nuevo camino hacia el destino.

Ahora que hemos aclarado el mecanismo de eliminación de una o más rutas, procederemos a efectuar una prueba para entender como reacciona nuestro sistema cuando es anunciada la eliminación de una ruta. El parámetro clave para éste tipo de análisis es el tiempo de convergencia, es decir el intervalo temporal que trascurre entre la recepción de un mensaje *update*, indicando un cambio topológico tal de modificar el siguiente salto para uno o más destinos, y el primer paquete IP recibido sobre el enlace directamente conectado al SUT y perteneciente al nuevo camino.

5.3.3.1 Descripción del experimento

Con el fin de analizar y validar el mecanismo de eliminación de una ruta en Xorp hemos realizado una prueba en la cual el ruteador 10.0.1.2 anuncia la red virtual 192.0.0.1 que puede ser alcanzada por R1 o por R2 (figura 5.21).

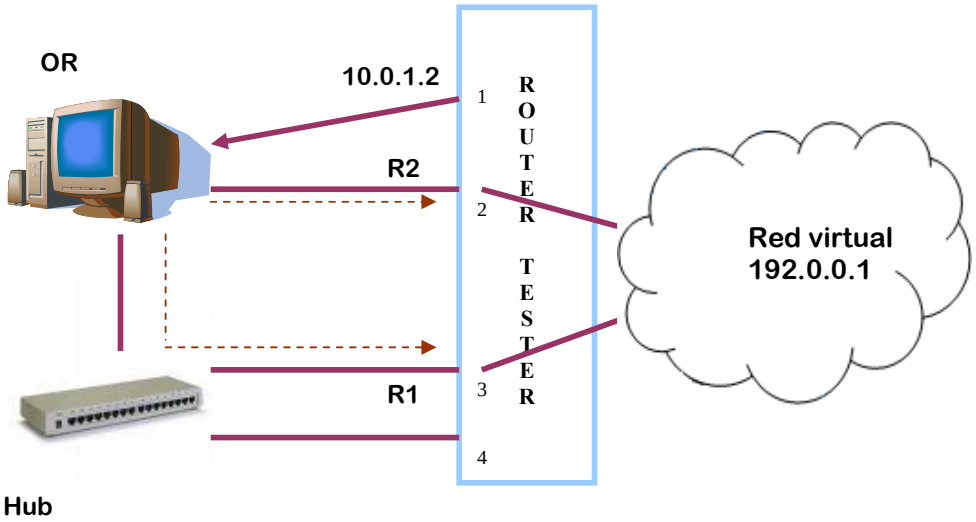


Fig 5.21: Topología de prueba para los experimentos correspondientes al tiempo de convergencia BGP

Como muestra figura 5.21 hemos usado 3 interfaces del Open Router, cada una conectada a una puerta Gigabit del Router Tester. Detrás de la puerta 1 hemos simulado el ruteador 10.0.1.2 que genera tráfico hacia la red virtual 192.0.01, alcanzable sea de R1 que de R2, ruteadores simulados detrás de la puerta 2 y 3 del Router Tester. Entre el ruteador R1 y el OR hemos colocado un Hub, empleado para capturar el tráfico desde el OR hacia R1 y viceversa.

5.3.3.2 Resultados

Una vez establecida una sesión BGP entre pares correspondientes, 10.0.1.2 comienza a enviar tráfico a la red virtual 192.0.0.1 y elige el camino vía R1. Cuando el enlace R1-red virtual es deshabilitado, R1 envía un mensaje *update* al OR indicando a R2, mediante un mensaje *update*, que ha ocurrido un cambio topológico, y ambos recalculan sus tablas de enrutamiento.

Cuando el OR actualiza su tabla de enrutamiento, elegirá R2 como el nuevo camino para enviar paquetes hasta 192.0.0.1

Con la ayuda del Hub conectado a R1 hemos logrado capturar el tráfico en ambos sentidos para analizar los paquetes BGP que atraviesan el enlace. Procediendo en este modo logramos obtener el tiempo en el que R2 recibe el primer paquete IP, una vez que el OR ha actualizado su tabla de enrutamiento con el nuevo camino.

En la figura 5.22 mostramos los resultados obtenidos en la simulación realizada según el esquema de figura 5.21.

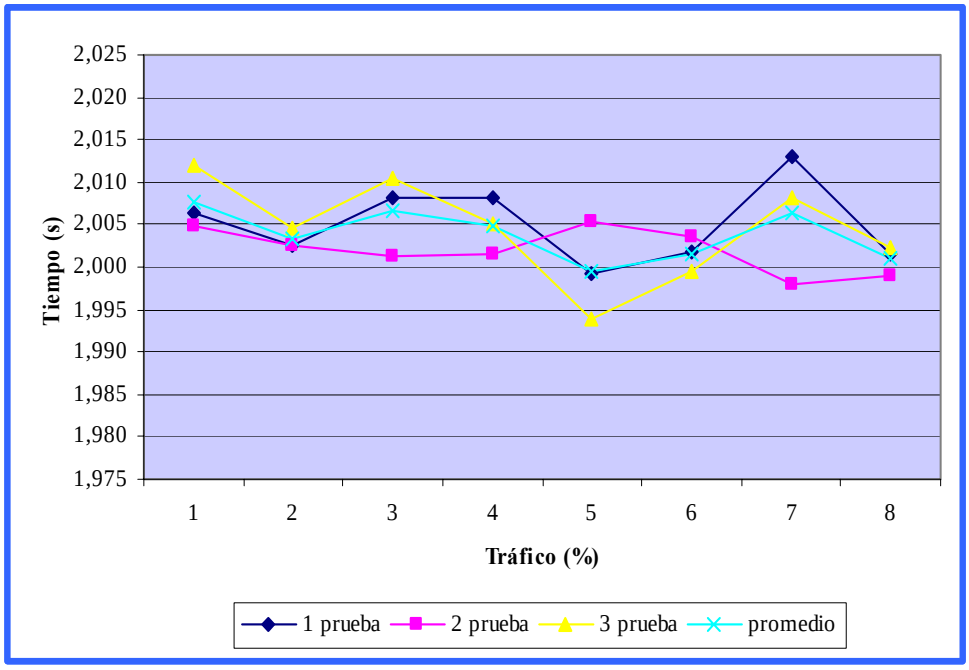


Fig 5.22: Tiempo de convergencia BGP

Como se ve en figura 5.22 hemos realizado tres pruebas con las mismas condiciones, variación de tráfico constante y eliminación del enlace R1-red virtual. Como podemos observar existe una variación de milisegundos entre los diferentes puntos debido a la variación de pérdida de paquetes, ésta pérdida no es constante depende de la capacidad de las colas y de la rapidez con que el Open Router sirve a las mismas, al recibir una menor cantidad de paquetes el Open Router reacciona de manera más rápida. La curva de la prueba 3 es considerada el peor de los casos por los valores altos obtenidos, debido a esta variación sacamos un promedio del tiempo de convergencia.

El tiempo de convergencia promedio es cercano a 2 segundos, tiempo razonable para recalcular un nuevo camino, considerando el hecho que BGP por eficiencia solo envía las diferencias de la actualización precedente.

Si usáramos una escala más extensa obtendríamos como resultado una línea recta; si analizamos este resultado podemos ver que en este caso el tiempo no depende mucho del tráfico pues los paquetes que ingresan a la cola del Open Router son servidos eficientemente lo que no da opción a pérdidas mayores de paquetes, y nos da una diferencia de milisegundos con el aumentar del tráfico

Otro parámetro que podemos medir para comprender el comportamiento de nuestro sistema es el tiempo que nosotros llamamos Δt *unfeasible* (Δt *irrealizable)*, es decir, el intervalo entre el mensaje update enviado desde R1 indicando que ha habido un cambio topológico (por ejemplo la eliminación del enlace R1) y el mensaje *update* que recibe R2. En la figura 5.23 mostramos los resultados obtenidos en la simulación realizada según el esquema de figura 5.21 al variar el tráfico, y en la figura 5.24 en ausencia de tráfico.

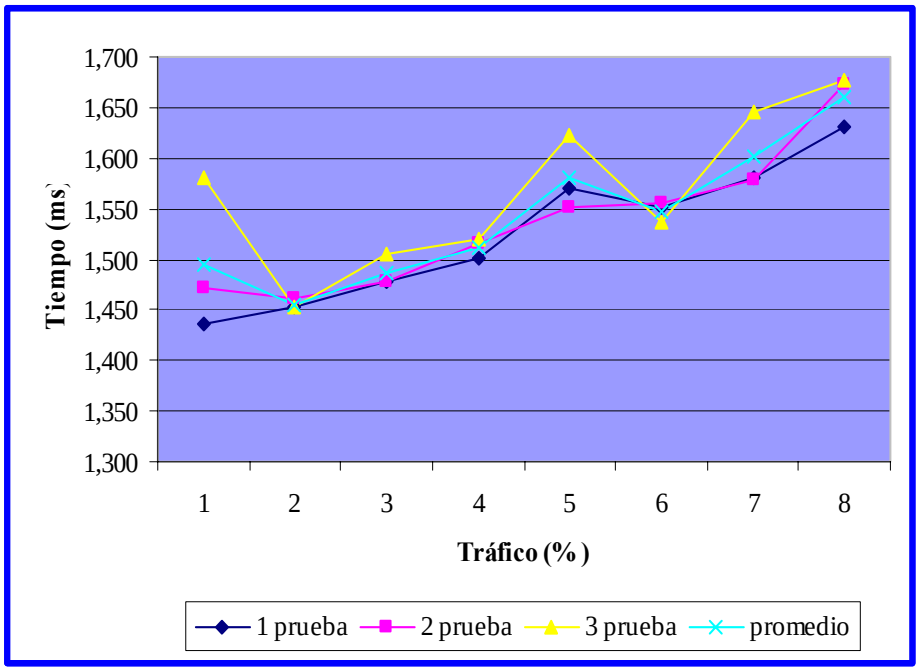


Fig 5.23: Δt unfeasible con tráfico

Las pruebas efectuadas para este experimento son las mismas pruebas del experimento “tiempo de convergencia”, como podemos ver la curva de la prueba 3 dio resultados altos en comparación a las otras pruebas (peor de los casos) por lo que tuvimos que sacar un promedio de dicho parámetro. Como podemos ver el Δt *unfeasible* aumenta al aumentar el tráfico. A causa de motivaciones legadas a la capacidad del buffer del router *tester Agilent* no hemos podido efectuar pruebas con tráfico más alto. Otra observación interesante concerniente al Δt *unfeasible*, es que resulta relativamente bajo (en el orden de los milisegundos) sea en la prueba con tráfico que en la prueba sin tráfico; éste fenómeno es explicable teniendo presente que la única acción cumplida por el OR cuando recibe el mensaje *update* de R1, es de transmitirlo a R2.

Éste se limita, por lo tanto, a propagar la información del cambio topológico para que R2 pueda actualizar su tabla de enrutamiento. Sucesivamente después de que ambos han cambiado sus propias tablas de enrutamiento, proceden a intercambiarse los mensajes update para trasferirse información de enrutamiento.

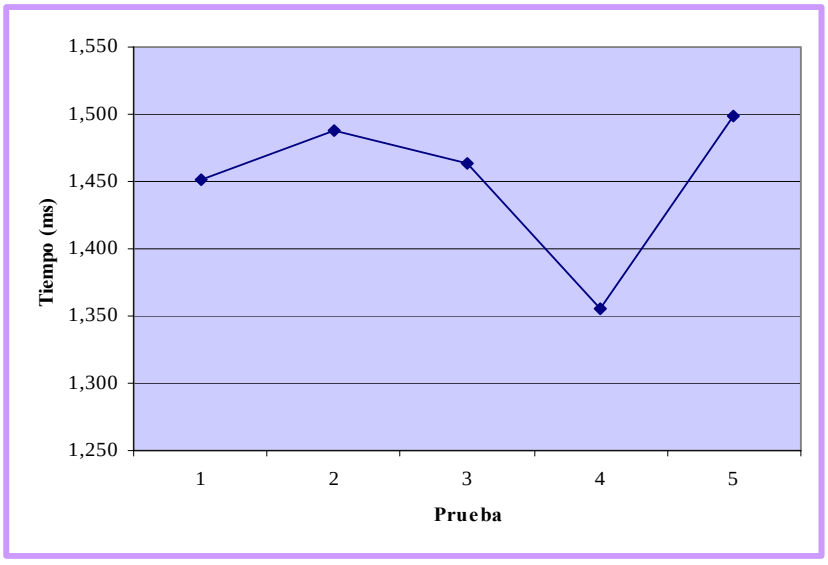


Fig 5.24: Δt unfeasible sin tráfico

Estas pruebas se realizaron en el Departamento de Telemática e Informática de la Universidad de Génova en Italia, donde hasta el momento no se habían realizado pruebas sobre el Open Router con el protocolo BGP. Las pruebas aquí detalladas son las primeras que describen el desempeño del dispositivo antes nombrado, por lo tanto, datos que nos permitan comparar nuestros resultados con resultados de pruebas previas no existen.

Para comparar nuestros resultados con los de un ruteador comercial acudimos a documentos científicos que detallan pruebas realizadas con BGP y exponen las diferentes características y el desempeño de dicho protocolo.

Por datos obtenidos del Ing. Albert Espinal, Director Academia Regional Cisco, el tiempo de convergencia de BGP es de aproximadamente 60 segundos, pero recalcó que no es un valor estándar, dicho tiempo depende de la topología de la red, es decir de la cantidad de AS que forman la red.

Grupos interesados en networking han analizado el impacto del tiempo de convergencia en distintas redes, han creado modelos para dicho estudio. En [46] se establece que el tiempo de convergencia de BGP oscila entre 30 hasta 400 segundos, esto depende de la cantidad de AS que forman la red y de los modelos a seguir según los resultados que se desean obtener.

Comparar nuestro resultado con los apenas citados es un poco complicado, pues el ambiente en que se desarrollaron las pruebas es muy diferente, en [46] se expone que se realizaron pruebas con una red de 60 AS mientras nosotros usamos 3 AS.

Al existir una diferencia grande en la dimensión de las topologías usadas hacen que la confrontación sea imposible, pues el crecimiento del tiempo de

convergencia es exponencial y no lineal, lo que impide hacer una ponderación con respecto a los 60 AS usados en [46].

Si tomamos en cuenta el hecho de tener una topología pequeña y de usar plataformas de código abierto en vía de desarrollo podemos considerar que el tiempo de 2 segundos obtenidos en nuestra prueba demuestra que el OR se desempeña favorablemente y que es un óptimo objetivo de futuras investigaciones para mejorar dicho tiempo con redes más grandes y complejas. El avance en el estudio de plataformas de software abierto, como Linux y Xorp, ayudarán a dicho objetivo.

5.4 Testbed experimentales sobre OSPF

Como dijimos antes OSPF es un protocolo de enrutamiento intra-dominio ampliamente usado en las redes IP.

Los retrasos de elaboración de las tablas de enrutamiento, en respuesta a cambios topológicos en las implementaciones de OSPF, tienen un gran impacto sobre las prestaciones y sobre el nivel de fiabilidad de las redes en tecnología TCP/IP.

A continuación explicaremos en detalle las pruebas realizadas para analizar los índices de prestación de nuestro sistema.

5.4.1 Tiempo de desconexión en una comunicación OSPF

El principal objetivo perseguido en este párrafo, ha sido aquel de proveer una estimación del perfomance del tiempo de desconexión en una comunicación OSPF.

5.4.1.1 Descripción del experimento

En lo que concierne el escenario de pruebas de referencia, hemos elegido un setup con complejidad creciente utilizando una configuración de ruteador de núcleo (*core-router)* compuesta por un Linux Open Router equipado con diferentes interfaces de red a alta velocidad (Gigabit Ethernet).

Con más detalle, hemos operado nuestra prueba usando la siguiente configuración (figura 5.25):

- Un solo flujo monodireccional que atraviesa el OR desde una puerta Gigabit a otra.

Fig 5.25: Configuración de prueba de referencia

Como dijimos antes, los paquetes hello son enviados periódicamente para establecer y mantener relaciones con el vecino. El intervalo entre un paquete y otro es de 10 segundos.

Un parámetro importante incluido en el paquete hello es el Router Dead Interval que especifica el número de segundos que pasa antes de declarar un ruteador inactivo, este tiempo comúnmente es de 40 segundos, es decir si el ruteador no recibe al menos 1 paquete hello en éste tiempo el estado de la comunicación OSPF pasará al estado Down porque considerará que el ruteador vecino no es alcanzable, y cuando el ruteador recibe un paquete hello donde no se ha incluido en el campo neighbour Router el ID del receptor lleva

automáticamente la NDS al estado Init, y por lo tanto el otro ruteador no está en grado de mantener la comunicación bidireccional activa.

Para conocer el tiempo que permanece inactiva una comunicación OSPF debemos conocer la probabilidad de pérdida de un paquete hello. Con el número de paquetes enviados (*offered*) y el número de paquetes recibidos (*througput*) podemos obtener la probabilidad deseada (PrDrop). Para obtener la probabilidad de pérdida de un paquete hello usamos la expresión matemática:

$$\Pr Drop = \frac{Offered - Througput}{Offered}$$

Y si deseamos obtener la probabilidad de pérdida consecutiva de *n* paquetes hello:

$$\Pr Drop(n) = \Pr Drop + \Pr Drop^2 + \Pr Drop^3 + ... + \Pr Drop^n$$

Donde *n* es la cantidad de paquetes hello perdidos.

La tabla VI muestra la probabilidad de pérdida desde 1 hasta 6 paquetes hello.

OFFERED	0,9	0,8	0,7	0,6
THROUGPUT	0,4	0,4	0,4	0,4
PR DROP	0,55556	0,5	0,428571	0,333333
1	0,55556	0,5	0,428571	0,333333
2	0,30864	0,25	0,183673	0,111111
3	0,17147	0,125	0,078717	0,037037
4	0,09526	0,0625	0,033736	0,012346
5	0,05292	0,03125	0,014458	0,004115
6	0,0294	0,015625	0,006196	0,001372

Tab VI: Probabilidad de pérdida de los paquetes hello

Si perdemos 4 paquetes hello consecutivos, los ruteadores se desconectarán; esto puede suceder por algunos segundos mientras la conexión OSPF está establecida, para tener una idea del tiempo que permanece inactiva la comunicación, hagamos uso de la siguiente expresión matemática:

$$TiempoDeDesconexión = \Pr Drop(n) * TiempodeConexión$$

5.4.1.2 Resultados

En nuestra prueba hemos establecido una comunicación OSPF por 15 minutos (900 segundos).

En estos 900 segundos la comunicación OSPF se perdió por decenas de segundos. Esta prueba le realizamos a diferentes cantidades de tráfico. La tabla VII muestra los valores teóricos y prácticos del tiempo de desconexión de nuestras pruebas.

Tráfico (%)	90	80	70	60
Teórico (s)	85,7339	56,25	30,36235	11,11111
Práctico (s)	90	70	15	10

Tab VII: Tiempo de desconexión en el tiempo de 15 minutos

El kernel Linux 2.6 usado en nuestras pruebas hace que el rendimiento de procesamiento (*throughput)* sea solo del 40% del tráfico enviado. El OR genera una gran cantidad de interrupciones, que son considerados prioritarios para su correcto funcionamiento y por esto descarta paquetes que considera no importantes, entre ellos los paquetes hello.

Si no recibe ningún paquete hello dentro de 40 segundos el ruteador considera al vecino no alcanzable y la comunicación OSPF se pierde. Según el tráfico aumenta, la probabilidad de perder los paquetes hello es mayor, por lo tanto, el tiempo de desconexión es mayor como muestra la figura 5.26 donde se confrontan los resultados obtenidos teóricamente con los resultados obtenidos en nuestra prueba.

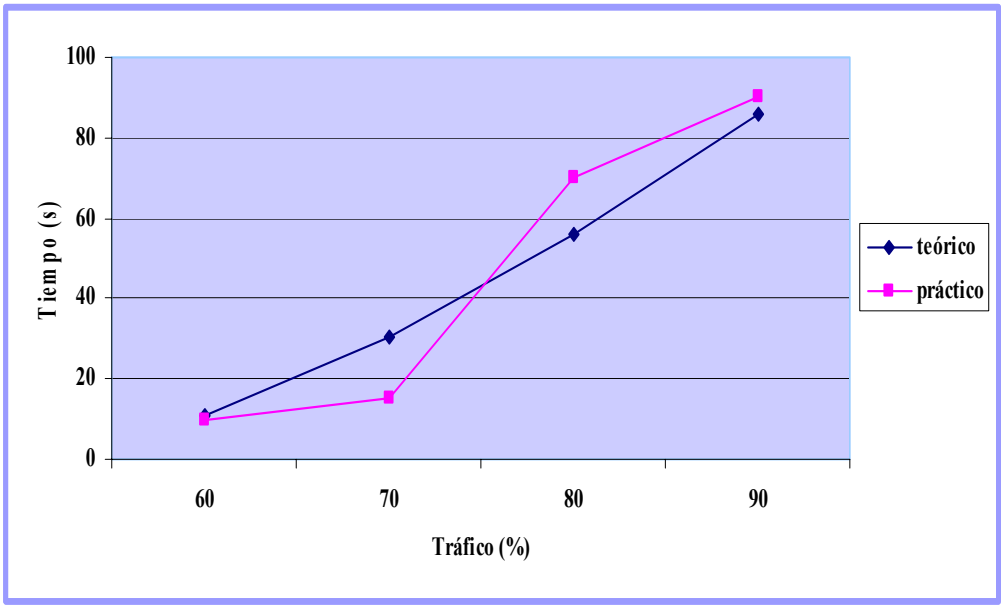


Fig 5.26: Tiempo de desconexión OSPF en el tiempo de 15 minutos.

5.4.2 Tiempo de convergencia

Como explicamos en la prueba efectuada con el protocolo BGP, el tiempo de convergencia es definido como el intervalo de tiempo entre la recepción de un mensaje que anuncia un cambio en la topología, y el primer paquete IP recibido sobre el enlace perteneciente al nuevo camino.

5.4.2.1 Descripción del experimento

Para efectuar esta prueba hemos usado 3 interfaces del OR conectadas a 3 puertas Gigabit del Router Tester, como muestra figura 5.27.

Para alcanzar la red simulada por el Router Tester, el OR utilizará caminos óptimos que incluyen los enlaces A y B.

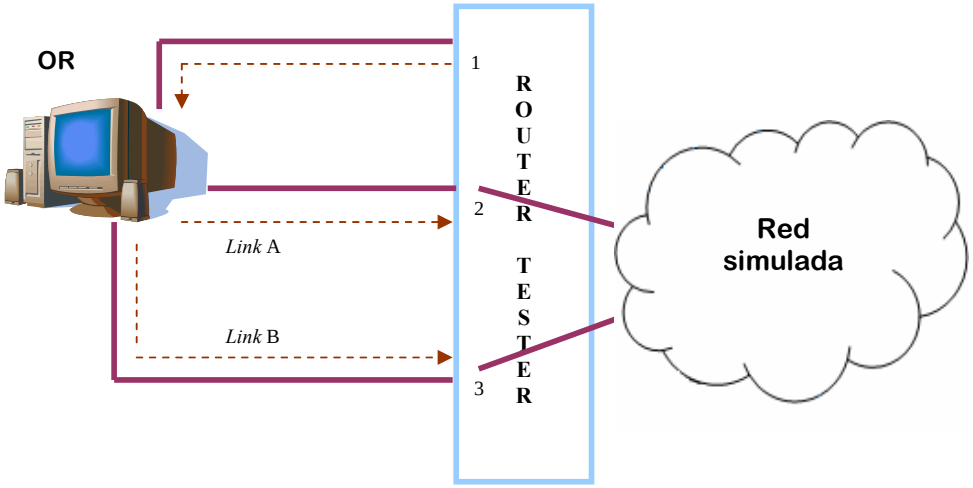


Fig 5.27: Topología de prueba para los experimentos concernientes al tiempo de convergencia OSPF

El OR es atravesado por un flujo de tráfico de datos proveniente de la red simulada detrás de la puerta 1 del Router Tester, que es destinado a la red simulada detrás de las puertas 2 y 3.

Consideremos la siguiente condición inicial: el peso del enlace A es menor al del enlace B, por lo tanto, el camino elegido será el enlace A para llegar a la red simulada por el Router Tester.

Si la métrica del enlace A sufre un cambio, se genera un LSA que será enviado al OR. Si este valor de métrica es mayor al del enlace B, la tabla de enrutamiento será modificada para enviar los paquetes hacia la red de destino sobre el nuevo camino.

5.4.2.2 Resultados

Las pruebas mostradas en este párrafo han considerado el estudio de las variaciones del tiempo de convergencia en función del aumento de los nodos pertenecientes a la red simulada.

Decidimos realizar estos experimentos manteniendo constante el tráfico de datos generado por el Router Tester y a carga baja por motivos de capacidad del Router Tester Agilent, para lograr capturar los tiempos en que se producen los cambios topológicos y el tiempo en que el nuevo camino recibe el primer paquete IP.

La operación de conmutación del tráfico sobre el nuevo camino está constituida por el conjunto de operaciones descritas en el capítulo 4, que representa una de las acciones más laboriosas que el ruteador deba realizar.

En la figura siguiente mostramos el resultado obtenido de las simulaciones efectuadas según el esquema de figura 5.27.

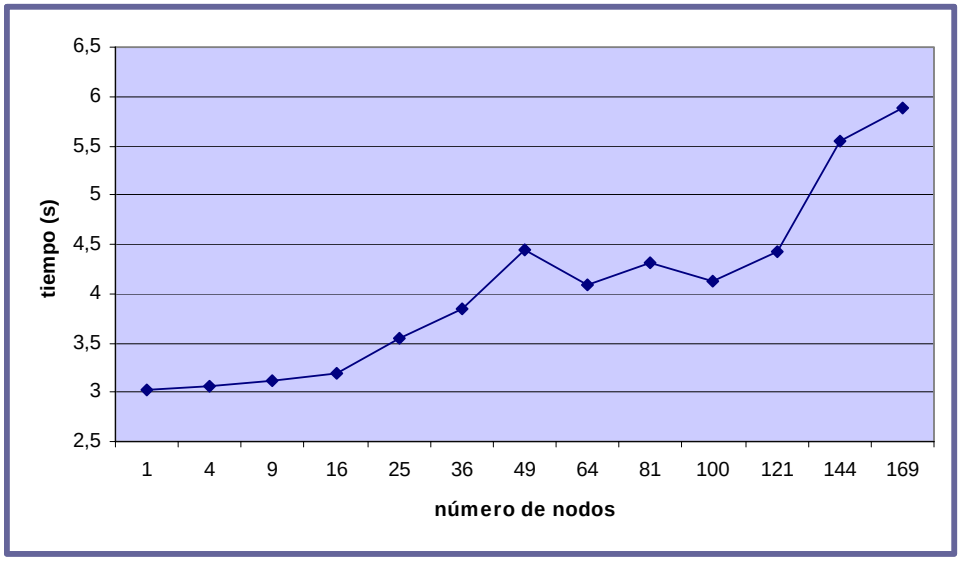


Fig 5.28: Tiempo de convergencia con tráfico al 10%

En la figura 5.28 podemos notar que los valores del tiempo de convergencia (en el orden de los segundos) son crecientes con el número de nodos de la red (en nuestra prueba simulamos una red de máximo 169 nodos), seguramente atribuible al hecho que, la estructura de memorización ordenada de los nodos de Xorp fue originalmente implementada mediante un árbol binario, y para conmutar el tráfico sobre el nuevo camino de costo mínimo, se debe realizar el cálculo de las rutas hacia cada destino.

5.4.3 Tiempo de recálculo de la tabla de enrutamiento

Después que el OR recibe el LSA, que contiene el nuevo valor de métrica, éste debe recalcular toda la tabla de enrutamiento, y después, enviar un LSA para anunciar su LS.

El tiempo que pasa entre el LSA recibido y el LSA enviado, es el tiempo de recálculo de la tabla de enrutamiento.

5.4.3.1 Descripción del experimento

Para realizar esta prueba usaremos el esquema de figura 5.27, descrito en sección 5.4.2.1.

Con el fin de obtener este tiempo, hemos colocado un contador que nos permitirá tomar el tiempo en que llega el anuncio de un LSA al OR, y el tiempo en que el OR envía un LSA con la información del recálculo de su tabla de enrutamiento.

5.4.3.2 Resultados

En las figuras siguientes mostramos los resultados obtenidos, de las simulaciones realizadas según el esquema de figura 5.27.

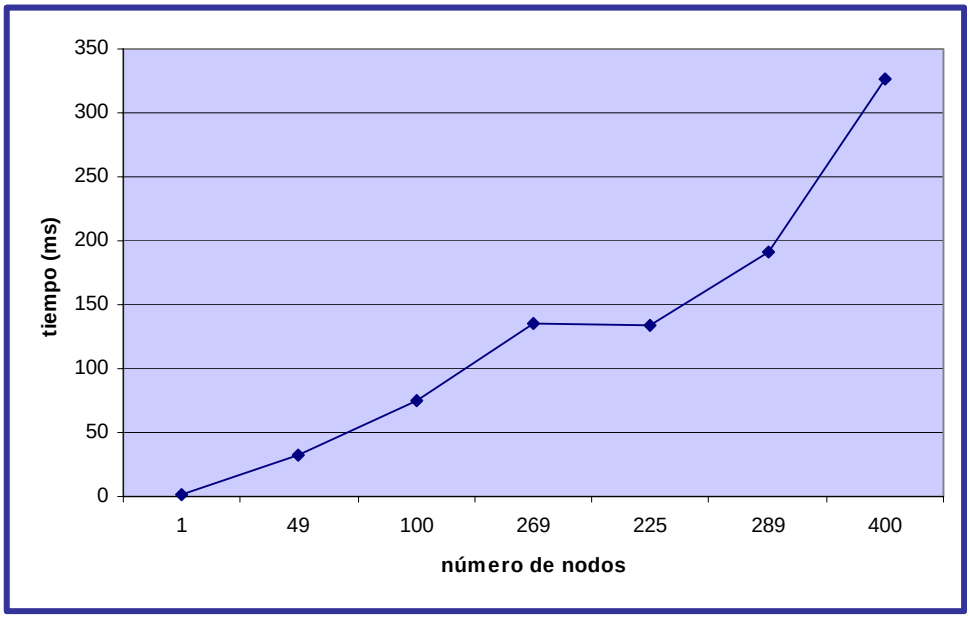


Fig 5.29: Tiempo de recálculo de la tabla de enrutamiento sin tráfico.

La figura 5.29 nos muestra el tiempo de recálculo de la tabla de enrutamiento del OR sin tráfico (0%), y la figura 5.30 con tráfico al 30%.

Como podemos notar en ambas figuras el tiempo de recálculo (en el orden de los milisegundos) es creciente con el número de nodos de la red, resultado similar al de la prueba del tiempo de convergencia, debido a que, la topología es la misma y, por lo tanto, la estructura de memorización de los nodos de Xorp está implementada mediante un árbol binario, y como dijimos antes es necesario recalcular todas las rutas hacia cada destino, con el inconveniente que Xorp no soporta (hasta el momento de la escritura de esta tesis) un algoritmo incremental para el cálculo de los Spf.

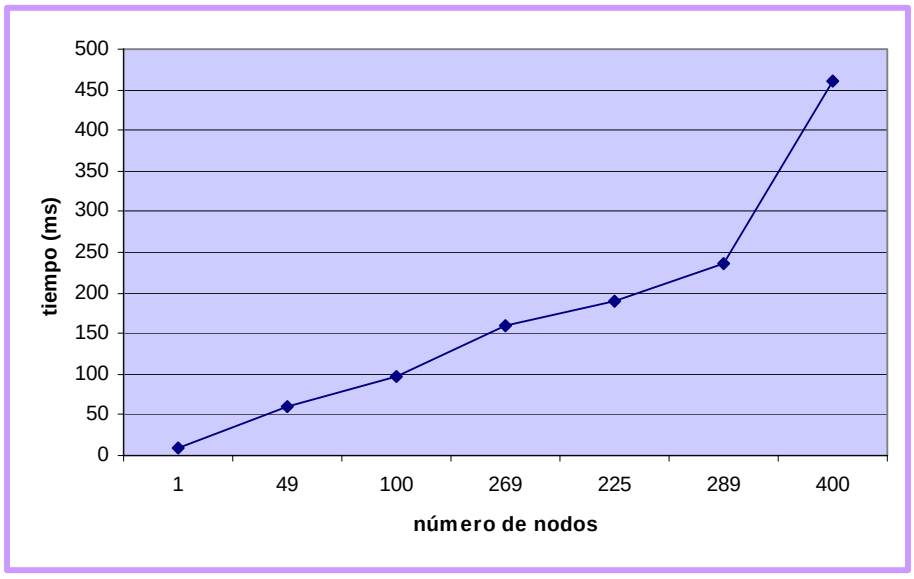


Fig 5.30: Tiempo de recálculo de la tabla de enrutamiento con tráfico al 30%

En el caso del experimento “tiempo de convergencia” con el protocolo OSPF podemos confrontar nuestros resultados con los resultados obtenidos en

pruebas realizadas en un ruteador comercial [47]. Dicha prueba fue previamente realizada por un estudiante de tesis de la Universidad de Génova para determinar dicho tiempo en el ruteador comercial Juniper M10.

El anexo B expone la tabla y el gráfico de resultados de las pruebas realizadas sobre dicho ruteador.

La prueba fue realizada con una topología de 1800 nodos lo que representa un tiempo de convergencia de 5.4 segundos; en nuestro experimento el número máximo de nodos con los cuales realizamos la prueba es de 169 con un tiempo de convergencia de casi 6 segundos.

La diferencia en la dimensión de la red es atribuible a las limitaciones en la simulación de la topología, debido a que el Open Router no dispone de varias interfaces detrás de las cuales se puedan simular redes de 100 nodos cada una como se realizó en [47]. La topología de 169 nodos fue la misma para las 2 interfaces del Open Router que usamos para dicha prueba.

Al momento de realizar el experimento tuvimos problemas con el dispositivo, a topologías mayores el Open Router dejaba de funcionar, debido a la memorización a árbol binario de los nodos en Xorp. El equipo comenzó a descartar una gran cantidad de paquetes que impedían su correcto funcionamiento.

Al realizar una ponderación lineal de los resultados obtendremos que el tiempo del ruteador Juniper para 169 nodos tiene un tiempo de 0.51 segundos, pero dicho valor no es seguro debido a que el crecimiento del tiempo de convergencia es exponencial.

Al confrontar dichos resultados obtendríamos que el Open Router para este caso se desempeña de manera ineficiente, pues la diferencia de valores es significativa, pero debemos tomar en cuenta que la plataforma sobre la cual OSPF funciona es de una arquitectura mejor estructurada que permite un mejor desempeño del equipo mientras la plataforma que usa el Open Router está todavía en desarrollo, pero que ha demostrado ser capaz de competir con una arquitectura cerrada, pues al obtener un tiempo de convergencia de 6 segundos para una topología relativamente grande de 169 nodos demuestra que dichas plataformas tienen futuro en el desarrollo de redes mucho más grandes.

Conclusiones y Recomendaciones

1. Los resultados presentados en esta tesis, han contribuido al estudio de los dispositivos de Internet y de arquitectura abierta.

2. El objetivo principal de este trabajo ha sido validar el nivel de perfomance obtenible para este tipo de dispositivos. Nos focalizamos en particular sobre las actividades de optimización y de análisis de las prestaciones del proceso de control de una arquitectura PC basada en la plataforma Linux.

3. Los experimentos realizados en este trabajo han sido ejecutados con conformidad a la RFC 2544 [48]. El estudio de la estructura modular de Xorp nos ha permitido analizar el código fuente, individuar donde está implementada la actualización de las rutas, y de observar los tiempos empleados para el recálculo de las tablas de enrutamiento.

4. Con el aporte de la shell de Xorp pudimos monitorear y hacer un seguimiento del proceso para entablar una conexión entre ruteadores BGP. Pudimos establecer el archivo de configuración de Xorp, visualizamos la tabla de enrutamiento y conocimos los detalles de dicha conexión.

5. El tiempo de convergencia para la conexión BGP entre 3 AS diferentes fue de 2 segundos, tiempo considerable para la selección de un nuevo camino.

6. Establecimos que el tiempo que permanece desconectada una comunicación OSPF entre dos ruteadores depende de la cantidad de tráfico que los atraviesa, a mayor tráfico mayor es el tiempo en que dicha comunicación permanece en el estado Down.

7. Para una red de 169 nodos el tiempo en que un ruteador OSPF converge al producirse un cambio en la métrica de la red, es de 6 segundos.

8. Determinamos que el tiempo que un ruteador OSPF demora para recalcular su tabla de enrutamiento aumenta con la cantidad de nodos de la topología de red y la cantidad de tráfico que los atraviesa. Para una red de 400 nodos el tiempo fue 450 ms, y para la misma red sin tráfico el resultado fue de 330 ms.

9. Las limitaciones en la simulación de la topología de red nos impiden realizar una buena comparación entre nuestros resultados con los de un dispositivo comercial.

10. Realizar una ponderación entre los 30 segundos del tiempo de convergencia de BGP con una red de 60 AS [46], con nuestros resultados, 2 segundos con una red de 3 AS, no es factible debido a que el crecimiento del tiempo de convergencia en una red tiene un comportamiento exponencial y no lineal. Lo mismo ocurre con OSPF.

11. Tomando en consideración nuestras topologías de red y las condiciones en que realizamos nuestras pruebas podemos decir que el Open Router tuvo un desempeño considerable, y que el continuo estudio y actualización de estas

plataformas permitirán el progreso del código para un mejor servicio de los aparatos de arquitectura abierta.

12. Debemos indicar que en lo que concierne la plataforma de enrutamiento Xorp, al final del desarrollo de esta tesis fue anunciada la salida de un nuevo release del software, que provee numerosas integraciones del código y la posibilidad de implementar OSPFv3.

13. Los resultados de este trabajo representan una buena base de partida para futuros desarrollos que podrían considerar, por ejemplo, el impacto del plano de control sobre las prestaciones globales del sistema, y de sucesivas actividades de investigación de otras plataformas.

Bibliografía

[1] Vint Cerf "*A Brief History of the Internet and Related Networks*". Novembre 2001
URL: http://www.isoc.org/internet/history/cerf.shtml

[2] Internet Software Consortium.
URL: http://www.isc.org

[3] Information Sciences Institute, University of Southern California. "*Request for Comments 791: Internet Protocol. DARPA Internet Program Protocol Specification*". Settembre 1981.
URL: http://www.ietf.org/rfc/rfc791.txt

[4] S. Deering, R. Hinden. "*Request for Comments 2460: Internet Protocol, Version 6 (IPv6) Specification*" Dicembre 1998.
URL: http://www.ietf.org/rfc/rfc2460.txt

[5] J. Postel. "*Request for Comments 792: Internet Control Message Protocol. DARPA Internet Program Protocol Specification*". Settembre 1981.
URL: http://www.ietf.org/rfc/rfc791.txt

[6] V. Fuller, T. Li, J. Yu, K. Varadhan. "*Request for Comments 1519: Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*". Settembre 1993.
URL: http://www.ietf.org/rfc/rfc1519.txt

[7] D. Plummer. "*Request for Comments 826: An Ethernet Address*

Resolution Protocol". Novembre 1982.
URL: http://www.ietf.org/rfc/rfc826.txt

[8] R. Finlayson, T. Mann, J. Mogul, M. Theimer. "*Request for Comments 903: A Reverse Address Resolution Protocol*". Giugno 1984
URL: http://www.ietf.org/rfc/rfc903.txt

[9] C. Hedrick. "*Request for Comments 1058: Routing Information Protocol*". Giugno 1988
URL: http://www.ietf.org/rfc/rfc1058.txt

[10] G. Malkin. "*Request for Comments 1388: Routing Information Protocol Version 2*". Gennaio 1993.
URL: http://www.ietf.org/rfc/rfc1388.txt

[11] Cisco Systems. "*Interior Gateway Routing Protocol*".
URL:
http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/igrp.htm

[12] J. Moy. "*Request for Comments 1247: OSPF*". Luglio 1991.
URL: http://www.ietf.org/rfc/rfc1247.txt

[13] J. Moy. "*Request for Comments 1583: OSPF Version 2*". Marzo 1994
URL: http://www.ietf.org/rfc/rfc1583.txt

[14] D.L. Mills. "*Request for Comments 904: Exterior Gateway Protocol Formal Specification*". Aprile 1984.
URL: http://www.ietf.org/rfc/rfc904.txt

[15] K. Lougheed, Y. Rekhter. "*Request for Comments 1105: A Border Gateway Protocol (BGP)*". Giugno 1989
URL: http://www.ietf.org/rfc/rfc1105.txt

[16] K. Lougheed, Y. Rekhter. "*Request for Comments 1163: A Border Gateway Protocol 2 (BGP-2)*". Giugno 1990
URL: http://www.ietf.org/rfc/rfc1163.txt

[17] K. Lougheed, Y. Rekhter. "*Request for Comments 1267: A Border Gateway Protocol 3 (BGP-3)*". Ottobre 1991
URL: http://www.ietf.org/rfc/rfc1267.txt

[18] Y. Rekhter, T. Li. "*Request for Comments 1771: A Border Gateway Protocol 4 (BGP-4)"*. Marzo 1995
URL: http://www.ietf.org/rfc/rfc1771.txt

[19] B. Hubert et al. "Linux advanced *routing* & traffic control HOWTO".
URL: http://lartc.org

[20] S. Radhakrishnan. "Linux – Advanced *networking* overview".

[21] Berkeley University. "*FreeBSD*".
URL: http://www.freebsd.org

[22] P. Gray, A. Betz, "*Performance Evaluation of Copper-Based Gigabit Ethernet Interfaces*", 27th Annual IEEE Conference on Local Computer *Networks* (LCN'02), Tampa, Florida, Novembre 2002, pp.679-690.

[23] Intel Corporation. "The Intel PRO/1000 XT Server Adapter".
URL:
http://www.intel.com/network/connectivity/products/pro1000xt.htm

[24] Intel Corporation. "Overview, Installing and Using Priority *Packet*".
URL:
http://support.intel.com/support/network/adapter/ppack/013748.htm

[25] K. Nichols, S. Blake, F. Baker, D. Black, "Request for Comments 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 *Headers*". Dicembre 1998.
URL: http://www.ietf.org/rfc/rfc2474.txt

[26] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "*Request for comments 2475: An Architecture for Differentiated Service*". Dicembre 1998

URL: http://www.ietf.org/rfc/rfc2475.txt

[27] Sun Micro*systems*. "*Solaris*".
URL: http://www.sun.com

[28] Apple Computers. "*OS X*".
URL: http://www.apple.com/macosx

[29] The Linux Kernel Archives.
URL: http://www.kernel.org

[30] Debian GNU/Linux.
URL: http://www.debian.org

[31] GNU Zebra.
URL: http://www.zebra.org

[32] Quagga Software Routing Suite.
URL: http://www.quagga.net

[33] Xorp Open Source IP Router.
URL: http://www.xorp.org

[34] M. Handley, O. Hodson, E. Kohler. "*XORP: an open platform for network research*". ACM SIGCOMM Computer Communication Review, Vol. 33 Issue 1, Gennaio 2003, pp. 53-57.

[35] J. H. Salim, R. Olsson, A. Kuznetsov. "*Beyond Softnet*". Proceedings of the 5th annual linux Showcase & Conference, Novembre 2001, Oakland California.

[36] A. Cox. "*Network Buffers and Memory Management*". Linux Journal, Ottobre 1996.
URL: http://www2.linuxjournal.com/lj−issues/issue30/1312.html

[37] The descriptor recycling patch.
URL: ftp://robur.slu.se/pub/Linux/net-development/skb_recycling/

[38] J. Hawkinson, T. Bates. "*Request for Comments 1930: Guidelines for creation, selection, and registration of an Autonomous System (AS)*". Marzo 1996
URL: http://www.ietf.org/rfc/rfc1930.txt

[39] ICANN 2002 "International Conference on Artificial Neuronal Networks" Agosto 27-30
URL: http://www.ii.uam.es/icann2002/

[40] Nuove prospettive di utilizzo di Xorp
URL:
http://money.cnn.com/magazines/business2/business2_archive/2006/03/01/8370567/index.htm

[41] Mark Handley, Orion Hodson, Eddie Kohler "*Xorp: An Open Platform for Network Research*"

[42] CVS repository di Xorp
URL: http://www.xorp.org/downloads.html

[43] Xorp Open Source IP *Router*
URL: http://www.xorp.org/

[44] J. Moy: "*Request for Comments: 2328 OSPF Version 2*" April 1998
URL: http://www.faqs.org/rfcs/rfc2328.html

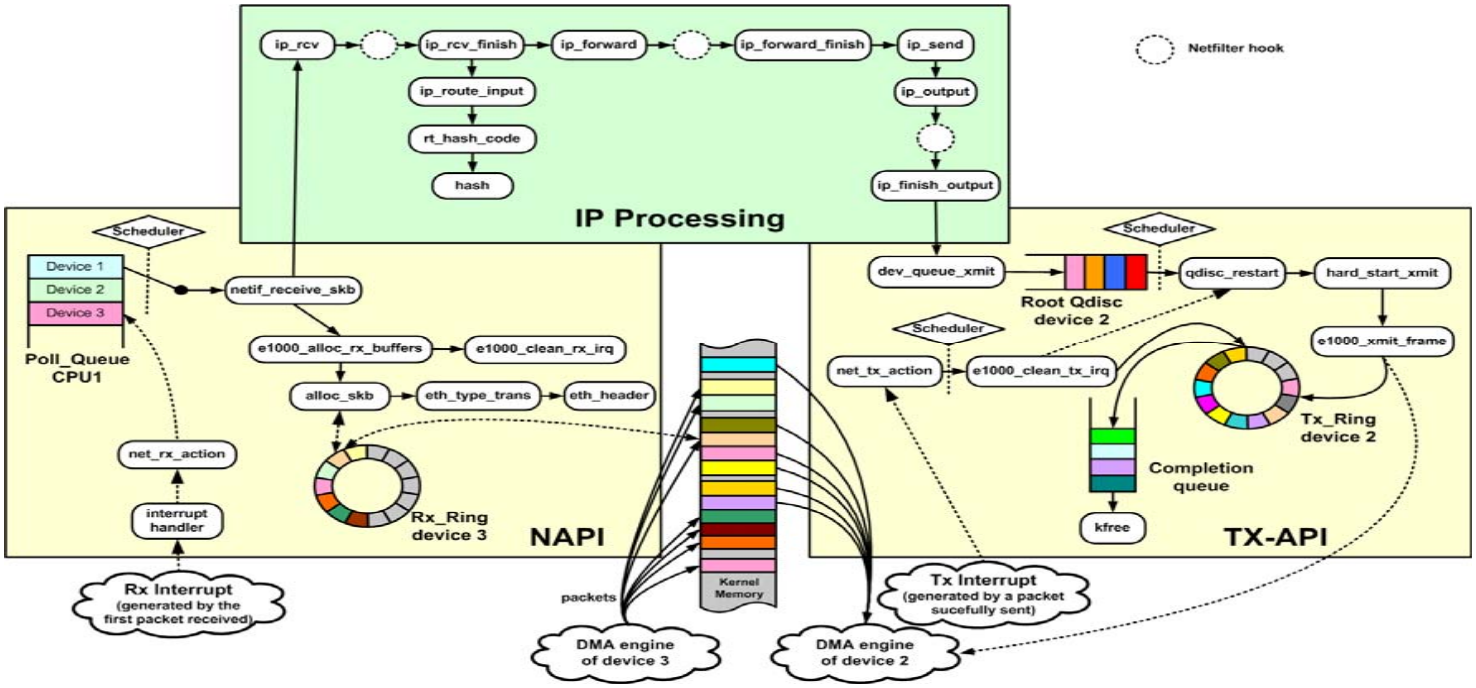
[45] *Router* Tester Agilent
URL: http://advanced.comms.agilent.com/n2x/

[46] Dan Pei, Xiaoliang Zhao, Lan Wang, Daniel Massey, Allison Mankin, S. Felix Wu, Lixia Zhang "*Improving BGP Convergence Through Consistency Assertions*"

[47] Andrea Rolleri "*Analisi di prestazioni del piano di controllo e di inoltro in Core Router ad architettura aperta*". Febrero 2006

[48] S. Bradner, J. McQuaid. "*Request for Comments 2544: Benchmarking Methodology for Network Interconnect Devices"*. Marzo 1999
URL: http://www.ietf.org/rfc/rfc2544.txt

ANEXO A



Esquema detallado de las operaciones de forwarding en el kernel 2.6 NAPI.

ANEXO B

Nodos	Vertices	s.t.juniper.2000F/s
6	12	0.144143667
100	280+11	0.202107
400	1120+14	0.527391667
700	1960+17	0.799297333
1300	3640+23	1.47078633333333
1800	5040+18	5.406004333

Tabla del tiempo de convergencia OSPF router Juniper M10

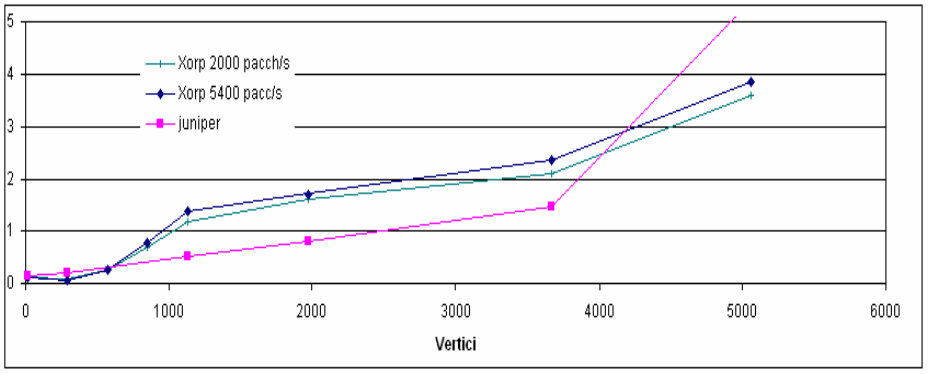


Grafico del tiempo de convergencia OSPF router Juniper M10