UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"ANÁLISIS DE LOS MODELOS DE SERVICIOS DIFERENCIALES Y SERVICIOS INTEGRALES PARA BRINDAR QoS EN INTERNET"

TESIS:
QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN

PRESENTA:
THELMA GARCÍA REYES

DIRECTORES DE TESIS:
M. C. GABRIEL GERÓNIMO CASTILLO
M. C. RICARDO RUIZ RODRÍGUEZ

HUAJUAPAN DE LEÓN, OAXACA, JUNIO DE 2007

«ÍNDICE»

Capítulo 3 Servicios Integrales

Capítulo 4 Modelos Híbridos

Capítulo 5 Modelos Aplicados en los Sistemas GNU/Linux

«LISTA DE FIGURAS»

«LISTA DE TABLAS»

«CAPÍTULO I»

INTRODUCCIÓN

1.1 Antecedentes

En la actualidad se busca que en las redes de computadoras, así como en los servicios ofrecidos a los usuarios sean eficientes y eficaces. Sin embargo una red homogénea y unificada requiere un cierto nivel de calidad. Para solucionar este problema, se han buscado implementar modelos para satisfacer la demanda de Calidad de Servicio (QoS). El concepto de QoS *(Quality of Service - Calidad de Servicio)* puede variar dependiendo del autor, según la ISO/IEC DIS 13236 mencionada en [1] y [3], QoS es un conjunto de cualidades relacionadas con la provisión de un servicio hacia un usuario, hoy en día los usuarios de servicios de Internet pueden ser tanto humanos como programas de aplicación, buscadores, bases de datos distribuidas, multimedia, telefonía entre otros. En [13] se define la QoS como, la capacidad para proporcionar aseguramiento del recurso y diferenciación del servicio en una red. En un principio la QoS que se implementaba en las redes se basaban únicamente en la integridad de los datos que viajaban a través de la red, esto significaba no perder el contenido ni la secuencia de los datos, dado que las redes fueron creadas en primera instancia para transportar de forma óptima y segura el tráfico de datos pero no en tiempo real.

Debido a que en la actualidad las aplicaciones son diferentes, los requerimientos también suelen ser diferentes para cada usuario. Por ejemplo el manejo de audio y video ha incrementado las posibilidades de comunicación interactiva entre los usuarios, por tal motivo se han propuesto modelos que satisfagan los niveles de QoS para cada necesidad, siendo así que cada modelo existente maneja sus propios métodos para llevar a cabo esta tarea. Estos modelos necesitan considerar otros parámetros de calidad tales como la latencia y el ancho de banda.

Aplicar QoS en las redes IP da la capacidad de asegurar el sistema utilizado, al mismo tiempo reduce los costos de ofrecer servicios con un grado de fiabilidad establecida, cumpliendo los requisitos de tráfico (en términos de perfil y ancho de banda) para un flujo de información dado. Esto quiere decir que ofrecer QoS en un sistema, permite administrar el ancho de banda que se ofrezca, garantizando que se lleven a cabo los servicios cuando haya periodos de congestión alta y optimizando así el desempeño de la red. Por esta razón se necesitan mecanismos de señalización que propicien tener bajo control los parámetros de calidad y garantía de los datos.

Existe una organización que establece reglamentos para transferir información a través de la red, la IETF *(Internet Engineering Task Force - Grupo de Trabajo sobre Ingeniería de Internet)* quien propone algunos modelos para satisfacer la demanda de QoS en las redes IP, como se muestran en [3]:

- Int–Serv *(Integrated Services - Servicios Integrales)*
- Diff–Serv (Differentiated Services - Servicios Diferenciales)
- MPLS *(Multi Protocol Label Switching - Multiprotocolo de Conmutación de Etiquetas)*
- Ingeniería de Tráfico
- Constraint – Based Routing *(Ruteo Basado en Restricciones)*
- SBM *(Subset Bandwidth Management - Administración del Ancho de Banda de la Subred)*

Estos mecanismos proporcionan una serie de herramientas que el administrador de redes puede utilizar para disponer de forma eficaz el uso de los recursos de la red, siempre y cuando se analicen con respecto a los requerimientos de los usuarios.

Actualmente se utilizan dos versiones del protocolo de Internet, el protocolo versión 4 o IPv4 y el protocolo IPv6, en cada uno de ellos se puede aplicar QoS, utilizando los campos de sus encabezados.

En IPv4 se utiliza el campo ToS *(Type of Service - Tipo de Servicio)* y en IPv6 se utilizan los campos TC *(Traffic Control - Control de Tráfico)* y el campo Etiqueta de Flujo.

En IPv4 la transmisión de bloques de datos llamados datagramas tiene un cierto formato de encabezado que contiene en forma general cuatro puntos clave para proporcionar un servicio: tipo de servicio, tiempo de vida, opciones y suma de verificación.

Asimismo el Campo ToS se utiliza para la aplicación de QoS, este campo tiene una longitud de 8 bits y se encuentra dividido en 6 subcampos, (Figura 1) [2]. En donde D significa Retardo, T Desempeño, R Integridad y C Costo.

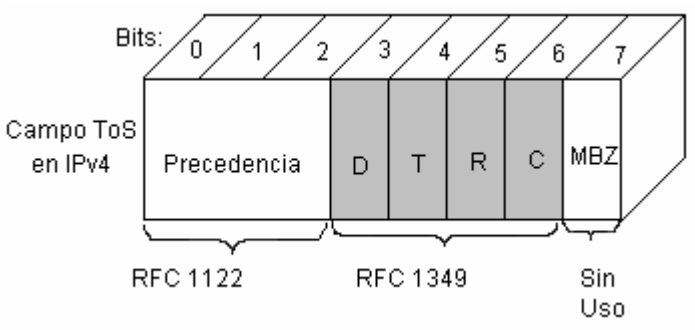

Figura 1. Formato del campo ToS en IPv4

Los tres primeros bits indican la importancia o prioridad del datagrama y los siguientes 4 bits tienen una interpretación (Tabla 1) que se deben tomar en cuenta en cada una de las pasarelas.

Tabla 1. Bits del 3 al 6 del campo ToS

D	T	R	C	INTERPRETACION
1	0	0	0	Solicita un procesamiento con mínimo retardo
0	1	0	0	Solicita máximo desempeño
0	0	1	0	Solicita máxima confiabilidad
0	0	0	1	Solicita mínimo costo monetario
0	0	0	0	Especifica un trato normal para el datagrama

En IPv6 el manejo de QoS se hace mediante los campos Etiqueta de Flujo y Clase de Tráfico (TC). El campo Etiqueta de flujo tiene una longitud de 20 bits que pueden ser usados por la fuente para etiquetar la secuencia de paquetes, los cuales requieren un trato especial por los routers, tales como un servicio de calidad diferente de lo normal o algún servicio de tiempo real. El campo TC tiene 8 bits y es usado por los nodos y/o routers para identificar y distinguir los paquetes enviados con diferentes clases o diferentes prioridades. Los valores que pueden tomar estos bits dependen de los nodos que envían los paquetes, y su valor por omisión es cero [2].

1.2 Marco de referencia

Durante la última década se han realizado investigaciones con los modelos propuestos por la IETF [1][2][3], por mencionar algunos, en estas investigaciones se han plasmado las características de cada uno de estos modelos tratando de encontrar alguno que cumpla con todas las expectativas de QoS en redes IP, pero aún no se ha logrado que alguno sea totalmente aceptado.

Como trabajo relacionado con esta tesis se mencionan algunas investigaciones de los modelos propuestos para el ofrecimiento de QoS que se han realizado anteriormente. Existe una investigación [3] en donde se realiza un análisis de la QoS del modelo de Servicios Integrales. El análisis toma en cuenta los requerimientos de los usuarios y los servicios ofrecidos por el modelo Servicios Integrales. Este análisis muestra que el modelo de Servicios Integrales satisface y

garantiza principalmente requerimientos de rendimiento, de seguridad, de funcionamiento y algunos requerimientos temporales. Sin embargo no satisface ningún requerimiento de seguridad en las comunicaciones. Además del análisis mencionado anteriormente, en esta investigación se realiza un modelo de los Servicios Integrales usando UML (*Unified Modeling Language - Lenguaje Unificado de Modelado*) con el objetivo de ampliarlo y poder satisfacer y garantizar aquellos requerimientos de los usuarios.

Otra investigación que sirve como pauta para esta tesis es [2] en donde se hace un análisis de la QoS desde el punto de vista de los servicios proporcionados por los protocolos de Internet IPv4 e IPv6. Analiza también los Servicios Diferenciales que ofrece IP por medio de los campos ToS y TC, y finaliza mencionando cómo aplicar QoS en los hosts y los routers.

Otra investigación que se tomará en cuenta como base para esta tesis es [1]. Esta investigación hace un análisis de la arquitectura de QoS en Internet, dicho análisis se hizo por un lado, considerando los requerimientos definidos por la mayoría de las aplicaciones de Internet y por otro lado considerando las características que proveen los protocolos de Internet (IPv4 e IPv6), los cuales son las bases de los modelos propuestos por la IETF.

1.3 Objetivos

El objetivo general de este trabajo de tesis es el de ayudar a satisfacer la demanda de QoS en una red, por medio de un estudio detallado de dos de los modelos propuestos por la IETF: los Diff–Serv y los Int–Serv.

Asimismo los objetivos particulares que se pretenden alcanzar son los siguientes:

- Investigar y documentar los modelos de Diff–Serv e Int–Serv
- Realizar una propuesta de representación con UML para los Diff–Serv
- Realizar una propuesta de representación con UML para los Int–Serv
- Investigar y documentar los modelos híbridos
- Identificar los mecanismos que se utilizan para brindar QoS en los sistemas GNU/Linux a nivel kernel

- Aplicar la representación propuesta con UML para saber como se lleva a cabo en el kernel de GNU/Linux

1.4 Justificación

Este trabajo de estudio y análisis será realizado para alcanzar los objetivos mencionados en la sección anterior, tratando de plasmar las ventajas y desventajas que pueden proporcionar los dos modelos que se tratan en este trabajo, en cuanto al ofrecimiento de QoS en Internet.

Ya que aun no hay un modelo que sea totalmente aceptado, se verá a lo largo de este trabajo cuales son las características que hacen ver como propuestas para satisfacer la QoS a los Diff–Serv como los Int–Serv.

Al analizar sus características y ventajas, veremos cual de los dos modelos es el que se vislumbra como la mejor opción para ofrecer QoS en Internet.

Asimismo ocupando algunos de los artefactos de UML se propondrán en esta tesis, los diagramas de casos de uso, diagramas de actividades y diagramas de secuencia de los modelos de servicio Diff–Serv e Int–Serv, con el propósito de ilustrar de una manera gráfica y a un nivel de representación más abstracta y entendible el funcionamiento de dichos modelos de servicio, para así comprender mejor su arquitectura, ya que en la actualidad existe una propuesta que se puede ver en [3] que es confusa al tratar de plasmar el funcionamiento de la arquitectura de los Int–Serv. Con lo anterior no se pretende presentar un modelado orientado a objetos ya que no es la implementación de un sistema, sino más bien se presentan los actores y los elementos identificados en los modelos analizados durante la presente tesis y la interacción que existe entre ellos.

En cuanto a la versión de Linux que se propone para identificar los mecanismos para proporcionar QoS mediante Diff–Serv e Int–Serv, se optó por la distribución Fedora Core 5 con el kernel 2.6.16.18 ya que es la distribución más utilizada en nuestra institución actualmente.

«CAPÍTULO 2»

SERVICIOS DIFERENCIALES

2.1 Introducción

Durante años se han buscado nuevas formas de proporcionar calidad en las redes de computadoras, por tal motivo la IETF ha propuesto modelos que ofrecen QoS en diferentes modalidades. El modelo Diff–Serv es uno de los que se ha vislumbrado como una de las propuestas más alentadoras para proporcionar QoS en redes grandes como lo es Internet. Por tal motivo en este capítulo se estudia y analiza este modelo, sus características, su arquitectura y las expectativas que se tienen al proponer su implementación en las redes. Además en el último apartado de este capítulo se propone un modelado mediante UML para comprender mejor su arquitectura.

2.2 Modelo de Servicios Diferenciales

La propuesta de este modelo es garantizar la mayor QoS en las redes IP de gran tamaño como lo es Internet, añadiendo la facilidad de implementación y el bajo costo ya que no hay necesidad de implementar grandes cambios en la estructura de las redes actuales.

Los Diff–Serv son un conjunto de tecnologías por medio de las cuales los proveedores de servicios de red pueden ofrecer distintos niveles de QoS para diferentes clientes y tráfico de información.

Se basan en la división del tráfico en diferentes clases y en la asignación de prioridades a los paquetes, llamando a este proceso tratamiento diferenciado de

los paquetes IP en los routers. Las características de los paquetes se pueden especificar en términos cuantitativos o estadísticos del rendimiento, de la demora, de la inestabilidad, y/o de la pérdida, o de otro modo se puede especificar en términos de alguna prioridad relativa del acceso a los recursos de la red.

Este modelo sigue una estrategia que facilita la estabilidad y el despliegue en las redes, ya que no necesita que en todos los nodos de la red tengan implementado este tipo de arquitectura.

Con el marcado de paquetes que proponen los Diff–Serv hace que los paquetes que pertenezcan a una misma clase reciban un mismo trato por parte de la red. Cuanto mayor sea la prioridad o el ancho de banda asignado a la clase, mejor será el trato que reciba el paquete [URL, 6].

La diferenciación de servicios se lleva a cabo mediante la definición de comportamientos específicos para cada clase de tráfico entre dispositivos de interconexión, a este hecho se le conoce como PHB *(Per Hop Behavior - Comportamiento por Salto)*.

En este modelo se introduce el concepto de PHB, el cual define cuánto tráfico le corresponde a un paquete en particular. En las cabeceras de los paquetes IP, el PHB no es indicado como tal, si no que se maneja mediante los valores del campo DSCP *(Differentiated Services Code Point – Punto de Codificación de Servicios Diferenciados)* [7].

El PHB especifica la prioridad del encaminamiento de paquetes en los routers y otros dispositivos de la red. Los valores estándares de PHB son [URL, 4][URL, 5][8]: BE (*Best Effort, Por Omisión o Mejor Esfuerzon*), CS (*Selector Class, Selector de Clase*), EF (*Expedited Forwarding, Encaminamiento Expedito*), AF (*Assured Forwarding, Encaminamiento Asegurado*).

2.3 Campos que proveen QoS en IPv4 e IPv6

Para proporcionar Diff–Serv en las redes IP, es necesario explotar los diferentes campos que permiten darle prioridad a los datagramas tanto en IPv4 como en IPv6.

El esquema de servicios diferenciales delimita las funciones que se tienen que realizar en los nodos de ingreso a la red y en los nodos internos de la red. Los nodos de acceso a la red se encargan de la clasificación y especificación del contenido del campo DS *(Diferencial Service - Servicio Diferencial)*. Los nodos interiores se encargan del reenvío de los paquetes dependiendo del contenido del campo DS.

La clasificación que se haga del paquete, queda especificada en el contenido del campo ToS en el encabezado del paquete IP.

En la arquitectura de Diff–Serv, los paquetes son clasificados únicamente en el dispositivo de acceso a la red, y ya dentro de la red el tipo de procesamiento que reciban los paquetes va a depender del contenido del encabezado.

Para tomar la información de la cabecera de cada paquete, se ha designado un byte en cada una de las versiones de IP para manejar los servicios diferenciales. En IPv6 contempla este marcado de paquetes mediante el campo TC. Y en IPv4 este marcado se realiza a través del byte ToS. En ambos casos se utiliza éste campo como el byte DS.

2.3.1 Campo Tipo de Servicio (ToS)

La transmisión de bloques de datos o datagramas en IPv4 se lleva a cabo siguiendo un formato de encabezado con las características que se muestran en la Figura 2.

El campo ToS se utiliza en el encabezado de los paquetes, para proporcionar QoS. Este campo tiene una longitud de ocho bits y se encuentra dividido en seis subcampos como se muestra en la Figura 1del capítulo anterior.

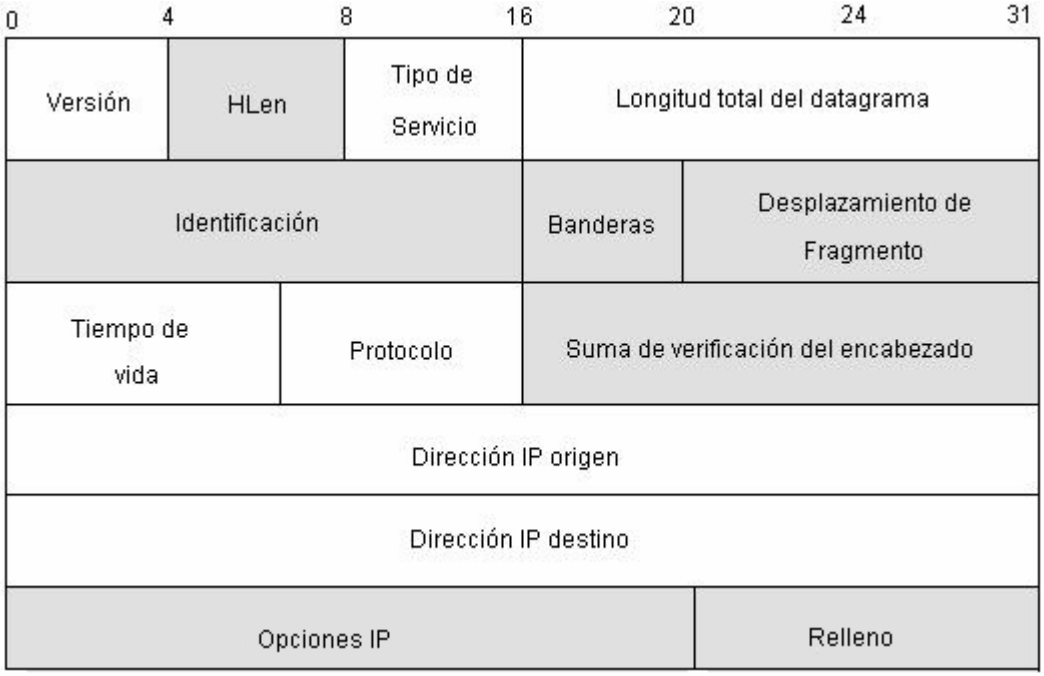

Figura 2. Encabezado en IPv4

En el campo ToS se utilizan los tres primeros bits para obtener ocho posibles niveles de precedencia, que se muestran en la tabla 2 [URL, 9].

Tabla 2. Niveles de precedencia IP

Valor de precedencia IP	**Nombre Descriptivo**
000	Normal o rutinario
001	Prioritario
010	Inmediato
011	Urgente
100	Muy urgente
101	Crítico
110	Control entre redes
111	Control de red

Los bits D, T, R y C (mostrados en la Figura 1, del capítulo 1), especifican el tipo de transporte deseado para el datagrama. Cuando el bit D está activo quiere decir que se está solicitando un procesamiento con mínimo retardo, si el bit T está activo se está solicitando máximo desempeño, el bit R activo solicita una máxima confiabilidad y el bit C activo solicita un mínimo costo monetario [2]. Los paquetes o datagramas de menor precedencia pueden ser descartados a favor de una precedencia mayor cuando exista una congestión en la red. Además cada paquete puede ser marcado para recibir uno de dos niveles de D, T y R. Sin embargo este esquema tiene limitaciones cruciales:

- El esquema de precedencia IP permite sólo especificación de prioridad relativa de un paquete. Por ejemplo un administrador de red podría dar la misma prioridad alta a paquetes HTTP y Telnet. Sin embargo, cuando haya una congestión en la red el administrador querrá descartar los paquetes Telnet antes que los HTTP, lo cual no es posible con este esquema.
- El manejo de sólo tres bits restringe el número de posibles clases de prioridad a ocho.
- Ninguna precedencia IP, ni los bits DTR han sido implementados consistentemente por los vendedores de redes hoy en día.

En adición, en el RFC 1349 se redefine el subcampo del ToS, utilizando los bits 3, 4, 5 y 6 y elimina el concepto DTR [URL, 10].

2.3.2 Campo Etiqueta de Flujo

En IPv6 la QoS se maneja mediante los campos: Etiqueta de Flujo y Clase de Tráfico (Figura 3). El campo Etiqueta de Flujo en IPv6 tiene el objetivo de clasificar los paquetes de acuerdo a su destino y a su servicio.

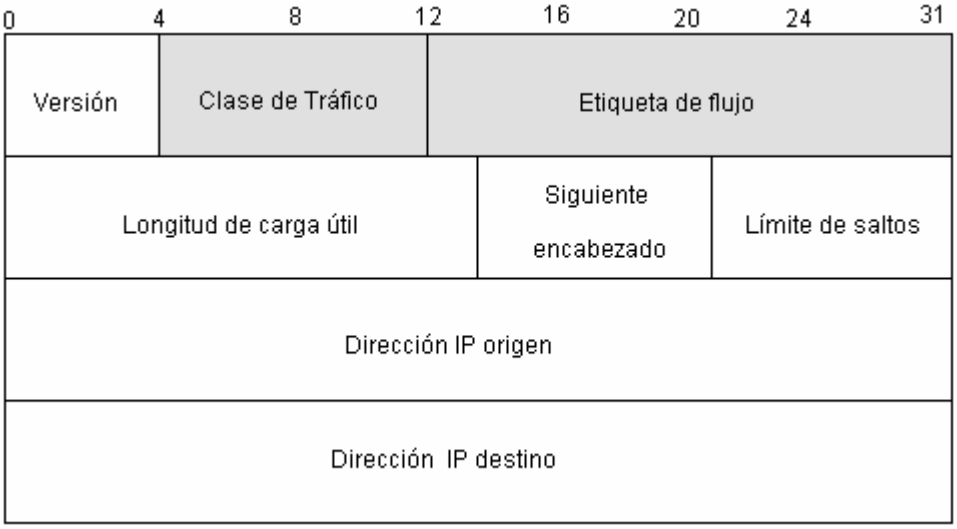

Figura 3. Encabezado de IPv6

El campo Etiqueta de Flujo tiene una longitud de 20 bits y es usado por el remitente para indicar que sus paquetes sean tratados de forma especial por los routers, como en servicios de alta calidad o en tiempo real. Es decir, se entiende el flujo como un conjunto de paquetes que requieren un tratamiento especial. Todos los paquetes pertenecientes al mismo flujo deben tener valores similares en el campo dirección de origen, dirección destino, prioridad y etiqueta de flujo. Mediante este campo se puede asignar a un flujo de tráfico un nivel específico de seguridad, retardo de propagación, caso de transmisiones VSAT *(Very Small Aperture Terminal – Terminal de apertura muy Pequeña)* o un costo.

En la Figura 4 se puede ver un ejemplo del trato que se le da a los flujos de datos mediante el campo etiqueta de flujo. En la primera parte de la imagen se observa que se identificaron dos tipos de flujo enviados por un remitente, Flujo 1 y Flujo 2, quienes viajaran a través de los routers de una red, a dichos flujos se les brindará un trato diferente es decir, al Flujo 1 se le asigna un tipo de calidad Q1 y al flujo 2 se le asigna un tipo de calidad Q2. En la segunda parte de la Figura 4 se puede observar como pasan los datagramas del Flujo 1 a través de los routers de la red, recibiendo así un trato adecuado, en cada router hasta llegar a su destino final.

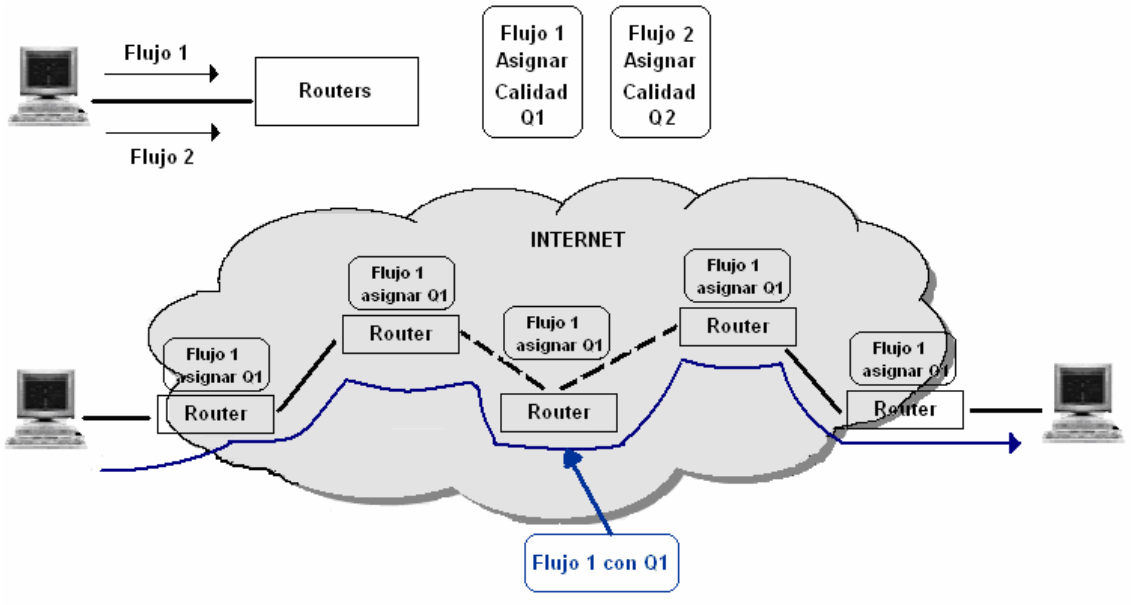

Figura 4. Tratamiento de paquetes mediante el campo Etiqueta de Flujo

2.3.3 Campo Clase de Tráfico (TC)

En IPv6 el campo TC es equivalente al campo ToS de IPv4, es un campo compuesto por 8 bits (un byte) que ayuda a identificar o diferenciar el tipo de tráfico que circula por la red. Usado por los remitentes o routers reenviantes para identificar y distinguir entre las diferentes clases o prioridades de paquetes IPv6.

En los Diff–Serv este campo es el que clasifica el tráfico más importante, asignándole mayor prioridad, actuando de forma similar a los bits de precedencia que se utilizan en IPv4.

2.3.4 Campo Servicio Diferencial (DS)

Para aplicar la diferenciación de servicios se definió un campo que sustituyera las actuales definiciones del campo ToS en IPv4 y el TC del IPv6, este campo es conocido como DS.

El campo DS consta de 8 bits, los primeros 6 bits son utilizados como parte del código mientras que los últimos dos bits deben ser ignorados por los nodos que

tengan implementado Diff-Serv. La estructura del campo DS se muestra en la Figura 5.

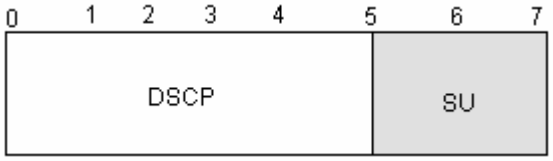

Figura 5. Campo DS

Donde:
DSCP = Diferential Service Code Point
SU = Sin uso

El campo DS permite definir hasta 64 posibles categorías de tráfico, aunque en la práctica se utilizan menos. Como se muestra en la Tabla 3, los valores de DS son divididos en tres grupos.

Tabla 3. Grupos de CodePoint del campo DS

CodePoint	Posibles valores	Uso
xxxxx0	32	Estándar
xxxx11	16	Local/Experimental
xxxx01	16	Reservado

Los valores de DSCP estándares son de la forma xyzab0, donde xyz coinciden con el campo de precedencia de IPv4 (001, 010, 011 ó 100) y ab es 01, 10 ó 11.

Cada código mapea un PHB determinado. Actualmente existen 4 PHB especificados para ser usados dentro de una red de servicios diferenciales:

1. Comportamiento por omisión (Default Behavior),
2. Selector de clase,
3. Encaminamiento expedito (Expedited forwarding), y
4. Encaminamiento asegurado (Assured forwarding)

2.4 Tipos de PHB

2.4.1 Comportamiento por omisión (BE PHB)

Es el comportamiento que todas las redes que implementen Diff–Serv deben de incorporar. Este comportamiento equivale a un servicio de mejor esfuerzo. Todos los paquetes que no tengan especificado un comportamiento, utilizan el servicio de mejor esfuerzo para moverse a través de la red. El código que representa el comportamiento por omisión es el 0x000000.

2.4.2 Selector de clase (CS PHB)

Este comportamiento define hasta ocho clases distintas en la red. El formato del código toma en cuenta los primeros 3 bits del octeto xxx000. Los tres primeros bits representan un número del 0 al 7. El número de menor valor representa una prioridad menor (es decir, los tres primeros bits son cero, el cual corresponde al comportamiento por omisión o de mejor esfuerzo) mientras que un número mayor representa una prioridad mayor. No es necesario que un nodo (puede ser un nodo interno) soporte las ocho clases (Tabla 4). Se pueden agrupar las clases para soportar por ejemplo dos prioridades. Los códigos con número 1 al 3 pueden representar una prioridad baja, mientras que los códigos con los números del 4 al 7 representan una prioridad alta. De esta forma, el nodo sigue siendo compatible con la especificación Diff-Serv, aún sin tener ocho clases definidas.

Tabla 4. Códigos para el selector de clase

Clase	Código	Servicio Correspondiente
0	000000	Mejor Esfuerzo
1	001000	Encaminamiento Asegurado Clase 1
2	010000	Encaminamiento Asegurado Clase 2
3	011000	Encaminamiento Asegurado Clase 3
4	100000	Encaminamiento Asegurado Clase 4
5	101000	Encaminamiento Expedito
6	110000	Reservado
7	111000	Reservado

Los nodos que utilizan este tipo de PHB pueden implementarlo utilizando los manejadores de colas SPQ *(Strictic Priority Queing – Encolamiento de Prioridad Estricta)*, WFQ *(Weighted Fair Queing – Encolamiento Ponderado)*, WRR *(Weighted Round Robin – Round Robin por Peso)* o CBQ *(Class Based Queing - Colas Basadas en Clases)*.

2.4.3 Encaminamiento expedito (EF PHB)

Este PHB tiene asociado una tasa de transmisión, la cual es definida por el ISP *(Internet Service Provider – Proveedor de Servicios de Internet)*. La función de este PHB es proveer las herramientas necesarias para proporcionar un servicio de extremo a extremo con bajas pérdidas, bajo retardo, bajo jitter[1] y un ancho de banda asegurado dentro de un dominio Diff-Serv.

El principio de operación de este PHB, es que la tasa de partida de los paquetes debe ser igual o mayor a una tasa configurada por el administrador. Esta tasa no puede ser menor que la tasa de llegada de paquetes. Esto significa que se tiene una serie de paquetes del mismo tamaño que llegan a un nodo, éstos saldrán del nodo con la misma tasa de entrada. La idea es reducir el exceso de retardo y jitter en lo posible.

Lo anterior se puede observar en la figura 6, que muestra el modo de operación de este tipo de PHB.

[1] jitter = Variación en el tiempo en la llegada de los paquetes, causada por la congestión de la red.

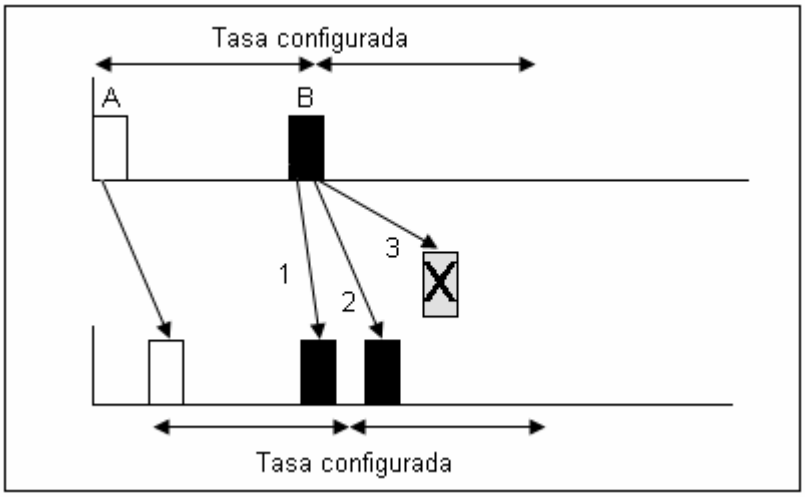

Figura 6. Modelado y descarte de paquetes

Cuando el paquete se aproxima antes de su tiempo programado de llegada, existen tres opciones en los nodos de ingreso e internos para su tratamiento:

1. Reenviar el paquete inmediatamente
2. Reenviar el paquete en el tiempo configurado
3. Descartar el paquete

En la figura 6 se muestran de manera general estas tres opciones, en la opción 1, se reenvía el paquete de forma inmediata, esto porque se trata de un paquete al que se esta proporcionando un servicio de ancho de banda garantizado, a través de una baja latencia, baja pérdida y bajo jitter, en la opción 2 se reenvía el paquete en el siguiente tiempo configurado, esto porque es un paquete que necesita un servicio con prioridad menor que el primero y en la opción 3 se está descartando un paquete porque se le esta dando prioridad a los dos servicios críticos anteriores.

Las opciones que toman los nodos de acceso e internos son diferentes: los nodos de acceso por lo general tomaran las opciones 2 y 3 para evitar que la fuente se apropie de un mayor ancho de banda del que se tiene configurado. Para los nodos internos, es altamente recomendada la opción 1, ya que la aplicación de la opción dos podría provocar retardos acumulados.

El EF PHB requiere un alto control sobre la tasa de transmisión de paquetes en los nodos de acceso a la red y de un rápido reenvío de paquetes en los nodos internos de la red.

De la especificación y de lo anteriormente visto se puede definir la región de acción en el modelo de servicio. Ya que es necesario un estricto control de la tasa de transmisión, el servicio ofrecido por este PHB es muy similar al de una línea dedicada.

La finalidad de EF PHB es la de proveer enlaces de alta calidad, con respecto a retardo y pérdidas. Puede ser utilizado para proveer enlaces que simulen enlaces dedicados, con bajos retardos y bajas variaciones en el ancho de banda.

2.4.4 Encaminamiento asegurado (AF PHB)

Este PHB define cuatro clases, a las cuales se les tiene que asignar espacio en el buffer y ancho de banda de manera independiente en cada nodo. A cada una de estas clases se le especifican tres niveles de descarte (Tabla 5). Es importante señalar que no es necesario implementar los tres niveles de descarte. Si el operador de la red, no espera que existan muchas condiciones de congestión, el número de niveles de descarte se puede compactar a dos [8].

Tabla 5. Códigos DS recomendados para AF PHB

	Clase 1	**Clase 2**	**Clase 3**	**Clase 4**
Baja probabilidad de descarte	001010	010010	011010	100010
Media probabilidad de descarte	001100	010100	011100	100100
Alta probabilidad de descarte	001110	010110	011110	100110

El tráfico de la red se divide entre los tres niveles de la siguiente manera:

- **Clase Oro – baja probabilidad de descarte**: El tráfico de esta categoría dispone del 50% del ancho de banda.
- **Clase Plata – media probabilidad de descarte**: El tráfico en esta categoría dispone del 30% del ancho de banda.

- **Clase Bronce – alta probabilidad de descarte**: Reserva el 20% del ancho de banda.

2.5 Tipos de nodos en una red Diff-Serv

En una red Diff–Serv, los nodos externos de la red (nodos externos o nodos extremos) y los nodos internos de la red (nodos internos o nodos centrales) tienen diferentes responsabilidades (Figura 7). Los nodos externos realizan funciones tales como el control de admisión, vigilancia, acondicionamiento y contabilidad del tráfico, en cambio el comportamiento de los nodos internos dependerá únicamente de la clase de servicio asociada a cada paquete [URL, 12][URL, 13].

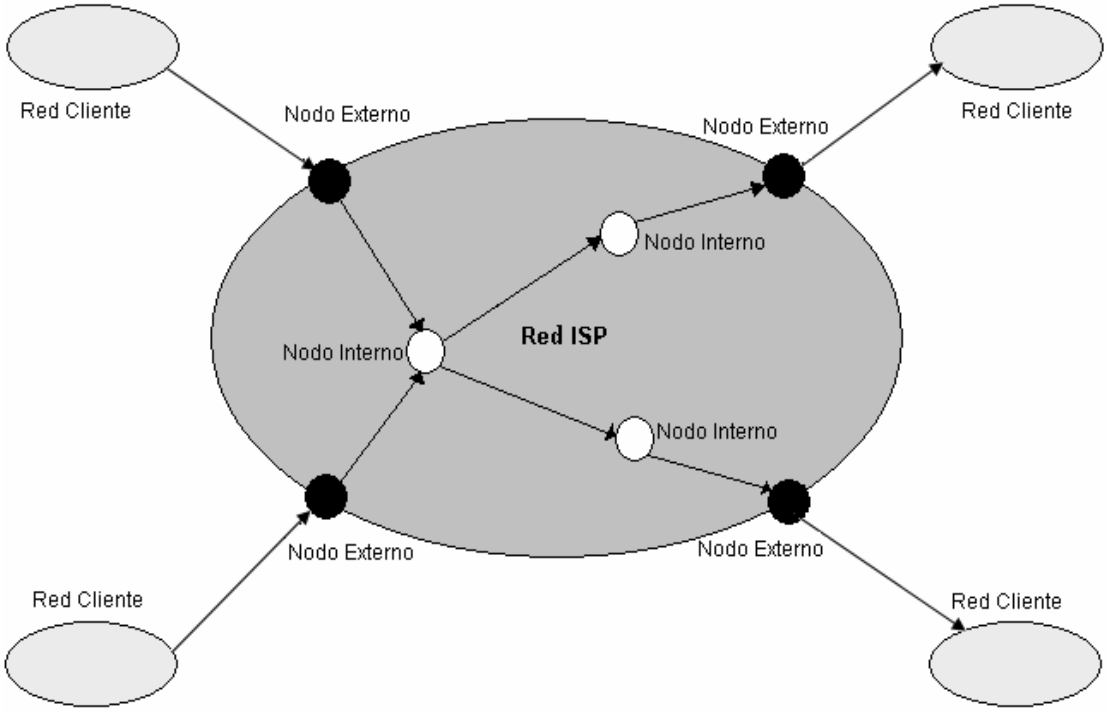

Figura 7. Nodos externos y nodos internos.

2.5.1 Nodos externos DS

Los nodos externos realizan funciones complejas, tales como el acondicionamiento de tráfico entre los dominios Diff–Serv interconectados y la clasificación de paquetes. De esta manera se debe clasificar y establecer las condiciones de ingreso de los flujos de tráfico en función de: dirección IP y puerto

(origen y destino), protocolo de transporte y DSCP. Este clasificador se conoce como MF *(Multi-Field Classifier - Clasificador Multi Campo)*.

Una vez que los paquetes han sido marcados adecuadamente, los nodos internos deberán seleccionar el PHB definido para cada flujo de datos. Los nodos DS de entrada serán responsables de asegurar que el tráfico de entrada cumple los requisitos de algún TCA *(Traffic Conditioning Agreement – Acuerdo de Acondicionamiento de Tráfico)*, que es un derivado del SLA *(Service Level Agreement – Acuerdos de Nivel de Servicio)*, entre los dominios interconectados. Por otro lado los nodos DS de salida deberán realizar funciones de acondicionamiento de tráfico o Tc *(Traffic Conformation – Conformación de tráfico)* sobre el tráfico transferido al otro dominio DS conectado.

2.5.2 Nodos internos DS

Los nodos internos, realizan funciones más simples que los nodos externos, tales como remarcado de DSCP. Los nodos DS internos sólo se conectan a nodos internos o a nodos externos de su propio dominio. En estos nodos la selección de PHB es realizada sólo analizando el contenido DSCP conocido como BA *(Behavior Agregate Clasifier – Clasificador de Comportamiento Agregado)* [URL, 11].

Los nodos DS de entrada son responsables de asegurar que el tráfico de entrada esté conformado por algún acuerdo de condición entre los dominios a conectarse. Los nodos DS de salida deben de realizar las funciones de acondicionamiento de tráfico sobre los flujos de datos transferidos al siguiente dominio DS.

2.6 Características de los Diff–Serv

Los Diff–Serv plantean la asignación de prioridades a cada uno de los paquetes que son enviados a la red. Cada router debe analizar y dar un tratamiento diferencial a cada uno de los paquetes. En este enfoque no se necesita asignar ningún estado ni establecer algún proceso de señalización en cada nodo. Esta es la razón principal del porqué Diff–Serv ofrece mejor

escalabilidad que Int–Serv. El grupo de trabajo sobre Diff–Serv de la IETF, define el campo DS válido tanto para IPv4 e IPv6, como el campo donde se asignan las prioridades a los paquetes asimismo describen las técnicas para implementar y escalar este modelo en Internet.

Diff–Serv permite ofrecer servicios de red basados en un conjunto de reglas de salto perfectamente definidas en los routers que componen la red del proveedor Diff–Serv. Esas reglas se pueden aplicar a flujos agregados de tráfico mediante la asignación de un código de Diff–Serv (DSCP) a los paquetes del flujo de tráfico que corresponden a la regla de salto. Entre un proveedor de red Diff–Serv y sus clientes se negocian contratos de nivel de servicio SLA. Esos contratos especifican los servicios que el proveedor debe ofrecer al cliente. Además, un SLA incluye un contrato de configuración del tráfico TCA que establece los requisitos de configuración que debe cumplir el tráfico para que estén disponibles los servicios especificados. Los contratos de requisitos de configuración del tráfico especifican:

- Quién debe asignar el código de Diff–Serv correspondiente a un servicio determinado a los paquetes del cliente, así como las reglas en las que se basa esa asignación.
- Qué perfil debe tener el tráfico que se envía para un servicio determinado.

2.6.1 Elementos de la arquitectura Diff–Serv

La arquitectura de los Diff–Serv comprende varios elementos, que se muestran en la figura 8 y que se describen a continuación:

- **Clasificador:** Tiene la función de guiar los paquetes con características similares hacia los procedimientos de condicionamiento de tráfico. Los clasificadores deben ser configurados por medio de un procedimiento de administración de acuerdo con el TCA apropiado.

- **Medidor:** Los medidores pasan su información hacia las funciones de acondicionamiento para determinar alguna acción en particular para cada paquete que se encuentre ya sea dentro o fuera del perfil de tráfico.

- **Marcador:** Puede ser configurado para marcar los paquetes mediante un conjunto de códigos DS para seleccionar un PHB dentro de un grupo de PHB's, de acuerdo a las mediciones realizadas por el módulo anterior. Cuando el marcador cambia el código DS de un paquete se dice que este paquete fue remarcado.

- **Acondicionador:** El acondicionador regularmente posee un tamaño de cola finito, procediendo al descarte de paquetes cuando no exista suficiente espacio dentro de la cola, con la finalidad de mantener el retardo de los paquetes suficientemente pequeño.

- **Descartador:** Se encarga de descartar paquetes pertenecientes a un flujo de tráfico con la finalidad de evitar congestionamiento y de que el flujo cumpla con los requisitos acordados dentro del perfil de tráfico.

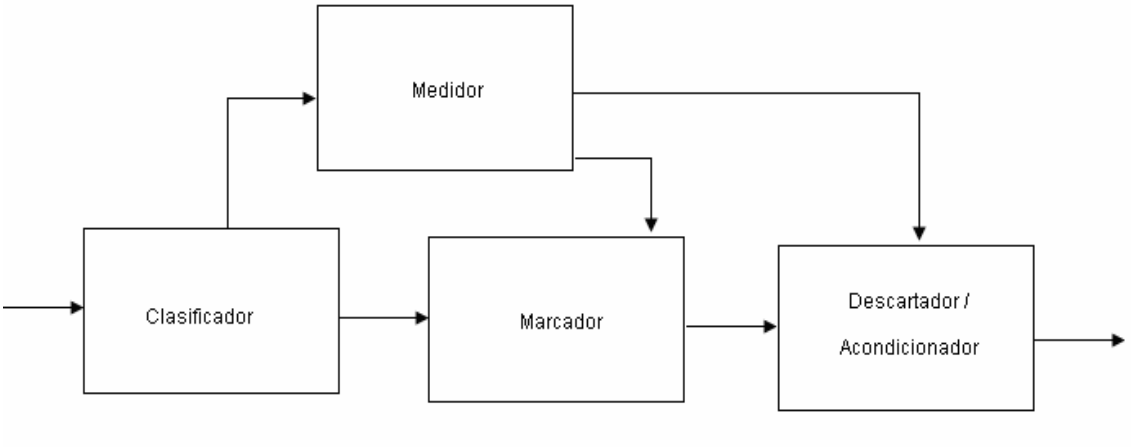

Figura 8. Arquitectura de Diff–Serv

2.7 Representación UML de los Diff–Serv

Para realizar la propuesta de representación del modelo Diff–Serv, se utilizaron sólo tres de los diagramas que utiliza UML para la representación de un sistema, se utilizarán los diagramas de casos de uso, de actividades y de secuencia para ilustrar de una manera gráfica y a un nivel de representación más abstracta y entendible el funcionamiento de la arquitectura de Diff–Serv.

Con esto no se pretende presentar un modelado orientado a objetos sino la representación de las características de los actores y los elementos identificados en el modelo analizado en este capítulo.

2.7.1 Diagrama de Casos de Uso

En el diagrama de casos de uso del modelo Diff–Serv se muestran los actores que se identificaron de manera general y que son los que llevan a cabo el otorgamiento de QoS en una red IP (Figura 9).

En la Tabla 6, se muestra la descripción de los actores identificados en el diagrama de casos de uso del modelo Diff–Serv.

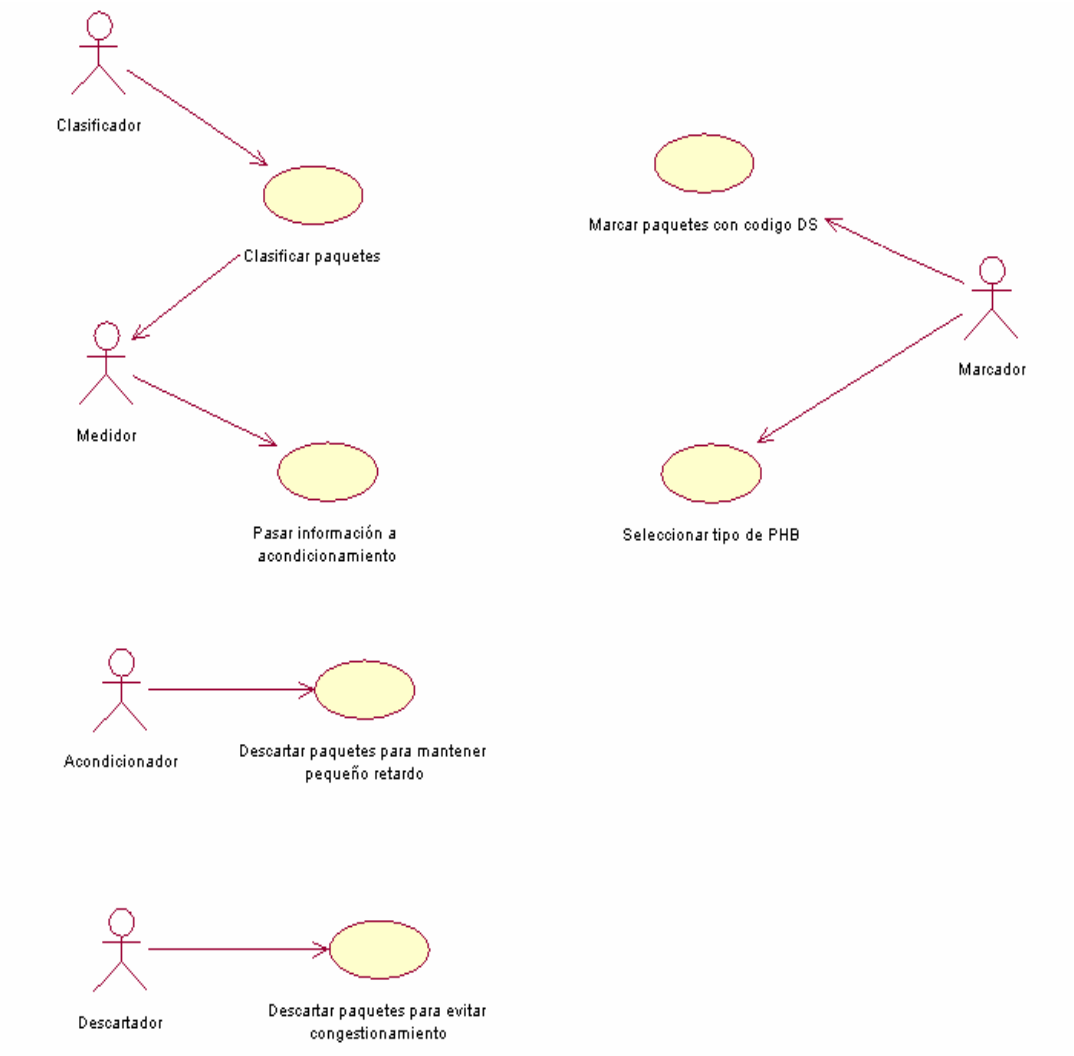

Figura 9. Diagrama de casos de uso del modelo Diff–Serv

Tabla 6. Descripción de actores de los casos de uso del modelo Diff–Serv

Actor	Descripción
Clasificador	Una vez que el emisor envía un paquete, el clasificador tiene la función de guiar paquetes similares.
Medidor	Pasa información hacia las funciones de acondicionamiento para determinar alguna acción en particular para cada paquete
Marcador	Marca a los paquetes con los códigos DS para seleccionar un PHB.
Descartador	Descarta paquetes pertenecientes a un flujo, con la finalidad de evitar congestionamiento en la red.
Acondicionador	Acondiciona a los paquetes recibidos para después enviarlos al siguiente nodo, procediendo al descarte en caso de que no exista suficiente espacio dentro de la cola.

2.7.2 Diagrama de Actividades

En el diagrama de actividades se muestra de una forma más detallada las actividades que realizan cada uno de los actores representados e identificados en los casos de uso, así mismo se puede observar en que momento se va realizando la reserva de recursos hasta llegar al objetivo que es el otorgamiento de QoS o en su caso el descarte de los paquetes (Figura 10).

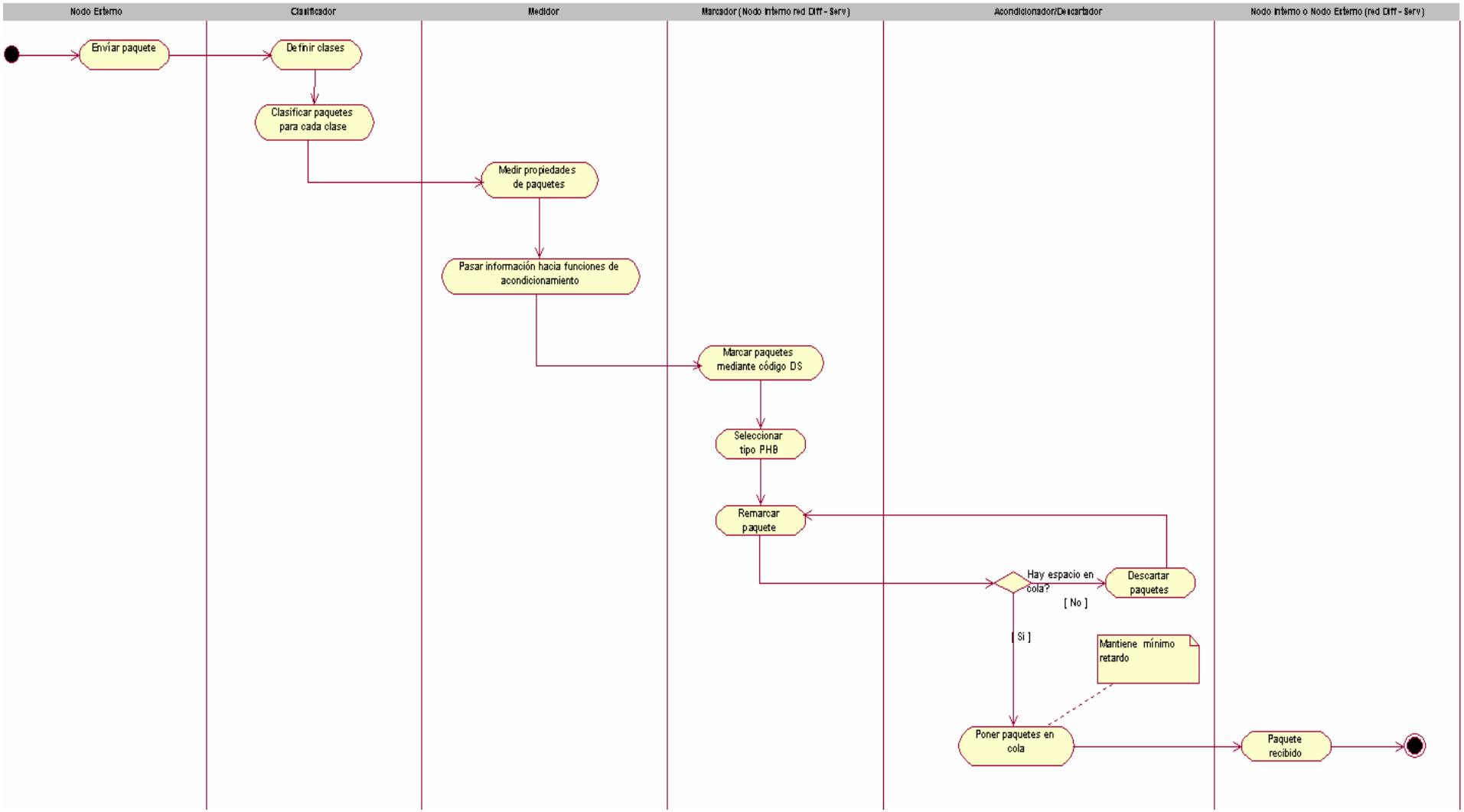

Figura 10. Diagrama de actividades del modelo Diff–Serv

2.7.3 Diagrama de Secuencia

El diagrama de secuencia realizado para el modelo Diff–Serv, se presenta de forma descriptiva, representa el envío de mensajes y la secuencia de acciones que realiza cada uno de los actores identificados en los diagramas anteriores y que colaboran en el proceso de otorgamiento de QoS dentro de una red con la arquitectura estudiada en el presente capitulo (Figura 11).

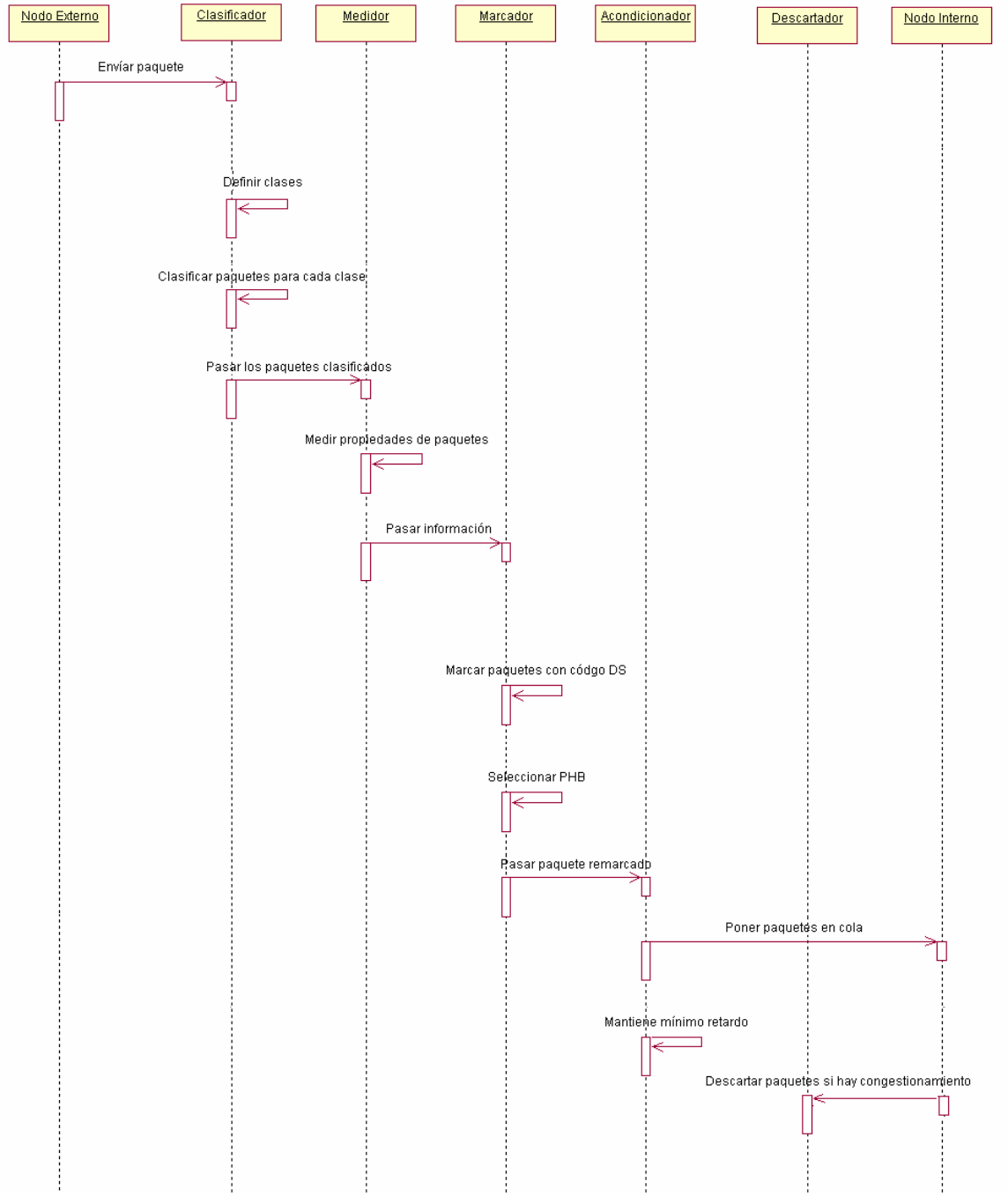

Figura 11. Diagrama de secuencia del modelo Diff–Serv

2.8 Ventajas y desventajas de los Diff–Serv

Las ventajas y desventajas que se han observado al estudiar y analizar este modelo se muestran en [URL, 22]:

Ventajas

- **Escalabilidad**: Mejora la complejidad de mantener información de estado de muchos circuitos virtuales. Proporciona soporte escalable para flujos de datos y voz sobre una única infraestructura.
- **Funcionamiento**: El contenido de los paquetes se inspecciona sólo una vez para clasificarlo. En ese momento, el paquete es marcado y todas las decisiones de QoS posteriores se hacen de acuerdo al valor en un campo fijo de la cabecera, reduciendo los requerimientos de procesamiento.
- **Flexibilidad**: Permite definir muchos tipos de tráfico.
- **Sencillez de señalización**: Es mucho más sencillo que en el protocolo RSVP de Int–Serv.
- **Costo**: Costos de gestión reducidos.

Desventajas

- No hay reservaciones de ancho de banda extremo a extremo, por lo tanto, las garantías de servicio pueden ser imparciales en los nodos de la red que no implemente los PHB's correctamente sobre enlaces congestionados o por nodos que no están correctamente diseñados para el volumen de tráfico esperado de una clase específica.
- La ausencia de control de admisión por flujo o por sesión hace posible que las aplicaciones se congestionen unas con otras.

2.9 Comentarios Finales

Como se ha observado a lo largo de este capítulo los Diff–Serv son una propuesta bastante alentadora para el ofrecimiento de QoS en redes IP, ya que con la arquitectura que propone se puede observar que no hay necesidad de incrementar grandes cambios en las redes existentes para proporcionar QoS.

Por otro lado, la forma en que maneja los campos de las cabeceras de los protocolos IPv4 e IPv6, hace que sea una de las mejores soluciones que se propone hoy en día en cuestión de asignación de QoS.

«CAPÍTULO 3»

SERVICIOS INTEGRALES

3.1 Introducción

Como respuesta a la necesidad de ofrecer QoS en las redes IP, se han creado modelos que proporcionen mecanismos para el buen funcionamiento de éstas, soportando las nuevas y futuras necesidades de los usuarios como son: telefonía, realidad virtual, audio y video en tiempo real, entre otras. Todo esto manejando la infraestructura de las redes implementadas actualmente, tratando de no alterarlas completamente.

Por lo anterior en 1994 la IETF empezó a definir la arquitectura de los Int–Serv que pretendía ampliar la arquitectura IP existente para soportar actividades en tiempo real, manteniendo el servicio Best Effort que existía hasta el momento [14].

En este capítulo se estudiará y analizará este modelo, sus características, su arquitectura, ventajas y desventajas que pueda tener en comparación con los Diff–Serv y otros modelos.

3.2 Modelo de Servicios Integrales

En la arquitectura Int–Serv el concepto de flujo juega un papel fundamental. El flujo se define como un tráfico continuo de datagramas relacionados entre sí que se produce como consecuencia de una acción del usuario y que requiere una misma QoS. Un flujo es unidireccional y es la entidad más pequeña a la que se le puede aplicar una determinada QoS. Los flujos pueden agruparse en clases; todos los flujos pertenecientes a una misma clase reciben la misma QoS. [URL, 15]

En IPv4 un flujo se identifica por las direcciones de origen y destino, el puerto de origen y destino, y el protocolo de trasporte utilizado (TCP o UDP). En IPv6 la identificación se puede hacer de la misma forma que en IPv4, o mediante las direcciones de origen y destino, y el valor del campo Etiqueta de Flujo.

En la arquitectura Int–Serv se definen tres tipos de servicio [14]:

- **Servicio Garantizado**: Garantiza un caudal mínimo y un retardo máximo. Cada router que se encuentre en el trayecto del datagrama debe ofrecer las garantías solicitadas, aunque a veces esto no es posible por las características del medio físico.
- **Servicio de Carga Controlada**: Este servicio debe ofrecer una calidad comparable a la de una red de datagramas poco cargada, en general deben proporcionar un buen tiempo de respuesta, pero sin garantías estrictas. Eventualmente se pueden producir retardos grandes.
- **Servicio Mejor Esfuerzo**: Este servicio no tiene ninguna garantía, ni ofrece QoS.

El modelo Int–Serv dispone del protocolo RSVP (*Resource reSerVation Protocol, Protocolo de Reservación de Recursos*), se basa en este protocolo para señalizar y reservar la QoS deseada para cada flujo de datos en la red. Debido a que la información de estados para cada reservación necesita ser mantenida por cada router a lo largo de la ruta del datagrama.

Int–Serv provee a las aplicaciones de un nivel garantizado de servicio, negociando parámetros de red de punto a punto. La aplicación solicita el nivel de servicio necesario para ella con el fin de operar apropiadamente, y se basa en la QoS para que se reserven los recursos de red necesarios antes de que la aplicación comience a operar. Estas reservaciones se mantienen habilitadas hasta que la aplicación termina o hasta que el ancho de banda requerido por ésta sobrepase el límite reservado para la aplicación.

3.3 Componentes básicos de los Int–Serv

Los cuatro componentes básicos de la arquitectura Int–Serv se describen a continuación [14][16][17][URL, 19]:

- **El control de admisión**: Comprueba que existen recursos suficientes para soportar el servicio solicitado.

- **Clasificador de paquetes**: Analiza los campos de direcciones y puertos para determinar la clase a la que pertenece el paquete.

- **Planificador de paquetes**: Aplica algoritmos de encolado que gestionan la transmisión de los paquetes por un enlace de salida.

- **El protocolo RSVP**: Para que una aplicación pida un determinado servicio a la red. El protocolo entrega la petición al control de tráfico de cada router, que comprobará si es viable la petición.

3.4 Protocolo de Reservación de Recursos (RSVP)

El protocolo RSVP fue creado en 1990 por la IETF, definiendo un modelo de asignación de QoS en el cual cada receptor es responsable de elegir su propio nivel de reserva de recursos, iniciando la reserva y manteniéndola activa el tiempo que sea necesario.

RSVP es un protocolo de señalización que permite el establecimiento de los Int–Serv. Es uno de los protocolos más complejos en cuanto a tecnologías de QoS se trata, tanto para los sistemas finales como para los routers de la red que seleccionen este modelo para el ofrecimiento de QoS. Este protocolo opera bajo IPv4 e IPv6.

Ya que no es un protocolo de encaminamiento, está pensado para trabajar conjuntamente con éstos. Los protocolos de encaminamiento determinan dónde se

reenvían los paquetes mientras que RSVP se preocupa de la QoS de los paquetes reenviados de acuerdo con el encaminamiento.

RSVP es un protocolo orientado a conexión, ya que los routers tienen que guardar una cierta información de estado de cada flujo para el que efectúa la reserva, algo equivalente a un circuito virtual [URL, 15]. Gracias a este protocolo los receptores pueden reservar recursos de la red para un flujo de datos específico. Por ejemplo para realizar una sesión de videoconferencia se utiliza este mecanismo, siempre y cuando la red IP por la que se quiera transmitir soporte esta aplicación, tratando de garantizar un determinado ancho de banda.

También representa el mayor cambio con relación al servicio Best Effort de IP, RSVP tiene el mayor nivel de QoS en términos de servicios garantizados y también la mayor granularidad de los mismos. RSVP es un protocolo situado a nivel 4 o de transporte.

Para implementar RSVP los routers deberán incorporar cuatro elementos como se muestra en las Figuras 12 y 13:

- **Control de Admisión**: Comprueba si la red tiene los recursos suficientes para satisfacer la petición.

- **Política de Control**: Determina si el usuario tiene los permisos adecuados para la petición realizada. La comprobación se puede realizar consultando una base de datos mediante el protocolo COPS (*Common Open Policy Service - Política de Servicios Comúnmente Abiertos*).

Si los dos elementos anteriores se cumplen se activa el elemento Clasificador de Paquetes, y si alguno falla, se genera una notificación de error que se envía a la aplicación que ha solicitado la reserva.

- **Clasificador de Paquetes**: Clasifica los paquetes en categorías de acuerdo con la QoS a la que pertenecen. Cada categoría tendrá una cola y un espacio propio para buffers en el router.

- **Planificador de Paquetes**: Organiza el envío de los paquetes dentro de cada categoría.

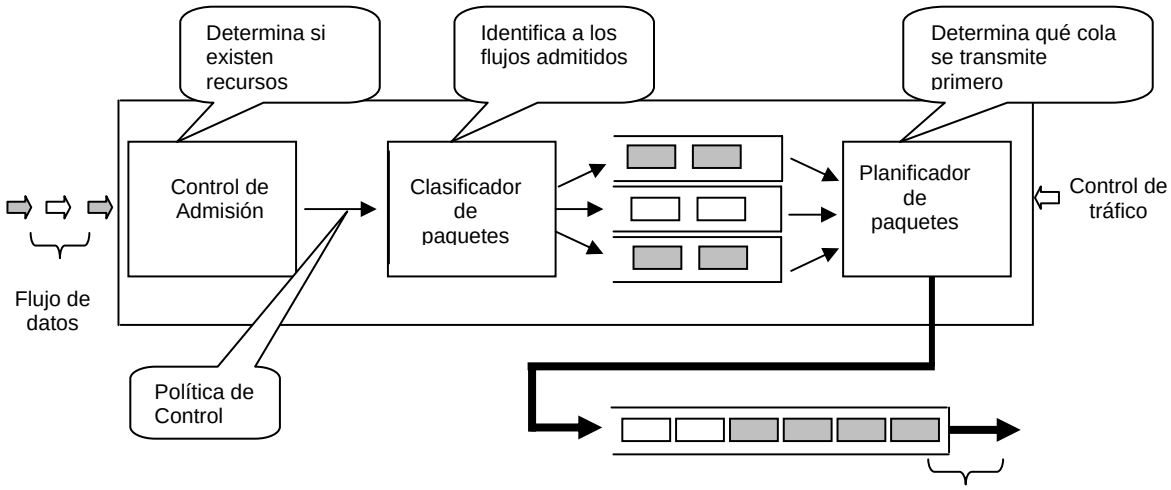

Figura 12. Arquitectura de Int–Serv

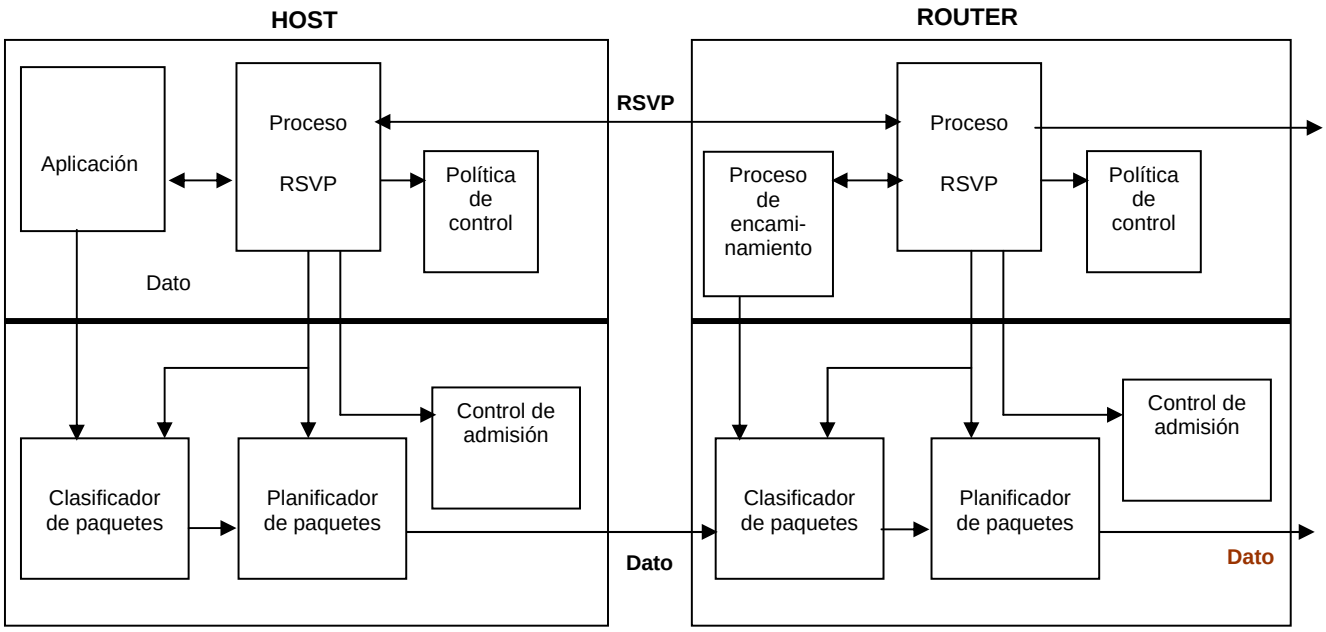

Figura 13. Elementos del protocolo RSVP en Hosts y Routers

3.4.1 Características del protocolo RSVP

El protocolo RSVP cuenta con las siguientes características [18]:

- Realiza reservas de recursos para aplicaciones unicast y multicast, adaptando dinámicamente los cambios en miembros y rutas.
- Se hacen reservas para flujos de datos unidireccionales.
- Mantiene el soft state[2] en routers y hosts.
- No es un protocolo de encaminamiento, pero depende de los protocolos de encaminamiento.
- RSVP transporta y mantiene el control de tráfico y los parámetros de la política de control que son opacos para RSVP.
- Mantiene y transporta parámetros de control de tráfico, que le son transparentes.
- Proporciona varios modelos o estilos de reserva para adaptarse a una gran variedad de aplicaciones.
- RSVP provee operaciones transparentes a través de routers que no lo soporten.
- Soporta IPv4 e IPv6.

3.4.2 Funcionamiento del protocolo RSVP

En el protocolo RSVP se manejan varios mensajes para llevar a cabo la reservación de recursos, los mensajes primarios son *Path* y *Resv* [14][URL, 20].

El emisor envía un mensaje denominado *Path*, con su especificación de tráfico, hacia el destino o destinos. El propósito del mensaje *Path* es el de marcar la ruta entre emisor y receptor además de recolectar información sobre que tan viable es la solicitud a lo largo del camino. La especificación que se realiza incluye los valores máximo y mínimo de ancho de banda, retardo y variación del mismo. Cada router va guardando la ruta por la que va circulando el mensaje de *Path*, añadiendo la dirección IP de donde viene el mensaje, para que después pueda reconstruirse la ruta de regreso. Al llegar el mensaje *Path* al receptor o receptores, pueden medir qué tipo de servicio puede soportar la red (Tabla 7).

[2]Soft state = Estados en los routers y hosts extremos, refrescados por los mensajes Path y Resv del protocolo RSVP.

Tabla 7. Información que incluye un mensaje *PATH*

Mensaje PATH	**Descripción**
Phop	Dirección del último nodo RSVP capaz de enviar este mensaje Path.
Sender Template	Contiene la dirección IP del emisor y opcionalmente el puerto.
Sender Tspec	Define las características del tráfico.
Adspec	Opcional. Contiene información que es actualizada por cada router.

Cada router intercepta los mensajes *Path* y verifica su validez. Si se detecta un error el router enviará un mensaje *PathErr* para informar al emisor que no puede realizar ninguna acción apropiada. Si el mensaje es válido el router hace lo siguiente:

- Actualiza el estado de la entrada de la ruta para el emisor identificado en el *Sender Template*. Si no existe la ruta la crea. El estado de la ruta incluye *Sender Tspec*, dirección. *Phop* es necesaria para encaminar los mensajes *Resv* en el sentido contrario.
- Actualiza los contadores de limpieza de rutas a su valor inicial.

RSVP agrega un protocolo de mensaje de refresco periódico para mantener un estado de los routers intermedios para proporcionar fiabilidad y seguridad. Para ello, cada entrada en el router tiene un controlador asociado que cuando llega a cero se elimina la conexión. Para que esto no ocurra las rutas activas tienen que recibir un refresco por medio de mensajes *Path* en intervalos regulares. El periodo de refresco debe ser menor al tiempo de los contadores de limpieza para que no produzca desconexiones. Además de la eliminación de las rutas de forma automática, RSVP incluye el mensaje *PathTear* para eliminar la ruta de forma activa.

Es el receptor o receptores los que realmente hacen la reserva de recursos, al enviar un mensaje *Resv,* (Tabla 8). Dicho mensaje incluye además de la especificación de tráfico recibida del emisor, la especificación requerida por el receptor, que consta del tipo de servicio integral solicitado y un filtro que selecciona los paquetes con una determinada característica (por ejemplo protocolo

y número de puerto) a los que se va aplicar la reserva. El identificador de sesión que utilizan los routers está compuesto por el tipo de servicio integral y el filtro.

Tabla 8. Parámetros de los mensajes *RESV*

Mensaje RESV	**Descripción**
Estilo de Reserva	Indica el estilo de reserva a utilizar.
FilterSpec	Especificación de filtro para identificar a los emisores.
FlowSpec	Se compone del Rspec y la especificación de tráfico, Tspec.
ResvConf	Opcionalmente se envía un objeto de confirmación conteniendo la dirección IP del receptor.

Cuando un router recibe un mensaje tipo *Resv*, usa el control de admisión para aceptar o no la reserva. En caso positivo se hace la reserva y el mensaje *Resv* progresa hacia el siguiente router en la dirección del emisor. En caso contrario se envía un mensaje de error *ResvErr* al receptor.

- El mensaje *Resv* se envía de regreso por la ruta que ha recorrido. Los mensajes se pueden fusionar con otros mensajes *Resv* con la misma interfaz, obteniendo un nuevo *FlowSpec* y *FilterSpec*.

Si el router no soporta RSVP retransmite los mensajes RSVP de forma transparente. En estos enlaces no se puede garantizar la calidad de servicio, lo que implica que puede perderse la calidad de servicio extremo a extremo. Si el último router efectúa la reserva envía un mensaje de confirmación al receptor. Cuando la sesión termina debe indicarse, para liberar los recursos de la reserva.

Se necesita una interfaz para que las aplicaciones se comuniquen con RSVP. Las aplicaciones suministran la especificación de tráfico, inician el proceso de reserva y reciben la correspondiente notificación acerca de lo que ha ocurrido con la misma. También deben ser informadas de lo que pueda suceder a lo largo de la existencia de la sesión.

Las reservas las efectúa el receptor, para soportar grandes y heterogéneos grupos receptores de multidifusión. Como se ha indicado anteriormente, RSVP permite a una aplicación especificar la mayor granularidad y la mejor calidad de servicio posible. El precio que hay que pagar por ello es una mayor complejidad y

procesamiento, lo cual no es apropiado para muchas aplicaciones y partes de la red.

3.4.3 Mensajes de establecimiento de rutas

Como se vio en la sección anterior existen varios mensajes utilizados por los emisores y routers para llevar a cabo la reservación de recursos, en esta sección se muestra un ejemplo en donde se involucra un emisor y tres receptores, con la finalidad de entender en que dirección se envían los mensajes RSVP, (Figura 14) [URL, 21].

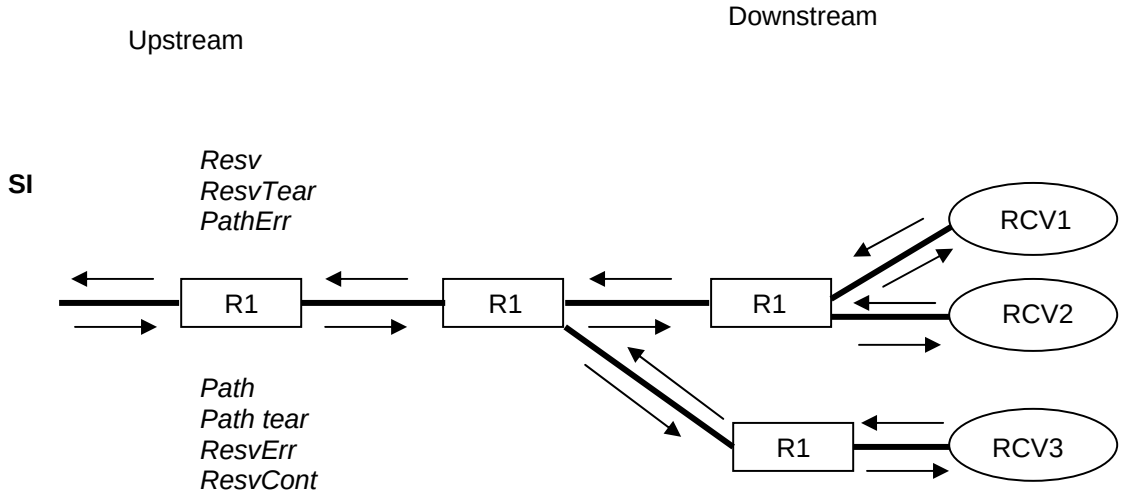

Figura 14. Dirección en la que viajan los mensajes RSVP

El emisor SI envía un mensaje primario Path a tres receptores RCV1, RCV2 y RCV3, el cual va a instalar un estado de encaminamiento inverso a través de los routers y además proporcionará a los receptores información sobre las características del tráfico a enviar y de la ruta para que se puedan realizar las peticiones de reserva adecuadas, al llegar a los receptores ellos enviarán mensajes Resv que realizarán las peticiones de reserva en los routers a lo largo de la ruta, así mismo se irán enviando los mensajes Resv Tear, Path Err, Path tear, ResvErr y ResvCont, dependiendo del tipo de reserva que se vaya produciendo y de los errores que se vayan generando en caso de no llevarse a cabo la reserva de recursos.

3.5 Representación UML de los Int–Serv

Una vez más al igual que en la sección 2.7 se utilizarán sólo se utilizaron tres de los diagramas que utiliza UML para representar el funcionamiento del modelo Int–Serv, dicho diagramas son, diagramas de casos de uso, de actividades y de secuencia, con el propósito de ilustrar como se mencionó anteriormente, de manera gráfica y a un nivel de representación más abstracta y entendible, recalcando que al igual que en el modelo Diff–Serv, solo se muestra la representación de los actores y los elementos identificados para la comprensión de la lógica del modelo estudiado en el presente capítulo.

3.5.1 Diagrama de Casos de Uso

En el diagrama de casos de uso del modelo Int–Serv se muestran los cuatro actores que se identificaron de manera general y que son los que llevan a cabo el protocolo RSVP, mismos que ya fueron mencionados en secciones anteriores en el presente capítulo. Esta representación nos va a ayudar a darnos una idea más clara de cómo se va llevando a cabo la reserva de recursos para otorgar QoS a los paquetes a través de una red IP (Figura 15).

En Tabla 9 se muestra una descripción de los actores que fueron identificados en los casos de uso para el modelo analizado en el presente capítulo.

Tabla 9. Descripción de actores de los casos de uso del modelo Int–Serv

Actor	Descripción
Control de Admisión	Una vez que se recibe el paquete, comprueba que haya recursos disponibles en la red para iniciar la reserva.
Política de control	Determina si el usuario tiene suficientes permisos para realizar la reserva.
Clasificador de paquetes	Clasifica los paquetes en diferentes categorías.
Planificador de paquetes	Una vez que están los paquetes clasificados, se encarga de organizar como los irá enviando al siguiente nodo.

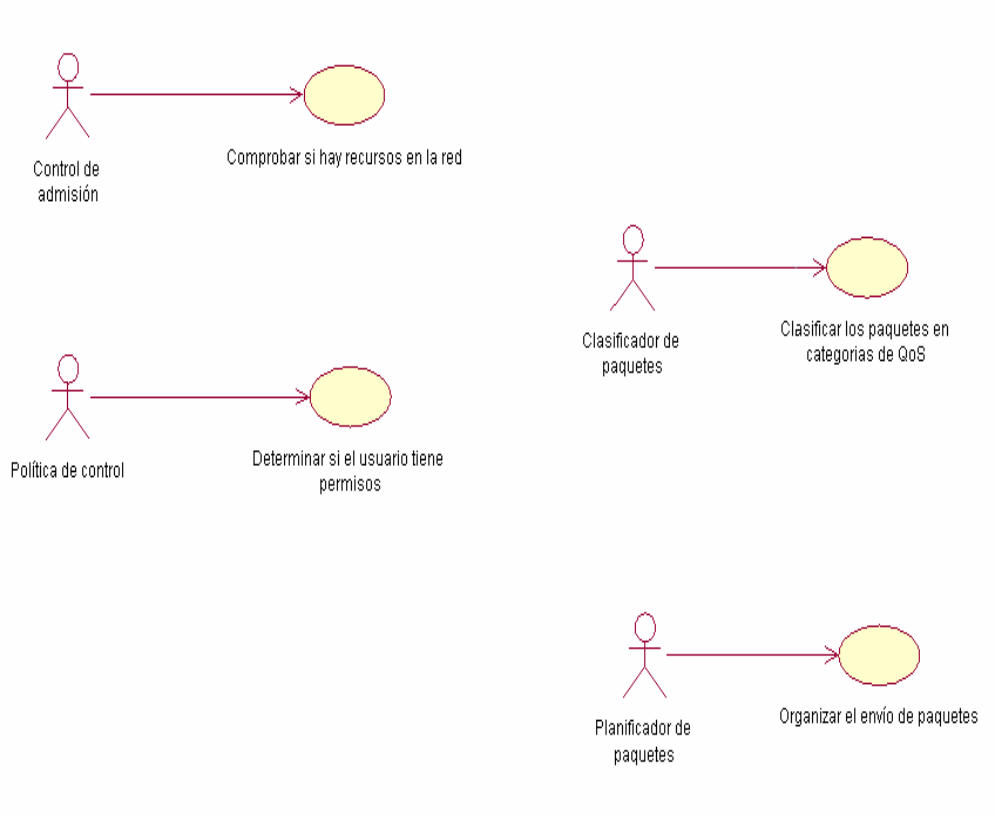

Figura 15. Diagrama de casos de uso del modelo Int–Serv

3.5.2 Diagrama de Actividades

En el diagrama de actividades para el modelo Int–Serv se muestra de una forma más detallada las acciones que realizan las entidades identificadas en el protocolo RSVP, parte que es más complicada de comprender del modelo estudiado en el presente capitulo y que es el protocolo que ayuda a los actores a realizar la reserva de QoS, así mismo se puede observar en que momento se va realizando la reserva de recursos hasta cumplir con el objetivo del modelo que es la entrega del paquete al receptor o en su caso el descarte de los paquetes si no se realiza la reserva de recursos (Figura 16).

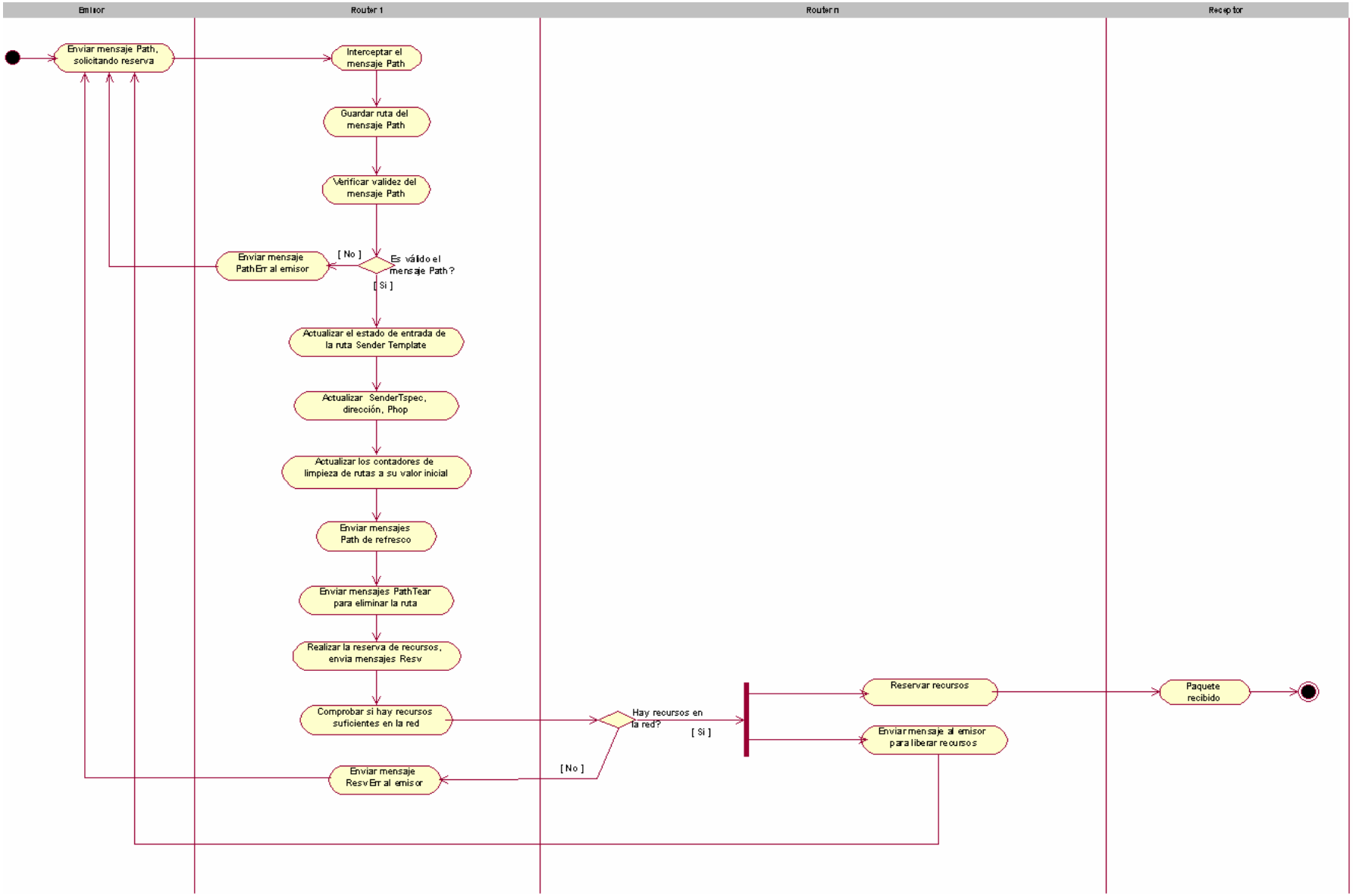

Figura 16. Diagrama de actividades del modelo Int – Serv

3.5.3 Diagrama de Secuencia

El diagrama de secuencia realizado para este modelo, se presenta de forma descriptiva, no representa un envío de mensajes, sino la secuencia de acciones que realiza cada una de las entidades identificadas dentro de la arquitectura Int–Serv y que ayudan al funcionamiento del protocolo RSVP al momento de otorgar QoS a los paquetes dentro de una red IP (Figura 17).

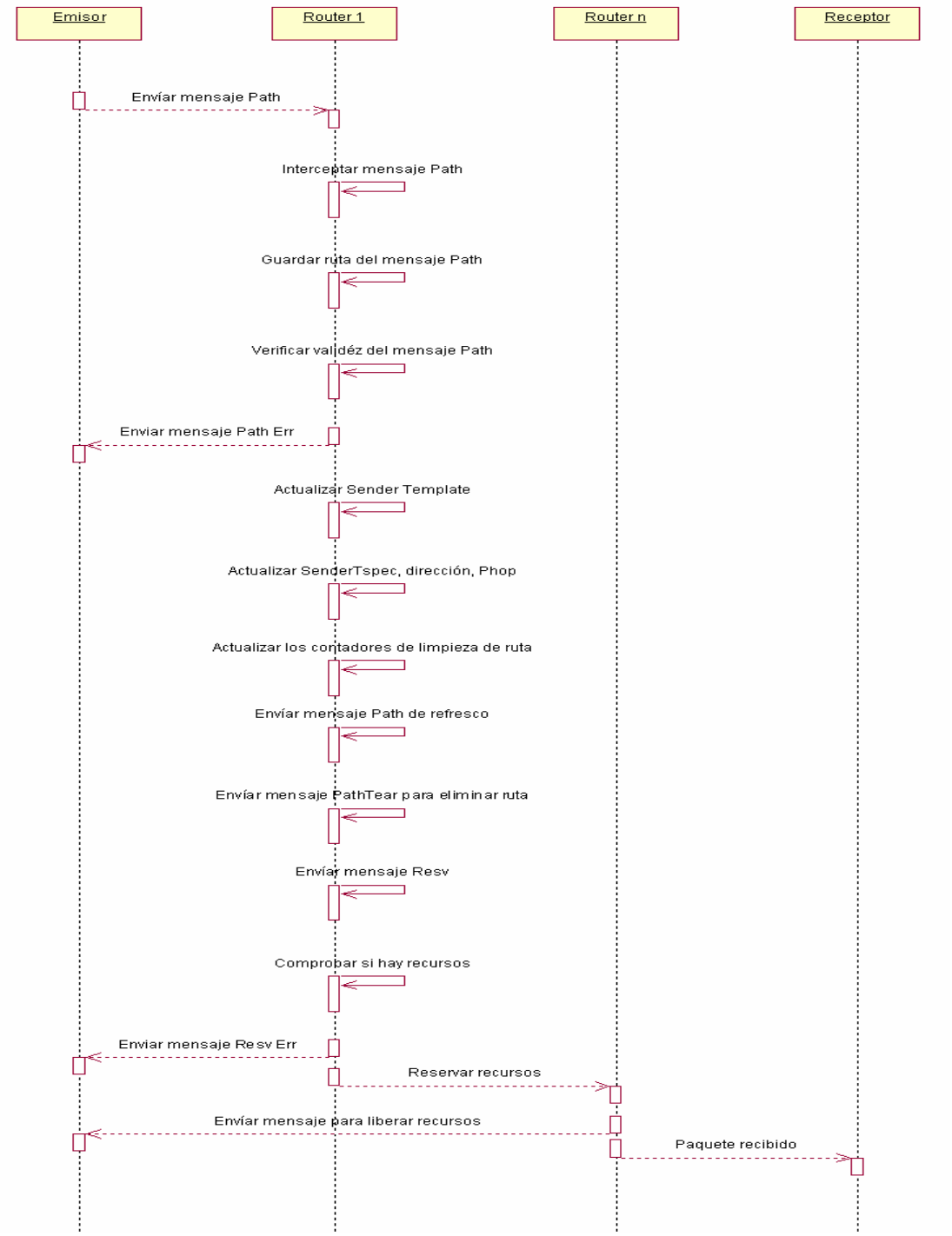

Figura 17. Diagrama de secuencia del modelo Int – Serv

3.6 Ventajas y desventajas de los Int–Serv

En esta sección se mostrarán las ventajas y desventajas que se han observado al estudiar y analizar el modelo de los Int–Serv [URL ,15] [URL, 22].

Ventajas:

- Simplicidad conceptual, facilitando la integración con la administración de políticas de la red.
- QoS discreta por flujo, haciéndolo arquitecturalmente conveniente para control de admisión de llamadas de voz, lo cual permite avisar a los puntos finales si el ancho de banda necesario está disponible.

Desventajas:

- Falta de escalabilidad, es decir, el costo de la implementación de este modelo crece de forma lineal con la complejidad de la red.
- Protocolo RSVP orientado a conexión, lo que provoca que los routers guarden información de los estados de los flujos activos que pasan por ellos.
- La información de los estados pueden ser aceptados sin problemas en los routers que se encuentran en los extremos de la red, pero resulta muy complicado y costoso soportarlos en los routers del centro de la red, que tienen que soportar miles de conexiones activas.
- En las versiones actuales no tienen implementados mecanismos de seguridad, que eviten robos de servicios ni tampoco políticas de control para autenticar y autorizar a las aplicaciones y a los usuarios.
- No existe un protocolo que sirva para hallar una ruta que soporte la QoS solicitada por el protocolo RSVP.
- Todos los elementos de la red deben mantener e intercambiar mensajes de estado y señalización por cada flujo, lo cual puede requerir de una cantidad de ancho de banda significativo en redes grandes.
- Se utilizan mensajes de refresco periódicos, lo cual puede requerir protección frente a pérdida de paquete para mantener la(s) sesión(es) intactas.

Debido a los aspectos arriba mencionados la IETF creo un nuevo modelo que permitiera evitar los problemas del modelo Int–Serv, este nuevo modelo es Diff–Serv.

3.7 Comentarios Finales

Al analizar los Int–Serv, se puede observar que es una propuesta para ofrecer QoS bastante compleja, ya que en cada router de la trayectoria se tienen que almacenar la información del estado de los flujos de datos, lo que hace que se congestione cada punto de la red, además que en cada uno de ellos debe de tener éste modelo para poder ofrecer QoS, lo que al final hace que las redes que implementan éste modelo pueden llegar a ser muy costosas en comparación con las redes Diff–Serv.

«CAPÍTULO 4»

MODELOS HÍBRIDOS

4.1 Introducción

Para ofrecer una mejor QoS en las redes IP, se han manejado combinaciones de los modelos propuestos por la IETF, ya que haciendo estudios y pruebas del funcionamiento de cada uno de los modelos por separado, se ha encontrado que no cubren las necesidades totales de todos los usuarios, por lo que si se combinan algunos modelos y los hacen trabajar conjuntamente se pueden ofrecer mejores respuestas dependiendo de las necesidades que tenga cada usuario. La combinación de modelos se ha realizado con el objetivo principal de mejorar el proceso del manejo de ancho de banda y de las necesidades en las aplicaciones que tengan los clientes de las redes IP [URL, 23].

En este capítulo se hablará de algunas combinaciones de modelos que han dado buenos resultados en el ofrecimiento de QoS en las redes IP actuales y que se vislumbran como propuestas para mejorar el funcionamiento de las redes.

4.2 Combinaciones de los modelos que ofrecen QoS

Los modelos propuestos por la IETF analizados en los capítulos anteriores no se utilizan de forma absoluta, sino que están diseñados para poder ser utilizados en forma conjunta con otros modelos para poder ofrecer un mejor soporte de QoS extremo a extremo.

La mayoría de estas combinaciones no están todavía estandarizadas pero ya se vislumbran como posibilidades de arquitecturas para soportar QoS; en la Figura

18, se pueden observar como pueden interactuar las diferentes combinaciones en cada nivel de los protocolos TCP/IP [URL, 23].

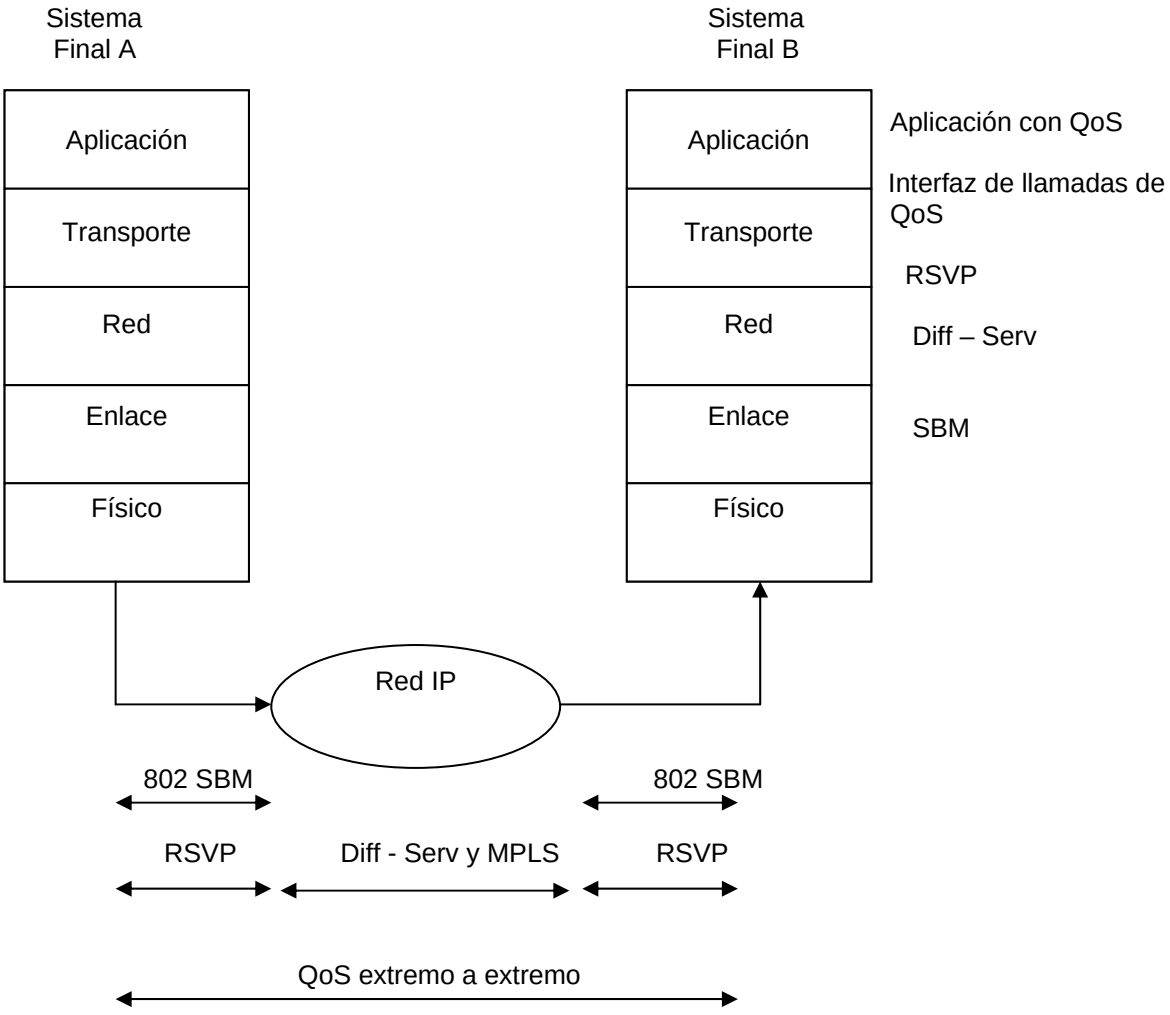

Figura 18. Arquitectura de QoS con diferentes modelos

4.3 Int–Serv con Diff–Serv

Como ya se analizaron las arquitecturas de los Diff–Serv y los Int–Serv en los capítulos 2 y 3, se puede decir que la arquitectura de los Int–Serv es más compleja y exigente, tal que puede ocasionar que se afecte de manera negativa la operación de los routers internos de la red IP, así que para evitar éste problema es mejor utilizar el modelo Diff–Serv en dichos routers.

En los Int–Serv la QoS es manejada por flujos. Un flujo es identificado con una granularidad que permite la distinción de una simple aplicación. Cada flujo es sometido a un apropiado control de admisión y en caso de ser aceptado, se le

permite recibir una cierta QoS especificada por la asociación de clase de servicio. En particular, la clase de servicio garantizado, proporciona un nivel asegurado de ancho de banda y un límite de retardo punto a punto. Por otro lado, la clase de los servicios controlados provee sólo un servicio cualitativo y está pensado para aplicaciones que pueden tolerar una cierta cantidad de retardo. Con esta solución el usuario necesita la capacidad de ofrecer los requerimientos de QoS por flujo a la red; esto está hecho en el control planeado por un protocolo de reservación apropiado. Como una consecuencia del manejo del flujo y la señalización, aumentan los problemas concernientes a la escalabilidad en la arquitectura de los Int–Serv.

La arquitectura de los Diff–Serv explota la agregación de flujo y maneja las agregaciones basándose en la información contenida en el campo DS. Cada paquete perteneciente a una cierta clase de Diff–Serv es marcado de acuerdo a un PHB y tratado de acuerdo a la red Diff–Serv. Actualmente sólo dos tipos de PHB están estandarizados: EF PHB y AF PHB [URL, 23] [URL, 28].

La ínter operación de estos dos modelos parece ser una solución prometedora para proveer QoS punto a punto de una manera escalable. La idea básica es usar el modelo Diff–Serv en el centro de la red y el protocolo RSVP de Int–Serv en el acceso a la red. En este escenario, un papel muy importante es el que juegan los dispositivos interworking, llamados ED (*Edge Devices, Dispositivos de Extremo*), colocados en los bordes entre estos dominios (Figura 19).

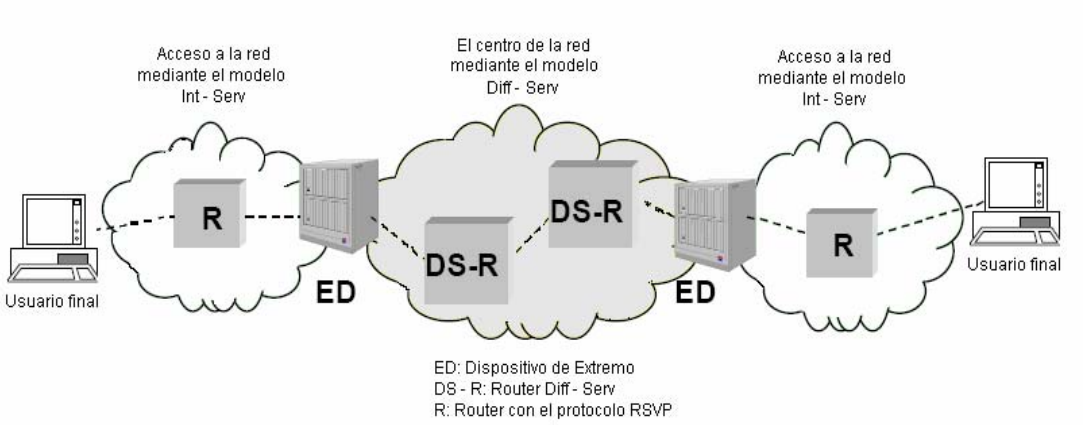

Figura 19. Escenario de ínter operación Int–Serv/Diff–Serv

Como se muestra en la figura anterior, en el ingreso los ED reciben mensajes PATH del protocolo RSVP, desde la fuente RSVP, el ED almacena el estado PATH y envía el mensaje hacia su destino. El router DS–R en el centro de la red simplemente ignora el mensaje RSVP y lo envía transparentemente. Cuando los mensajes RSVP alcanzan la salida del ED, vuelven a ser mensajes RSVP, este es interpretado y enviado hacia su último destino. Debemos notar que el concepto de ingreso o egreso de los ED dependen de la dirección del flujo; un ED puede ser simultáneamente ED de ingreso para un flujo y egreso para otro.

El mismo procedimiento se aplica para los mensajes RESV del protocolo RSVP, los cuales son recibidos por el ED de egreso y enviados hacia el siguiente ED de ingreso. En la recepción del primer mensaje RESV relacionado a un flujo de ingreso ED ejecuta un procedimiento de admisión de control de flujo relacionado a la nube Diff–Serv. Si el procedimiento de admisión tiene éxito, el ED de ingreso envía de regreso el mensaje RESV hacia el host que lo envió [URL, 28].

Por otro lado, en el plano del usuario, el ED de ingreso recibe el paquete IP relacionado a un flujo dado, marca estos paquetes en la clase Diff–Serv apropiada, y los envía en la red Diff–Serv hacia el ED de egreso. Los routers que se encuentran dentro de la red Diff–Serv dirigen los paquetes IP hacia el ED de salida basándose en la planificación Diff–Serv. Finalmente el ED de salida se comporta como un router normal RSVP y maneja los paquetes IP basado por flujo.

En un escenario genérico, el acceso a la red Diff–Serv es operado por un dispositivo llamado Border Router (Router de Extremo) el cual actúa como un agente de control de admisión a la región Diff–Serv. [29]

Para simplificar, se considera que los ED pueden también jugar el papel de routers de frontera (Figura 20).

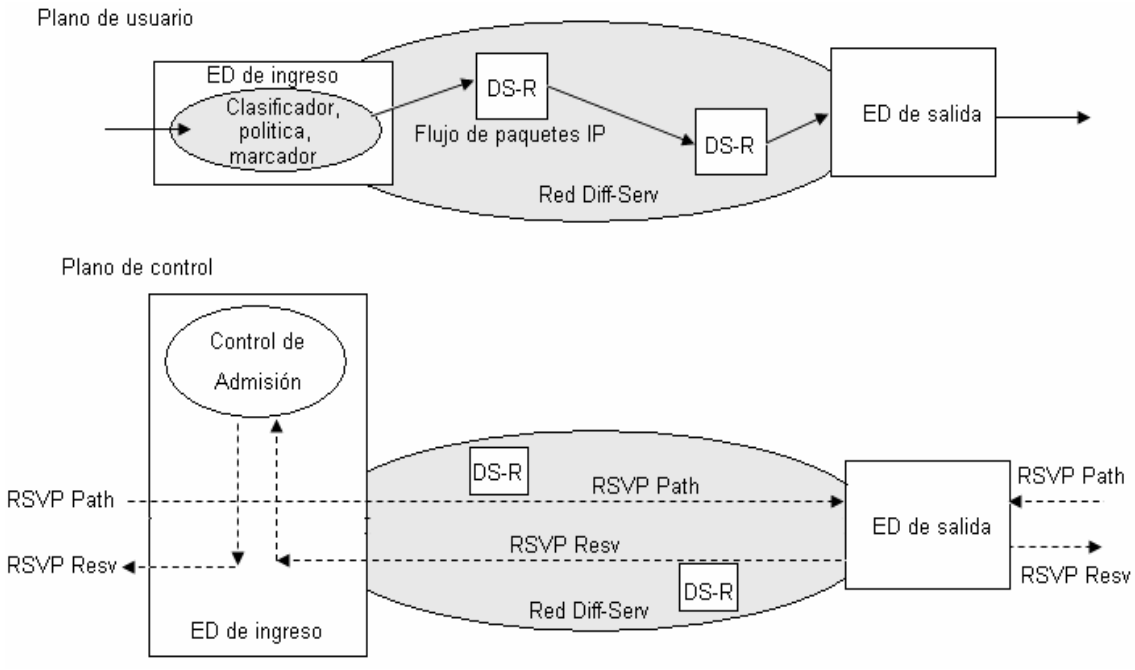

Figura 20. Soporte de RSVP sobre una red Diff–Serv

4.4 MPLS con Int–Serv

Antes de empezar a analizar la combinación de estos dos modelos se describirá brevemente el funcionamiento del modelo MPLS (*Multiprotocol Label Switching, Multiprotocolo de Conmutación de Etiquetas*), para observar la forma de interacción con el modelo Int–Serv analizado en el capítulo anterior.

4.4.1 MPLS

La base de MPLS está en la asignación e intercambio de etiquetas, que permiten el establecimiento de los LSP[3] (*Label Switched Path, Camino Conmutado de Etiquetas*) por la red, los cuales han sido establecidos para un sólo sentido del tráfico en cada punto de entrada de la red. Cada LSP se crea a base de concatenar uno o más saltos en los que se intercambian las etiquetas de modo que cada paquete se envía de un LSR[4] (*Label Swiching Router, Conmutador de Etiquetas*) a otro, a través del dominio MPLS.

[3] LSP = Funciona como un túnel para transporte a través de una red MPLS

[4] LSR = Es un router especializado en el envío de paquetes etiquetados por MPLS

En MPLS la asignación de un paquete a una FEC (*Forwarding Equivalence Classes, Clases Equivalentes de Envío*), puede realizarse no sólo en su dirección IP destino como sucede en las redes IP, sino también en la dirección IP fuente, en el puerto fuente o destino, o en el código Diff–Serv del encabezado IP.

Al momento que un paquete entra a la red, es asignado a una FEC. La FEC a la cual es asignado el paquete es codificada mediante un valor pequeño de longitud fija llamado etiqueta. La etiqueta se envía con el paquete hacia el siguiente nodo [24]. Una vez que el paquete es asignado a una FEC, los routers subsecuentes no analizan el encabezado del paquete, el envío de los paquetes es ahora controlado por las etiquetas.

La trayectoria del paquete a través de la red está determinada por la etiqueta que le asigna el LSR cuando ingresa a la red. Cada LSR en la trayectoria cuenta con una tabla de asignación de etiquetas, esta tabla toma la interfaz de entrada y el valor de la etiqueta de entrada y les asigna una interfaz de salida y el valor de la etiqueta de salida a los paquetes.

Los routers analizan el encabezado IP del paquete no sólo para decidir el siguiente nodo, sino también para determinar sus requerimientos de calidad de servicio. MPLS permite que la prioridad o la clase de servicio del paquete puedan ser determinadas mediante el valor de la etiqueta.

4.4.2 Encabezado MPLS

El encabezado MPLS está formado por 4 octetos y se encuentra entre los encabezados de la capa de enlace de datos y de la capa de red. Es creado por el LSR de ingreso y utilizado por este mismo para determinar la FEC, la cual a su vez es utilizada para crear la etiqueta del paquete. En la Figura 21 se muestran los campos que forman al encabezado MPLS y la descripción de cada uno de ellos [URL, 25].

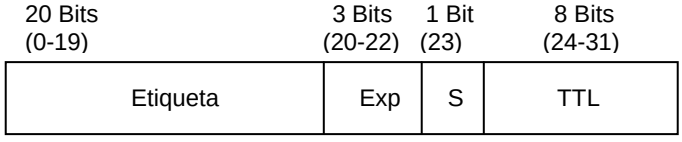

Figura 21. Encabezado MPLS

- **Etiqueta**: Se utiliza para ver la información del siguiente salto del paquete, la etiqueta de salida, la interfaz de salida.

- **Experimental (Exp)**: Reservado para uso experimental, aunque comúnmente es utilizado para ofrecer Calidad de Servicio. Este campo también es conocido como CoS (Class of Service, Clase de Servicio).

- **Pila de Etiquetas (S)**: Informa a los enrutadores el estado de la pila de etiquetas. Cuando tiene el valor de 1, indica que el paquete sólo tiene una etiqueta, en caso que tenga el valor 0, indica que el paquete tiene varias etiquetas.

- **Tiempo de Vida (TTL)**: Define un límite en el número de saltos que puede hacer un paquete MPLS dentro de la red. Reemplaza al campo de límite de saltos del encabezado IP.

4.4.3 Interoperabilidad de MPLS con Int – Serv

Existe el propósito de usar un objeto en RSVP para predeterminar el camino a tomar por parte de las sesiones RSVP con etiquetas. Estas sesiones usan las conexiones establecidas por los routers MLPS. Incluso sin este objeto es posible que MPLS asigne etiquetas con arreglo a las especificaciones de RSVP. En cualquier caso, la consecuencia es una simplificación del funcionamiento de Int–Serv en los routers MPLS.

4.5 MPLS con Diff–Serv

MPLS está diseñado para poder soportar el modelo Diff–Serv, según lo establecido por la IETF. Este modelo define una variedad de mecanismos para poder clasificar el tráfico en un reducido número de clases de servicio, con diferentes prioridades, como se pudo observar en la sección anterior donde se analiza su arquitectura. MPLS se adapta perfectamente a Diff–Serv ya que las etiquetas MPLS tienen el campo EXP para poder propagar la clase de servicio en el correspondiente LSP.

La combinación de Diff–Serv y MPLS resuelve gran parte de los problemas de QoS en las redes IP. Diff–Serv como ya se analizó en el capítulo 2, se apoya del campo DS clasificando los tráficos en diferentes clases en los nodos de ingreso al dominio Diff–Serv. MPLS realiza en cierta forma una clasificación similar, sólo que este modelo los clasifica y agrupa en FEC's para garantizar QoS. Ambos modelos emplean etiquetas, en Diff–Serv son conocidas como DSCP y en MPLS como etiqueta. La etiqueta MPLS determina la ruta que tomará un paquete, esto permite optimizar el ruteo dentro de una red. Además es factible aplicar la Ingeniería de Tráfico, la cual garantiza la asignación de circuitos virtuales con ciertas garantías de ancho de banda para igual número de etiquetas que lo requieran. Por otro lado, el valor DSCP determina el comportamiento de los nodos de acuerdo a esquemas de colas (*Queuing*).

Al emplear esquemas de Calidad de Servicio metropolitano con Diff–Serv y MPLS, es muy importante tener en cuenta un adecuado esquema de colas. El seleccionar erróneamente uno de estos esquemas, podría ser la causa principal de bajo desempeño en las aplicaciones sensibles a retardo [URL, 26]. Como cabría esperar, dada la similitud entre MPLS y Diff–Serv, la traslación del tráfico Diff–Serv a conexiones MPLS es sencilla en las redes IP.

4.5.1 Interoperabilidad de MPLS con Diff – Serv

Para que un proveedor de servicios tenga la capacidad de ofrecer Diff–Serv a través de una red MPLS, es necesario contar con uno o varios mecanismos que aseguren que cada clase de servicio reciba su correspondiente PHB en cada LSR por el cual circule dentro del dominio MPLS. Debido a que el código Diff–Serv se encuentra en el encabezado IP para enviar los paquetes, se necesita de un mecanismo que pueda determinar el PHB apropiado a partir del encabezado MPLS. Con este propósito se definen dos mecanismos [27]:

- **LSP con Clase de Calendarización PHB deducida del campo EXP (E-LSP)**: El PHB del paquete es determinado por el valor del campo EXP del encabezado MPLS. Cada LSR en la trayectoria utiliza el valor de la etiqueta

para determinar el siguiente salto del paquete y el valor del campo EXP para determinar el PHB que debe aplicar al paquete durante el proceso de envío. Los E-LSP's permiten utilizar hasta un máximo de 8 PHB's por FEC, lo cual pone al descubierto una de las limitaciones de los E-LSP, ya que Diff–Serv permite hasta 64 PHB's gracias a que el campo DS cuenta con 6 bits, en tanto el campo EXP sólo tiene un tamaño de 3 bits.

- **LSP con Clase de Calendarización PHB deducida sólo de la etiqueta (L-LSP)**: Utilizan el valor de la etiqueta para decidir el siguiente salto del paquete y el PHB a aplicar a este mismo; la prioridad de descarte es llevada en el campo EXP. En lugar de establecer una sola LSP entre el LSR de ingreso y el LSR de egreso, un LSR de ingreso puede establecer múltiples LSP's entre él y un LSR de egreso, donde cada LSP es configurado con los requerimientos de QoS de una clase de calendarización en particular. En comparación con las trayectorias E-LSP, las trayectorias L-LSP no se encuentran limitadas a 8 PHB's por FEC, si no que pueden establecer un conjunto de L-LSP's que permitan utilizar los 64 PHB's que establece Diff–Serv. Sus desventajas son expuestas al momento de su configuración, la cual es compleja, además de que tener varios L-LSP's complica la administración de la red.

4.6 Comentarios Finales

Como se observó en el presente capítulo, para ofrecer QoS cumpliendo con la mayor eficiencia y eficacia, ha sido necesario el unir los esfuerzos en la combinación de modelos propuestos por la IETF.

Durante los últimos años se han hecho propuestas de modelos híbridos que han llegado a ser buenas opciones. En este capítulo se analizaron sólo tres de ellos, haciendo notar que no son los únicos existentes, pero son los mas conocidos y que han dado mejores resultados.

«CAPÍTULO 5»

MODELOS APLICADOS EN LOS SISTEMAS GNU/LINUX

5.1 Introducción

Actualmente la mayoría de redes IP son montadas en sistemas GNU/Linux en sus diferentes distribuciones, dependiendo de los conocimientos de cada uno de los administradores o de las necesidades que se requieran cubrir; se ha optado por utilizar éste sistema operativo por ser un sistema que ofrece más seguridad [URL, 38]. Para el caso práctico de esta tesis, se ocupará la distribución Fedora Core 5 FC5 con el kernel 2.6.18.3.

Para observar el funcionamiento de los modelos analizados en las secciones anteriores, se mostrará en el presente capítulo la forma en que hay que modificar el Kernel de Fedora Core 5 para obtener QoS, con los modelos estudiados en los capítulos 2 y 3.

5.2 El Kernel de GNU/Linux

Antes de empezar a ver qué partes de la configuración del kernel de Fedora Core 5 se tiene que modificar para ofrecer QoS mediante Int–Serv y Diff–Serv, se definirá lo que es un kernel y cual es su función en el Sistema Operativo GNU/Linux.

5.2.1 Definición del Kernel

El kernel o núcleo de GNU/Linux es el corazón de éste sistema operativo. Es el encargado de que el software y el hardware de una computadora puedan trabajar juntos [URL, 30].

Las funciones más importantes del kernel, son las siguientes:

1. Administración de la memoria, para todos los programas en ejecución.
2. Administración de procesos.
3. Es el encargado de que se pueda acceder a los dispositivos de E/S.

Existen dos versiones del kernel de Linux:

- *Versión de producción*: La versión de producción, es la versión estable hasta el momento. Esta versión es el resultado final de las versiones de desarrollo o experimentales. Cuando el equipo de desarrollo del kernel experimental, decide que ha conseguido un núcleo estable y con la suficiente calidad, se lanza una nueva versión de producción. Esta versión es la que se debe utilizar para un uso normal del sistema, ya que son las versiones consideradas más estables y libres de fallos en el momento de su lanzamiento.
- *Versión de desarrollo*: Esta versión es experimental y es la que utilizan los desarrolladores para programar, comprobar y verificar nuevas características, correcciones, entre otras cosas. Estos núcleos suelen ser inestables.

5.2.2 Tipos de Kernel

El kernel se puede clasificar dependiendo de su tamaño y de las funcionalidades que posea. Existe una tendencia básica a reducir el tamaño del kernel proporcionando menos funcionalidades, que son desplazadas a módulos que se cargan en tiempo de ejecución [URL, 31] [URL, 32].

Por lo anterior existen tres tipos de kernel:

- **Kernel monolítico:** Este tipo de kernel concentra todas las funcionalidades posibles: planificación, sistema de archivos, redes, controladores de

dispositivos, gestión de memoria, etc., dentro de un gran programa. Se trata de un programa de tamaño considerable que se recompila completamente cada vez que se quiera añadir algo nuevo. Esta es la estructura original de GNU/Linux. Todos los componentes funcionales del kernel tienen acceso a todas sus estructuras de datos internas y a sus rutinas. Un error en una rutina puede propagarse a todo el kernel. Todos sus componentes se encuentran integrados en un único programa que se ejecuta en un único espacio de direcciones. En este tipo de kernel, todas las funciones que ofrece el sistema operativo se ejecutan en modo supervisor (Figura 22).

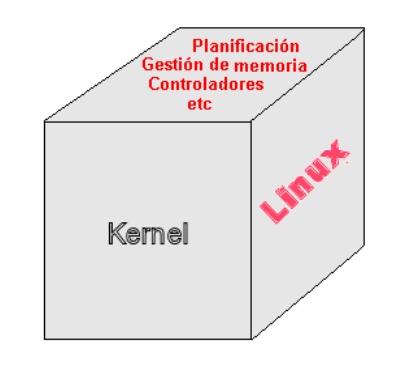

Figura 22. Kernel Monolítico, adaptado de [URL, 31]

- **Kernel modular:** En el kernel se centran las funcionalidades esenciales como la administración de memoria, la planificación de procesos, etc. Este tipo de kernel usa menos memoria y es más rápido además que los módulos pueden ser compilados por separado y añadidos al kernel en tiempo de ejecución (Figura 23).

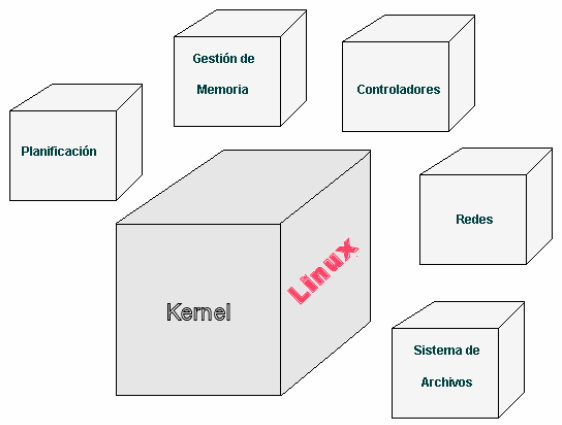

Figura 23. Kernel Modular, adaptado de [URL, 31]

- **Estructura de microkernel:** Provee un conjunto de primitivas o llamadas mínimas al sistema, para implementar servicios básicos como espacios de direcciones, comunicación entre procesos y planificación básica. Los servicios como gestión de memoria, sistema de archivos, operaciones de entrada/salida, etc, que en general son proveídos por el kernel, se ejecutan como procesos servidores en espacio de usuario. Algunos ejemplos de sistemas operativos con microkernel son: Minix, Hurd, NEXTSTEP, L4, NetKernel entre otros (Figura 24).

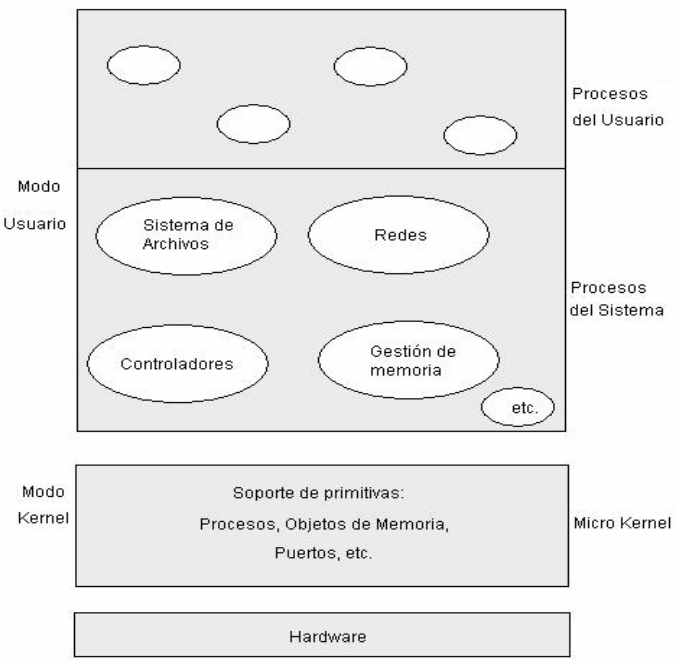

Figura 24. Esquema de un microkernel, adaptado de [URL, 31]

5.2.3 Componentes del Kernel de GNU/Linux

El núcleo de Linux, pese a ser monolítico, está organizado como un conjunto de gestores fuertemente interrelacionados. Cada uno de estos gestores tiene un subdirectorio en el directorio principal de las fuentes de FC 5 [URL, 33].

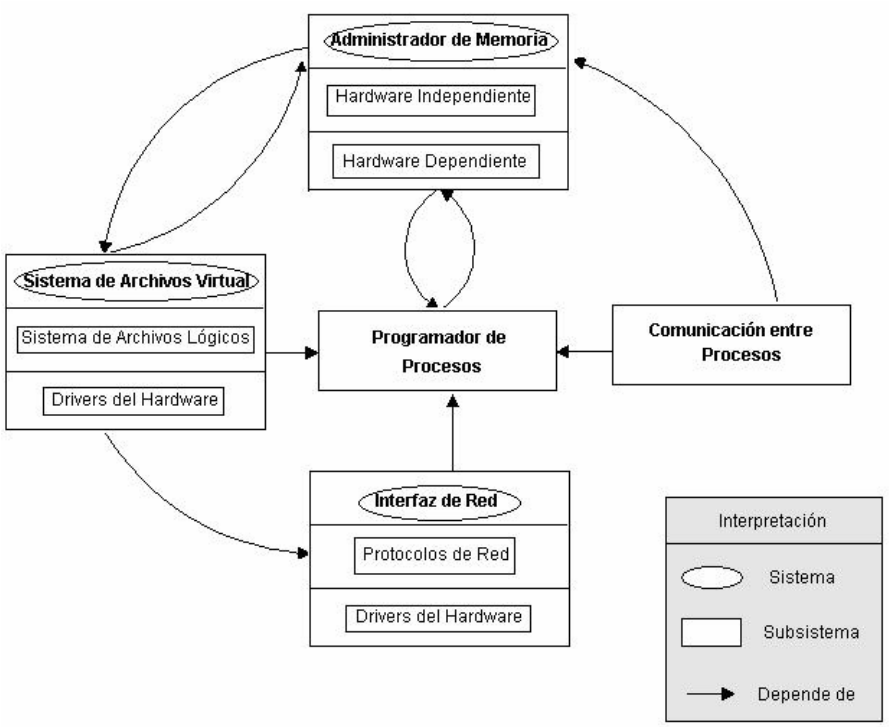

Figura 25. Esquema interno del kernel

El kernel está compuesto por cinco subsistemas (Figura 25):

- **Programador de Procesos***:* Es el corazón del sistema operativo, sus responsabilidades son: permitirle a los proceso crear nuevas copias de sí mismos, determinar qué proceso tendrá acceso a la CPU, recibir interrupciones y desviarlas hacia el subsistema respectivo, enviar mensajes a los procesos de usuario, manejar el reloj de Hardware y liberar recursos cuando un programa los haya desocupado.

- **Administrador de Memoria**: Provee las siguientes capacidades para sus clientes: *gran espacio de direcciones*, es decir, los programas pueden requerir más memoria de la que físicamente existe, *protección*, la memoria asignada a un proceso, es privada y no puede ser leída o modificada por otro proceso, también el administrador de memoria prevé procesos de código de sobre escritura y sólo lectura de datos, *mapeo de memoria*, los clientes pueden mapear un archivo dentro de un área de memoria virtual y acceder al archivo como memoria, *acceso limpio a la memoria física*, el administrador de memoria asegura que los procesos puedan usar transparentemente todos los recursos

de la máquina, asegurando además un rendimiento aceptable, *memoria compartida*, permite que los procesos puedan compartir trozos de la memoria asignada.

- **Sistema de Archivos Virtual**: GNU/Linux puede ínter operar fácilmente con otros sistemas operativos. El sistema de archivos de GNU/Linux tiene los siguientes objetivos: *múltiples dispositivos de hardware*, provee acceso a diferentes dispositivos de hardware, *múltiples sistemas de archivos lógicos*, *múltiples tipos de archivos ejecutables*, tales como .out, ELF, java, entre otros, *homogeneidad*, es decir, una interfaz común entre los sistemas de archivos lógicos y el hardware, *rendimiento*, *seguridad de datos*, no perder o corromper datos, *seguridad de acceso*, restricción a los archivos, cotas, permisos, etc. También se usan dos interfaces para el manejo del Sistema de Archivos. Una interfaz para llamadas de usuarios y otra para los subsistemas del kernel.

- **Interfaz de Red**: El sistema de red de GNU/Linux permite la conectividad entre distintas máquinas de una red y un modelo de conexión vía sockets. Existen dos tipos de sockets, los BSD y los INET. GNU/Linux posee dos protocolos de transporte con distintos tipos de modelo de comunicación y calidad de servicio. Están los mensajes no confiables, basados en el protocolo UDP y los mensajes confiables basados en TCP. Estos dos están implementados sobre el protocolo de transporte y el protocolo IP.

- **Comunicación entre Procesos**: Es un mecanismo en el cual los procesos que se están ejecutando poseen medios para compartir recursos, sincronizarse y compartir datos entre ellos. GNU/Linux implementa todas las formas de Comunicación Inter – Proceso a través de recursos compartidos, estructuras de datos de kernel y colas de espera. Estas formas son a través de, señales, colas de espera, bloqueos de archivos, tuberías *(pipes)* y tuberías renombradas *(named pipes)*, que permiten una transferencia de datos bidireccional y orientada a la conexión entre dos procesos.

El kernel de GNU/Linux ofrece una amplia variedad de funciones de control de tráfico. La Figura 26, muestra de manera general cómo se procesan los datos

que se recibe en la red en el kernel y cómo se generan nuevos datos para ser enviados sobre la red.

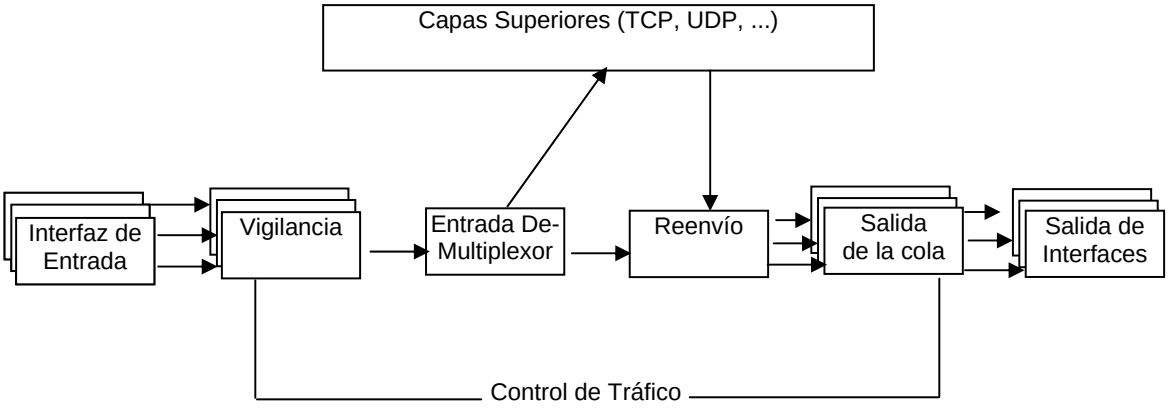

Figura 26. Procesamiento de datos de la red

Los paquetes llegan por una interfaz de entrada, donde pueden ser susceptibles a vigilancia. La vigilancia descarta paquetes indeseados, por ejemplo, si el tráfico está llegando demasiado rápido. Después de la vigilancia, los paquetes son directamente reenviados a la red, o son pasados a capas superiores. Esas capas superiores pueden generar datos por sí solas y entregarlo a capas inferiores para tareas como encapsulamiento, ruteo, y eventualmente transmisión. El reenvío incluye la selección de una interfaz de salida, la selección del próximo salto, encapsulamiento, etc. Una vez que todo esto ha sido realizado, los paquetes son encolados en la interfaz de salida respectiva. Este segundo punto es donde el control de tráfico entra en juego. El control de tráfico puede, entre otras cosas, decidir si los paquetes son encolados o si son descartados.

5.3 Diff–Serv en GNU/Linux

En Linux, se han diseñado módulos que pueden proporcionar QoS mediante los dos modelos estudiados en la presente tesis, dichos módulos a su vez se auxilian de otros y de algoritmos que ayudan a complementar cada una de las arquitecturas. Los módulos que se utilizarán para obtener QoS con los Diff–Serv son: DSMARK (*Differentiated Services Marker, Marcador de Servicios Diferenciales*), GRED (*Generic Random Early Detection, Detección Genérica*

Aleatoria Temprana), HTB (*Hierarchical Token Bucket, Pila de Tokens Jerárquico*) y TC_INDEX (*Traffic Control_Index, Índice de Control de Tráfico*).

Como se mencionó en el capítulo 2, existen dos tipos de PHB importantes al momento de ofrecer QoS con Diff–Serv: AF y EF, en FC 5 estos dos tipos de PHB se llevan a cabo mediante la unión de algunos módulos para realizar el encaminamiento de los paquetes a través de los nodos de la red. Para el AF, se utilizarán los módulos GRED en combinación con DSMARK para repartir el ancho de banda según prioridades entre diferentes tipos de tráfico, para implementar EF se va a usar HTB en combinación con DSMARK. En las siguientes secciones se explicará cómo funciona cada uno de los módulos.

Para comprender mejor cómo es que se lleva a cabo el modelo de Diff–Serv en el kernel de FC5, en la Tabla 10 se muestran los actores identificados en la representación UML que realizan cada una de las partes de la arquitectura, los módulos, el directorio en donde se encuentran éstos, y los archivos que realizan dichas funciones. Finalmente la Figura 27 ilustra los módulos que intervienen y que llevan a cabo cada una de las partes de esta arquitectura.

Tabla 10. Análisis de Diff–Serv en FC5

Actores UML	Módulo en FC 5	Directorio del archivo	Archivo
Clasificador	N/A	-	-
Medidor	N/A	-	-
Marcador	DSMARK	/usr/src/linux2.6.18.3/net/sched	sch_dsmark.c
Acondicionador	DSMARK/HTB y DSMARK/GRED	/usr/src/linux2.6.18.3/net/sched	sch_htb.mod.c sch_gred.mod.c
Descartador	TCINDEX	/usr/src/linux2.6.18.3/net/sched	cls_tcindex.mod.c

Nota: N/A se refiere a que en FC5 si se puede llevar a cabo estas dos partes de la arquitectura, utilizando algunas herramientas que se tienen que configurar e instalar, tales como IPTABLES, IPROUTE2, entre otras cumpliendo así completamente con el modelo Diff–Serv.

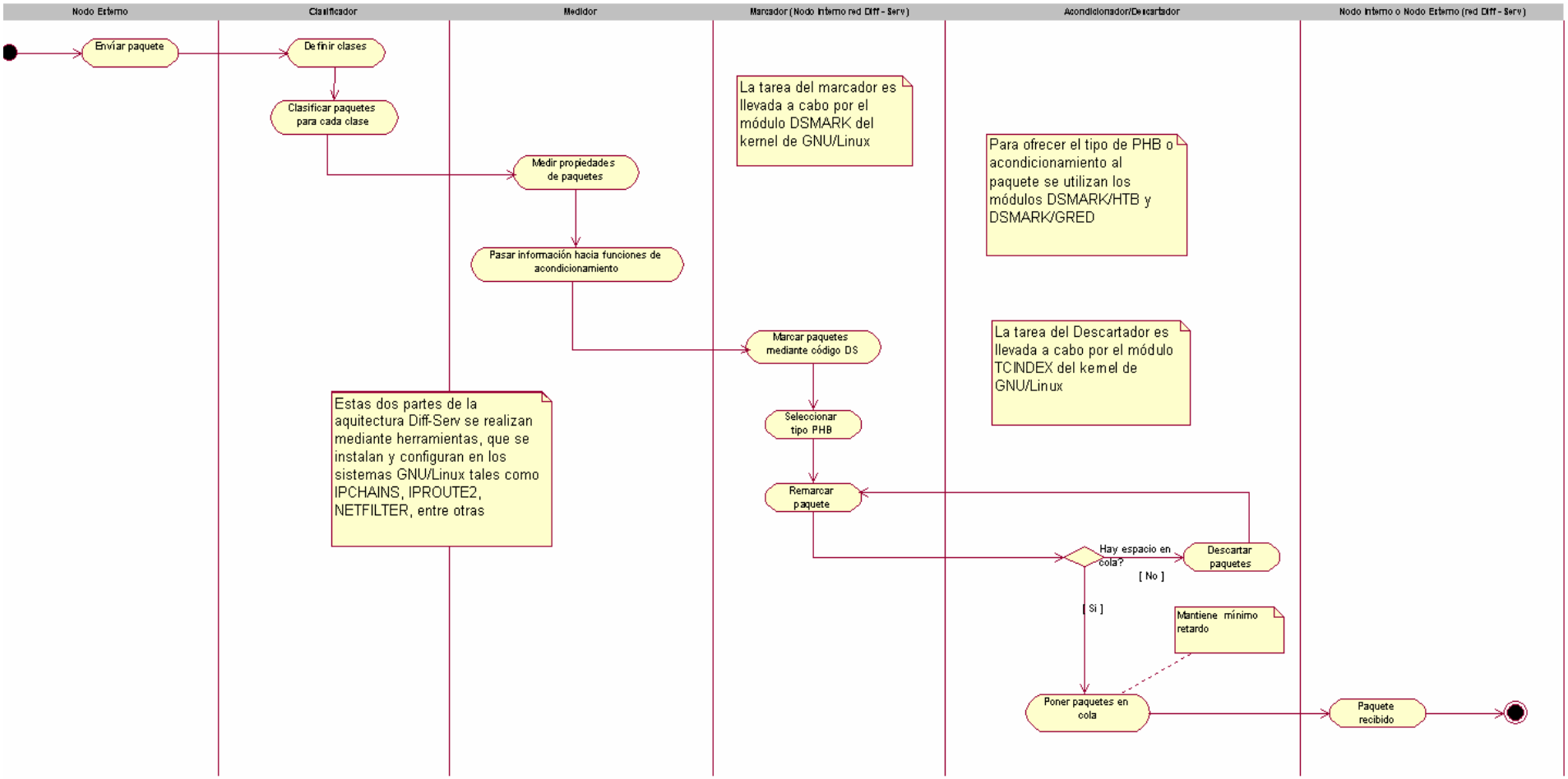

Figura 27. Módulos del kernel GNU/Linux que ofrecen QoS en Diff–Serv

5.3.1 Módulo DSMARK

La disciplina de cola DSMARK fue diseñada especialmente para cumplir completamente con las especificaciones de los Diff–Serv. Se encarga del marcado de paquetes dentro de la implementación de la arquitectura Diff–Serv. Su comportamiento es muy simple cuando se compara con otras disciplinas de cola. Contrariamente a otras disciplinas, DSMARK no hace modelado *(shapping)*, control o vigilancia *(policing)* de tráfico. Tampoco prioriza, retarda, reordena o descarta paquetes. Sólo marca paquetes usando el campo DS.

DSMARK, es una disciplina de cola full class. Las clases son enumeradas del 1 hasta *n-1*, donde *n* es un parámetro que define el tamaño de una tabla interna necesaria para implementar el comportamiento de la disciplina de cola. Este parámetro se denomina índice. Siendo *q* el número de cola, el elemento *q:0* es la cola principal misma. Elementos a partir de *q:1* a *q:n-1* son las clases de la disciplina de cola. Cada una de esas clases puede ser seleccionada utilizando un filtro conectado a la disciplina de cola; los paquetes que están siendo seleccionados por el filtro son puestos en su clase DSMARK respectiva. Los paquetes son marcados en el campo DS, con un valor entero que se define para cada clase cuando se configura la disciplina de cola. Los paquetes son marcados justo antes de dejar la disciplina de cola para ser colocados en la interfaz de salida (Figura 28) [URL, 34].

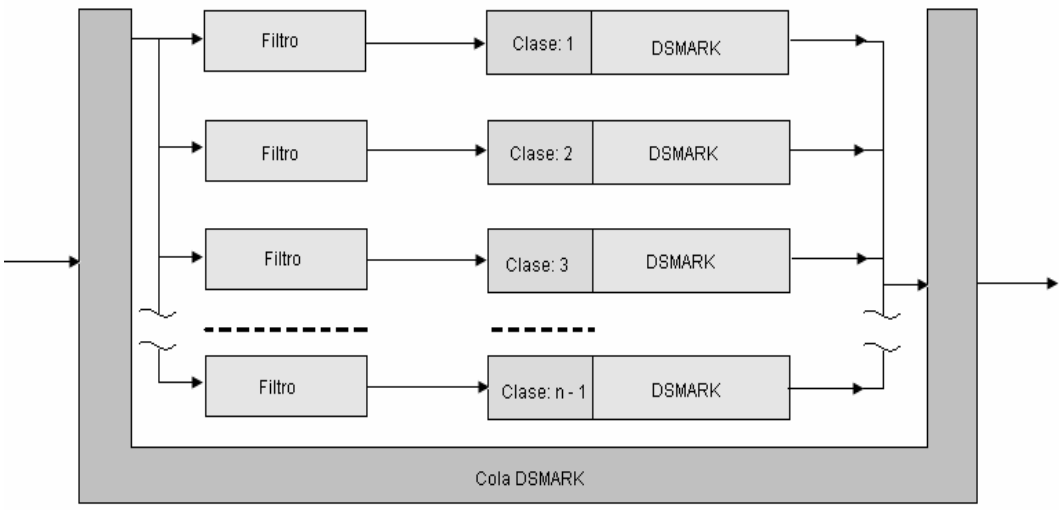

Figura 28. Disciplina de colas DSMARK

5.3.2 Módulo HTB

HTB tiene una funcionalidad muy similar a la de CBQ *(Class Based Queing – Cola Basada en Clases)*, aunque su implementación es completamente distinta y su configuración mucho más sencilla. HTB surge como una disciplina más entendible, intuitiva y más rápida para reemplazar a CBQ en Linux. HTB permite controlar el uso del ancho de banda saliente sobre un enlace dado. Permite que se utilice un sólo enlace físico para simular varios enlaces más pequeños y para enviar diferentes tipos de tráfico sobre diferentes enlaces simulados. Se necesita especificar cómo dividir el enlace físico en enlaces simulados y cómo decidir qué enlace simulado tiene que utilizar un paquete dado para ser enviado.

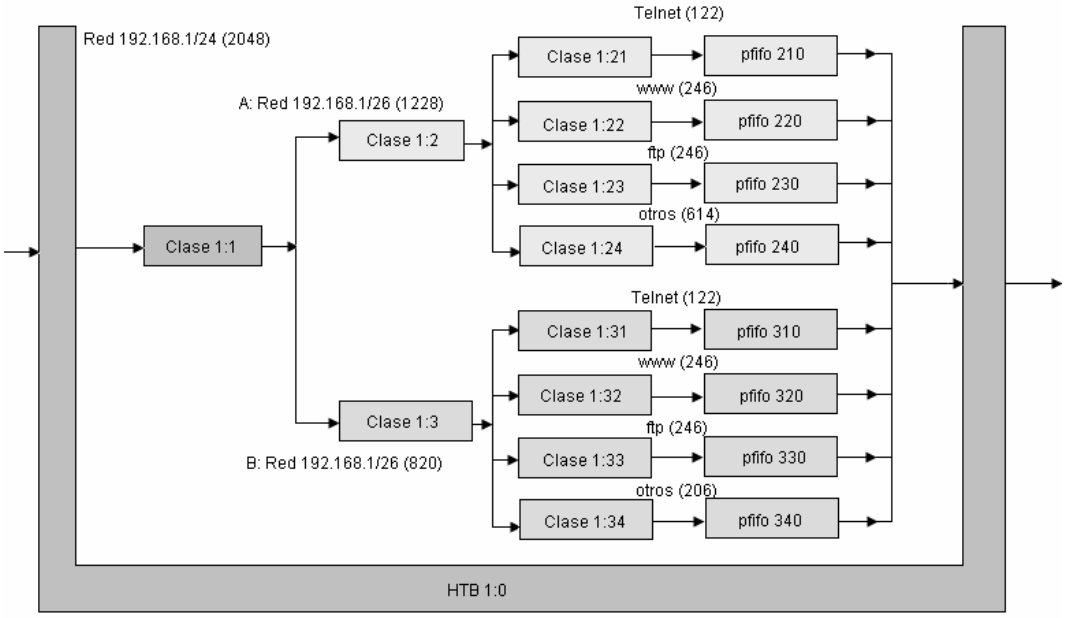

Figura 29. Disciplina de colas HTB

HTB es una disciplina de colas de compartición jerárquica del enlace. La jerarquía puede tener varios niveles. Puede haber varias clases en cada cola que pueden tener a su vez asociadas colas con varias clases (Figura 29). A cada clase se le garantiza un mínimo nominal de ancho de banda, de forma que en cada momento se le concede a cada clase el mínimo de entre el nominal y el solicitado. Si algunas clases no utilizan todo su ancho de banda, las otras lo pueden tomar prestado para tener más de lo que les corresponde mientras esas clases no lo requieran. El orden de prioridad para optar a esos recursos prestados también está establecido. De esta forma, cada clase de tráfico puede aprovechar el ancho de banda disponible en la medida en que no lo requiera un tráfico más prioritario, y siempre tiene garantizado un mínimo en cualquier caso. La principal ventaja de ésta disciplina es que consigue una gran calidad a la hora de conformar

el tráfico en el caso de tener un ancho de banda variante. La principal desventaja que tiene es que obliga a definir un ancho de banda total para cada enlace, lo que en el caso de emplear una red inalámbrica resulta un gran inconveniente, puesto que dicho parámetro se ve modificado de forma casi impredecible a lo largo del tiempo. Dado que el parámetro ancho de banda total del enlace sirve para definir a su vez, el ancho de banda otorgado a cada clase, no es posible a priori, determinar el comportamiento de HTB, puesto que el ancho de banda que hayamos otorgado a un determinado servicio, puede tornarse insuficiente o demasiado alto al bajar las prestaciones en su conjunto [URL, 35].

5.3.3 Módulo GRED

Para proporcionar AF se necesita una disciplina de colas que soporte múltiples prioridades de descarte, por esto, se diseñó la disciplina GRED que es una generalización de la disciplina RED *(Random Early Detection – Detección Aleatoria Temprana)* (Figura 30).

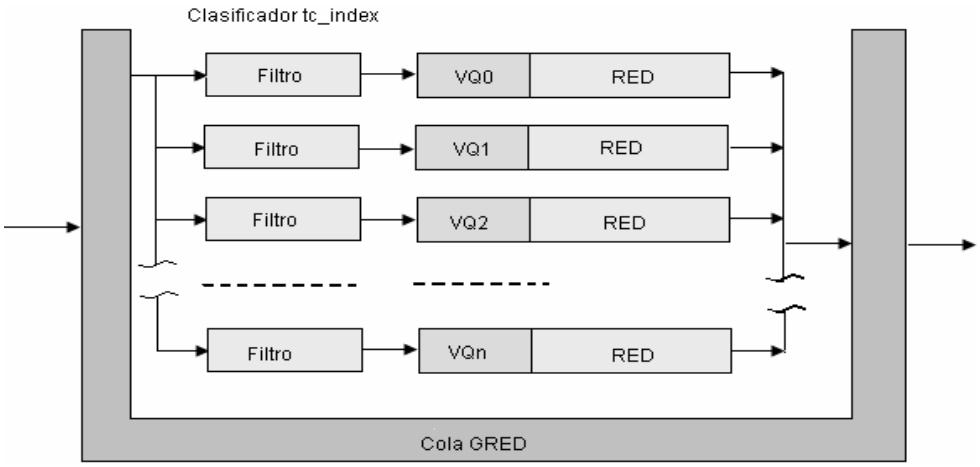

Figura 30. Disciplina de colas GRED

Para Diff–Serv no se usa la disciplina RED, en lugar de eso se toman las ventajas de los mecanismos de descarte probabilística para implementar las clases AF. Cada una de estas clases necesita soportar una precedencia de tres niveles de descarte, siendo ésta la diferencia entre los niveles de probabilidad de descarte. RED sólo ofrece una cola y una probabilidad de descarte, en comparación la disciplina GRED ofrece diez y seis niveles.

La disciplina GRED funciona de forma coordinada con la disciplina DSMARK. Al usar el algoritmo GRED se definen diversas colas virtuales RED con sus correspondientes curvas de probabilidad de servicio de descarte. La selección de la cola virtual adecuada para cada paquete se realiza en función de los 4 bits menos significativos del campo skb->tc_index de la cabecera interna del paquete. Dicho campo debe haber sido inicializado por la disciplina de servicio DSMARK.

Esta disciplina funciona como una clase de cola full class. Cada clase tiene su propia cola virtual. También tiene un tipo de filtro para seleccionar la cola que le corresponde a cada paquete. Los diseñadores extendieron el buffer de paquetes en GNU/Linux para agregar un nuevo campo llamado tc_index. El buffer de paquetes es accedido usando un indicador llamado skb. Para acceder al campo se usa skb->tc_index. El nuevo campo usa los cuatro bits menos significativos para seleccionar las colas virtuales (VQ). Cuando se utiliza la disciplina GRED, los cuatro bits menos significativos del campo tc_index están ordenados adecuadamente como corresponde para sus valores (0, 1, 2...15), uno de los diez y seis valores de VQs es seleccionado para recibir el paquete. Es importante aclarar que el campo tc_index no está en la cabecera del paquete, pero si en la memoria del buffer asignado para la administración del paquete dentro del proceso del control de tráfico.

Para instalar GRED, en primer lugar, se debe realizar una configuración general de la disciplina invocando a tc de la siguiente manera:

tc qdisc add dev eth0 root gred ***setup DPs <VQs>*** *default <VQ> [grio]*

Con el comando anterior se configura el demonio GRED como una cola de root de interfaz ethernet 0. <VQs> es el número de colas virtuales que se necesitarán (pueden configurarse desde 1 hasta 16 colas virtuales RED). El parámetro default establece la cola en la cual deben ubicarse los paquetes cuyos 4 bits menos significativos del campo skb->tc_index indiquen una cola inválida o bien, si se activa la opción grio (parámetro opcional), se puede asignar una prioridad a cada VQ. El rango de prioridades va de 1 a 16 siendo 1 el rango mayor. Cuando se indica la opción *grio*, para el cómputo de la ocupación de la cola, al número de paquetes almacenados en dicha cola, se le suman los paquetes almacenados en la colas de menor prioridad de esta forma, si se definen las curvas de probabilidad

de descarte de forma idéntica para todas las colas, aquéllas con mayor prioridad presentarán menores pérdidas [URL, 39].

En segundo lugar, se debe configurar cada una de las colas virtuales mediante el comando tc change:

tc qdisc change gred limit <bytes> min <bytes> \ max <bytes> burst <packets> avpkt <bytes> bandwidth <kbps>\ DP <vq> probability <number> [prio <value>]

Donde DP define el identificador de la cola virtual que se pretende configurar. Los demás valores si no se especifican los toma por default como si estuviera trabajando con la disciplina de colas RED.

5.3.4 Filtro TC_INDEX

El filtro TC_INDEX está ligado con el módulo DSMARK y es un clasificador especialmente diseñado para implementar la arquitectura Diff–Serv en GNU/Linux. El clasificador TC_INDEX basa su comportamiento en el campo skb->tc_index situado en la estructura de buffer sk_buff. Este espacio de buffer es creado para cada paquete entrante. Debido a esto, el clasificador TC_INDEX debe ser usado solo con disciplinas de cola que puedan reconocer y fijar el campo skb->tc_index, estas disciplinas de colas son GRED, DSMARK e INGRESS [URL,37].

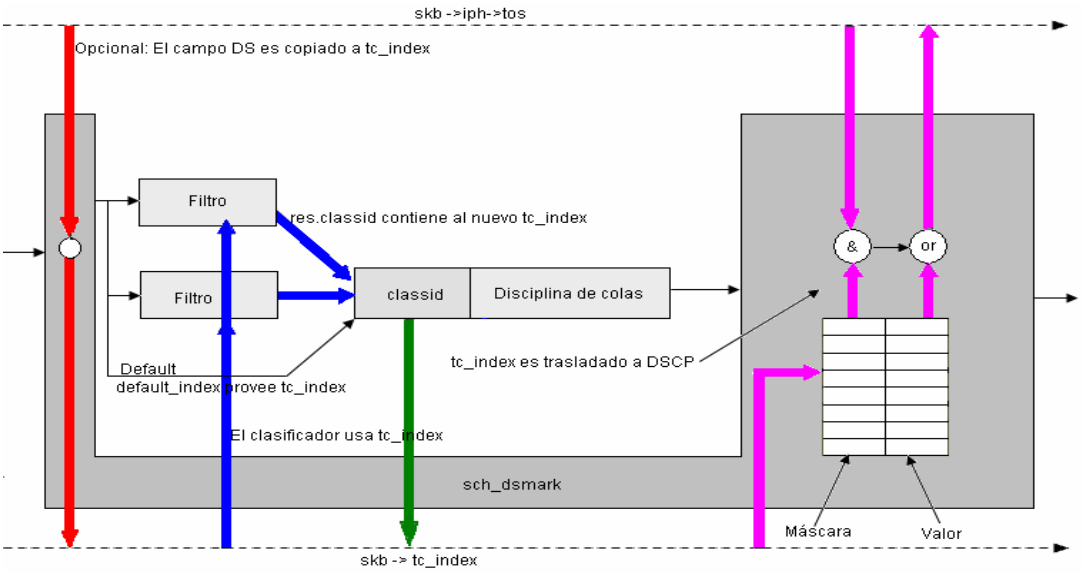

Figura 31. Funcionamiento del filtro TC_INDEX

El valor skb->tc_index es fijado por la disciplina de cola DSMARK cuando se fija el parámetro set_tc_index. El skb->iph->tos, contiene el valor del campo DS de los paquetes y es copiado en el campo skb->tc_index. En la Figura 31 éste proceso se representa por la línea roja que va de arriba hacia abajo en la entrada DSMARK.

El valor del campo skb->tc_index es leído por el clasificador tcindex, el filtro aplica una operación binaria sobre una copia del valor del campo skb->tc_index (el valor original no se modifica); el valor final obtenido de la operación se pasa a los elementos del filtro para obtener un valor compatible. Teniendo un valor compatible, el identificador de clase correspondiente para éste elemento es regresado y pasado a la disciplina de cola como el resultado del identificador de clase. En la Figura 31, este proceso está representado por la línea azul vertical que va de abajo hacia los elementos del filtro y del filtro hacia el identificador de clase.

Los 4 bits menos significativos del identificador de clase regresa a la disciplina de cola DSMARK por el clasificador tcindex, éste es copiado y regresado por la disciplina de colas en el campo skb->tc_index. En la Figura 31 éste proceso está representado por la línea vertical de color verde que va del rectángulo classid al campo skb->tc_index.

En la disciplina de cola DSMARK el valor de skb->tc_index es usado como un índice para la tabla interna de pares valor-máscara para obtener el par que será usado. El par seleccionado es usado cuando hay una salida en la cola, para modificar el valor del campo DS de los paquetes se usa una combinación de operaciones binarias and/or. En la Figura 31 éste proceso entero es representado por las flechas color púrpura y la representación de la tabla DSMARK.

5.4 Int–Serv en GNU/Linux

Para llevar a cabo la QoS mediante los Int–Serv, en el kernel de FC 5 se activarán los módulos IPv4 RESOURCE RESERVATION PROTOCOL RSVP (*IPv4 Protocolo de Reservación de Recursos RSVP*) e IPv6 RESOURCE RESERVATION PROTOCOL RSVP6 (*IPv6 Protocolo de Reservación de Recursos RSVP6*), estos dos módulos ocupan el archivo cls_rsvp.h, el cual se encarga de proporcionar QoS mediante Int–Serv.

Para comprender mejor la manera en cómo funciona esta arquitectura en el kernel de GNU/Linux, en la Tabla 11 se muestra un resumen de los actores identificados en la representación UML, el módulo que se encarga de realizar las tareas para llevar a cabo Int–Serv en el kernel y el directorio en donde se encuentran los archivos que trabajan para ofrecer QoS. En la Figura 32 se muestra el módulo que se identificó y las funciones de las que se apoya para llevar a cabo la arquitectura Int–Serv.

Tabla 11. Análisis de Int–Serv en FC5

Actores UML	Módulo FC5	Directorio del archivo	Archivo
Control de tráfico	Módulo cls_rsvp	/usr/src/linux2.6.18.3/net/sched	cls_rsvp.h cls_rsvp.c
Control de Admisión	Módulo cls_rsvp	/usr/src/linux2.6.18.3/net/sched	cls_rsvp.h cls_rsvp.c
Política de control	Módulo cls_rsvp	/usr/src/linux2.6.18.3/net/sched	cls_rsvp.h cls_rsvp.c
Clasificador de paquetes	Módulo cls_rsvp	/usr/src/linux2.6.18.3/net/sched	cls_rsvp.h cls_rsvp.c
Planificador de paquetes	Módulo cls_rsvp	/usr/src/linux2.6.18.3/net/sched	cls_rsvp.h cls_rsvp.c

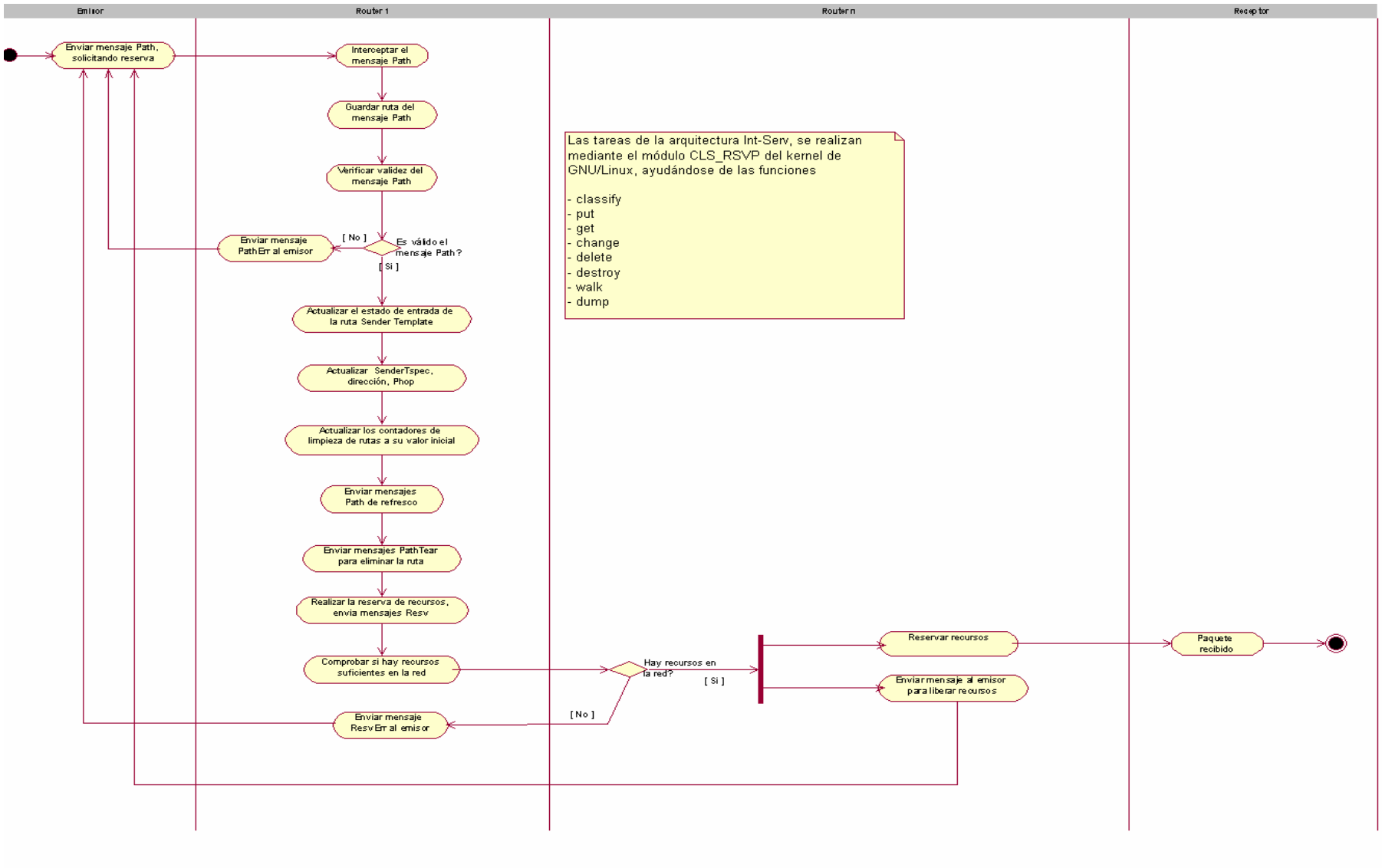

Figura 32. Módulo del kernel GNU/Linux que ofrece QoS en Int–Serv

5.4.1 Módulo CLS_RSVP

El archivo que lleva a cabo todo el trabajo de la arquitectura Int - Serv es llamado CLS_RSVP, este módulo soporta la arquitectura Int – Serv. Este permite que los sistemas finales obtengan una tasa mínima o máxima de flujo de datos para una conexión. Esto es importante para datos en tiempo real como son audio y video. CLS_RSVP funciona como un filtro que es controlado vía funciones definidas en la estructura tcf_proto_ops, los modulos (Ver Anexo I. Estructuras para QoS en FC5) mediante los cuales se lleva a cabo la arquitectura Int–Serv son [URL, 39]:

classify: Realiza la clasificación y devuelve uno de los valores de *sk_buff*. También devuelve la ID de la clase seleccionada. Si la ID interna de la clase no se omite, se devuelve y se almacena en la estructura *tcf_result*; si no el valor cero se almacena en *res.*

get: Busca un elemento filtrado y regresa el ID del filtro interno o el número menor (cero significa una qdisc).

put: Es invocado cuando un elemento filtrado previamente hace referencia con *get.*

change: Configura un nuevo filtro o cambia un filtro existente. Registra la adición de un nuevo filtro o un elemento filtrado a una clase llamando a *tcf_bind.*

delete: Remueve un elemento filtrado.

destroy: Remueve un filtro entero.

walk: Pasa por todos los elementos de un filtro e invoca una función de rellamada automática para cada uno de ellos.

dump: Retorna el diagnóstico del dato para un filtro o para los elementos del filtro.

5.5 Comentarios Finales

Como se pudo observar en éste capítulo, para llevar a cabo QoS mediante los modelos estudiados en la presente tesis utilizando como herramienta la distribución de Linux, FC 5, fue necesario identificar los módulos que se encargan de llevar ésta labor, mismos que han sido plasmados en este capítulo para su mejor entendimiento y comprensión, dejando en claro también que no es la única forma de llevar a cabo QoS en los sistemas GNU/Linux.

«CAPÍTULO 6»

CONCLUSIONES Y TRABAJO A FUTURO

6.1 Conclusiones

En conclusión se puede decir que se cumplió con los objetivos planteados al inicio de la tesis, el objetivo general de ayudar a satisfacer la demanda de QoS en una red, por medio de la identificación de los módulos que proveen QoS en GNU/Linux para los dos modelos que se estudiaron y analizaron en la tesis, utilizando como herramienta de estudio la distribución de GNU/Linux, FC 5 con el kernel 2.6.18.3.

Asimismo los objetivos particulares:

- Investigar y documentar los modelos de Diff–Serv e Int–Serv

Para llevar a cabo este punto, se analizó la información de cada uno de los modelos y se plasmó en la presente tesis lo más importante y relevante de las arquitecturas estudiadas, con el objetivo de que los lectores comprendan el funcionamiento de éstos modelos.

- Realizar una propuesta de representación con UML para los Diff–Serv
- Realizar una propuesta de representación con UML para los Int–Serv

Para lograr el éxito de los dos puntos anteriores, que es el de proponer una representación UML para los Int–Serv y Diff–Serv, se ocupó la herramienta Rational Rose Enterprise, con la cual se desarrollaron las representaciones que se presentan en los capítulos 2 y 3 de la presente tesis y que muestran de una

manera abstracta el modo de operación de éstas dos arquitecturas, de esta forma se busca una mejor comprensión del funcionamiento de éstos modelos, sin que con ello se haya pretendido representar en UML su implementación, sino más bien su lógica.

- Investigar y documentar los modelos híbridos

En la actualidad, por las crecientes necesidades de servicios en tiempo real de los usuarios de las redes IP, se ha buscado una mejor forma de llevar a cabo la QoS y así mantener las redes en óptimas condiciones, es por tal motivo que se han propuesto los llamados modelos híbridos, de los cuales en la presente tesis se presentaron los que están teniendo más auge en la actualidad y los que se vislumbran como respuestas a las necesidades de los usuarios finales.

- Identificar los mecanismos que se utilizan para brindar QoS en los sistemas Linux a nivel kernel
- Aplicar la representación propuesta con UML para saber cómo se lleva a cabo en el kernel de Linux

Para cumplir con los dos últimos objetivos particulares se analizó el kernel de FC 5 para ver de qué manera lleva a cabo la QoS mediante los modelos estudiados, logrando identificar los módulos descritos en las secciones 5.3 y 5.4, y considerando la forma en que trabajan se puede decir que FC 5 satisface en un ochenta por ciento (tomando en cuenta el número de actividades que cumplen en cada uno de los modelos) las arquitecturas de los dos modelos estudiados, ya que a pesar de contar con los módulos descritos anteriormente y comparándolos con las representaciones UML, no cubren todas las partes de las dos arquitecturas, por lo cual, se puede observar también que ésta no es la única forma de ofrecer QoS mediante Int–Serv y Diff–Serv, ya que se pueden explotar algunas herramientas para cubrir todas las necesidades de los usuarios de las redes IP, pero ese será otro trabajo de investigación.
Cabe mencionar que la representación UML propuesta fue de gran ayuda para la identificación de los módulos en GNU/Linux, ya que sin contar con ellas hubiera sido un poco más complicada.

También para llevar a cabo este objetivo, durante el desarrollo de la tesis se estuvo en comunicación vía e-mail con uno de los desarrolladores[5] de los módulos del kernel que ayudan a proveer QoS, obteniendo como respuesta que los Int–Serv se han quedando obsoletos y que la tendencia para que sigan en auge es que se combinen con otros modelos propuestos por la IETF.

6.2 Trabajo a Futuro

Como se pudo observar, al utilizar sólo los módulos del kernel de FC 5, no se cubre en un cien por ciento todas las características de las arquitecturas estudiadas en la presente tesis, por lo que se propone como trabajo a futuro estudiar, analizar y explotar, las herramientas que existen para llevar a cabo la QoS mediante Diff–Serv e Int–Serv para GNU/Linux, tales como las herramientas IPCHAINS, IPROUTE 2, NETFILTER, IPTABLES, entre otras, con el objetivo de proporcionar QoS a los usuarios finales de una red IP.

Asimismo otra línea de investigación que se puede tomar para un nuevo trabajo de tesis, es el estudio y análisis del modelo Int–Serv combinado con otro de los modelos establecidos por la IETF para ofrecer QoS, ya que por sí sólo este modelo se ha visto que ha quedado prácticamente obsoleto.

Otra investigación que se puede realizar es estudiar, analizar y proponer una representación UML para los modelos híbridos que se trataron en la presente tesis.

Uno más seria analizar cómo funcionan los modelos híbridos en sistemas GNU/Linux.

[5] Werner Almesberger - werner@almesberger.net

«GLOSARIO»

AF: Assured Forwarding – Encaminamiento Asegurado

BA: Behavior Abregate Classifier – Clasificador de Comportamiento Agregado

BE: Best Effort – Mejor Esfuerzo

CBQ: Class based Queing – Disciplina de Colas Basadas en Clases

COPS: Common Open Policy Service – Política de Servicios Comúnmente Abiertos

Diff–Serv: Differentiated Services – Servicios Diferenciales

DS: Diferencial Service – Servicio Diferencial

DSCP: Differentiated Services Code Point – Punto de Codificación de servicios Diferenciados

DSMARK: Differential Service Marker – Marcador de Servicios Diferenciales. Módulo del Kernel de FC5 para llevar a cabo QoS mdiante Diff–Serv

ED: Edge Devices – Dispositivos de Extremo

EF: Expedited Forwarding – Encaminamiento Expedito

FEC: Forwarding Equivalente Classes – Clases Equivalentes de Envío

FC5: Fedora Core 5 – Distribución de Linux, utilizada en la presente tesis

GRED: Generic Random Early Detection – Detección Genérica Aleatoria Temprana. Módulo que se utiliza en combinación con DSMARK para ofrecer QoS, mediante Diff–Serv

HTB: Hierarchical Token Bucket – Pila de Tokens Jerárquico. Módulo que en combinación con DSMARK ofrece QoS.

IETF: Internet Engineering Task Force – Grupo de trabajo sobre Ingeniería de Internet

Int–Serv: Integrated Services – Servicios Integrales

ISP: Internet Service Provider – Proveedor de Servicios de Internet

LSP: Label Switched Path – Camino Conmutado de Etiquetas

LSR: Label Swiching Router – conmutador de Etiquetas

MF: Multi - Field Classifier – Clasificador Multi Campo

MPLS: Multiprotocol Label Switching – Protocolo de Conmutación de Etiquetas

PHB: Per Hop Behavior – Comportamiento por Salto

QDISC: Queuing Discipline – Disciplina de colas

QoS: Quality of Service – Calidad de Servicio

RED: Random Early Detection – Disciplina de colas de Detección Aleatoria Temprana

RSVP: Resource reSerVation Protocol – Protocolo de Reservación de Recursos

SBM: Subset bandwidth Management – Administración del Ancho de Banda de la Subred

SLA: Service Level Agreement – Acuerdos de Nivel de Servicio

SPQ: Strictic Priority Queing – Encolamiento de Prioridad Estricta

Tc: Traffic Conformation – Conformación de Tráfico

TC: Traffic Control – Control de Tráfico

TCA: Traffic Conditioning Agreement – Acuerdo de Acondicionamiento de Tráfico

TC_INDEX: Traffic Control Index – Índice de Control de Tráfico, es un filtro para el control de Tráfico para el modelo Diff–Serv

ToS: Type of Service – Tipo de Servicio

UML: Unified Modeling Language – Lenguaje Unificado de Modelado

VQ: Virtual Queing – Colas Virtuales

VSAT: Very Small Aperture Terminal – Terminal de Apertura muy Pequeña

WFQ: Weighted Fair Queing – Encolamiento Ponderado

WRR: Weighted Roung Robin – Round Robin por Peso

«REFERENCIAS»

[1] M. León. *Quality of Service of the Internet Protocols*. International Symposium on Advanced Distributed System. ISADS '02. Noviembre 2002.

[2] G. Gerónimo, E. Rocha. *Análisis de la Calidad de Servicio por Medio del Modelo Diferencial*. XXX Aniversario FCC – BUAP. Diciembre 2003. ISBN 968 863 711 4. Pags. 256-260.

[3] M. León. *Modelo UML del Modelo de Servicios Integrados de Internet*. XII Congreso Internacional de Computación. Octubre 2003. ISBN 970-36-0099-9.

[URL, 4] *Calidad de Servicios en Redes QoS*, http://dis.eafit.edu.co/cursos/st780/material/bdigital/articulos/2003/QoS.pdf, última fecha de acceso: Junio 2006

[URL, 5] *Diff–Serv Provides Quality of Service*, http://www.nwfusion.com/news/tech/2002/1104techupdate.html, última fecha de acceso: Junio 2006

[URL, 6] *Diff Serv: Servicios Diferenciados*, *Monografía de Evaluación de Performance en Redes de Telecomunicaciones*, http://iie.fing.edu.uy/ense/asign/perfredes/trabajos/trabajos_2003/diffserv/Trabajo%20Final.pdf, última fecha de acceso: Junio 2006

[7] Brim S., *Per hop Behavior Identification Codes*, RFC 2836, Mayo 2000

[8] Heinanen J., *Assured Forwarding PHB Group*, RFC 2597, Junio 1999

[URL, 9] *Quality of Service*, http://www.rhyshaden.com/qos.htm, última fecha de acceso: 17 de junio de 2006

[URL, 10] *Diff–Serv – The scalable End to End QoS Model*, http://www.cisco.com/en/US/tech/tk543/tk766/technologies_white_paper09186a00800a3e2f.shtml, última fecha de acceso: Junio 2006

[URL, 11] *Diff–Serv Como solución a la provisión de QoS en Internet*, http://www.ist-mobydick.org/publications/cita2002.pdf, última fecha de acceso: Junio 2006

[URL, 12] *Calidad de servicio en redes de servicios diferenciados*, http://www.tid.es/documentos/revista_comunicaciones_i+d/numero24.pdf, última fecha de acceso: Julio 2006

[13] Wang Z., *Internet QoS Architectures and Mechanisms for Quality of Services*, Morgan Kaufmann Publishers, USA, 2001

[14] R. Braden, *Integrated Services in the Internet Architecture: an Overview*, RFC 1633, Junio 1994.

[URL, 15] *Calidad de Servicio (QoS)*, http://www.uv.es/~montanan/ampliacion, última fecha de acceso: agosto de 2006.

[16] J. Wroclawski, *Specification of the Controlled-Load Network Element Service*, RFC 2211, September 1997.

[17] S. Shenker, *Specification of Guaranteed Quality of Service*, RFC 2212, September 1997.

[18] B. Braden, *Resourse ResServation Protocol (RSVP) – Version 1 Functional Specification*, RFC 2205, September 1997

[URL, 19] *Voip: Una Visión General* http://www.ravel.ufrj.br/arquivosPublicacoes/nelson_voip.pdf, última fecha de acceso: agosto de 2006.

[URL, 20] *Señalización para QoS en redes IP* http://www.ahciet.net/comun/portales/1000/10002/10007/10300/docs/qos.pdf, última fecha de acceso: agosto de 2006.

[URL, 21] *Análisis y propuesta de un esquema de Calidad de Servicio (Qos) para la red de la Universidad de Colima* http://digeset.ucol.mx/tesis_posgrado/Pdf/Arturo%20Torres%20Gutierrez.pdf, última fecha de acceso: agosto de 2006.

[URL, 22] *Controlador de Ancho de Banda* http://www.um.edu.ar/nuke6/imagenes-contenido/UM-MTI-NavarroD.pdf, última fecha de acceso: agosto de 2006.

[URL, 23] Propuesta de optimización de la interconexión de redes con calidad de servicio para aplicaciones multimedia, http://it.aut.uah.es/josema/publicaciones/tesis.pdf, última fecha de acceso: 07 de marzo de 2006.

[24] E. Rosen, Multiprotocol Label Switching Architecture, RFC 3031, January 2001.

[URL, 25] Arquitectura de MPLS, http://www.fi.upm.es/~jgarcia/Curso_MPLS/capitulo4.html, última fecha de acceso: marzo de 2006.

[URL, 26] Retos de MPLS y DiffServ en Redes Metropolitanas Ethernet,

http://www.cujae.edu.cu/eventos/citel2004/Trabajos/Trabajos/CIT009.pdf, última fecha de acceso: marzo de 2006.
[27] F. Le Faucheur, RFC 3270: Multiprotocol Label Switching (MPLS) Support of Differentiated Services, May 2002.
[URL, 28] Combining IntServ and DiffServ under Linux, http://citeseer.ist.psu.edu/cache/papers/cs/26483/http:zSzzSzlcawww.epfl.chzSzPublicationszSzGiordanozSzGiordanoAMSS00.pdf/combining-intserv-and-diffserv.pdf, última fecha de acceso: marzo de 2006.
[29] Y. Bernet, RFC 2998: A Framework for Integrated Services Operation over Diffserv Networks, November 2000.
[URL, 30] *Qué es el kernel/núcleo?,* http://www.hospedajeydominios.com/mambo/documentacion-manual_linux-pagina-35.html, última fecha de acceso: octubre de 2006.
[URL, 31] *Tipos de Kernel,* http://laurel.datsi.fi.upm.es/~rpons/personal/trabajos/lpractico/node60.html, última fecha de acceso: octubre de 2006
[URL, 32] *DPD Kernel,* http://wiki.gleducar.org.ar/wiki/DPD_Kernel, última fecha de acceso: octubre de 2006
[URL, 33] *Concrete Architecture of the Linux Kernel,* http://plg.uwaterloo.ca/~itbowman/CS746G/a2, última fecha de acceso: octubre de 2006
[URL, 34] *DSMARK queuing discipline*, http://www.opalsoft.net/qos/DS-29.htm, última fecha de acceso: enero de 2007
[URL, 35] *Red Experimental DiffServ Utilizando Router-PCs con Sistema Operativo Linux*, http://jogiguz.webs.upv.es/papers/JTRedIris03.pdf, última fecha de acceso: enero de 2007
[URL, 36] *HTB queuing discipline*, http://www.opalsoft.net/qos/DS-28.htm, última fecha de acceso: enero de 2007
[URL, 37] *TCINDEX classifier*, http://www.opalsoft.net/qos/DS-210.htm, última fecha de acceso: enero de 2007
[URL, 38] *Seguridad en Linux*, http://www.iec.csic.es/CRIPTonOMICon/linux/ Última fecha de acceso: enero de 2007
[URL, 39] GRED *queuing discipline*, http://www.opalsoft.net/qos/DS-27.htm, última fecha de acceso: enero de 2007

«ANEXO I»

ESTRUCTURAS PARA QoS EN FC5

Estructura tcf_proto_ops

Estructura de la que se basa el módulo cls_rsvp, para llevar a cabo QoS mediante el modelo Int – Serv.

```
struct tcf_proto_ops
{
        struct tcf_proto_ops    *next;
        char                    kind[IFNAMSIZ];
        int                     (*classify)(struct sk_buff*, struct tcf_proto*,
                                struct tcf_result *);
        int                     (*init)(struct tcf_proto*);
        void                    (*destroy)(struct tcf_proto*);
        unsigned long           (*get)(struct tcf_proto*, u32 handle);
        void                    (*put)(struct tcf_proto*, unsigned long);
        int                     (*change)(struct tcf_proto*, unsigned long,
                                u32 handle, struct rtattr **, unsigned long *);
        int                     (*delete)(struct tcf_proto*, unsigned long);
        void                    (*walk)(struct tcf_proto*, struct tcf_walker *arg);
        /* rtnetlink specific */
        int                     (*dump)(struct tcf_proto*, unsigned long,
                                struct sk_buff *skb, struct tcmsg*);
        struct module           *owner;
};
```

Estructura del módulo cls_rsvp

Estructura que utilizan los módulos DSMARK, GRED, HTB y TC_INDEX para llevar a cabo QoS mediante Diff – Serv.

```
struct sk_buff_head
{
   struct sk_buff       * next;
   struct sk_buff       * prev;
   __u32                qlen;

#if CONFIG_SKB_CHECK
   int                  magic_debug_cookie;
#endif
};
```

```
struct sk_buff
{
   struct sk_buff       * next;              /* Siguiente buffer en la lista*/
   struct sk_buff       * prev;              /* Previo buffer en la lista*/
   struct sk_buff_head * list;
#if CONFIG_SKB_CHECK
   int                  magic_debug_cookie;
#endif
   struct sk_buff       *link3;              /* Link para el protocolo IP*/
   struct sock          *sk;
   unsigned long        when;
   struct timeval       stamp;               /* Tiempo de llegada*/
   struct device        *dev;                /*Dispositivo de llegada o de salida*/
   union
   {
            struct tcphdr       *th;
            struct ethhdr       *eth;
            struct iphdr        *iph;
            struct udphdr       *uh;
            unsigned char       *raw;
            /* para pasar encabezados de archivos en dominio unix */
            void *filp;
   } h;

   union
   {
                                unsigned char       *raw;
            struct ethhdr       *ethernet;
   } mac;

   struct iphdr         *ip_hdr;
   unsigned long        len;                 /* Tamaño del dato actual*/
   unsigned long        csum;                /* Checksum */
   __u32                saddr;               /* Dirección IP fuente*/
   __u32                daddr;               /* Dirección IP destino*/
   __u32                raddr;               /*Dirección IP del siguiente salto*/
   __u32                seq;                 /* Número de secuencia TCP */
   __u32                end_seq;
   __u32                ack_seq;
   unsigned char        proto_priv[16];
   volatile char        acked,
                        used,
                        free,
                        arp;
   unsigned char        tries,
                        lock,
                        localroute,
                        pkt_type,            /* Clase de paquetes*/
                        pkt_bridged,
                        ip_summed;
#define PACKET_HOST                 0                   /* Para el host*/
```

```
#define PACKET_BROADCAST        1               /* Para todos*/
#define PACKET_MULTICAST        2               /* Para un grupo*/
#define PACKET_OTHERHOST        3               /* Para alguien*/
   unsigned short     users;
   unsigned short     protocol;
   unsigned short     truesize;

   atomic_t count;
   struct sk_buff     *data_skb;
   unsigned char      *head;
   unsigned char      *data;
   unsigned char      *tail;
   unsigned char      *end;
   void               (*destructor)(struct sk_buff *);
   __u16              redirport;
};
```