

UNIVERSIDAD DE MÁLAGA  
ESCUELA TÉCNICA SUPERIOR DE  
INGENIERÍA DE TELECOMUNICACIÓN



PROYECTO FIN DE CARRERA:

**IMPLEMENTACIÓN DE UN ENCAMINADOR  
IP CON CALIDAD DE SERVICIO  
PARA WINDOWS**

INGENIERÍA DE TELECOMUNICACIÓN

MÁLAGA, 2001

ÓSCAR LÓPEZ MARTÍN



UNIVERSIDAD DE MÁLAGA  
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE  
TELECOMUNICACIÓN

Titulación: Ingeniería de Telecomunicación

Reunido el tribunal examinador en el día de la fecha, constituido por:

D. \_\_\_\_\_

D. \_\_\_\_\_

D. \_\_\_\_\_

para juzgar el Proyecto Fin de Carrera titulado:

**IMPLEMENTACIÓN DE UN ENCAMINADOR IP  
CON CALIDAD DE SERVICIO PARA WINDOWS**

del alumno D. **Óscar López Martín**

dirigido por D. **Francisco Javier González Cañete**

ACORDÓ POR: \_\_\_\_\_ OTORGAR LA CALIFICACIÓN DE:

\_\_\_\_\_  
y, para que conste, se extiende firmada por los componentes del tribunal, la presente diligencia.

Málaga, a \_\_\_\_\_ de \_\_\_\_\_ de 2001

El Presidente:

El Secretario:

El Vocal:

Fdo: \_\_\_\_\_

Fdo: \_\_\_\_\_

Fdo: \_\_\_\_\_



UNIVERSIDAD DE MÁLAGA  
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE  
TELECOMUNICACIÓN

# IMPLEMENTACIÓN DE UN ENCAMINADOR IP CON CALIDAD DE SERVICIO PARA WINDOWS

REALIZADO POR:

Óscar López Martín

DIRIGIDO POR:

Francisco Javier González Cañete

DEPARTAMENTO DE: Tecnología Electrónica

TITULACIÓN: Ingeniería de Telecomunicación

Palabras claves: Encaminador, Router, QoS, RSVP, IP, Reserva de Recursos, Señalización, Calidad de Servicio, Garantías, Tráfico, Retardo, Saturación, Multimedia, Internet, Borland C++ Builder, CPPBuilder, Multitarea, Ethernet, Windows.

## RESUMEN:

Cada vez más la red Internet es usada para la transferencia de contenido multimedia. Este tipo de conferencias no están tan extendidas debido al problema que presentan las comunicaciones de tráfico en tiempo real a través de una red de conmutación de tipo datagrama como es IP y por lo tanto Internet. Este tipo de comunicaciones necesitan una garantía en cuanto al ancho de banda, retraso máximo y variación del retardo, entre otros parámetros, para poder ser realizadas satisfactoriamente. Para garantizar una calidad de servicio se ha desarrollado la versión 1.0 del protocolo de señalización RSVP que garantiza los parámetros críticos de comunicaciones en tiempo real mediante la reserva de recursos de conmutación en los encaminadores. Este proyecto implementará un encaminador IP completamente operativo que soporte el protocolo RSVP. Se decide la implementación para entornos Windows al no existir para él ningún software con estas características siendo el S.O. más utilizado para aplicaciones de multimedia en tiempo real por la mayoría de los usuarios. Además, este software proporcionará datos estadísticos y la configuración de parámetros internos para poder comparar las ventajas que aporta el soporte de RSVP a un encaminador convencional.

Málaga, Noviembre 2001



**A mis Padres**

*“El final de una cosa es sólo el comienzo de otra”*

# Agradecimientos

Estas son las últimas líneas que escribo de este documento, el último de una carrera que comencé en 1994 tras dar bastantes tumbos por toda España y parte del extranjero. Si a alguien tengo que agradecer que esté en esta situación es, ante todo, a mis padres, que confiaron en mí y en mi responsabilidad. Sin su apoyo moral, espiritual y económico no hubiera podido ni siquiera empezar estos estudios que tanto me han dado y que elegí mucho antes de entrar en la universidad.

Quiero agradecer a Nathalie las horas que estudiamos juntos ya que sus asignaturas de programación concurrente me sirvieron para entender los problemas de la multitarea y cómo se solucionaban. A su familia por acogerme como un hijo cuando no tuve a mis padres, a Iciar por aguantarme en época de exámenes y durante los meses más intensos de este proyecto. A todos los amigos que se han portado conmigo como si fueran mis hermanos: mi amigo, compañero y tutor Francisco Javier González, a mis dos compañeros y amigos, que durante la carrera de Telecomunicación hemos compartido tantas experiencias Ismael y en especial a Juan de Dios.

En estos momentos me gustaría que todos los que significan algo para mí pudieran sentir la gran satisfacción que siento y que no hubiera podido alcanzar sin ellos, los que se perdieron en las aguas del olvido y los que ya no están con nosotros, como Mi profesor de física de C.O.U. en Lisboa, Sebastián Santolito, o la desgracia que sufrimos al perder a Marienka.

Agradezco también los buenos ratos con mis amigos de la red y de las Partys que han hecho olvidar lo malo de esta carrera y afrontar el futuro con optimismo, a Los Simpsons, TRON y The Matrix por lo que han sido para mí.

Gracias a todos.

# Índice de Contenidos

<i>Índice de contenidos</i>	<i>i</i>
<i>Relación de Acrónimos</i>	<i>vii</i>
<b>CAPÍTULO 1: Introducción</b>	<b>1</b>
<b>CAPÍTULO 2: Estudio del Problema</b>	<b>4</b>
<b>2.1 Redes de Comunicaciones y Protocolo TCP/IP</b>	<b>4</b>
2.1.1 La Red	4
2.1.2 TCP/IP	5
2.1.3 Funcionamiento de la Red de Datagramas TCP/IP	6
2.1.4 El Modelo De 4 Capas de Windows	8
<b>2.2 Los Problemas del tráfico en tiempo real</b>	<b>11</b>
2.2.1 Pérdida de Paquetes	12
2.2.2 El <i>Jitter</i>	12
2.2.3 Elevado Tiempo de Tránsito	13
<b>2.3 La Solución es la Reserva de Recursos</b>	<b>14</b>
<b>2.4 Necesidad del Encaminador</b>	<b>15</b>
<b>2.5 El Lenguaje de Programación</b>	<b>16</b>
<b>2.6 Conclusión</b>	<b>17</b>
<b>CAPÍTULO 3: Protocolos Usados</b>	<b>18</b>

<b>3.1</b>	<b>Introducción</b>	<b>18</b>
<b>3.2</b>	<b>Protocolos de la Capa de Acceso al Medio</b>	<b>19</b>
3.2.1	El estandar 802	19
3.2.2	Direcciones Físicas	23
3.2.3	Formato de la Trama	24
3.2.4	Velocidades	25
3.2.5	Tipos de Adaptadores	26
<b>3.3</b>	<b>Protocolo IP4</b>	<b>27</b>
3.3.1	Introduccion	27
3.3.2	El Datagrama IP	28
3.3.3	Formato del Datagrama IP	28
3.3.4	Fragmentación	31
3.3.5	Direcciones IP	33
3.3.6	Direcciones IP Especiales y Reservadas	35
3.3.7	Encaminamiento IP	37
3.3.8	Encaminamiento con Salto al Siguiente.	38
3.3.9	Algoritmo de Encaminamiento IP	40
3.3.10	Manejo de Datagramas Entrantes.	41
3.3.11	Direccionamiento sin Clase	41
3.3.12	CIDR Enrutamiento Inter-Dominio Sin Clases (Classless Inter-Domain Routing)	42
<b>3.4</b>	<b>ICMP: Protocolo de Mensajes de Control de Internet (Internet Control Message Protocol) 45</b>	
3.4.1	Formato del Mensaje ICMP	46
3.4.2	Mensajes Solicitud de Eco y Respuesta al Eco	46
3.4.3	Mensaje Destino Inaccesible.	47
3.4.4	Mensaje de Choque	48
3.4.5	Mensaje Redireccionar	49
3.4.6	Mensaje Tiempo Excedido	50
3.4.7	Mensaje Problema de Parámetros	50
3.4.8	Mensaje Solicitud de Timestamp y Respuesta de Timestamp	51
3.4.9	Mensaje Solicitud de Mascara de Subred y Respuesta de Mascara de Subred	51
3.4.10	PING para Comprobar la Comunicación IP entre dos Ordenadores	52
3.4.11	Utilidad de PING para Diagnosticar Errores en una Red Aislada	53
3.4.12	Utilidad de PING para Diagnosticar errores en un Router	54
3.4.13	Utilidad de los Mensajes de Tiempo Excedido Para Marcar la Ruta	55
<b>3.5</b>	<b>Protocolo ARP</b>	<b>57</b>
3.5.1	Descripción	57

3.5.2	Resolviendo una Dirección IP remota.	59
3.5.3	La <i>caché</i> ARP	63
<b>3.6</b>	<b>Protocolo UDP</b>	<b>64</b>
3.6.1	Introducción	64
3.6.2	Descripción	64
3.6.3	Formato del Mensaje UDP	65
3.6.4	Interfaz de Usuario	66
3.6.5	Interfaz IP	67
3.6.6	Aplicación del Protocolo	67
<b>3.7</b>	<b>Protocolo TCP</b>	<b>68</b>
3.7.1	Descripción	68
3.7.2	Asegurar la Fiabilidad	69
3.7.3	Identificador de Conexiones	71
3.7.4	Formato del Segmento TCP	72
3.7.5	Establecimiento de una Conexión	74
3.7.6	Cierre de una Conexión	76
<b>3.8</b>	<b>Protocolo RSVP</b>	<b>77</b>
3.8.1	Características	78
3.8.2	Funcionamiento del Protocolo	81
3.8.3	Estructura de una Implementación RSVP	82
3.8.4	Modelo de Reservas	86
3.8.5	Creación del Camino de Datos	92
3.8.6	Realización de la Reserva	93
3.8.7	Los <i>'Soft State'</i>	95
3.8.8	One Pass with Advertising (OPWA)	100
3.8.9	Zonas no RSVP	103
3.8.10	Formato de los Mensajes RSVP	106
3.8.11	Los Objetos RSVP	110
<b>3.9</b>	<b>El Interfaz PACKET32</b>	<b>125</b>
3.9.1	Estructuras de Datos	126
3.9.2	Funciones	128
<b><i>CAPÍTULO 4: Desarrollo del Encaminador IP Convencional.</i></b>		<b>133</b>
<b>4.1</b>	<b>Introducción</b>	<b>133</b>

## Índice de Contenidos

---

<b>4.2</b>	<b>Estructura del Router</b>	<b>134</b>
4.2.1	El Concepto Inicial	134
4.2.2	Identificando Procesos	135
4.2.3	Estructura Definitiva	137
<b>4.3</b>	<b>Recepción y Envío de Datos</b>	<b>138</b>
4.3.1	Introducción	138
4.3.2	El Interfaz con PACKET32	139
<b>4.4</b>	<b>Uso de Multitarea</b>	<b>140</b>
4.4.1	Ventajas	140
4.4.2	Inconvenientes	140
4.4.3	Sincronización entre Procesos	141
4.4.4	Acceso a Zonas de Memoria Comunes	142
4.4.5	Creación y Destrucción de Hebras de Proceso	143
<b>4.5</b>	<b>Los Trabajadores (hebras)</b>	<b>143</b>
4.5.1	Ciclo de Vida	144
4.5.2	Proceso Receptor	145
4.5.3	Proceso Transmisor	145
<b>4.6</b>	<b>Las Colas de Paquetes</b>	<b>147</b>
4.6.1	El Constructor	149
4.6.2	La Inserción de Paquetes	149
4.6.3	La extracción de Paquetes	149
4.6.4	El destructor	150
<b>4.7</b>	<b>El Clasificador de Paquetes</b>	<b>150</b>
<b>4.8</b>	<b>La cache ARP</b>	<b>151</b>
4.8.1	Pidiendo Datos a la Cache ARP	152
4.8.2	Insertando una Asociación en la Cache ARP	152
<b>4.9</b>	<b>Los Adaptadores</b>	<b>153</b>
4.9.1	Interfases Relacionados	153
4.9.2	Obteniendo el Mejor Adaptador	155
<b>4.10</b>	<b>Otras Clases Usadas</b>	<b>155</b>
4.10.1	La Clase Paquete	156
4.10.2	La clase TDirMAC	156
4.10.3	La Clase TRouter	156
<b>4.11</b>	<b>El interfaz de usuario</b>	<b>157</b>

<b>4.12 Pruebas de Funcionamiento</b>	<b>157</b>
<b><i>CAPÍTULO 5: Integración de la gestión RSVP</i></b>	<b>160</b>
<b>5.1 Introducción</b>	<b>160</b>
<b>5.2 Estructura básica</b>	<b>161</b>
<b>5.3 El Gestor RSVP</b>	<b>162</b>
5.3.1 Punto de entrada al gestor RSVP	162
5.3.2 Interpretar los Mensajes de Señalización RSVP	163
5.3.3 El Gestor de Mensajes	164
5.3.4 Encaminamiento de los Paquetes de una Sesión	168
5.3.5 Problemas de Accesos Concurrentes al Gestor	168
<b>5.4 Limitando en ancho de banda</b>	<b>169</b>
<b>5.5 Planificación de colas</b>	<b>171</b>
5.5.1 First-In-First-Out (FIFO)	171
5.5.2 GPS (Generalized Processor Sharing)	173
5.5.3 SQ (Statistic Queuing)	173
5.5.4 WFQ (Weighted Fair Queueing)	175
5.5.5 WF <sup>2</sup> Q (Worst-case Fair Weighted Fair Queueing)	176
<b>5.6 Estimación del Ancho de Banda</b>	<b>178</b>
5.6.1 Ethernet es un Medio de Acceso en Contienda	178
5.6.2 Clase TBenchMark	179
<b>5.7 Creación del entorno de usuario</b>	<b>180</b>
5.7.1 Ventana Principal	180
5.7.2 Pestaña Cola de Entrada	182
5.7.3 Pestaña Clasificador	183
5.7.4 Pestaña Adaptadores	185
5.7.5 Pestaña QoS	187
<b>5.8 Pruebas de funcionamiento</b>	<b>189</b>
<b><i>CAPÍTULO 6: Manual de Usuario</i></b>	<b>192</b>
<b>6.1 Introducción</b>	<b>192</b>
<b>6.2 Requerimientos</b>	<b>193</b>

## Índice de Contenidos

---

<b>6.3</b>	<b>Instalación</b>	<b>193</b>
<b>6.4</b>	<b>Funcionamiento Interno</b>	<b>195</b>
<b>6.5</b>	<b>Interfaz de Usuario</b>	<b>196</b>
6.5.1	Pestaña Cola de Entrada	197
6.5.2	Pestaña Clasificador	198
6.5.3	Pestaña Adaptadores	200
6.5.4	Pestaña QoS	202
<b><i>CAPÍTULO 7: Conclusiones y líneas futuras</i></b>		<b><i>205</i></b>
<b>7.1</b>	<b>Conclusiones finales</b>	<b>205</b>
<b>7.2</b>	<b>Líneas futuras</b>	<b>207</b>
<b><i>Bibliografía</i></b>		<b><i>209</i></b>

## Relación de Acrónimos

<b>ARP</b> .....	Address Resolution Protocol.
<b>ATM</b> .....	Asynchronous Transfer Mode.
<b>BCPPB</b> .....	Borland C++ Builder.
<b>BGP</b> .....	Border Gateway Protocol.
<b>CBR</b> .....	Constant Bit Rate.
<b>CIDR</b> .....	Classless Inter-Domain Routing.
<b>CPU</b> .....	Central Processing Unit.
<b>CSMA/CD</b> ...	Carrier Sense Multiple Access / Collision Detect.
<b>DDK</b> .....	Driver Development Kit.
<b>DLL</b> .....	Dynamic Link Library
<b>DNS</b> .....	Domain Name System.
<b>DQDB</b> .....	Distributed Queue Dual Bus.
<b>E/S</b> .....	Entrada/Salida.
<b>FIFO</b> .....	First In - First Out.
<b>FTP</b> .....	File transfer Protocol.

<b>GPS</b>	Generalized Processor Sharing.
<b>IEEE</b>	Institute of Electric and Electronic Engineers.
<b>IETF</b>	Internet Engineering Task Force.
<b>ICMP</b>	Internet Control Message Protocol.
<b>IGMP</b>	Internet Group Management Protocol.
<b>IGP</b>	Interior Gateway Protocol.
<b>IP</b>	Internet Protocol.
<b>IPX</b>	Internetwork Packet eXchange.
<b>IS</b>	Integrated Services.
<b>LAN</b>	Local Area Network.
<b>LED</b>	Light Emitting Diode.
<b>LCC</b>	Logical Link Control.
<b>MAC</b>	Medium Access Control.
<b>MPS</b>	Multiproceso paralelo simétrico.
<b>MS</b>	MicroSoft.
<b>MSDN</b>	MicroSoft Developer Network.
<b>MTU</b>	Maximun Transfer Unit.
<b>NAPT</b>	Network Address and Port Tranlation.
<b>NM</b>	NetMeeting.
<b>OPWA</b>	One Pass with Advertising.
<b>OSI</b>	Open System Interconnection.
<b>PC</b>	Personal Computer.
<b>PPP</b>	Point to Point Protocol.
<b>QoS</b>	Quality of Service.
<b>RDSI</b>	Red Digital de Servicios Integrados.
<b>RFC</b>	Request For Comment.
<b>ROADS</b>	Running Out of Address Space
<b>RSVP</b>	Resource reSerVation Protocol.
<b>RTB</b>	Red Telefónica Básica.
<b>SDK</b>	Software Development Kit.
<b>SFQ</b>	Statistic Fair Queuing.
<b>SNMP</b>	Simple Network Management Protocol.

<b>SPX</b> .....	Simple Packet eXchange.
<b>STP</b> .....	Shielded Twister Pair.
<b>TCP</b> .....	Transmisión Control Protocol.
<b>TFTP</b> .....	Trivial File Transfer Protocol.
<b>TTL</b> .....	Time To Live.
<b>TV</b> .....	Televisión.
<b>UDP</b> .....	User Datagram Protocol.
<b>UTP</b> .....	Unshielded Twisted Pair.
<b>WAN</b> .....	Wide Area Network.
<b>WFQ</b> .....	Weighttted Fair Queuing.
<b>WF<sup>2</sup>Q</b> .....	Worst-case Fair Weighted Fair Queueing.



# **CAPÍTULO 1: Introducción**

Gracias a la popularidad adquirida por Internet y al gran crecimiento sufrido por esta en los últimos años, las redes de ordenadores que usan protocolos IP (Internet Protocol) son las más extendidas hoy día. Por esta razón, en la actualidad se realizan multitud de investigaciones encaminadas a mejorar la red, intentando solucionar nuevos problemas que se van detectando y adaptándola a los nuevos tipos de aplicaciones que van apareciendo.

Una de estas mejoras es hacer posible la comunicación en tiempo real de contenido multimedia, ya que en un futuro cercano este tipo de transferencias se harán muy comunes, haciendo posible la recepción de audio y vídeo de muy alta calidad o videoconferencia sobre el protocolo de red IP. La solución más extendida es la que hace uso del protocolo de

reserva de recursos RSVP (ReSerVation Protocol). Cada vez hay más aplicaciones multimedia que hacen uso de este sistema, y la mayoría para el sistema operativo Windows, ya que es indudablemente el más extendido en el mundo del ordenador personal.

El problema es que cambiar o actualizar la flota de encaminadores que forman Internet es caro y lento. Para hacer que esta tarea no sea tan aparatosa, este proyecto crea un encaminador IP con soporte para aplicaciones que hagan uso del protocolo RSVP. Este encaminador o *Router* es capaz de saber la calidad de servicio o QoS (Quality of Service) que necesitan las transferencias que hacen uso de RSVP. Para que realmente aporte una solución, este software se ha desarrollado para ser usado bajo Windows, ya que puede ser instalado en cualquier máquina con procesadores Pentium o superiores y de esta manera reemplazar rápidamente y sin un alto coste los routers tradicionales sin soporte para RSVP.

Para aprovechar toda la potencia del PC (Personal Computer) tanto de los más antiguos como de los más rápidos se ha realizado usando técnicas de multiproceso. Así mismo se ha optimizado el código para una velocidad óptima y un eficiente uso de el/los procesadores y dispositivos de red.

El trabajo comienza planteando los problemas actuales de Internet para transmitir de manera eficiente información en tiempo real, analizando las fuentes de cada uno de ellos, llevándonos a RSVP como la solución más completa. Continuaremos con un análisis en profundidad del protocolo RSVP, viendo sus características y funcionamiento. Se describirán todos los objetos de los que hace uso y la estructura básica de encaminador RSVP. Los siguientes capítulos describirán detalles de la implementación usando la herramienta de programación orientada a objetos Borland C++ Builder 5. En estos capítulos se detallan las fases correspondientes a la creación del código:

1<sup>a</sup> Desarrollo del encaminador IP convencional:

Se implementa la capacidad de encaminamiento IP incluyendo la búsqueda en tabla de encaminamiento, gestión de colas, capacidad multiproceso y envío y recepción de datos a la red.

2<sup>a</sup> Comprobación del encaminador IP:

En esta fase se comprueba el correcto funcionamiento de la aplicación en un entorno real formado por varios PC gestionando tráfico convencional con política *Best-Effort* (Mejor esfuerzo).

3<sup>a</sup> Ampliación de la aplicación para soporte de RSVP:

Una vez comprobada la funcionalidad convencional del *router*, se procede a incorporar la capacidad de reserva de recursos que gestiona el protocolo RSVP, así como distintas políticas de gestión de colas y generación de datos estadísticos disponibles en un entorno gráfico de usuario.

4<sup>a</sup> Pruebas y Optimización:

La última fase de este proyecto consistirá en la comprobación del correcto funcionamiento del router usando tráfico en tiempo real con RSVP junto con tráfico convencional. Para ello no se pudo usar la red del departamento, ya que no permite la creación del entorno de pruebas adecuado, así que se usó una red privada conectada a internet mediante un router NAPT.

Este software además liberará al investigador de tener que realizar simulaciones en un PC y probar realmente cómo se comporta la transferencia de señal multimedia a través de un Router con soporte QoS en distintos escenarios, aportando datos estadísticos reales de su operación.

## **CAPÍTULO 2: Estudio del Problema**

### **2.1 REDES DE COMUNICACIONES Y PROTOCOLO TCP/IP**

#### **2.1.1 LA RED**

Una red es un conjunto de ordenadores, conectados entre si, que pueden comunicarse compartiendo datos y recursos. Los ordenadores suelen estar conectados mediante cables.

Pero si la red es extensa, las conexiones pueden realizarse a través de líneas telefónicas, radio, microondas, e incluso satélites.

Las redes se clasifican en redes de área local (LAN: *Local Area Network*) y redes de área amplia (WAN: *Wide Area Network*). Las redes LAN abarcan una zona no demasiado grande, mientras que las WAN pueden abarcar varios países. Internet es una red, que a su vez se compone de otras redes, que pueden ser LAN (conectadas mediante Ethernet y otros protocolos de redes como pueden ser IPX/SPX, AppleTalk o por supuesto el estándar de internet : TCP/IP) o WAN (mediante módems a través de líneas telefónicas , conexiones ISDN , etc...).

### 2.1.2 TCP/IP

Para que la comunicación entre ordenadores sea posible, es necesaria la existencia de un protocolo. Un protocolo es un conjunto de convenciones que determinan cómo se realiza el intercambio de datos entre dos ordenadores o aplicaciones. El protocolo usado por todas las redes que forman parte de Internet se llama abreviadamente **TCP/IP**. Mientras que las grandes organizaciones tienen sus propias LANs y *Gateways*, los usuarios particulares deben conectarse mediante módems. Existen dos formas para la conexión de estos últimos: La conexión por Dial y por el nivel **IP**. La primera consiste en conectarse con el servidor del ordenador del proveedor del servicio y tener acceso a los programas y utilidades que ofrece este servidor, para este tipo de conexión solo se necesita un terminal y un módem, ejemplos de proveedores de este servicio son CIX, CompuServe y Delphi.

Una conexión de nivel **IP** es mucho más complicada, pero ofrece mucha mayor flexibilidad. Se necesita de la instalación de una serie de drivers de red en el ordenador local, una pila de protocolos TCP/IP y un controlador de bajo nivel de módem. Una vez configurada esta pila, ya se puede ejecutar cualquier tipo de software **TCP/IP** que lo reconozca y tener acceso directo a Internet. Internet es una red, a través de la cual se

encuentran interconectadas una gran cantidad de redes de ordenadores, de forma que cada ordenador puede comunicarse con cualquier otro independientemente del tipo o del sistema operativo que utilice. Por eso el protocolo común de comunicaciones usado en Internet es el **TCP/IP** [Wright].

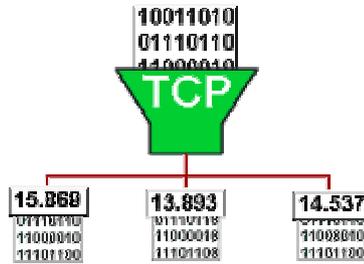
### **2.1.3 FUNCIONAMIENTO DE LA RED DE DATAGRAMAS TCP/IP**

Cuando se transmite información de un ordenador a otro, esta no es transmitida de una sola vez, sino que se divide en paquetes pequeños, evitando de esta manera la monopolización de los recursos de la red por un solo usuario. Por los cables de la red viajan paquetes de información provenientes de diferentes ordenadores y con destinos también diferentes. Para alcanzar su destino, estos paquetes atraviesan en el camino cierto número de ordenadores y otros dispositivos que hacen que la transmisión sea posible. Las distintas partes que forman Internet están conectadas por un conjunto de ordenadores llamados *Routers* o encaminadores, cuya misión principal es redirigir los paquetes de información que reciben por el camino adecuado para que alcancen su destino. Esto es posible ya que cada paquete lleva la información del remitente y el destinatario, al igual que cualquier correo tradicional.

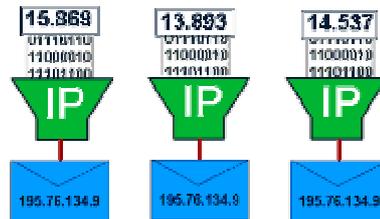
Resulta curioso comprobar cómo el funcionamiento de una red de ordenadores tan grande como Internet se basa en una idea conceptualmente sencilla: dividir la información en trozos o *paquetes*, que viajan de manera independiente hasta su destino, donde conforme van llegando se ensamblan de nuevo para dar lugar al contenido original. Estas funciones las realizan los protocolos TCP/IP: el *Transmission Control Protocol* se encarga de fragmentar y unir los paquetes y el *Internet Protocol* tiene como misión hacer llegar los fragmentos de información a su destino correcto.

El protocolo **TCP** (Transmission Control Protocol) se encarga de dividir la información en paquetes de tamaño adecuado, numerarlos para que puedan volver a unirse en el orden

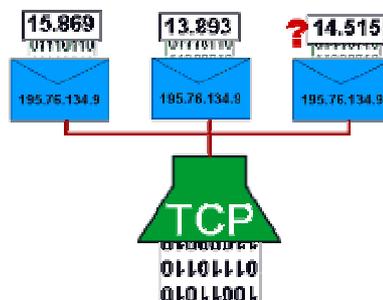
correcto y añadir cierta información extra necesaria para la transmisión y posterior decodificación del paquete. Además se encarga de retransmitir paquetes de los que no llega confirmación.



Mientras, el protocolo **IP** (Internet Protocol) se encarga de etiquetar cada paquete de información con la dirección apropiada. Cada ordenador conectado tiene una dirección Internet (IP Address) única y exclusiva, que está formada por cuatro números separados por puntos, cada uno de los cuales puede tomar valores entre 0 y 255.



A medida que se “ensobran”, los paquetes son enviados mediante Routers, que deciden en cada momento cuál es el camino más adecuado para llegar a su destino. Dado que la carga de Internet varía constantemente, los paquetes pueden ser enviados por distintas rutas, llegando en ese caso desordenados.

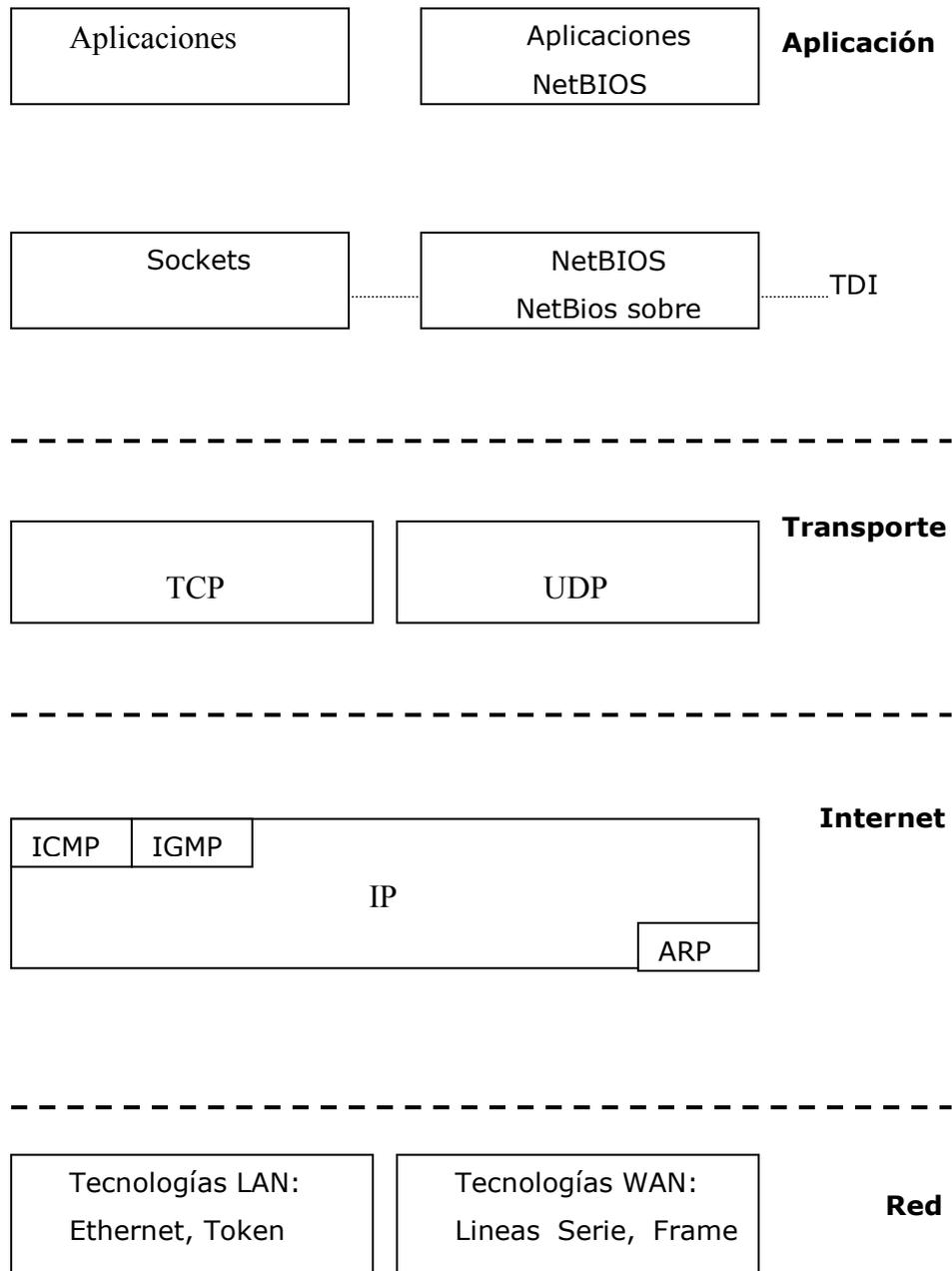


Con la llegada de paquetes a su destino, se activa de nuevo el protocolo TCP, que realiza una nueva suma de comprobación y la compara con la suma original. Si alguna de ellas no coincide, detectándose así pérdida de información en el trayecto, se solicita de nuevo el envío del paquete desde el origen. Por fin, cuando se ha comprobado la validez de todos los paquetes, el TCP los une formando el mensaje inicial.

En 1982, **TCP/IP** Internet incluía unos cuantos cientos de ordenadores concentrados principalmente en Norte América. En la primavera de 1993, ya había más de 1.200.000 ordenadores conectados a Internet en 45 países repartidos por los 7 continentes, y su tamaño sigue duplicándose cada 10 meses. Al principio los usos básicos que ofrecía la red eran los de correo electrónico, transferencia de ficheros y conexiones remotas. En la actualidad existe una gran cantidad de usuarios que diseñan protocolos de aplicación para construir sus aplicaciones software. La variedad de las aplicaciones que usan **TCP/IP** consisten en sistemas de monitorización y control de plataformas industriales, sistemas de control de inventarios de almacén, aplicaciones que permiten compartir el acceso a archivos entre sistemas alejados geográficamente, como también posibilitar teleconferencias y aplicaciones de sistemas multimedia.

### **2.1.4 EL MODELO DE 4 CAPAS DE WINDOWS**

Aunque este modelo es general en todas las implementaciones del TCP/IP, a lo largo del presente documento, vamos a ceñirnos a su implementación en Microsoft Windows [MSDN] que analizaremos a continuación.



**Capa de Red:** La base de este modelo de capas, es la capa de *interface* de red. Esta capa es la responsable de enviar y recibir *frames* (estructuras o marcos. Pero se prefiere a partir de ahora optar por el termino inglés, ya que es ampliamente aceptado en la terminología informática), los cuales son los paquetes de información que viajan en una red como una ‘unidad simple’. La capa de red, envía *frames* a la red, y recupera *frames* de la red.

**Capa de Internet:** Este protocolo encapsula paquetes en *datagramas* internet (no es tampoco la palabra *datagrama* una palabra castellana, pero es también aceptada en la terminología informática como ‘paquete de datos’) y además esta capa ejecuta todos los algoritmos de enrutamiento (*routing*) de paquetes. Los cuatro protocolos Internet son: *Internet Protocol* (IP), *Address Resolution Protocol* (ARP), *Internet Control Message Protocol* (ICMP) y *Internet Group Management Protocol* (IGMP).

- IP es el responsable del envío y enrutamiento de paquetes entre máquinas y redes.
- ARP obtiene las direcciones de hardware de las máquinas situadas en la misma red física. Recordemos que la dirección física de cada tarjeta de red es única en el mundo. Dicha dirección “física” ha sido implementada vía hardware por el fabricante de la tarjeta de red, y dicho fabricante, lo selecciona de un rango de direcciones único asignado a él y garantiza la unicidad de dicha tarjeta. Este caso es el más corriente y es el de las tarjetas de Red Ethernet. Existe para otras topologías de red, igualmente una asignación única hardware de reconocimiento de la tarjeta.
- ICMP envía mensajes e informa de errores en el envío de paquetes.
- IGMP se utiliza para la comunicación entre *routers* (enrutadores de Internet).

**Capa de Transporte:** La capa de transporte, nos da el nivel de “sesión” en la comunicación. Los dos protocolos posibles de transportes son TCP (*Transmission Control Protocol*) y UDP (*User Datagram Protocol*). Se puede utilizar uno u otro protocolo dependiendo del método preferido de envío de datos.

El TCP nos da un tipo de conectividad “orientada a conexión”. Típicamente se utiliza para transferencias de largas cantidades de datos de una sola vez. Se utiliza también en aplicaciones que requieren un “reconocimiento” o validación (ACK : *acknowledgment*) de los datos recibidos.

El UDP proporciona conexión de comunicación y no garantiza la entrega de paquetes. Las aplicaciones que utilicen UDP normalmente envían pequeñas cantidades de datos de una sola vez. La aplicación que lo utilice, es la responsable en este caso de la integridad de los paquetes y debe establecer sus propios mecanismos para pedir repetición de mensaje, seguimiento, etc, no existiendo ni garantía de entrega ni garantía del orden de entrega en la maquina destino.

**Capa de Aplicación:** En la cima de este modelo, está la capa de aplicación. Esta es la capa que las aplicaciones utilizan para acceder a la red. Existen muchas utilidades y servicios en la capa de aplicación, por ejemplo: FTP, Telnet, SNMP y DNS.

## 2.2 LOS PROBLEMAS DEL TRÁFICO EN TIEMPO REAL

El tráfico en tiempo real es muy sensible a los problemas que acarrea la red IP y que ni siquiera un protocolo de transporte como TCP puede solucionar. Ya que estas comunicaciones no admiten pérdidas ni retrasos [Casado00]. Es por ello por lo que hasta ahora las comunicaciones en tiempo real se han venido realizando a través de redes de conmutación de circuitos o del tipo *Frame Relay* que aseguran la llegada de los datos en orden y a la velocidad contratada. Como ejemplo basta citar la red telefónica básica o los

enlaces de televisión vía satélite. A continuación veremos porqué las redes de conmutación de datagramas no son las adecuadas para este tipo de tráfico, viendo los problemas y su causa.

### **2.2.1 PÉRDIDA DE PAQUETES**

La señal en tiempo real sufre de la pérdida de paquetes al igual que el resto de tráfico en situaciones de saturación en los que los encaminadores tienen sus buffers llenos de paquetes. Evidentemente este problema sólo ocurre en los encaminadores en un escenario de saturación. El tráfico que no es en tiempo real admite este tipo de problemas ya que el protocolo de transporte TCP (Transport Control Protocol) tiene en cuenta estas situaciones y únicamente sufrirá un retardo un poco más alto. En las comunicaciones en tiempo real esto no vale ya que es importante que los paquetes lleguen ordenados para de esta manera evitar esperas inútiles de paquetes perdidos.

### **2.2.2 EL *JITTER***

A primera vista tal y como hemos dicho, las redes de datagrama (no orientadas a conexión) no parecen adecuadas para tráfico en tiempo-real. Los paquetes son enrutados independientemente a través de redes compartidas, de forma que los tiempos de tránsito varían significativamente. A este efecto de las variaciones en el tiempo de tránsito se le llama *jitter* y es uno de los principales problemas para las aplicaciones de tiempo-real. Otro importante problema es la pérdida de paquetes. El TCP evita que se pierdan paquetes, pero no es un protocolo válido para tiempo-real porque al realizar una retransmisión se está introduciendo un retardo inviable para este tipo de aplicaciones. Además, en las aplicaciones de tiempo-real (principalmente audio y vídeo), no es importante la pérdida de

paquetes aislados, aunque si lo es la pérdida masiva de paquetes. Existe una clase de aplicaciones de este tipo llamadas *aplicaciones de reproducción* que intentan resolver este problema realizando una precarga de los datos, de forma que se empieza a reproducir cuando ya se tienen suficientes para que nunca se quede vacío el buffer de entrada. Estas aplicaciones se adaptan a los retardos variables y por tanto funcionan bien en redes de datagrama moderadamente cargadas, pudiendo soportar el *jitter* causado por pequeñas ráfagas e incluso tolerar ocasionales pérdidas de paquetes durante periodos breves de congestión. La reproducción de video a través de internet se vislumbra como una actividad común en un futuro a medio plazo. Estando hoy disponible multitud de contenido multimedia para reproducción online.

Sin embargo, hay zonas en Internet que están a menudo muy congestionadas, debido a la gran cantidad de tráfico que tienen que soportar. El precio pagado por compartir el ancho de banda es la congestión, que genera *jitter* y alta tasa de pérdida de paquetes. Aunque el tráfico de tiempo-real contribuye en gran medida a la congestión debido a sus requerimientos de ancho de banda, es también el más perjudicado por ésta ya que los datos se convierten en obsoletos si no llegan a tiempo por lo que hay que desecharlos mientras que el único efecto que produce en el tráfico de no tiempo-real es que una transferencia tarda más tiempo en realizarse. Como consecuencia, las aplicaciones de tiempo-real entregan peor calidad durante los periodos de congestión.

### **2.2.3 ELEVADO TIEMPO DE TRÁNSITO**

El problema subyacente es que diferentes tipos de aplicaciones necesitan diferentes servicios. Por ejemplo, una transferencia de ficheros requiere que una cantidad de datos sea transferida en un tiempo aceptable mientras que la telefonía IP necesita que la mayor parte de los paquetes lleguen al receptor en menos de 300 milisegundos. Si hay bastante ancho de banda disponible, el servicio básico por el que se rigen las redes IP actuales (mejor-

esfuerzo o *best-efford*) es suficiente para ambos tipos de datos. Sin embargo, cuando los recursos escasean, el tráfico de tiempo-real debe ser tratado de forma diferente.

Es muy común comprobar el efecto negativo del tiempo de tránsito en conversaciones tanto de señales de control como de videoconferencia. Como ejemplo primero baste citar la telecirugía: Para realizar una intervención quirúrgica a distancia, el doctor se encuentra en un sitio y el quirófano en otro. El doctor actúa sobre lo que está viendo, si hay un retardo, puede que no aplique el bisturí en el sitio correcto porque el órgano intervenido se haya movido en ese tiempo. Además es una comunicación doble, el médico tiene que actuar sobre lo que está viendo en un monitor así que mientras la señal llega del quirófano al doctor y retorna la actuación sobre el robot, el tiempo total transcurrido es la suma de los 2 tiempos de tránsito. Un ejemplo de este fenómeno lo podemos ver en cualquier videoconferencia vía satélite de los informativos de TV con sus corresponsales en otros continentes. El efecto es que hay que esperar un tiempo hasta que la otra parte contesta.

### **2.3 LA SOLUCIÓN ES LA RESERVA DE RECURSOS**

Está claro que los problemas son provocados por la congestión en algún punto del camino, así que una solución sería reservar un pequeño ancho de banda en los encaminadores intermedios para el envío de señal en tiempo real. De esta manera en caso de congestión, la comunicación se realiza en un tiempo y con una variación de retardo mínimos. El grupo de trabajo de Servicios Integrados (*Integrated Services o IS*) del IETF (Internet Engineering Task Force) ha desarrollado un servicio de Internet mejorado que incluye el servicio mejor-esfuerzo y el servicio de tiempo-real. Junto con el protocolo de reserva de recursos (RSVP) y otros protocolos adaptados a la transmisión de datos de tiempo real, esta arquitectura proporciona una aproximación válida que permite que las

aplicaciones usen el tipo de servicio que necesitan y con la calidad que eligen. Los Servicios Integrados se van a usar de la siguiente forma: si usando el servicio de mejor esfuerzo no se obtienen resultados satisfactorios, el usuario puede realizar una reserva de recursos. Si hay suficiente ancho de banda reservable en la ruta, la reserva es aceptada. En otro caso, se rechaza, lo que equivaldría a una señal de ocupado, y el usuario seguiría con el servicio de mejor-esfuerzo. Dependiendo del tipo de servicio elegido, una aplicación de tiempo-real obtendrá un ancho de banda garantizado con un retardo limitado o un servicio que, durante las etapas de congestión, proporciona una calidad tan buena como si la red sólo estuviese ligeramente cargada. Para poder realizar las reservas, los encaminadores presentes en el camino de los datos deben poder distinguir entre flujos de datos normales (o mejor-esfuerzo) y flujos de datos reservados. Un *flujo* es una sucesión de paquetes relacionados que van desde una fuente a uno o múltiples destinos. Los encaminadores utilizan sofisticadas técnicas para proporcionar el servicio de acuerdo a las reservas y a otras consideraciones. Sin embargo, no todo son ventajas en este modelo, ya que el establecimiento y gestión de las reservas consume ancho de banda y potencia de proceso lo que puede causar problemas de escalado. Este es uno de los motivos por el cual el protocolo RSVP es un protocolo aún sin terminar de especificar.

## **2.4 NECESIDAD DEL ENCAMINADOR**

El protocolo RSVP necesita de un router que lo soporte específicamente ya que no se puede implementar sobre un encaminador existente sin tener que modificar el código del mismo. Es por ello por lo que se necesita crear un router QoS desde cero. Para hacer que se pueda extender rápidamente este tipo de encaminadores, se elige la plataforma PC ya que es la más usada en la mayoría de las redes. Este tipo de plataformas usan mayormente Windows como sistema operativo, aunque hay un creciente número de ellas que son

gestionadas por Linux. Como la mayor parte de la flota de PC usa Windows, el router de estar diseñado para esta plataforma.

Con esto se consigue que este tipo de encaminadores se extienda rápidamente, vital para la expansión del soporte RSVP por el mundo. Este tipo de encaminadores realizan muchas tareas con lo que es necesario que sea rápido y que funcione óptimamente en ordenadores desde el más lento al más actual, ofreciendo las mejores prestaciones en cualquier escenario.

## **2.5 EL LENGUAJE DE PROGRAMACIÓN**

Para la implementación del encaminador se necesitará una herramienta de programación lo suficientemente potente como para optimizar el código para obtener la mayor velocidad posible. Debido a este requerimiento se optó por el lenguaje C que sin duda es el que ofrece el código más optimizado en cuanto a velocidad.

No obstante este proyecto puede servir en un futuro para nuevos e interesantes proyectos así que se optará por un lenguaje orientado a objetos como es el C++. De esta manera el código queda mucho más encapsulado y fácil de comprender, así mismo es sencillo reutilizar parte de los objetos en posteriores proyectos sin necesidad de adaptación, y en todo caso serían mínimas.

La elección del compilador tiene que ver con la velocidad del código generado y con la simplicidad del uso de los objetos. En este punto había 2 opciones MS Visual C++ 6.0 o Borland C++ Builder 5. Se eligió el segundo ya que la creación y uso de los objetos es mucho más simple y fácil de entender que en la herramienta de C++ de Microsoft, y

además el entorno de usuario también es mucho más sencillo de crear y modificar en C++ Builder 5, así que en este punto ya tenemos una herramienta de programación para Windows, que genera un código rápido y eficiente y con una organización sencilla del código para poder ser usado posteriormente con un coste de tiempo mínimo: Borland C++ Builder 5 Enterprise.

## 2.6 CONCLUSIÓN

En este capítulo hemos visto los problemas que tiene el protocolo de red IP que usa Internet para poder transportar información en tiempo real en todos los escenarios, más concretamente en situaciones de saturación o elevada carga en algún punto del camino de datos. Para solucionar los 3 grandes problemas que sufre este tipo de tráfico se decide reservar ancho de banda en el camino de datos con lo cual se elimina el efecto de la carga de los encaminadores sobre este tipo de tráfico. Otro problema que se presenta es la elección de una herramienta para programar el router, eligiendo el Borland C++ Builder 5 frente a otras alternativas siguiendo el razonamiento expuesto. Para dar soporte a la calidad de servicio se hará uso del protocolo RSVP que junto con el resto de protocolos usados en el encaminador, será el objeto de estudio del siguiente capítulo.

## **CAPÍTULO 3: Protocolos Usados**

### **3.1 INTRODUCCIÓN**

En el estudio del problema veíamos cómo la red Internet está construida usando el protocolo de red IP pero hay más protocolos que se van apilando de forma que los de capas superiores aprovechan los servicios de los protocolos de capas inferiores. Durante una transmisión cada protocolo se comunica con su homónimo del otro extremo sin preocuparse de los protocolos de otras capas. IP a su vez también tiene protocolos de capas interiores con los que también tiene que interactuar.

Una de las decisiones más importantes que debemos tomar a la hora de diseñar una red es elegir un protocolo de la capa de acceso al medio y otro de las capas de red y transporte. A continuación estudiamos los distintos protocolos. Adelantamos, no obstante, que la combinación más interesante para redes locales nuevas es Ethernet + TCP/IP. Los siguientes protocolos son los que se han necesitado o usa el router en alguna parte de su código. Se detallan los mismos desde la capa más baja hasta la más alta.

## **3.2 PROTOCOLOS DE LA CAPA DE ACCESO AL MEDIO**

En la capa de acceso al medio se determina la forma en que los puestos de la red envían y reciben datos sobre el medio físico. Se responden preguntas del tipo: ¿puede un puesto dejar información en el cable siempre que tenga algo que transmitir?, ¿debe esperar algún turno?, ¿cómo sabe un puesto que un mensaje es para él?

### **3.2.1 EL ESTANDAR 802**

Un organismo de normalización conocido como IEEE (Instituto de Ingenieros Eléctricos y Electrónicos) ha definido los principales protocolos de la capa de acceso al medio, conocidos en conjunto como estándares 802. Los más importantes son los IEEE 802.3 y IEEE 802.5 que se estudian a continuación.

El estándar 802.1 es una introducción al conjunto de estándares y define algunos aspectos comunes. El estándar 802.2 describe la parte superior de la capa de enlace de datos del modelo OSI (entre la capa de acceso al medio y la capa de red) que puede proporcionar control de errores y control de flujo al resto de estándares 802 utilizando el protocolo LLC (Logical Link Control, control lógico de enlace). Las normas 802.3 a 802.5 definen protocolos para redes LAN. El estándar 802.4 que no se va a estudiar por su escasa implantación se conoce como Token Bus (bus con paso de testigo). Finalmente, 802.6 es un estándar adecuado para utilizarse en redes MAN. Se trata de DQDB (*Distributed Queue Dual Bus*, bus doble de colas distribuidas).

El protocolo utilizado en esta capa viene determinado por las tarjetas de red que instalemos en los puestos. Esto quiere decir que si adquirimos tarjetas Ethernet sólo podremos instalar redes Ethernet. Y que para instalar redes *Token ring* necesitaremos tarjetas de red especiales para *Token ring*. Actualmente en el mercado únicamente se comercializan tarjetas de red Ethernet (de distintas velocidades y para distintos cableados).

### **3.2.1.1 Token ring (802.5)**

Las redes *Token ring* (paso de testigo en anillo) fueron utilizadas ampliamente en entornos IBM desde su lanzamiento en el año 1985. En la actualidad es difícil encontrarlas salvo en instalaciones antiguas de grandes empresas.

El cableado se establece según una topología de anillo. En lugar de utilizar difusiones, se utilizan enlaces punto a punto entre cada puesto y el siguiente del anillo. Por el anillo *Token ring* circula un mensaje conocido como *token* o ficha. Cuando una estación desea transmitir espera a recibir el *token*. En ese momento, lo retira de circulación y envía su

mensaje. Este mensaje circula por el anillo hasta que lo recibe íntegramente el destinatario. Entonces se genera un *token* nuevo.

Las redes *Token ring* utilizan una estación monitor para supervisar el funcionamiento del anillo. Se trata de un protocolo complejo que debe monitorizar en todo momento el buen funcionamiento del *token* (que exista exactamente uno cuando no se transmiten datos) y sacar del anillo las tramas defectuosas que no tengan destinatario, entre otras funciones.

Las redes *Token ring* de IBM pueden funcionar a 4 Mbps o a 16 Mbps utilizando cable par trenzado o cable coaxial.

### 3.2.1.2 Ethernet (802.3)

Las redes Ethernet son actualmente las únicas que tienen interés para entornos LAN. El estándar 802.3 fue diseñado originalmente para funcionar a 10 Mbps, aunque posteriormente ha sido perfeccionado para trabajar a 100 Mbps (802.3u) o 1 Gbps.

Una red Ethernet tiene las siguientes características:

- *Canal único*. Todas las estaciones comparten el mismo canal de comunicación por lo que sólo una puede utilizarlo en cada momento.
- Es de *difusión* debido a que todas las transmisiones llegan a todas las estaciones (aunque sólo su destinatario aceptará el mensaje, el resto lo descartarán).
- Tiene un *control de acceso distribuido* porque no existe una autoridad central que garantice los accesos. Es decir, no hay ninguna estación que supervise y asigne los turnos al resto de estaciones. Todas las estaciones tienen la misma prioridad para transmitir.

### 3.2.1.3 Ethernet vs Token Ring

En Ethernet cualquier estación puede transmitir siempre que el cable se encuentre libre; en *Token ring* cada estación tiene que esperar su turno. Ethernet utiliza un canal único de difusión; *Token ring* utiliza enlaces punto a punto entre cada estación y la siguiente. *Token ring* tiene siempre una estación monitor que supervisa el buen funcionamiento de la red; en Ethernet ninguna estación tiene mayor autoridad que otra. Según esta comparación, la conclusión más evidente es que, a iguales velocidades de transmisión, *Token ring* se comportará mejor en entornos de alta carga y Ethernet, en redes con poco tráfico.

En las redes Ethernet, cuando una estación envía un mensaje a otra, no recibe ninguna confirmación de que la estación destino haya recibido su mensaje. Una estación puede estar enviando paquetes Ethernet a otra que está desconectada y no advertirá que los paquetes se están perdiendo. Las capas superiores (y más concretamente, TCP) son las encargadas de asegurarse que la transmisión se ha realizado de forma correcta.

El protocolo de comunicación que utilizan estas redes es el CSMA/CD (*Carrier Sense Multiple Access / Collision Detect*, acceso múltiple con detección de portadora y detección de colisiones). Esta técnica de control de acceso a la red ha sido normalizada constituyendo el estándar IEEE 802.3. Veamos brevemente el funcionamiento de CSMA/CD:

Cuando una estación quiere transmitir, primero escucha el canal (detección de portadora). Si está libre, transmite; pero si está ocupado, espera un tiempo y vuelve a intentarlo. Sin embargo, una vez que una estación ha decidido comenzar la transmisión puede darse el caso de que otra estación haya tomado la misma decisión, basándose en que el canal estaba libre cuando ambas lo comprobaron. Debido a los retardos de propagación en el cable, ambas señales colisionarán y no se podrá completar la transmisión de ninguna de las dos estaciones. Las estaciones que están transmitiendo lo advertirán (detección de colisiones) e interrumpirán inmediatamente la transmisión. Después esperarán un tiempo

aleatorio y volverán a intentarlo. Si se produce una nueva colisión, esperarán el doble del tiempo anterior y lo intentarán de nuevo. De esta manera, se va reduciendo la probabilidad de nuevas colisiones.

Debemos recordar que el canal es único y por lo tanto todas las estaciones tienen que compartirlo. Sólo puede estar una estación transmitiendo en cada momento, sin embargo pueden estar recibiendo el mensaje más de una.

**Nota:** La existencia de colisiones en una red no indica que exista un mal funcionamiento. Las colisiones están definidas dentro del protocolo Ethernet y no deben ser consideradas como una situación anómala. Sin embargo, cuando se produce una colisión, el canal se desaprovecha porque ninguna estación logra transmitir en ese momento. Debemos tratar de reducir el número de colisiones que se producen en una red. Esto se consigue separando grupos de ordenadores mediante un switch o un router. Podemos averiguar las colisiones que se producen en una red observando el correspondiente LED de nuestro hub.

### 3.2.2 DIRECCIONES FÍSICAS

¿Cómo sabe una estación que un mensaje es para ella? Está claro, que hay que distinguir unas estaciones de otras utilizando algún identificador. Esto es lo que se conoce como *direcciones físicas*.

Los adaptadores Ethernet tienen asignada una dirección de 48 bits de fábrica que no se puede variar. Los fabricantes nos garantizan que no puede haber dos tarjetas de red con la misma dirección física. Si esto llegase a ocurrir dentro de una misma red la comunicación

se volvería imposible. Los tres primeros bytes corresponden al fabricante (no puede haber dos fabricantes con el mismo identificador) y los tres últimos al número de serie (no puede haber dos tarjetas del mismo fabricante con el mismo número de serie). Por ejemplo,

5D:1E:23:10:9F:A3

Los bytes 5D:1E:23 identifican al fabricante y los bytes 10:9F:A3 al número de serie del fabricante 5D:1E:23

**Nota:** Los comandos *ipconfig / all /more* y *winipcfg* muestran la dirección física de nuestra tarjeta de red Ethernet. Obsérvese que estos comandos pueden recoger también información relativa al adaptador virtual "PPP Adapter" (se corresponde con el módem o adaptador RDSI) además de la referente a la tarjeta de red real.

No todas las direcciones representan a máquinas aisladas, algunas de ellas se utilizan para enviar mensajes de multidifusión. Esto es, enviar un mensaje a varias máquinas a la vez o a todas las máquinas de la red. Ethernet permite que el mismo mensaje pueda ser escuchado por más de una máquina a la vez.

### 3.2.3 FORMATO DE LA TRAMA

La comunicación entre una estación y otra a través de una red Ethernet se realiza enviando tramas Ethernet [MS3com]. El mensaje que se quiere transmitir se descompone en una o más tramas con el siguiente formato:

8 bytes	6 bytes	6 bytes	2 bytes	64-1500 bytes	4 bytes
Preámbulo	Dirección física destino	Dirección física origen	Tipo de trama	Datos de la trama	CRC

Las *direcciones origen* y *destino* son las direcciones físicas de los adaptadores de red de cada ordenador. El campo *Tipo de trama* indica el formato de los datos que se transfieren en el campo *Datos de la trama*. Por ejemplo, para un datagrama IP se utiliza el valor hexadecimal de 0800 y para un mensaje ARP el valor 0806. Todos los mensajes (*datagramas*) que se envíen en la capa siguiente irán encapsulados en una o más tramas Ethernet utilizando el campo *Datos de la trama*. Y esto mismo es aplicable para cualquier otro tipo de red distinta a Ethernet. Como norma general, cada mensaje que transmite una capa se coloca en el campo datos de la capa anterior. Aunque es muy frecuente que el mensaje no quepa en una sola trama y se utilicen varias.

### 3.2.4 VELOCIDADES

Ethernet puede funcionar a tres velocidades: 10 Mbps, 100 Mbps (*FastEthernet*) y 1 Gbps (1000 Mbps). 10 Mbps es la velocidad para la que se diseñó originalmente el estándar Ethernet. Sin embargo, esta velocidad se ha mejorado para adaptarse a las crecientes exigencias de las redes locales. La velocidad de 100 Mbps es actualmente la más utilizada en la empresa. Las redes a 1 Gbps están comenzando a ver la luz en estos momentos por lo que tardarán un tiempo en implantarse en el mercado (los precios son todavía muy altos).

Para crear una red que trabaje a 10 Mbps es suficiente con utilizar cable coaxial o bien, cable par trenzado de categoría 3 o superior. Sin embargo, es recomendable utilizar cables par trenzado de categoría 5 y concentradores con velocidades mixtas 10/100 Mbps. De esta forma, en un futuro se podrán ir cambiando gradualmente los adaptadores de 10 Mbps por unos de 100 Mbps sin necesidad de instalar nuevo cableado.

La mejor opción actualmente para redes nuevas es *FastEthernet*. Para conseguir velocidades de 100 Mbps es necesario utilizar cable par trenzado con una categoría mínima de 5, un concentrador que soporte esta velocidad y tarjetas de red de 100 Mbps. Generalmente, los cables sin malla UTP (*Unshielded Twisted Pair*) cumplen bien con su función pero en situaciones concretas que requieran el máximo rendimiento de la red o existan muchas interferencias, puede ser necesario un cableado mallado STP (*Shielded Twister Pair*).

### 3.2.5 TIPOS DE ADAPTADORES

La siguiente tabla resume los principales tipos de adaptadores Ethernet en función del cableado y la velocidad de la red. (T se utiliza para par trenzado, F para fibra óptica y X para *FastEthernet*).

	<b>10Base5</b>	<b>10Base2</b>	<b>10BaseT</b>	<b>10BaseFP</b>	<b>100BaseTX</b>	<b>100BaseFX</b>
<b>Cableado</b>	Coaxial		Par trenzado	Par de fibra óptica	Par trenzado	2 fibras ópticas
<b>Velocidad</b>	10 Mbps				100 Mbps	
<b>Topología</b>	Bus		Estrella			
<b>Longitud máxima segmento</b>	500 m	185 m	100 m	500 m	100 m	100 m
<b>Nodos por segmento</b>	100	30	2 (un extremo es el hub y el otro el ordenador)			

Los adaptadores pueden ser compatibles con varios de los estándares anteriores dando lugar a numerosas combinaciones. Sin embargo, lo habitual es encontrar en el mercado tarjetas de red de tan sólo estos dos tipos:

- *Tarjetas de red combo.* Tienen 2 conectores, uno para cable coaxial y otro para RJ45. Su velocidad máxima es de 10 Mbps por lo que soportan 10Base2 y 10BaseT. La tarjeta de red RTL8029 del fabricante Realtek pertenece a este tipo. Este grupo de tarjetas de red tienden a desaparecer (al igual que el cable coaxial).
- *Tarjetas de red 10/100.* Tienen sólo conector para RJ45. Se adaptan a la velocidad de la red (10 Mbps o 100 Mbps). Son compatibles con 10BaseT y 100BaseT. Como ejemplos de este tipo se encuentran las tarjetas Realtek RTL8139 y 3COM 3C905.

## 3.3 PROTOCOLO IP4

### 3.3.1 INTRODUCCION

El protocolo IP4 [RFC791] es el software que implementa el mecanismo de entrega de paquetes sin conexión y no confiable (técnica del mejor esfuerzo). El protocolo IP cubre tres aspectos importantes:

- Define la unidad básica para la transferencia de datos en una inter-red, especificando el formato exacto de un Datagrama IP.
- Realiza las funciones de enrutamiento.
- Define las reglas para que los Host y Routers procesen paquetes, los descarten o generen mensajes de error.

### 3.3.2 EL DATAGRAMA IP

El esquema de envío de IP es similar al que se emplea en la capa Acceso a red. En esta última se envían Tramas formadas por un Encabezado y los Datos. En el Encabezado se incluye la dirección física del origen y del destino. En el caso de IP se envían *Datagramas*, estos también incluyen un Encabezado y Datos, pero las direcciones empleadas son *Direcciones IP*.



### 3.3.3 FORMATO DEL DATAGRAMA IP

El datagrama IP es la unidad básica de transferencia de datos entre el origen y el destino. Viaja en el campo de datos de las tramas físicas (recuérdese la trama Ethernet) de las distintas redes que va atravesando. Cada vez que un datagrama tiene que atravesar un router, el datagrama *saldrá* de la trama física de la red que abandona y se *acomodará* en el campo de datos de una trama física de la siguiente red. Este mecanismo permite que un mismo datagrama IP pueda atravesar redes distintas: enlaces punto a punto, redes ATM, redes Ethernet, redes *Token Ring*, etc. El propio datagrama IP tiene también un campo de datos: será aquí donde viajen los paquetes de las capas superiores.

0				10								20								30											
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
VERS				HLEN				Tipo de servicio								Longitud total															
Identificación																Bandrs				Desplazamiento de fragmento											
TTL				Protocolo								CRC cabecera																			
Dirección IP origen																															
Dirección IP destino																															
Opciones IP (si las hay)																								Relleno							
Datos																															
...																															

Campos del datagrama IP:

- **VERS** (4 bits). Indica la versión del protocolo IP que se utilizó para crear el datagrama. Actualmente se utiliza la versión 4 (IPv4) aunque ya existe la siguiente versión, la 6 (IPv6).
- **HLEN** (4 bits). Longitud de la cabecera expresada en múltiplos de 32 bits. El valor mínimo es 5, correspondiente a 160 bits = 20 bytes.
- **Tipo de servicio** (*Type Of Service*). Los 8 bits de este campo se dividen a su vez en:
  - **Prioridad** (3 bits). Un valor de 0 indica baja prioridad y un valor de 7, prioridad máxima.
  - Los siguientes tres bits indican cómo se prefiere que se transmita el mensaje, es decir, son sugerencias a los encaminadores que se encuentren a su paso los cuales pueden tenerlas en cuenta o no.
  - **Bit D** (*Delay*). Solicita retardos cortos (enviar rápido).
  - **Bit T** (*Throughput*). Solicita un alto rendimiento (enviar mucho en el menor tiempo posible).
  - **Bit R** (*Reliability*). Solicita que se minimice la probabilidad de que el datagrama se pierda o resulte dañado (enviar bien).
  - Los siguientes dos bits no tienen uso.

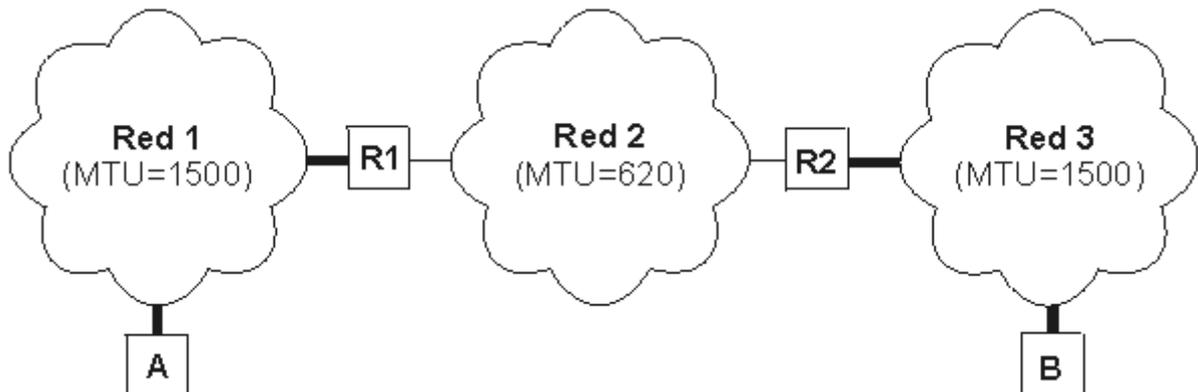
- **Longitud total** (16 bits). Indica la longitud total del datagrama expresada en bytes. Como el campo tiene 16 bits, la máxima longitud posible de un datagrama será de 65535 bytes.
- **Identificación** (16 bits). Número de secuencia que junto a la dirección origen, dirección destino y el protocolo utilizado identifica de manera única un datagrama en toda la red. Si se trata de un datagrama fragmentado, llevará la misma identificación que el resto de fragmentos.
- **Banderas** o indicadores (3 bits). Sólo 2 bits de los 3 bits disponibles están actualmente utilizados. El bit de *Más fragmentos* (**MF**) indica que no es el último datagrama. Y el bit de *No fragmentar* (**NF**) prohíbe la fragmentación del datagrama. Si este bit está activado y en una determinada red se requiere fragmentar el datagrama, éste no se podrá transmitir y se descartará.
- **Desplazamiento de fragmentación** (13 bits). Indica el lugar en el cual se insertará el fragmento actual dentro del datagrama completo, medido en unidades de 64 bits. Por esta razón los campos de datos de todos los fragmentos menos el último tienen una longitud múltiplo de 64 bits. Si el paquete no está fragmentado, este campo tiene el valor de cero.
- **Tiempo de vida** o TTL (8 bits). Número máximo de segundos que puede estar un datagrama en la red. Cada vez que el datagrama atraviesa un router se resta 1 a este número. Cuando llegue a cero, el datagrama se descarta y se devuelve un mensaje ICMP de tipo "tiempo excedido" para informar al origen de la incidencia.
- **Protocolo** (8 bits). Indica el protocolo utilizado en el campo de datos: 1 para ICMP, 2 para IGMP, 6 para TCP y 17 para UDP.
- **CRC cabecera** (16 bits). Contiene la suma de comprobación de errores sólo para la cabecera del datagrama. La verificación de errores de los datos corresponde a las capas superiores.
- **Dirección origen** (32 bits). Contiene la dirección IP del origen.
- **Dirección destino** (32 bits). Contiene la dirección IP del destino.

- **Opciones IP.** Este campo no es obligatorio y especifica las distintas opciones solicitadas por el usuario que envía los datos (generalmente para pruebas de red y depuración). Existen hasta 40 bytes extra en la cabecera del Datagrama IP que pueden llevar una o más opciones. Su uso es bastante raro.
  - Uso de Ruta Estricta (Camino Obligatorio)
  - Ruta de Origen Desconectada (Nodos Obligatorios)
  - Crear registro de Ruta
  - Marcas de Tiempo
  - Seguridad Básica del Departamento de Defensa
  - Seguridad Extendida del Departamento de Defensa
- **Relleno.** Si las opciones IP (en caso de existir) no ocupan un múltiplo de 32 bits, se completa con bits adicionales hasta alcanzar el siguiente múltiplo de 32 bits (recuérdese que la longitud de la cabecera tiene que ser múltiplo de 32 bits).

### 3.3.4 FRAGMENTACIÓN

Ya hemos visto que las tramas físicas tienen un campo de datos y que es aquí donde se transportan los datagramas IP. Sin embargo, este campo de datos no puede tener una longitud indefinida debido a que está limitado por el diseño de la red. El MTU (*Maximum Transfer Unit*) de una red es la mayor cantidad de datos que puede transportar su trama física. El MTU de las redes Ethernet es 1500 bytes y el de las redes *Token-Ring*, 8192 bytes. Esto significa que una red Ethernet nunca podrá transportar un datagrama de más de 1500 bytes sin fragmentarlo.

Un encaminador (*router*) fragmenta un datagrama en varios si el siguiente tramo de la red por el que tiene que viajar el datagrama tiene un MTU inferior a la longitud del datagrama. Veamos con el siguiente ejemplo cómo se produce la fragmentación de un datagrama.



Supongamos que el host A envía un datagrama de 1400 bytes de datos (1420 bytes en total) al host B. El datagrama no tiene ningún problema en atravesar la red 1 ya que  $1420 < 1500$ . Sin embargo, no es capaz de atravesar la red 2 ( $1420 \geq 620$ ). El router R1 fragmenta el datagrama en el menor número de fragmentos posibles que sean capaces de atravesar la red 2. Cada uno de estos fragmentos es un nuevo datagrama con la misma *Identificación* pero distinta información en el campo de *Desplazamiento de fragmentación* y el bit de *Más fragmentos (MF)*. Veamos el resultado de la fragmentación:

- **Fragmento 1:** Long. total = 620 bytes; Desp = 0; MF=1 (contiene los primeros 600 bytes de los datos del datagrama original)
- **Fragmento 2:** Long. total = 620 bytes; Desp = 600; MF=1 (contiene los siguientes 600 bytes de los datos del datagrama original)
- **Fragmento 3:** Long. total = 220 bytes; Desp = 1200; MF=0 (contiene los últimos 200 bytes de los datos del datagrama original)

El router R2 recibirá los 3 datagramas IP (fragmentos) y los enviará a la red 3 sin reensamblarlos. Cuando el host B reciba los fragmentos, recompondrá el datagrama original. Los encaminadores intermedios no reensamblan los fragmentos debido a que esto supondría una carga de trabajo adicional, aparte de memorias temporales. Nótese que el ordenador destino puede recibir los fragmentos cambiados de orden pero esto no supondrá

ningún problema para el reensamblado del datagrama original puesto que cada fragmento guarda suficiente información.

Si el datagrama del ejemplo hubiera tenido su bit *No fragmentar (NF)* a 1, no hubiera conseguido atravesar el router R1 y, por tanto, no tendría forma de llegar hasta el host B. El encaminador R1 descartaría el datagrama.

### 3.3.5 DIRECCIONES IP

La dirección IP es el identificador de cada host dentro de su red de redes. Cada host conectado a una red tiene una dirección IP asignada, la cual debe ser distinta a todas las demás direcciones que estén vigentes en ese momento en el conjunto de redes visibles por el host. En el caso de Internet, no puede haber dos ordenadores con 2 direcciones IP (públicas) iguales. Pero sí podríamos tener dos ordenadores con la misma dirección IP siempre y cuando pertenezcan a redes independientes entre sí (sin ningún camino posible que las comunique).

Las direcciones IP se clasifican en:

- **Direcciones IP públicas.** Son visibles en todo Internet. Un ordenador con una IP pública es accesible (visible) desde cualquier otro ordenador conectado a Internet. Para conectarse a Internet es necesario tener una dirección IP pública.
- **Direcciones IP privadas (reservadas).** Son visibles únicamente por otros hosts de su propia red o de otras redes privadas interconectadas por routers. Se utilizan en las empresas para los puestos de trabajo. Los ordenadores con direcciones IP privadas pueden salir a Internet por medio de un router (o proxy) que tenga una IP pública. Sin embargo, desde Internet no se puede acceder a ordenadores con direcciones IP privadas.

A su vez, las direcciones IP pueden ser:

- **Direcciones IP estáticas (fijas).** Un host que se conecte a la red con dirección IP estática siempre lo hará con una misma IP. Las direcciones IP públicas estáticas son las que utilizan los servidores de Internet con objeto de que estén siempre localizables por los usuarios de Internet. Estas direcciones hay que contratarlas.
- **Direcciones IP dinámicas.** Un host que se conecte a la red mediante dirección IP dinámica, cada vez lo hará con una dirección IP distinta. Las direcciones IP públicas dinámicas son las que se utilizan en las conexiones a Internet mediante un módem. Los proveedores de Internet utilizan direcciones IP dinámicas debido a que tienen más clientes que direcciones IP (es muy improbable que todos se conecten a la vez).

Las direcciones IP están formadas por 4 bytes (32 bits). Se suelen representar de la forma a.b.c.d donde cada una de estas letras es un número comprendido entre el 0 y el 255. Por ejemplo la dirección IP del servidor de IBM (www.ibm.com) es 129.42.18.99.

Las direcciones IP también se pueden representar en hexadecimal, desde la 00.00.00.00 hasta la FF.FF.FF.FF o en binario, desde la 00000000.00000000.00000000.00000000 hasta la 11111111.11111111.11111111.11111111.

Las tres direcciones siguientes representan a la misma máquina (podemos utilizar la calculadora científica de Windows para realizar las conversiones).

- (decimal) 128.10.2.30
- (hexadecimal) 80.0A.02.1E
- (binario) 10000000.00001010.00000010.00011110

¿Cuántas direcciones IP existen? Si calculamos 2 elevado a 32 obtenemos más de 4000 millones de direcciones distintas. Sin embargo, no todas las direcciones son válidas para asignarlas a hosts. Las direcciones IP no se encuentran aisladas en Internet, sino que pertenecen siempre a alguna red. Todas las máquinas conectadas a una misma red se

caracterizan en que los primeros bits de sus direcciones son iguales. De esta forma, las direcciones se dividen conceptualmente en dos partes: el *identificador de red* y el *identificador de host*.

Dependiendo del número de hosts que se necesiten para cada red, las direcciones de Internet se han dividido en tres **clases primarias**:

	0	1	2	3	8	16	24	31	
Clase A	0	red				host			
Clase B	1	0	red				host		
Clase C	1	1	0	red				host	

Clase	Formato (r=red, h=host)	Número de redes (aprox.)	Número de hosts por red (aprox.)	Rango de direcciones de redes	Máscara de subred
<b>A</b>	r.h.h.h	128	16 millones	0.0.0.0 - 127.0.0.0	255.0.0.0
<b>B</b>	r.r.h.h	16000	65000	128.0.0.0 - 191.255.0.0	255.255.0.0
<b>C</b>	r.r.r.h	2 millones	254	192.0.0.0 - 223.255.255.0	255.255.255.0

### 3.3.6 DIRECCIONES IP ESPECIALES Y RESERVADAS

No todas las direcciones comprendidas entre la 0.0.0.0 y la 223.255.255.255 son válidas para un host: algunas de ellas tienen significados especiales. Las principales direcciones especiales se resumen en la siguiente tabla. Su interpretación depende del host desde el que se utilicen.

Bits de red	Bits de host	Significado	Ejemplo
todos 0		Mi propio host	0.0.0.0
todos 0	host	Host indicado dentro de mi red	0.0.0.10
red	todos 0	Red indicada	192.168.1.0
todos 1		Multidifusión a mi red	255.255.255.255
red	todos 1	Multidifusión a la red indicada	192.168.1.255
127	cualquier valor válido de host	Loopback (mi propio host)	127.0.0.1

Multidifusión o *broadcasting* es el envío de un mensaje a todos los ordenadores que se encuentran en una red. La dirección de *loopback* (normalmente 127.0.0.1) se utiliza para comprobar que los protocolos TCP/IP están correctamente instalados en nuestro propio ordenador. Lo veremos más adelante, al estudiar el comando PING.

Las direcciones de redes siguientes se encuentran reservadas para su uso en redes privadas (intranets):

Clase	Rango de direcciones reservadas de redes
<b>A</b>	10.0.0.0
<b>B</b>	172.16.0.0 - 172.31.0.0
<b>C</b>	192.168.0.0 - 192.168.255.0

Por ejemplo, si estamos construyendo una red privada con un número de ordenadores no superior a 254 podemos utilizar una red reservada de clase C. Al primer ordenador le podemos asignar la dirección 192.168.23.1, al segundo 192.168.23.2 y así sucesivamente hasta la 192.168.23.254. Como estamos utilizando direcciones reservadas, tenemos la garantía de que no habrá ninguna máquina conectada directamente a Internet con alguna de nuestras direcciones. De esta manera, no se producirán conflictos y desde cualquiera de nuestros ordenadores podremos acceder a la totalidad de los servidores de Internet (si

utilizásemos en un ordenador de nuestra red una dirección de un servidor de Internet, nunca podríamos acceder a ese servidor).

### 3.3.7 ENCAMINAMIENTO IP

Una *red de redes* está formada por redes interconectadas mediante routers o encaminadores. Cuando enviamos un datagrama desde un ordenador hasta otro, éste tiene que ser capaz de encontrar la ruta más adecuada para llegar a su destino. Esto es lo que se conoce como *encaminamiento*.

Los *routers* (encaminadores) son los encargados de elegir las mejores rutas. Estas máquinas pueden ser ordenadores con varias direcciones IP o bien, aparatos específicos. Los routers deben conocer, al menos parcialmente, la estructura de la red que les permita encaminar de forma correcta cada mensaje hacia su destino. Esta información se almacena en las llamadas *tablas de encaminamiento*. Observemos que debido al sistema de direccionamiento IP esta misión no es tan complicada. Lo único que necesitamos almacenar en las tablas son los prefijos de las direcciones (que nos indican la red). Por ejemplo, si el destino es la máquina 149.33.19.4 con máscara 255.255.0.0, nos basta con conocer el encaminamiento de la red 149.33.0.0 ya que todas las que empiecen por 149.33 se enviarán hacia el mismo sitio.

En general se puede dividir el enrutamiento en ***Entrega Directa*** y ***Entrega Indirecta***. La Entrega Directa es la transmisión de un Datagrama de una máquina a otra dentro de la misma red física. La Entrega Indirecta ocurre cuando el destino no está en la red local, lo que obliga al Host a enviar el Datagrama a algún Router intermedio. Es necesario el uso de máscaras de subred para saber si el Host destino de un Datagrama está o no dentro de la misma red física.

### 3.3.8 ENCAMINAMIENTO CON SALTO AL SIGUIENTE.

La forma más común de enrutamiento requiere el uso de una *Tabla de Enrutamiento IP*, presente tanto en los Host como en los Routers. Estas tablas no pueden tener información sobre cada posible destino, de hecho, esto no es deseable. En vez de ello, se aprovecha el esquema de direccionamiento IP para ocultar detalles acerca de los Host individuales, además, las tablas no contienen rutas completas, sino solo la dirección del siguiente paso en esa ruta.

En general una tabla de encaminamiento IP tiene pares (Destino, Router), donde destino es la dirección IP de un destino particular y Router la dirección del siguiente Router en el camino hacia el destino. Nótese que el Router debe ser accesible directamente desde la maquina actual.

Este tipo de encaminamiento trae varias consecuencias, consecuencia directa de su naturaleza estática:

- Todo tráfico hacia una red particular toma el mismo camino, desaprovechando caminos alternativos y el tipo de tráfico.
- Solo el Router con conexión directa al destino sabe si este existe o esta activo.
- Es necesario que los Routers cooperen para hacer posible la comunicación bidireccional.

La orden **Route** muestra y modifica la tabla de encaminamiento de un host. Todos los hosts (y no sólo los routers) tienen tablas de encaminamientos. A continuación se muestra una tabla sencilla para un host con IP 192.168.0.2 / 255.255.255.0 y puerta de salida 192.168.0.1.

A>route print

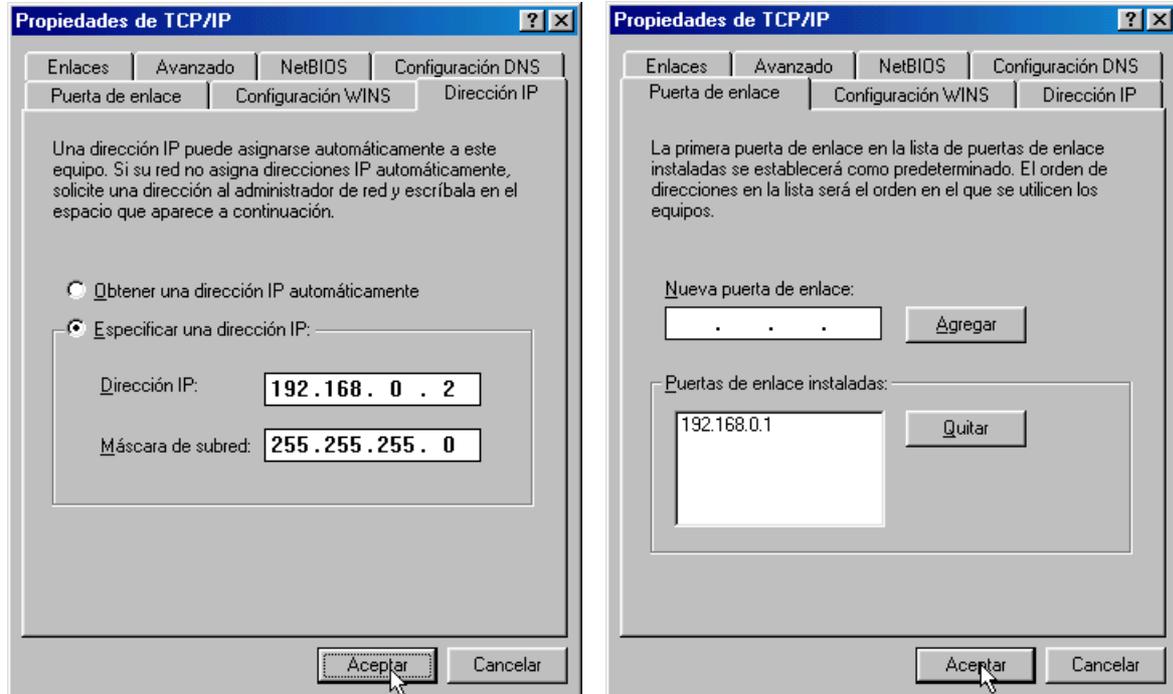
Rutas activas:

Dirección de red	Máscara de red	Puerta de enlace	Interfaz	Métrica	
0.0.0.0	0.0.0.0	192.168.0.1	192.168.0.2	1	(7)
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1	(6)
192.168.0.0	255.255.255.0	192.168.0.2	192.168.0.2	1	(5)
192.168.0.2	255.255.255.255	127.0.0.1	127.0.0.1	1	(4)
192.168.0.255	255.255.255.255	192.168.0.2	192.168.0.2	1	(3)
224.0.0.0	224.0.0.0	192.168.0.2	192.168.0.2	1	(2)
255.255.255.255	255.255.255.255	192.168.0.2	0.0.0.0	1	(1)

Esta tabla se lee de abajo a arriba:

- La línea (1) indica que los datagramas con destino "255.255.255.255" (dirección de multidifusión a la red del host) deben ser aceptados.
- La línea (2) representa un grupo de *multicasting*. La dirección "224.0.0.0" es una dirección de clase D, que no hemos estudiado, y se utiliza para enviar mensajes a una colección de hosts registrados previamente. Estas dos líneas se suelen pasar por alto: aparecen en todas las tablas de rutas.
- La línea (3) indica que todos los mensajes cuyo destinatario sea "192.168.0.255" deben ser aceptados (es la dirección de multidifusión a la red del host).
- La línea (4) se encarga de aceptar todos los mensajes que vayan destinados a la dirección del host "192.168.0.2".
- La línea (5) indica que los mensajes cuyo destinatario sea una dirección de la red del host "192.168.0.0 / 255.255.255.0" deben *salir* del host por su tarjeta de red para que se entreguen directamente en su subred.
- La línea (6) es la dirección de *loopback*: todos los paquetes con destino "127.0.0.0 / 255.0.0.0" serán aceptados por el propio host.
- Finalmente, la línea (7) representa a "todas las demás direcciones que no se hayan indicado anteriormente". En concreto son aquellas direcciones remotas que no pertenecen a la red del host. ¿A dónde se enviarán? Se enviarán a la puerta de salida (*gateway*) de la red "192.168.0.1".

Nótese que la tabla de rutas es la traducción de la configuración IP del host que habitualmente se escribe en las ventanas de Windows:



### 3.3.9 ALGORITMO DE ENCAMINAMIENTO IP

**Ruta** Datagrama (Datagrama)

```
{  
    Extrae de la Cabecera de Datagrama la dirección de destino D;  
    Extrae de D el prefijo de Red N;  
    Si N corresponde a cualquier dirección directamente conectada  
Entonces  
        Envía el Datagrama a D sobre la Red N;  
    Sino  
    Si en la tabla hay una ruta específica para D Entonces  
        Envía Datagrama al salto siguiente especificado;  
    Sino
```

```

Si En la tabla hay una ruta para la red N Entonces
    Envía Datagrama al salto siguiente especificado;
Sino
Si En la tabla hay una ruta por defecto Entonces
    Envía el Datagrama a la dirección por defecto;
Sino
    Declarar Fallo de Enrutamiento;
Fsi
}

```

### 3.3.10 MANEJO DE DATAGRAMAS ENTRANTES.

Cuando un Datagrama llega a un Host, el software de red lo entrega a IP. IP verifica la dirección de destino y si ésta concuerda con la de la maquina local, entonces acepta el Datagrama y lo entrega a las capas superiores. De no coincidir la dirección de destino, el Datagrama es descartado.

Por otra parte, un Router que reciba un Datagrama compara la dirección de destino con la suya propia. Si coinciden, el Datagrama pasa a las capas superiores, si no, se le aplica el algoritmo de encaminamiento y se reenvía el Datagrama.

### 3.3.11 DIRECCIONAMIENTO SIN CLASE

Mediante el empleo de Mascaras de subred, se logra convertir una única red (generalmente una Clase B) en múltiples redes lógicas interconectadas y administradas por la organización propietaria. El problema se presenta cuando el crecimiento explosivo de las redes locales produce el fenómeno ROADS (*Running Out of Address Space*), que consiste simplemente en el agotamiento del espacio de direcciones útil, causado por la gran demanda de las direcciones Clase B, de las cuales solo hay 16.384, mientras que las Clases

C permanecían sin asignar (pues aunque hay 2.097.152 de ellas, nadie las quiere por ser muy pequeñas).

Para enfrentar este problema se desarrollo el esquema de Direcciones sin Clase, que consiste en asignar a una misma organización un bloque continuo de direcciones de Clase C. De esta manera, una organización que requiera conectar a Internet un numero moderado de Hosts (digamos 3.800) puede recibir un bloque de 16 redes continuas de Clase C (por ejemplo, de la red Clase C 199.40.72.0 a la 199.40.87.0), con lo cual dispone de 4.096 direcciones IP validas para administrar.

### **3.3.12 CIDR ENRUTAMIENTO INTER-DOMINIO SIN CLASES (CLASSLESS INTER-DOMAIN ROUTING)**

El esquema de direcciones sin clase genera el problema de aumentar la información que debe incluirse en las tablas de enrutamiento. En el caso del ejemplo, se tendría que incluir 16 nuevas entradas en cada tabla de enrutamiento de cada Host y Router. CIDR resuelve el problema al incluir en las tablas información acerca del tamaño de los bloques y el numero de bloques, así, en las tablas de enrutamiento IP se tienen pares (Destino, Router), donde destino no es una dirección de Host o Red tradicional, sino que incluye información acerca del numero de redes que incluye el bloque (en nuestro ejemplo, 16) y el tamaño de cada una de esas redes (en el ejemplo, son Clases C, 256 direcciones cada una). El Direccionamiento sin clase modifica la estructura de una dirección IP, de esta manera:



Así, CIDR debe incluir en las tablas de enrutamiento cual es la primera red que compone el bloque, cuantos bits se emplean como Prefijo de Red y la mascara de subred

que se emplea. En nuestro ejemplo, las tablas de enrutamiento IP contendrían esta información:

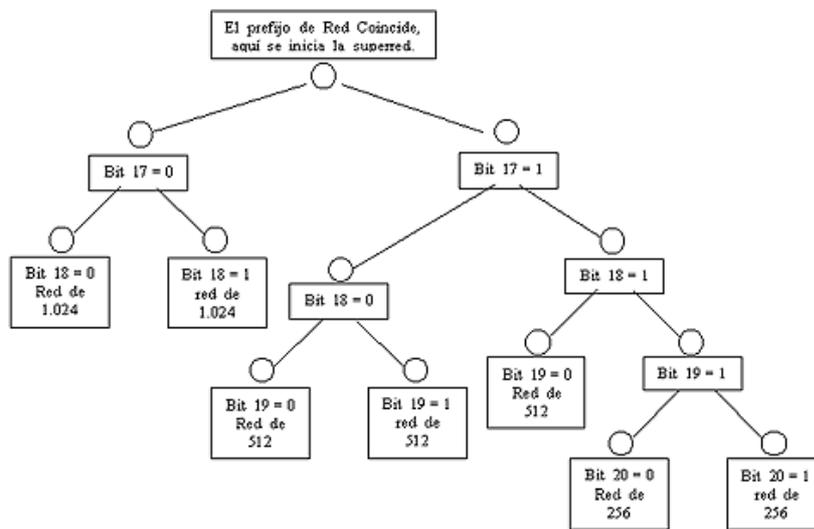
**199.40.72.0/20                      255.255.240.0**

Refiriéndose a un bloque que se inicia con la red 199.40.72.0 y que tiene 20 bits en el prefijo de red. La mascara 255.255.240.0 (11111111.11111111.1111**0000**.00000000) nos indica que se están usando 4 bits extra (los que se han resaltado) para identificar a las redes que componen al bloque. Nótese que cuatro bits permites agrupar precisamente 16 redes Clase C.

Un aspecto importante que hay que subrayar es que en ningún momento cambia el algoritmo básico de enrutamiento IP, lo que cambia es el contenido de las tablas. Además, las nuevas tablas contienen información resumida, por lo que buscar una dirección destino en la tabla se hace de otra manera, pero el algoritmo permanece inalterado.

El problema de buscar direcciones de destino en una tabla, consiste en que cualquier dirección cuya mascara de destino tenga menos bits, incluye a la que tiene mas bits. Con esto se quiere decir que si se encuentra una mascara de subred como 255.255.0.0 (**11111111.11111111**.00000000.00000000, es decir, 16 bits de prefijo de red), va a incluir dentro de ella misma a la mascara de subred 255.255.128.0 (**11111111.11111111.10000000**.00000000, 17 bits de prefijo de red) y esta a su vez incluye a la mascara 255.255.192.0 (**11111111.11111111.11000000**.00000000) y en general, entre menos bits tiene el prefijo de red, mas direcciones Host abarca. Por esta razón cuando se explora la tabla de enrutamiento IP en busca de una dirección de destino, se hace una búsqueda que inicia con las mascaras de más bits y termina en la de menos bits. Es decir, se inicia con mascaras como 255.255.255.255 (todo en uno) y se continua con la 255.255.255.254 (31 unos y un cero) y así sucesivamente. Esto quiere decir que tendrían que hacerse 32 recorridos secuenciales a la tabla, lo cual es muy ineficiente en cuanto a tiempo, pues además de ser un procedimiento demorado, se sabe ya que direcciones

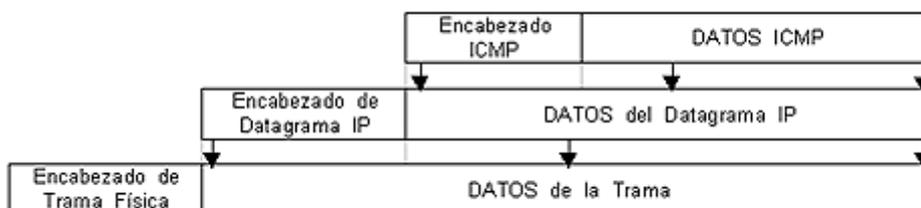
normales de Clase B (255.255.0.0) requieren 16 barridos a la tabla, además, hacen falta 32 barridos para notar que no hay una entrada en la tabla para esa dirección. Por esta razón se emplean otros métodos para hacer estas búsquedas en las tablas de enrutamiento IP. Un esquema muy popular emplea un Árbol Binario, en el cual cada bit representa una nueva rama en el árbol. Así, en nuestro ejemplo, podrían dividirse las direcciones asignadas a la organización (4.096) en subredes de esta forma: dos subredes de 1.024 direcciones cada una, tres de 512 y dos de 256 direcciones. De esta forma, el árbol resultante tendría esta forma:



### 3.4 ICMP: PROTOCOLO DE MENSAJES DE CONTROL DE INTERNET (INTERNET CONTROL MESSAGE PROTOCOL)

Si un Router no puede entregar un Datagrama, o si detecta una situación anómala que afecta su capacidad de hacerlo (por ejemplo, la congestión), tiene unos mecanismos normalizados para informar a la fuente original de que se está produciendo un error y realice las modificaciones necesarias para que evite o solucione el problema.

ICMP [RFC792] es un mecanismo para realizar esta operación y comunica la capa de red de una maquina con la misma capa en otra maquina. ICMP es un protocolo de *reporte de errores* (no los corrige), además, ICMP solo puede informar del error a la fuente del Datagrama, es esta maquina la que debe implementar mecanismos para enfrentar el problema.



Los mensajes de ICMP requieren doble encapsulación: Los mensajes ICMP viajan empaquetados en Datagramas IP. Aun así, no se considera a ICMP un protocolo de nivel superior a IP.

### 3.4.1 FORMATO DEL MENSAJE ICMP

Aunque cada tipo de mensaje tiene su propio formato, todos ellos comparten los primeros tres campos: TIPO (8 bits), CODIGO (8 bits) y CHECKSUM (16 bits).

El campo TIPO identifica al tipo de mensaje ICMP y determina su formato. Puede tener alguno de estos valores:

- 0 : Respuesta de Eco (*Echo Replay*)
- 3 : Destino Inaccesible (*Host Unreachable*)
- 4 : Paquete de Choque (*Source Quench*)
- 5 : Redireccionar (*Redirect*)
- 8 : Solicitud de Eco (*Echo Request*)
- 11 : Tiempo Excedido
- 12 : Problema de Parámetros
- 13 : Solicitud de *Timestamp*
- 14 : Respuesta de *Timestamp*
- 17 : Solicitud de mascara de subred
- 18 : Respuesta de mascara de subred

### 3.4.2 MENSAJES SOLICITUD DE ECO Y RESPUESTA AL ECO

Este es el tipo de mensaje que envía la maquina cuando se emplea el comando ping. Solicitud de Eco pide a la maquina destino que responda con una Respuesta de Eco con un numero de secuencia apropiado.

TIPO (8 o 0)	CODIGO (0)	CHECKSUM
Identificador		Numero de Secuencia
Datos Opcionales		

Los mensajes de solicitud y respuesta de eco, tipos 8 y 0 respectivamente, se utilizan para comprobar si existe comunicación entre 2 hosts a nivel de la capa de red. Estos mensajes comprueban que las capas física (cableado), acceso al medio (tarjetas de red) y red (configuración IP), Routers intermedios están correctos. Sin embargo, no dicen nada de las capas de transporte y de aplicación las cuales podrían estar mal configuradas; por ejemplo, la recepción de mensajes de correo electrónico puede fallar aunque exista comunicación IP con el servidor de correo.

### **3.4.3 MENSAJE DESTINO INACCESIBLE.**

Es el mensaje empleado para reportar que no es posible entregar un Datagrama. El campo CODIGO describe mejor el problema:

- 0 : Red Inaccesible
- 1 : Host Inaccesible
- 2 : Protocolo Inaccesible
- 3 : Puerto Inaccesible
- 4 : Necesita Fragmentación
- 5 : Falla en la Ruta de Origen
- 6 : Red de Destino Desconocida
- 7 : Host Destino Desconocido
- 8 : Host de Origen Aislado
- 9 : Comunicación con Red Destino Administrativamente Prohibida
- 10 : Comunicación con Host Destino Administrativamente Prohibida
- 11 : Red Inaccesible por el tipo de servicio
- 12 : Host Inaccesible por el tipo de servicio

TIPO (3)	CODIGO (0...12)	CHECKSUM
NO USADO (debe ser cero)		
Encabezado IP + Primeros 8 bytes de Datos IP		

Los errores de red inaccesible por lo general implican fallos de enrutamiento. Debido a que el mensaje ICMP contiene la cabecera del Datagrama que lo produjo (en el campo de datos), el origen sabrá cual destino es inaccesible.

### 3.4.4 MENSAJE DE CHOQUE

Debido a que IP funciona sin conexión, un Router no puede reservar memoria o recursos de comunicación antes de recibir los Datagramas. En consecuencia los Routers pueden verse repentinamente saturados por el trafico. A esta situación se le llama congestión.

El congestionamiento se da por que un Host de alta velocidad genera Datagramas mas rápido de lo que el Router puede manejar o porque muchos Host envían Datagrama a la misma dirección al mismo tiempo.

Cuando los Datagramas llegan mas rápido de lo que un Router puede manejarlos, este los coloca en un buffer. Si los Datagramas son parte de una ráfaga pequeña, esto soluciona el problema, pero si continúan llegando Datagramas se saturan los buffers y el Router debe descartar los nuevos Datagramas. Es entonces cuando el Router genera un mensaje ICMP de choque solicitando a éste reducir la tasa de envío de Datagramas. No existe un mensaje ICMP para revertir esta solicitud, en general poco después de bajar la tasa de envío, los Hosts la aumentan progresivamente hasta recibir otro mensaje de choque.

TIPO (4)	CODIGO (0)	CHECKSUM
NO – UTILIZADO (debe ser cero)		
Encabezado IP + 8 primeros bytes de Datos IP		

El objetivo de este mensaje era aliviar el problema de la congestión, pero no tuvo éxito. Se dejó al implementador decidir sobre cuándo enviar estos mensajes, por lo que cada fabricante emplea su política favorita sin que ninguna solucione el problema del todo. Por otra parte, ICMP informa al Host de origen que su Datagrama ha sido descartado, pero puede que este Host no sea el causante de la congestión. Además, ¿Como responder al mensaje ICMP? Documentos como *Requisitos para los Routers* (RFC 1812) estipulan que NO se deben enviar mensajes de choque. Se está trabajando en mecanismos más eficientes.

### 3.4.5 MENSAJE REDIRECCIONAR

Se asume que los Routers conocen rutas correctas. Los Host comienzan con información mínima de enrutamiento y aprenden nuevas rutas de los Routers. En caso de que un Host utilice una ruta no óptima, el Router que lo detecta envía un mensaje ICMP Redireccionar solicitándole que actualice su tabla de enrutamiento IP.

TIPO (5)	CODIGO (0...3)	CHECKSUM
Dirección IP del Router		
Encabezado de IP + 8 primeros bytes de Datos IP		

### 3.4.6 MENSAJE TIEMPO EXCEDIDO

Debido a que los Routers solo deciden sobre el próximo "Salto" usando tablas locales, errores en esas tablas pueden generar "ciclos de enrutamiento" para algún destino. Esto provoca que los Datagramas sean descartados por vencimiento de su TTL. Siempre que un Router descarte un Datagrama ya sea por vencimiento de TTL o por vencimiento del Tiempo de Reensamblado, envía un mensaje de Tiempo Excedido a la fuente.

TIPO (11)	CODIGO (0 o 1)	CHECKSUM
NO – UTILIZADO (debe ser cero)		
Encabezado de IP + 8 primeros bytes de Datos IP		

CODIGO = 0: Descartado por vencimiento de TTL

CODIGO = 1: Descartado por vencimiento de Tiempo de Reensamblado.

### 3.4.7 MENSAJE PROBLEMA DE PARÁMETROS

Cuando un Router o un Host encuentran un problema que no ha sido cubierto con los mensajes ICMP anteriores, envía este mensaje.

TIPO (12)	CODIGO (0 o 1)	CHECKSUM
Indicador	NO – Utilizado (debe ser cero)	
Encabezado de IP + 8 primeros bytes de Datos IP		

El campo indicador apunta al campo dentro del encabezado IP que generó el problema.

### 3.4.8 MENSAJE SOLICITUD DE TIMESTAMP Y RESPUESTA DE TIMESTAMP

Una técnica sencilla provista por TCP/IP para sincronizar relojes emplea ICMP para obtener la hora de la otra maquina. Una maquina envía a otra una solicitud de Timestamp, solicitándole que informe su valor actual para la hora del día. La otra maquina envía una respuesta de Timestamp con esa información.

TIPO (13 o 14)	CODIGO (0)	CHECKSUM
Identificador		Numero de Secuencia
Timestamp Origen		
Timestamp al Recibir		
Timestamp al Transmitir		

### 3.4.9 MENSAJE SOLICITUD DE MASCARA DE SUBRED Y RESPUESTA DE MASCARA DE SUBRED

Para aprender la mascara de subred utilizada por la red local, una maquina puede enviar un mensaje ICMP Solicitud de Mascara de Subred a un Router y esperar su Respuesta. Si la maquina no conoce la dirección del Router, puede enviar este mensaje por difusión.

TIPO (17 o 18)	CODIGO (0)	CHECKSUM
Identificador		Numero de Secuencia
Mascara de Subred		

### 3.4.10 PING PARA COMPROBAR LA COMUNICACIÓN IP ENTRE DOS ORDENADORES

La orden **PING** envía mensajes de solicitud de eco a un host remoto e informa de las respuestas. Veamos su funcionamiento, en caso de no producirse incidencias en el camino.

A envía un mensaje ICMP de tipo 8 (*Echo*) a B

B recibe el mensaje y devuelve un mensaje ICMP de tipo 0 (*Echo Reply*) a A

A recibe el mensaje ICMP de B y muestra el resultado en pantalla



```
A>ping 172.20.9.7 -n 1
```

```
Haciendo ping a 172.20.9.7 con 32 bytes de datos:
```

```
Respuesta desde 172.20.9.7: bytes=32 tiempo<10ms TDV=128
```

En la orden anterior hemos utilizado el parámetro "-n 1" para que el host A únicamente envíe 1 mensaje de solicitud de eco. Si no se especifica este parámetro se enviarían 4 mensajes (y se recibirían 4 respuestas).

Si el host de destino no existiese o no estuviera correctamente configurado recibiríamos un mensaje ICMP de tipo 11 (*Time Exceeded*).

```
A>ping 192.168.0.6 -n 1
```

```
Haciendo ping a 192.168.0.6 con 32 bytes de datos:
```

```
Tiempo de espera agotado.
```

Si tratamos de acceder a un host de una red distinta a la nuestra y no existe un camino para llegar hasta él, es decir, los routers no están correctamente configurados o estamos intentando acceder a una red aislada o inexistente, recibiríamos un mensaje ICMP de tipo 3 (*Destination Unreachable*).

```
A>ping 1.1.1.1 -n 1
```

```
Haciendo ping a 1.1.1.1 con 32 bytes de datos:
```

```
Respuesta desde 192.168.0.1: Host de destino inaccesible.
```

### 3.4.11 UTILIDAD DE PING PARA DIAGNOSTICAR ERRORES EN UNA RED AISLADA



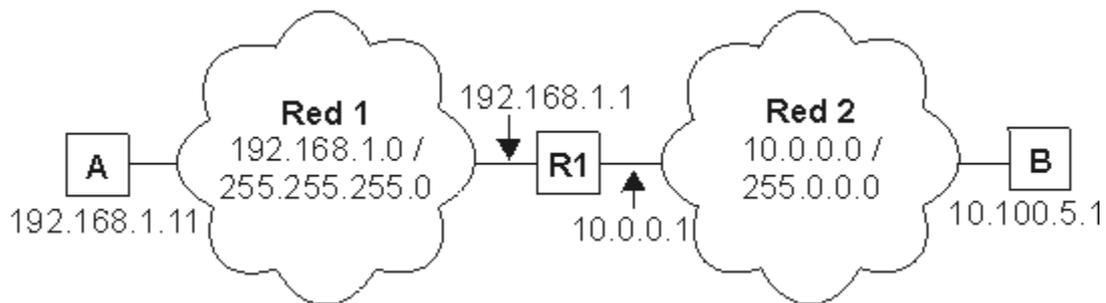
```
A>ping 192.168.1.12
```

- Respuesta. El cableado entre A y B, las tarjetas de red de A y B, y la configuración IP de A y B están correctos.
- Tiempo de espera agotado. Comprobar el host B y el cableado entre A y B.
- Host de destino inaccesible. Comprobar las direcciones IP y máscaras de subred de A y B porque no pertenecen a la misma red.
- Error. Probablemente estén mal instalados los protocolos TCP/IP del host A. Probar A>ping 127.0.0.1 para asegurarse.

**Nota:** El comando ping 127.0.0.1 informa de si están correctamente instalados los protocolos TCP/IP en nuestro host. No informa de si la tarjeta de red de nuestro host está correcta.

### 3.4.12 UTILIDAD DE PING PARA DIAGNOSTICAR ERRORES EN UN ROUTER

A continuación veremos un ejemplo para una red de redes formada por dos redes (1 solo router). La idea es la misma para un mayor número de redes y routers.



A>ping 10.100.5.1

- Respuesta. El cableado entre A y B, las tarjetas de red de A, R1 y B, y la configuración IP de A, R1 y B están correctos. El router R1 permite el tráfico de datagramas IP en los dos sentidos.
- Tiempo de espera agotado. Comprobar el host B y el cableado entre R1 y B. Para asegurarnos que el router R1 está funcionando correctamente haremos A>ping 192.168.1.1
- Host de destino inaccesible. Comprobar el router R1 y la configuración IP de A (probablemente la puerta de salida no sea 192.168.1.1). Recordemos que la puerta de salida (*gateway*) de una red es un host de su propia red que se utiliza para salir a otras redes.

- `ERROR`. Probablemente estén mal instalados los protocolos TCP/IP del host A. Probar `A>ping 127.0.0.1` para asegurarse.

En el caso producirse errores de comunicación en una red de redes con más de un router (Internet es el mejor ejemplo), se suele utilizar el comando PING para ir diagnosticando los distintos routers desde el destino hasta el origen y descubrir así si el fallo es responsabilidad de la red de destino, de una red intermedia o de nuestra red.

***Nota:** Algunos hosts en Internet tienen deshabilitadas las respuestas de eco (mensajes ICMP tipo 0) como medida de seguridad. En estos casos hay que utilizar otros mecanismos para detectar si responde (por ejemplo, la apertura de conexión a un puerto, como veremos en el capítulo siguiente).*

### **3.4.13 UTILIDAD DE LOS MENSAJES DE TIEMPO EXCEDIDO PARA MARCAR LA RUTA**

Los datagramas IP tienen un campo TTL (tiempo de vida) que impide que un mensaje esté dando vueltas indefinidamente por la red de redes. El número contenido en este campo disminuye en una unidad cada vez que el datagrama atraviesa un router. Cuando el TTL de un datagrama llega a 0, éste se descarta y se envía un mensaje ICMP de tipo 11 (*Time Exceeded*) para informar al origen.

Los mensajes ICMP de tipo 11 se pueden utilizar para hacer una traza del camino que siguen los datagramas hasta llegar a su destino. ¿Cómo? Enviando una secuencia de datagramas con TTL=1, TTL=2, TTL=3, TTL=4, etc... hasta alcanzar el host o superar el límite de saltos (30 si no se indica lo contrario). El primer datagrama caducará al atravesar el primer router y se devolverá un mensaje ICMP de tipo 11 informando al origen del router que descartó el datagrama. El segundo datagrama hará lo propio con el segundo router y así sucesivamente. Los mensajes ICMP recibidos permiten definir la traza.

La orden **TRACERT** (**traceroute** en entornos Unix) hace una traza a un determinado host. TRACERT funciona enviando mensajes ICMP de solicitud de eco con distintos TTL; *traceroute*, en cambio, envía mensajes UDP. Si la comunicación extremo a extremo no es posible, la traza nos indicará en qué punto se ha producido la incidencia. Existen algunas utilidades en Internet, como Visual Route, que determinan la localización geográfica de los routers de Internet. Esto permite dibujar en un mapamundi el recorrido que siguen los datagramas hasta llegar a un host.

A>**tracert 130.206.1.2**

Traza a la dirección sun.rediris.es [130.206.1.2]  
sobre un máximo de 30 saltos:

```
 1    1 ms    1 ms    1 ms    PROXY [192.168.0.1]
 2   122 ms   118 ms   128 ms   MADR-X27.red.retevision.es [62.81.1.102]
 3   143 ms   232 ms   147 ms   MADR-R2.red.retevision.es [62.81.1.92]
 4   130 ms   124 ms   246 ms   MADR-R16.red.retevision.es [62.81.3.8]
 5   590 ms   589 ms   431 ms   MADR-R12.red.retevision.es [62.81.4.101]
 6   612 ms   640 ms   124 ms   MADR-R10.red.retevision.es [62.81.8.130]
 7   259 ms   242 ms   309 ms   193.149.1.28
 8   627 ms   752 ms   643 ms   213.0.251.42
 9   137 ms   117 ms   118 ms   213.0.251.142
10  109 ms   105 ms   110 ms   A1-2-1.EB-Madrid00.red.rediris.es [130.206.224.81]
11  137 ms   119 ms   122 ms   A0-0-0-1.EB-Madrid3.red.rediris.es[130.206.224.86]
12  109 ms   135 ms   115 ms   sun.rediris.es [130.206.1.2]
```

Traza completa.

Ejemplo de Visual Route a una dirección IP de Taiwan (203.69.112.12):



## 3.5 PROTOCOLO ARP

### 3.5.1 DESCRIPCION

Dentro de una misma red, las máquinas se comunican enviándose tramas físicas. Las tramas Ethernet contienen campos para las direcciones físicas de origen y destino (6 bytes cada una):

8 bytes	6 bytes	6 bytes	2 bytes	64-1500 bytes	4 bytes
Preámbulo	Dirección física destino	Dirección física origen	Tipo de trama	Datos de la trama	CRC

El problema que se nos plantea es cómo podemos conocer la dirección física de la máquina destino. El único dato que se indica en los datagramas es la dirección IP de destino. ¿Cómo se pueden entregar entonces estos datagramas? Necesitamos obtener la dirección física de un ordenador a partir de su dirección IP. Esta es justamente la misión del protocolo ARP (*Address Resolution Protocol*, protocolo de resolución de direcciones).

ARP asocia las direcciones físicas de Hardware a cada dirección IP lógica asignada a una interfaz de red, una Máscara de Control de Acceso al Medio: es un identificador de Hardware único que es asignado por el fabricante de la NIC. Las MAC son adjuntadas a las cabeceras de IP origen y destino de manera de identificar la dirección IP mas la MAC de un sistema remoto.

Por ejemplo: **02-C9-00-P6-01-20**

Las MAC contienen 48bits de longitud y son únicas, es un componente superior de la Capa de Enlace de Datos (OSI).

Los sistemas de red local LAN usan ARP para descubrir su propia dirección física y la de los sistemas vecinos. Cuando un sistema necesita comunicarse con un host local busca la dirección de IP en su tabla de enrutamiento IP que normalmente se mantiene en memoria. Si no existe la IP en su tabla local, el HOST difunde una solicitud de ARP que contiene la dirección IP de destino.

Para resolver este problema, se diseñó el *protocolo de resolución de direcciones* (ARP, *Address Resolution Protocol*), válido para todas las redes que soportan multidistribución. La idea es simple, si una máquina A necesita saber la dirección física de una máquina B, envía por multidifusión un paquete especial que pide a la máquina con la dirección IP indicada que responda con su dirección física. Una vez recibida la respuesta, A puede enviar paquetes a B directamente, pues conoce su dirección física.

Debido a que la multidistribución es un recurso costoso (consume recursos de red, ya que todos los receptores deben procesar el paquete enviado), suele evitarse su uso lo más posible. Una de las formas de hacer esto es manteniendo en cada máquina una tabla relacionando direcciones IP con direcciones físicas. Además, como en cada petición ARP se encuentra la dirección IP y la dirección física del remitente, todas las máquinas activas pueden actualizar su tabla con el nuevo dato.

Al enviar un mensaje ARP de una máquina a otra, este debe viajar en una trama física. Para que la máquina destino identifique la trama como ARP, debe llevar un valor en el campo de tipo de trama que lo identifique como tal. En Ethernet, este valor es 0806h (en hexadecimal). El formato del mensaje ARP no es fijo, sino que depende que hardware de la red. El formato de un mensaje ARP para Ethernet es el siguiente:

Tipo de hardware		Tipo de protocolo
Long. Dir. física	long. dir. protocolo	Operación
Dirección física remitente (octetos 0 a 3)		
Dirección física remitente (octetos 4 a 5)		dirección IP remitente (octetos 0 y 1)
dirección IP remitente (octetos 2 y 3)		Dirección física destinatario (octetos 0 y 1)
Dirección física destinatario (octetos 2 a 5)		
Dirección IP destinatario (completa, octetos 0 a 3)		

- El campo ***tipo de hardware*** (16 bits) especifica el tipo de interfaz hardware del que se busca la dirección (1 para Ethernet). El campo ***tipo de protocolo*** (16 bits) indica el tipo de protocolo del que el origen ha enviado la dirección (0800h para IP).
- Los campos de ***longitud de direcciones física*** y de ***protocolo*** permiten usar ARP con diferente hardware y protocolos.
- El campo ***operación*** nos indica el tipo de operación en concreto, si es una petición ARP o una respuesta a una petición.
- El resto de los campos indican las direcciones IP y físicas tanto del remitente como del destinatario.

### 3.5.2 RESOLVIENDO UNA DIRECCIÓN IP REMOTA.

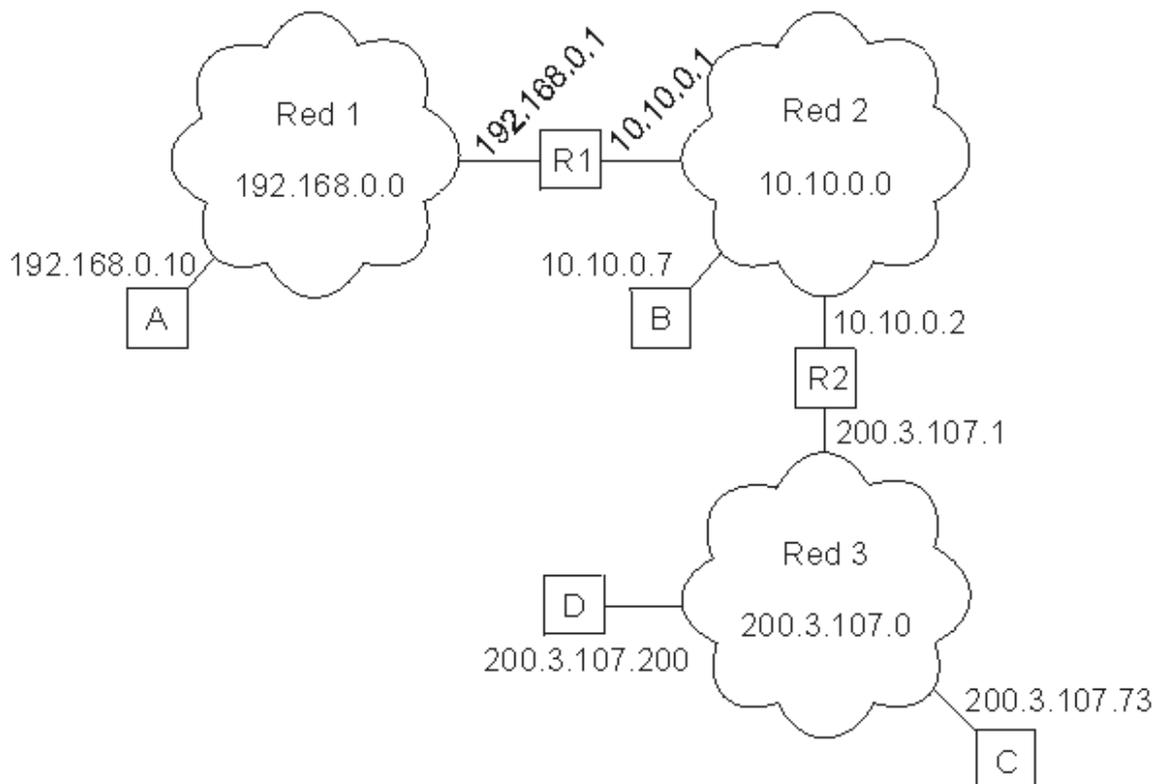
ARP también nos permite que dos máquinas de diferentes redes se comuniquen. En esta situación la petición ARP mediante *broadcasting* es para el *gateway* por defecto y no para la dirección IP de la máquina destino. Es decir la petición *broadcast* es para determinar el *router* que puede enviar los *datagramas* a la máquina destino en la red. Veamos el siguiente ejemplo:

- 1) Cuando iniciamos la petición, la dirección de destino IP se identifica como perteneciente a una red 'remota'. La maquina origen chequea su tabla de 'rutas' para encontrar un camino a la maquina o a la red destino. Si no encuentra coincidencia, la maquina origen determina la dirección del *gateway* por defecto. Chequea igualmente su *caché* ARP para la dirección IP / dirección hardware del *gateway* por defecto en este caso.
  
- 2) Si no encuentra coincidencia para el *gateway*, entonces se envía una petición ARP mediante *broadcast* para la dirección del *gateway* en vez de para la dirección de la maquina destino. El *router* responderá a la maquina origen con 'su' propia dirección hardware. La maquina origen, entonces envía los paquetes de datos al *gateway* para que este a su vez y siguiendo un proceso similar, los reenvíe a la maquina destino.
  
- 3) En el *router*, la dirección IP destino también es investigada para ver si es local o remota. Si es local, el *router* usa la técnica ARP (primero en la *caché* y luego por *broadcast*) para obtener su dirección hardware. Si es una dirección remota, el *router* chequea su tabla de rutas para encontrar un *gateway* para esa dirección y entonces usa ARP (*caché* o *boradcast*) para obtener la dirección hardware del siguiente *gateway* hasta el destino. El paquete se envía a la siguiente maquina de destino.
  
- 4) Después de que la maquina destino reciba la petición, esta responde con un mensaje de respuesta ICMP. Debido a que la maquina origen está en una red remota, la tabla de rutas local se chequea para encontrar un *gateway* para la dirección de la maquina origen. Cuando encuentra un *gateway*, ARP obtiene su dirección hardware.

- 5) Si la dirección hardware del *gateway* no está en la *caché* ARP una petición *broadcast* obtendrá esta. Una vez obtenida su dirección hardware, la respuesta ICMP es enviada al *router* que encaminará estos datos a la maquina origen.

Bien. En este caso el grafico de envío y peticiones es algo más complicado que el anterior ya que se hace intervenir a una maquina más: un *router*. Este proceso se puede ver con un ejemplo muy sencillo:

Suponemos que tenemos la siguiente topología:



Con las correspondencias entre host, adaptador de red, dirección IP y red a la que se conecta ese adaptador de la siguiente tabla:

Host	Dirección física	Dirección IP	Red
<b>A</b>	00-60-52-0B-B7-7D	192.168.0.10	Red 1
<b>R1</b>	00-E0-4C-AB-9A-FF	192.168.0.1	
	A3-BB-05-17-29-D0	10.10.0.1	Red 2
<b>B</b>	00-E0-4C-33-79-AF	10.10.0.7	
<b>R2</b>	B2-42-52-12-37-BE	10.10.0.2	
	00-E0-89-AB-12-92	200.3.107.1	Red 3
<b>C</b>	A3-BB-08-10-DA-DB	200.3.107.73	
<b>D</b>	B2-AB-31-07-12-93	200.3.107.200	

El host **A** envía un datagrama con origen 192.168.0.10 y destino 10.10.0.7 (**B**). Como el host **B** se encuentra en una red distinta al host **A**, el datagrama tiene que atravesar el router 192.168.0.1 (**R1**). Se necesita conocer la dirección física de **R1**.

Es entonces cuando entra en funcionamiento el protocolo ARP: **A** envía un mensaje ARP a todas las máquinas de su red preguntando "*¿Cuál es la dirección física de la máquina con dirección IP 192.168.0.1?*". La máquina con dirección 192.168.0.1 (**R1**) advierte que la pregunta está dirigida a ella y responde a **A** con su dirección física (00-E0-4C-AB-9A-FF). Entonces **A** envía una trama física con origen 00-60-52-0B-B7-7D y destino 00-E0-4C-AB-9A-FF conteniendo el datagrama (origen 192.168.0.10 y destino 10.10.0.7). Al otro lado del router **R2** se repite de nuevo el proceso para conocer la dirección física de **B** y entregar finalmente el datagrama a **B**. El mismo datagrama ha viajado en dos tramas físicas distintas, una para la red 1 y otra para la red 2.

Observemos que las preguntas ARP son multidifusión (se envían a todas las máquinas). Estas preguntas llevan además la dirección IP y dirección física de la máquina que pregunta. La respuesta se envía directamente a la máquina que formuló la pregunta.

### 3.5.3 LA CACHÉ ARP

Para intentar minimizar el número de *broadcast* a la red, el ARP mantiene siempre las direcciones de hardware conocidas y que fueron resueltas por primera vez mediante *broadcasting*.

Cada entrada en la *caché* de la ARP tiene un tiempo potencial de vida de 10 minutos. En cada entrada en la ARP, se guarda los datos de fecha / hora (*timestamp*). Si esta entrada no es usada en los dos primeros minutos, se borra de la *caché*. Si se utiliza será borrada después de los 10 primeros minutos. Si la *caché* del ARP alcanza su capacidad máxima antes de que las entradas anotadas en ella expiren, la entrada más antigua será borrada y la nueva será añadida en su lugar.

En algunas implementaciones del TCP/IP cuando una entrada de la *caché* del ARP es utilizada, se le añaden otros 10 minutos de vida. En Microsoft Windows no esta implementada esta característica. En este router se implementará este mecanismo independientemente de Windows para incrementar la velocidad de respuesta ya que se esta operación se realiza en cada paquete a enviar.

## 3.6 PROTOCOLO UDP

### 3.6.1 INTRODUCCIÓN

Este Protocolo de Datagramas de Usuario (UDP: User Datagram Protocol) [RFC768] se define con la intención de hacer disponible un tipo de Datagramas para la comunicación por intercambio de paquetes entre ordenadores en el entorno de un conjunto interconectado de redes de computadoras. Este protocolo asume que el Protocolo de Internet (IP: Internet Protocol) se utiliza como protocolo subyacente.

Este protocolo aporta un procedimiento para que los programas de aplicación puedan enviar mensajes a otros programas con un mínimo de mecanismo de protocolo. El protocolo se orienta a transacciones, y tanto la entrega como la protección ante duplicados no se garantizan. Las aplicaciones que requieran de una entrega fiable y ordenada de secuencias de datos deberían utilizar el Protocolo de Control de Transmisión (TCP: *Transmission Control Protocol*).

### 3.6.2 DESCRIPCIÓN

El protocolo UDP (*User Datagram Protocol*, protocolo de datagrama de usuario) proporciona una comunicación muy sencilla entre las aplicaciones de dos ordenadores. Al igual que el protocolo IP, UDP es:

- No orientado a conexión. No se establece una conexión previa con el otro extremo para transmitir un mensaje UDP. Los mensajes se envían sin más y éstos pueden duplicarse o llegar desordenados al destino.
- No fiable. Los mensajes UDP se pueden perder o llegar dañados.

UDP utiliza el protocolo IP para transportar sus mensajes. Como vemos, no añade ninguna mejora en la calidad de la transferencia; aunque sí incorpora los puertos origen y destino en su formato de mensaje. Las aplicaciones (y no el protocolo UDP) deberán programarse teniendo en cuenta que la información puede no llegar de forma correcta.

### 3.6.3 FORMATO DEL MENSAJE UDP

0										10										20										30			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
Puerto UDP origen																Puerto UDP destino																	
Longitud mensaje UDP																Suma verificación UDP																	
Datos																																	
...																																	

**Puerto de Origen** es opcional; cuando tiene sentido, indica el puerto del proceso emisor, y puede que se asuma que ése sea el puerto al cual la respuesta debería ser dirigida en ausencia de otra información. Si no se utiliza, se inserta un valor cero.

**Puerto de Destino** tiene significado dentro del contexto de una dirección de destino en un entorno Internet particular.

**Longitud** representa la longitud en octetos de este datagrama de usuario, incluyendo la cabecera y los datos. (Esto implica que el valor mínimo del campo Longitud es ocho.)

**Suma de Control** (*Checksum*) es el complemento a uno de 16 bits de la suma de los complementos a uno de las palabras de la combinación de una pseudo-

cabecera construida con información de la cabecera IP, la cabecera UDP y los datos, y rellena con octetos de valor cero en la parte final (si es necesario) hasta tener un múltiplo de dos octetos.

La pseudo-cabecera del nivel de enlace que imaginariamente antecede a la cabecera UDP contiene la dirección de origen, la dirección de destino, el protocolo y la longitud UDP. Esta información proporciona protección frente a datagramas mal encaminados. Este procedimiento de comprobación es el mismo que el utilizado en TCP.

0										10										20										30									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
Dirección origen																																							
Dirección destino																																							
cero										Protocolo										Logitud UDP																			

Si la suma de control calculada es cero, se transmite como un campo de unos (el equivalente en la aritmética del complemento a uno). Un valor de la suma de control transmitido como un campo de ceros significa que el emisor no generó la suma de control (para depuración o para protocolos de más alto nivel a los que este campo les sea indiferente).

Este es el protocolo 17 (21 en octal) cuando se utilice en el Protocolo de Internet (IP).

### 3.6.4 INTERFAZ DE USUARIO

Un interfaz de usuario debería permitir:

- la creación de nuevos puertos de recepción,

- operaciones de recepción en los puertos de recepción que devuelvan los octetos de datos y una indicación del puerto de origen y de la dirección de origen,
- y una operación que permita enviar un datagrama, especificando los datos y los puertos de origen y de destino y las direcciones a las que se debe enviar.

### **3.6.5 INTERFAZ IP**

El módulo UDP debe ser capaz de determinar las direcciones de origen y destino en un entorno internet así como el campo de protocolo de la cabecera del protocolo internet. Una posible interfaz UDP/IP devolvería el datagrama de internet completo, incluyendo toda la cabecera, en respuesta a una operación de recepción. Un interfaz de este tipo permitiría también al módulo UDP pasar un datagrama de internet completo con cabecera al módulo IP para ser enviado. IP verificaría ciertos campos por consistencia y calcularía la suma de control de la cabecera del protocolo internet.

### **3.6.6 APLICACIÓN DEL PROTOCOLO**

Los usos principales de este protocolo son el Servidor de Nombres de Internet y la Transferencia Trivial de Ficheros (TFTP: Trivial File Transfer).

## 3.7 PROTOCOLO TCP

### 3.7.1 DESCRIPCIÓN

El protocolo TCP (*Transmission Control Protocol*, protocolo de control de transmisión) [RFC793] está basado en IP que es no fiable y no orientado a conexión, y sin embargo es:

- Orientado a conexión. Es necesario establecer una conexión previa entre las dos máquinas antes de poder transmitir ningún dato. A través de esta conexión los datos llegarán siempre a la aplicación destino de forma ordenada y sin duplicados. Finalmente, es necesario cerrar la conexión.
- Fiable. La información que envía el emisor llega de forma correcta al destino.

El protocolo TCP permite una comunicación fiable entre dos aplicaciones. De esta forma, las aplicaciones que lo utilicen no tienen que preocuparse de la integridad de la información: dan por hecho que todo lo que reciben es correcto.

El flujo de datos entre una aplicación y otra viajan por un *circuito virtual*. Sabemos que los datagramas IP pueden seguir rutas distintas, dependiendo del estado de los encaminadores intermedios, para llegar a un mismo sitio. Esto significa que los datagramas IP que transportan los mensajes siguen rutas diferentes aunque el protocolo TCP logró la ilusión de que existe un único circuito por el que viajan todos los bytes uno detrás de otro (algo así como una tubería entre el origen y el destino). Para que esta comunicación pueda ser posible es necesario abrir previamente una conexión. Esta conexión garantiza que los todos los datos lleguen correctamente de forma ordenada y sin duplicados. La unidad de datos del protocolo es el *byte*, de tal forma que la aplicación origen envía bytes y la aplicación destino recibe estos bytes.

Sin embargo, cada byte no se envía inmediatamente después de ser generado por la aplicación, sino que se espera a que haya una cierta cantidad de bytes, se agrupan en un *segmento* y se envía el segmento completo. Para ello son necesarias unas *memorias intermedias* o *buffers*. Cada uno de estos segmentos viaja en el campo de datos de un

datagrama IP. Si el segmento es muy grande será necesario fragmentar el datagrama, con la consiguiente pérdida de rendimiento; y si es muy pequeño, se estarán enviando más cabeceras que datos. Por consiguiente, es importante elegir el mayor tamaño de segmento posible que no provoque fragmentación.

El protocolo TCP envía un *flujo de información no estructurado*. Esto significa que los datos no tienen ningún formato, son únicamente los bytes que una aplicación envía a otra. Ambas aplicaciones deberán ponerse de acuerdo para comprender la información que se están enviando.

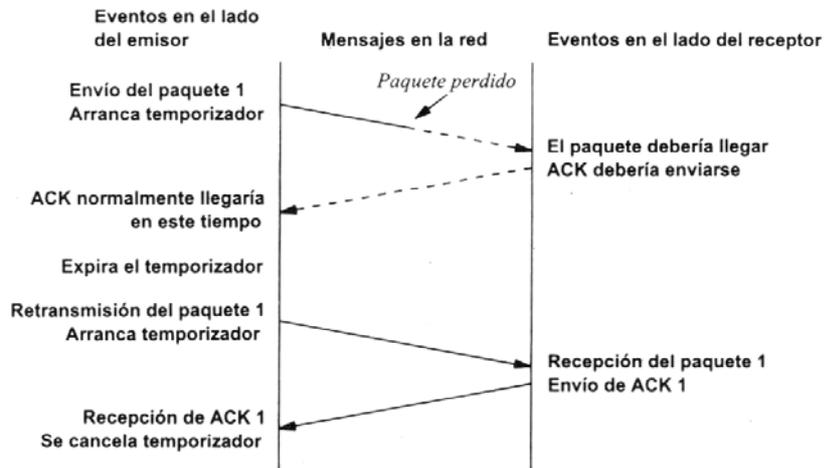
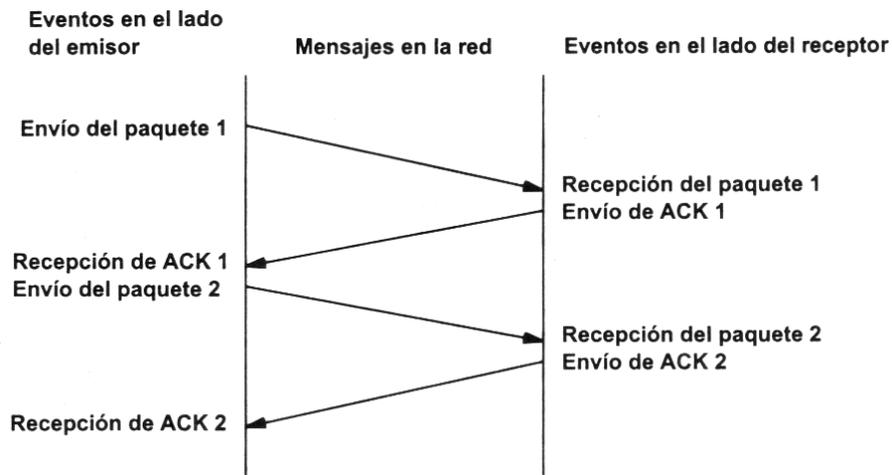
Cada vez que se abre una conexión, se crea un canal de comunicación bidireccional en el que ambas aplicaciones pueden enviar y recibir información, es decir, una conexión es *full-dúplex*.

### **3.7.2 ASEGURAR LA FIABILIDAD**

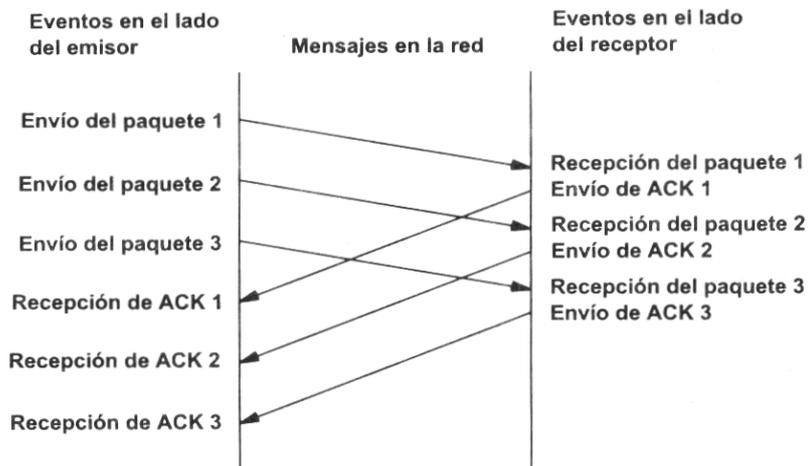
¿Cómo es posible enviar información fiable basándose en un protocolo no fiable? Es decir, si los datagramas que transportan los segmentos TCP se pueden perder, ¿cómo pueden llegar los datos de las aplicaciones de forma correcta al destino?

La respuesta a esta pregunta es sencilla: cada vez que llega un mensaje se devuelve una confirmación (*acknowledgement*) para que el emisor sepa que ha llegado correctamente. Si no le llega esta confirmación pasado un cierto tiempo, el emisor reenvía el mensaje.

Veamos a continuación la manera más sencilla (aunque ineficiente) de proporcionar una comunicación fiable. El emisor envía un dato, arranca su temporizador y espera su confirmación (ACK). Si recibe su ACK antes de agotar el temporizador, envía el siguiente dato. Si se agota el temporizador antes de recibir el ACK, reenvía el mensaje. Los siguientes esquemas representan este comportamiento:



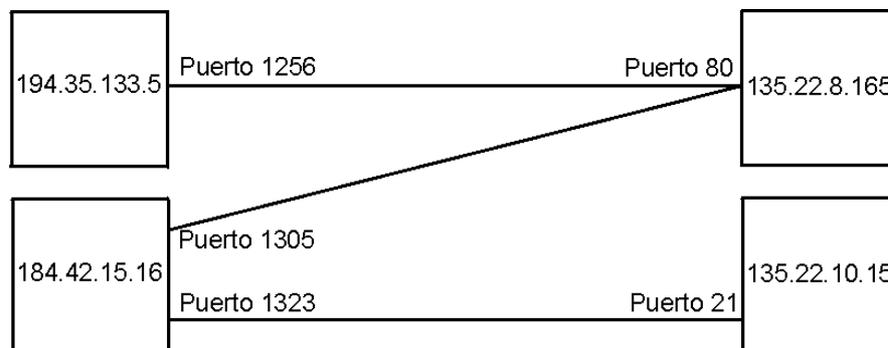
Este esquema es perfectamente válido aunque muy ineficiente debido a que sólo se utiliza un sentido de la comunicación a la vez y el canal está desaprovechado la mayor parte del tiempo. Para solucionar este problema se utiliza un *protocolo de ventana deslizante*, que se resume en el siguiente esquema. Los mensajes y las confirmaciones van numerados y el emisor puede enviar más de un mensaje antes de haber recibido todas las confirmaciones anteriores.



### 3.7.3 IDENTIFICADOR DE CONEXIONES

Una conexión son dos pares *dirección IP:puerto*. No puede haber dos conexiones iguales en un mismo instante en toda la Red. Aunque bien es posible que un mismo ordenador tenga dos conexiones distintas y simultáneas utilizando un mismo puerto. El protocolo TCP utiliza el concepto de conexión para identificar las transmisiones. En el siguiente ejemplo se han creado tres conexiones. Las dos primeras son al mismo servidor Web (puerto 80) y la tercera a un servidor de FTP (puerto 21).

Host 1	Host 2
194.35.133.5:1256	135.22.8.165:80
184.42.15.16:1305	135.22.8.165:80
184.42.15.16:1323	135.22.10.15:21



Para que se pueda crear una conexión, el extremo del servidor debe hacer una *apertura pasiva* del puerto (escuchar su puerto y quedar a la espera de conexiones) y el cliente, una *apertura activa* en el puerto del servidor (conectarse con el puerto de un determinado servidor).

*Nota:* El comando **NetStat** muestra las conexiones abiertas en un ordenador, así como estadísticas de los distintos protocolos de Internet.

### 3.7.4 FORMATO DEL SEGMENTO TCP

Ya hemos comentado que el flujo de bytes que produce una determinada aplicación se divide en uno o más segmentos TCP para su transmisión. Cada uno de estos segmentos viaja en el campo de datos de un datagrama IP. Para facilitar el control de flujo de la información los bytes de la aplicación se numeran. De esta manera, cada segmento indica en su cabecera el primer byte que transporta. Las confirmaciones o acuses de recibo (ACK) representan el siguiente byte que se espera recibir (y no el número de segmento recibido, ya que éste no existe).

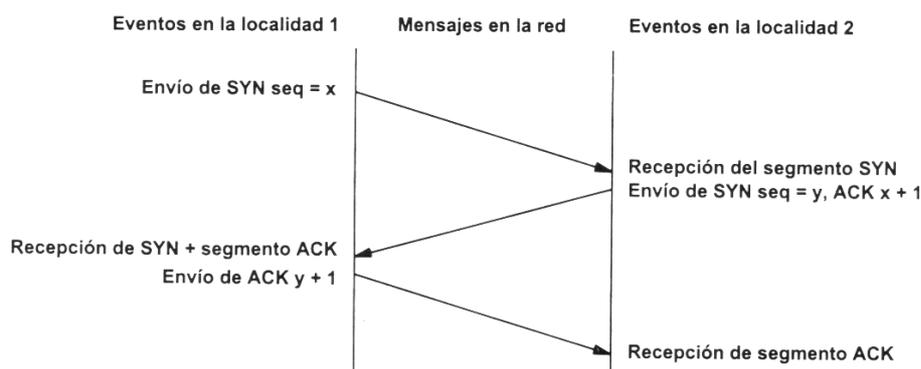
0										10										20										30	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Puerto TCP origen															Puerto TCP destino																
Número de secuencia																															
Número de acuse de recibo																															
HLEN					Reservado					Bits código					Ventana																
Suma de verificación															Puntero de urgencia																
Opciones (si las hay)																				Relleno											
Datos																															
...																															

- **Puerto fuente** (16 bits). Puerto de la máquina origen. Al igual que el puerto destino es necesario para identificar la conexión actual.
- **Puerto destino** (16 bits). Puerto de la máquina destino.
- **Número de secuencia** (32 bits). Indica el número de secuencia del primer byte que transporta el segmento.
- **Número de acuse de recibo** (32 bits). Indica el número de secuencia del siguiente byte que se espera recibir. Con este campo se indica al otro extremo de la conexión que los bytes anteriores se han recibido correctamente.
- **HLEN** (4 bits). Longitud de la cabecera medida en múltiplos de 32 bits (4 bytes). El valor mínimo de este campo es 5, que corresponde a un segmento sin datos (20 bytes).
- **Reservado** (6 bits). Bits reservados para un posible uso futuro.
- **Bits de código** o indicadores (6 bits). Los bits de código determinan el propósito y contenido del segmento. A continuación se explica el significado de cada uno de estos bits (mostrados de izquierda a derecha) si está a 1.
- **URG**. El campo *Puntero de urgencia* contiene información válida.
- **ACK**. El campo *Número de acuse de recibo* contiene información válida, es decir, el segmento actual lleva un ACK. Observemos que un mismo segmento puede transportar los datos de un sentido y las confirmaciones del otro sentido de la comunicación.
- **PSH**. La aplicación ha solicitado una operación *push* (enviar los datos existentes en la memoria temporal sin esperar a completar el segmento).
- **RST**. Interrupción de la conexión actual.
- **SYN**. Sincronización de los números de secuencia. Se utiliza al crear una conexión para indicar al otro extremo cual va a ser el primer número de secuencia con el que va a comenzar a transmitir (veremos que no tiene porqué ser el cero).

- **FIN.** Indica al otro extremo que la aplicación ya no tiene más datos para enviar. Se utiliza para solicitar el cierre de la conexión actual.
- **Ventana** (16 bits). Número de bytes que el emisor del segmento está dispuesto a aceptar por parte del destino.
- **Suma de verificación** (24 bits). Suma de comprobación de errores del segmento actual. Para su cálculo se utiliza una *pseudo-cabecera* que también incluye las direcciones IP origen y destino.
- **Puntero de urgencia** (8 bits). Se utiliza cuando se están enviando datos urgentes que tienen preferencia sobre todos los demás e indica el siguiente byte del campo *Datos* que sigue a los datos urgentes. Esto le permite al destino identificar donde terminan los datos urgentes. Nótese que un mismo segmento puede contener tanto datos urgentes (al principio) como normales (después de los urgentes).
- **Opciones** (variable). Si está presente únicamente se define una opción: el tamaño máximo de segmento que será aceptado.
- **Relleno.** Se utiliza para que la longitud de la cabecera sea múltiplo de 32 bits.
- **Datos.** Información que envía la aplicación.

### 3.7.5 ESTABLECIMIENTO DE UNA CONEXIÓN

Antes de transmitir cualquier información utilizando el protocolo TCP es necesario abrir una conexión. Un extremo hace una *apertura pasiva* y el otro, una *apertura activa*. El mecanismo utilizado para establecer una conexión consta de *tres vías*.



- 1) La máquina que quiere iniciar la conexión hace una apertura activa enviando al otro extremo un mensaje que tenga el bit SYN activado. Le informa además del primer número de secuencia que utilizará para enviar sus mensajes.
- 2) La máquina receptora (un servidor generalmente) recibe el segmento con el bit SYN activado y devuelve la correspondiente confirmación. Si desea abrir la conexión, activa el bit SYN del segmento e informa de su primer número de secuencia. Deja abierta la conexión por su extremo.
- 3) La primera máquina recibe el segmento y envía su confirmación. A partir de este momento puede enviar datos al otro extremo. Abre la conexión por su extremo.
- 4) La máquina receptora recibe la confirmación y entiende que el otro extremo ha abierto ya su conexión. A partir de este momento puede enviar ella también datos. La conexión ha quedado abierta en los dos sentidos.

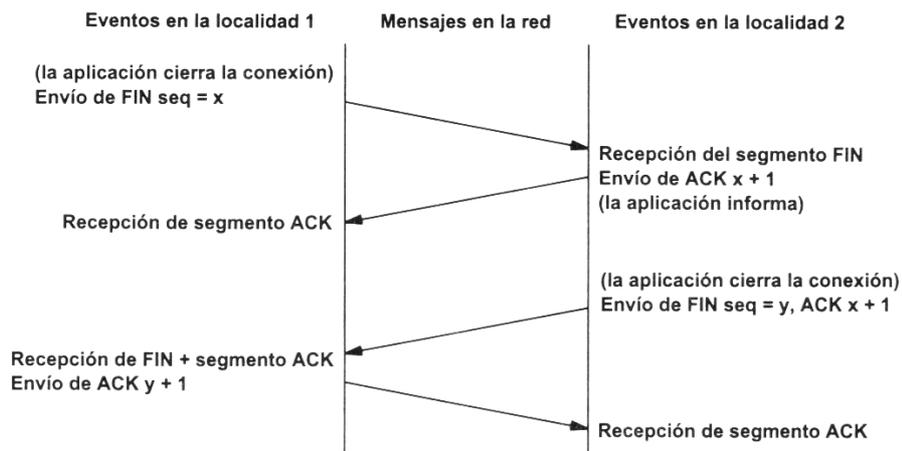
Observamos que son necesarios 3 segmentos para que ambas máquinas abran sus conexiones y sepan que la otra también está preparada.

**Números de secuencia.** — Se utilizan números de secuencia distintos para cada sentido de la comunicación. Como hemos visto el primer número para cada sentido se acuerda al establecer la comunicación. Cada extremo se inventa un número aleatorio y envía éste como inicio de secuencia. Observamos que los números de secuencia no comienzan entonces en el cero. ¿Por qué se procede así? Uno de los motivos es para

evitar conflictos: supongamos que la conexión en un ordenador se interrumpe nada más empezar y se crea una nueva. Si ambas han empezado en el cero es posible que el receptor entienda que la segunda conexión es una continuación de la primera (si utilizan los mismos puertos).

### 3.7.6 CIERRE DE UNA CONEXIÓN

Cuando una aplicación ya no tiene más datos que transferir, el procedimiento normal es cerrar la conexión utilizando una variación del mecanismo de 3 vías explicado anteriormente.



El mecanismo de cierre es algo más complicado que el de establecimiento de conexión debido a que las conexiones son *full-duplex* y es necesario cerrar cada uno de los dos sentidos de forma independiente.

- 1) La máquina que ya no tiene más datos que transferir, envía un segmento con el bit FIN activado y cierra el sentido de envío. Sin embargo, el sentido de recepción de la conexión sigue todavía abierto.
- 2) La máquina receptora recibe el segmento con el bit FIN activado y devuelve la correspondiente confirmación. Pero no cierra inmediatamente el otro sentido de la conexión sino que informa a la aplicación de la petición de cierre. Aquí se produce un lapso de tiempo hasta que la aplicación decide cerrar el otro sentido de la conexión.
- 3) La primera máquina recibe el segmento ACK.
- 4) Cuando la máquina receptora toma la decisión de cerrar el otro sentido de la comunicación, envía un segmento con el bit FIN activado y cierra la conexión.
- 5) La primera máquina recibe el segmento FIN y envía el correspondiente ACK. Observemos que aunque haya cerrado su sentido de la conexión sigue devolviendo las confirmaciones.
- 6) La máquina receptora recibe el segmento ACK.

## 3.8 PROTOCOLO RSVP

El protocolo de reserva de recursos o RSVP es un protocolo nuevo que se encuentra en fase de definición y que proporciona un mecanismo para que las aplicaciones puedan hacer reservas de una cierta calidad de servicio para sus flujos de datos en todos los encaminadores que pertenezcan al camino de datos.

La definición del protocolo se recoge básicamente en los siguientes documentos RFC (*Request For Comments*):

- RFC 2205 - "RSVP Version 1 Functional Specification" ([RFC 2205])
- RFC 2209 - "RSVP Version 1 Message Processing Rules" ([RFC 2209])
- RFC 2210 - "The Use of RSVP with IETF Integrated Services" ([RFC 2210])

RSVP es un protocolo de señalización completamente independiente del tipo de calidad de servicio que la red provea. Esto implica que en la reserva de los recursos de la red para una aplicación con este modelo intervengan dos agentes distintos:

- El protocolo de señalización (en este caso RSVP), el cual permite a una aplicación hacer las reservas de recursos necesarias, y mantenerlas a lo largo de todo el camino que deberá seguir el flujo de datos que esta genere.
- Los mecanismos que aseguren que, una vez hecha la reserva de una cierta calidad de servicio para un flujo de datos, los paquetes pertenecientes al mismo serán tratados de manera que se cumplan los parámetros de calidad aceptados en la reserva.

El protocolo RSVP define la manera de crear, mantener y eliminar reservas de calidad de servicio, pero no define la manera de llevar a cabo dichas reservas. Esta visión, y el hecho de que RSVP es independiente de los algoritmos de encaminamiento de la red y que esté pensado para trabajar con IPv6, hacen que RSVP sea un protocolo pensado de cara al futuro de la red.

### 3.8.1 CARACTERÍSTICAS

Las características básicas de RSVP son:

- Realiza reservas de recursos para aplicaciones unicast y multicast, adaptándose dinámicamente a los cambios de rutas y miembros de grupo. a RSVP es *simplex*,

esto es, hace reservas para flujos unidireccionales. El estado en routers y host es *'soft'*, es decir, deben ser refrescados periódicamente. a RSVP no es un protocolo de encaminamiento pero depende de ellos.

- RSVP mantiene y transporta parámetros de control de tráfico y *"policy control"* que son transparentes a RSVP. a Proporciona varios estilos de reserva para satisfacer a una variedad de aplicaciones. a Soporta tanto IPv4 como IPv6.

### 3.8.1.1 El Modelo de Reservas Basadas en el Receptor

Una ampliación trivial del modelo de entrega punto a punto sería un mecanismo en el cual el emisor se encarga de las reservas a lo largo de todo el árbol de distribución multicast.

Esta primera aproximación genera, no obstante, numerosos problemas. Por ejemplo, cada vez que un miembro abandona el árbol de distribución u otro se añadiera al mismo, sería responsabilidad del emisor el configurar una nueva conexión punto a multipunto con los receptores, lo cual crearía una gran carga en la red cuando el grupo fuera de grandes dimensiones.

Un protocolo iniciado por el emisor es incapaz de tratar con receptores heterogéneos, por ejemplo, ciertos receptores podrían usar mejor hardware, u otros tener una conexión con menor ancho de banda. Dado que el emisor no conoce a priori a los receptores, solo podría crear una reserva uniforme, lo que en muchos casos podría conducir a situaciones de desaprovechamiento de las reservas o situaciones injustas para algunos receptores.

RSVP es un protocolo con reservas basadas en el receptor. Esto significa que es responsabilidad del receptor el reservar un determinado nivel de recursos y mantener esta reserva mientras desee recibir el flujo de datos del emisor. Esto permite que cada receptor haga las reservas específicamente adaptadas a sus propias necesidades.

El modelo de reservas basadas en el receptor es una solución distribuida al problema de la reserva de recursos.

### 3.8.1.2 Receptores Heterogéneos

Es posible, y RSVP lo permite, que varios receptores hagan reservas de recursos diferentes para un mismo grupo de distribución multicast. Esto, que en principio no afecta en demasía al protocolo, complica la implementación de las aplicaciones.

Por ejemplo, unos receptores pueden recibir un flujo a 28'8 Kbps, otros a 128 Kbps, y otros a 10 Mbps o incluso más. Si un emisor está enviando un flujo de vídeo en multicast a estos receptores heterogéneos, ¿debe codificar el vídeo para calidad baja, es decir 28'8 Kbps, para calidad media a 128 Kbps, o para calidad alta a 10 Mbps? Si el vídeo se codificara a 10 Mbps, solo los receptores con esta capacidad podrían verlo. Por otro lado, si se codificara a 28'8 Kbps; los usuarios que disponen de 10 Mbps tendrían que ver el vídeo en baja calidad aunque podrían recibir una mejor calidad.

Bien, esta es una muy buena pregunta cuya solución no provee el protocolo RSVP, sino que es problema de las aplicaciones. Para este caso concreto se soluciona codificando la señal de vídeo en 2 capas (o *layers*): una capa base y una capa de mejora. La capa base podría codificarse a 20 Kbps mientras que la capa de mejora podría codificarse a 100 Kbps;

de esta manera los receptores con accesos de 28'8 Kbps podrían recibir las imágenes en baja calidad (solo con la capa base), mientras que los receptores con accesos de 128 Kbps o superiores podrían recibir las dos capas y reconstruir una imagen de alta calidad.

### **3.8.2 FUNCIONAMIENTO DEL PROTOCOLO**

Cada emisor que implementa el protocolo RSVP envía periódicamente mensajes PATH a los posibles receptores. Estos mensajes son encaminados mediante los routers/switches utilizando las tablas de encaminamiento normales hacia los receptores.

Los mensajes PATH contienen información sobre las características de tráfico que tendrán los datos enviados por el emisor y las características de recursos que tiene el camino de datos hasta cada uno de los receptores.

Cada receptor envía su petición de reserva, en forma de un mensaje RESV, en el camino inverso hacia los emisores. Estos mensajes son reenviados salto a salto por la red siguiendo el camino inverso al que siguieron los mensajes PATH (en realidad la principal utilidad de los mensajes PATH es la de crear este camino).

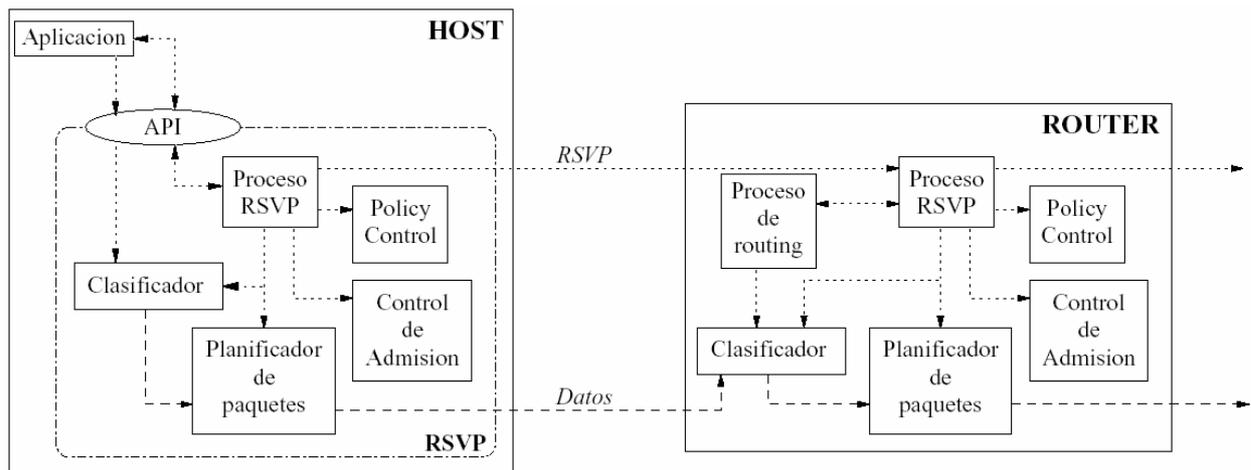
Cada uno de los nodos intermedios, a la recepción de un mensaje RESV, guarda la información necesaria para mantener esa reserva y la pasa al nodo anterior (en el orden establecido por el camino que siguieron los mensajes PATH).

Los estados de reserva y camino guardados en los nodos son automáticamente eliminados si no son refrescados antes de un cierto tiempo. Esto previene situaciones en las

que la caída de alguno de los nodos involucrados dejara reservas instaladas en los nodos intermedios. Es responsabilidad de emisores y receptores el refrescar periódicamente el estado del camino reservado.

### 3.8.3 ESTRUCTURA DE UNA IMPLEMENTACIÓN RSVP

Una implementación RSVP tiene la estructura de bloques mostrada en la siguiente figura.



Es necesario que el protocolo esté implementado en todos los nodos intermedios del camino de datos. Esto implica que el protocolo debe estar soportado en los hosts extremos (emisores y receptores) y en los routers por los que deberán pasar los flujos de datos.

Aun con todo, y dado que es imposible introducir un protocolo nuevo en toda la red al mismo tiempo, el protocolo RSVP contempla la posibilidad de que en la ruta de un flujo existan zonas sin soporte del protocolo.

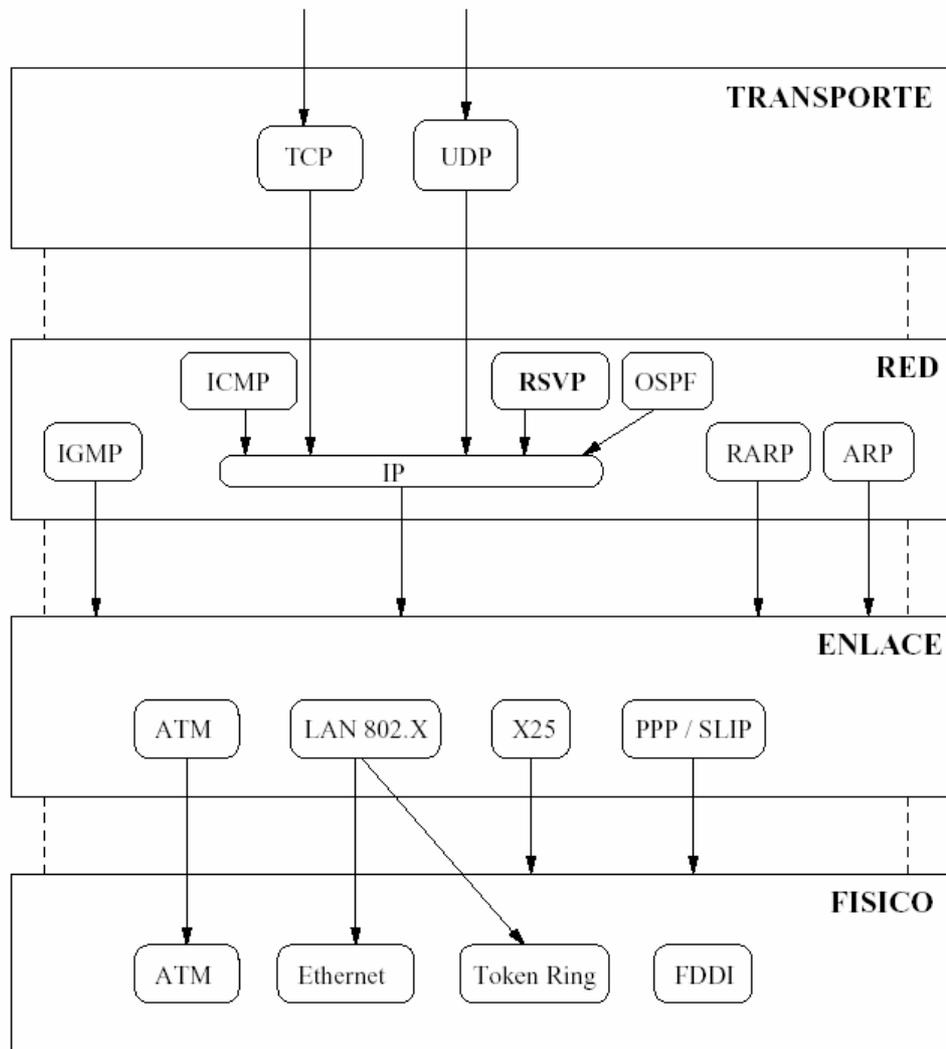
### 3.8.3.1 Transporte de los Mensajes RSVP

Al igual que otros algoritmos de control como ICMP o IGMP, RSVP se ejecuta en 'background' (o segundo plano) y no en el camino de datos.

Esto implica que los paquetes de datos que deberán beneficiarse de la reserva de recursos iniciada por RSVP no serán diferentes de aquellos que no pertenezcan al flujo reservado. Es responsabilidad de RSVP, y los mecanismos que implementen la calidad de servicio, el identificar a los paquetes que pertenezcan al flujo de datos para el que se hizo una reserva y darles el tratamiento necesario.

Todos los mensajes del protocolo RSVP se encapsulan dentro de datagramas IP (como *raw data*, protocolo 46), al igual, por ejemplo, que los mensajes del protocolo de control ICMP. La figura de la página siguiente muestra la posición lógica del protocolo dentro del modelo de capas de Internet.

Si el nodo no dispusiera de la capacidad de encapsular raw data dentro de datagramas IP (protocolo con id. 46), se ha previsto que se encapsule el mensaje RSVP dentro de un mensaje UDP (lo cual se especifica en el apéndice C del [RFC 2205])



### 3.8.3.2 Mecanismos de Control de Admisión

Dada una petición de reserva, cada nodo en el camino de datos debe decidir si acepta o rechaza dicha petición. Esta decisión la tornan en conjunto los siguientes módulos:

- **Policy control.** Determina qué usuarios tienen el permiso administrativo para realizar la reserva. Solo aquellos usuarios autorizados podrán instalar reservas en el nodo.

- **Control de Admisión.** Este módulo decide, dada una petición de reserva, si el nodo dispone de suficientes recursos para aceptar dicha reserva.

En otras palabras, a la recepción de una petición de reserva de recursos, un nodo RSVP debe cuestionarse las siguientes preguntas:

1. **¿Tiene el usuario/aplicación el permiso necesario?** Esta pregunta la responderá el módulo de Policy control.

2. **¿Dispone este nodo de los recursos suficientes para hacer la nueva reserva?** Cuya respuesta vendrá dada por el módulo de control de admisión.

### 3.8.3.3 Mecanismos de Control de Tráfico

Los 'mecanismos de control de tráfico' se encargan de implementar, para un determinado flujo de datos, la calidad de servicio reservada.

Estos mecanismos incluyen tres módulos fundamentales:

- **Clasificador de paquetes.** Dado un paquete de datos, determina qué tipo de QoS le pertenece (y tal vez la ruta que deba seguir)
- **Planificador de paquetes.** Determina qué paquete de datos debe enviarse en cada momento.

Los modelos de servicio de QoS implementados en RSVP son los definidos por el *Integrated Services Working Group*, que definen dos tipos de servicio: el de carga controlada y el garantizado (véase el [RFC 2210J])

### 3.8.4 MODELO DE RESERVAS

Los dos elementos fundamentales para hacer una reserva en RSVP son la 'sesión' y el 'descriptor de flujo'.

- La sesión es la unidad para la cual se hace una reserva, y equivale a una identificación del receptor.
- El descriptor de flujo se utiliza para describir qué calidad de servicio queremos para qué paquetes de datos de los que el emisor (o emisores) generará.

#### 3.8.4.1 Sesión RSVP

La sesión actúa como clave identificativa para los mensajes RSVP. Una sesión RSVP viene definida por los parámetros siguientes:

- *DestAddress*: contiene la dirección IP del destino, ya sea unicast o multicast.
- *ProtocolId* : identificador del protocolo IP utilizado (IPv4 o IPv6)
- *DstPort* (opcional): Puerto de destino generalizado. En teoría podría ser tanto un puerto TCP/UDP como uno de cualquier otro protocolo de nivel de aplicación, aunque en la práctica, y para la versión 1 de RSVP, se utilizan los puertos TCP/UDP.

Todo mensaje RSVP contiene un objeto de tipo SESSION que contiene estos datos.

Como puede comprobarse, una sesión identifica inequívocamente a una aplicación receptora (identificada por el campo *DstPort*) en un host determinado (identificado por la pareja *DstAddress+ProtocolId*).

### 3.8.4.2 Descriptor de Flujo

Una petición de reserva RSVP para una sesión determinada siempre incorpora un descriptor de flujo, dado que es la unidad de información necesaria para definir la calidad de servicio para un conjunto de datos del emisor.

Un descriptor de flujo se compone de dos conjuntos de parámetros:

- **FlowSpec** (Especificación del flujo) Define la calidad de servicio deseada, y aunque es imprescindible para RSVP, no depende propiamente del protocolo, sino del tipo de servicio utilizado (alguno de los definidos en el 'modelo de servicios integrados') De hecho, este parámetro es transparente para RSVP. Se divide en dos parámetros:
  - **Rspec** (Especificación de la Reserva) Describe propiamente la calidad de servicio requerida.
  - **Tspec** (Especificación del Tráfico) Describe las características del flujo de datos.
  
- **FilterSpec** (Especificación de Filtro) Define el conjunto de paquetes de datos que deberá recibir la calidad de servicio definida por el FlowSpec. Existen formatos diferentes para IPv4 y IPv6. Puede seleccionar arbitrariamente qué paquetes recibirán la QoS indicando:
  - Paquetes de un emisor dado.
  - Paquetes de un protocolo de alto nivel dado.
  - Paquetes en función de un campo de la cabecera

De todas maneras, en la versión actual se simplifica a sólo la dirección IP del emisor y opcionalmente el puerto TCP/UDP.

Dada su naturaleza, el *filterspec* es información necesaria para el módulo Clasificador de paquetes, ya que este módulo es el encargado de determinar si un paquete pertenece a alguna de las reservas instaladas en el nodo. De la misma manera, el *flowspec* es una información necesaria para el Planificador de paquetes, ya que es el encargado de implementar activamente la calidad de servicio (planificando en cada momento qué paquete debe salir por un interfaz de salida).

### 3.8.4.3 Reenvío Salto a Salto

Una de las características destacables del protocolo RSVP es el hecho de ser independiente del protocolo de encaminamiento usado por la red.

Esta característica a parte de ser deseable para cualquier nuevo protocolo de red, es necesaria para un protocolo de reserva de recursos. No tendría sentido que se estableciera una reserva de una cierta calidad de servicio en un camino determinado, y que a la hora de enviar los paquetes de datos estos no siguieran dicho camino (por ejemplo porque el algoritmo de encaminamiento decidiera una ruta diferente entre el origen y el destino). RSVP solventa este problema haciendo un envío de mensajes 'salto a salto' (*hop to hop*) el cual tiene tres fases diferenciadas:

1. **Creación de la ruta.** El emisor envía un mensaje PATH al receptor, el cual se encamina por la red siguiendo los algoritmos de encaminamiento propios de la misma. En cada nodo RSVP por el que pasa este mensaje se anota desde qué nodo ha llegado.

2. **Petición de reserva.** Por su parte, un receptor realiza una petición de reserva mediante un mensaje RESV. Este mensaje se envía salto a salto hacia el emisor siguiendo el estado de ruta que creó el anterior mensaje PATH. Si se enviara directamente al emisor, el mensaje podría ser encaminado por una ruta diferente. Cada nodo RSVP al que llega un mensaje RESV realiza la reserva localmente, consulta cual es el salto anterior (Previous Hop) y reenvía el mensaje a dicho nodo. De esta manera se asegura que los mensajes de reserva seguirán exactamente el camino inverso.
3. **Envío de datos.** Cada nodo RSVP que recibe un paquete de datos susceptible de pertenecer a una de las reservas que tiene instaladas, debe reenviar el paquete de datos siguiendo el camino en el que se ha instalado la reserva, evidentemente también salto a salto (sin dar posibilidad al protocolo de encaminamiento de cambiar dicho camino).

#### 3.8.4.4 Estilos de Reserva

RSVP incorpora una novedosa característica de diseño por la cual la reserva de recursos (*flowspec*) está separada de para quién se hace dicha reserva (*filterspec*).

Las especificaciones de filtro, o *filterspec*, pueden cambiarse dinámicamente por los receptores durante la vida de la reserva, sin necesidad de cambiar la calidad de servicio que reciben.

Existen diferentes maneras de especificar los receptores de los cuales se quiere recibir datos, lo que se conoce por 'estilos de reserva'.

Desde el punto de vista del tipo de reservas que se efectúan, una reserva puede ser de dos tipos:

- *Shared* (compartida). Todas las reservas para cada emisor deben ser tomadas como una sola.
- *Distinct* (diferenciada). Para cada emisor existe una reserva diferenciada del resto.

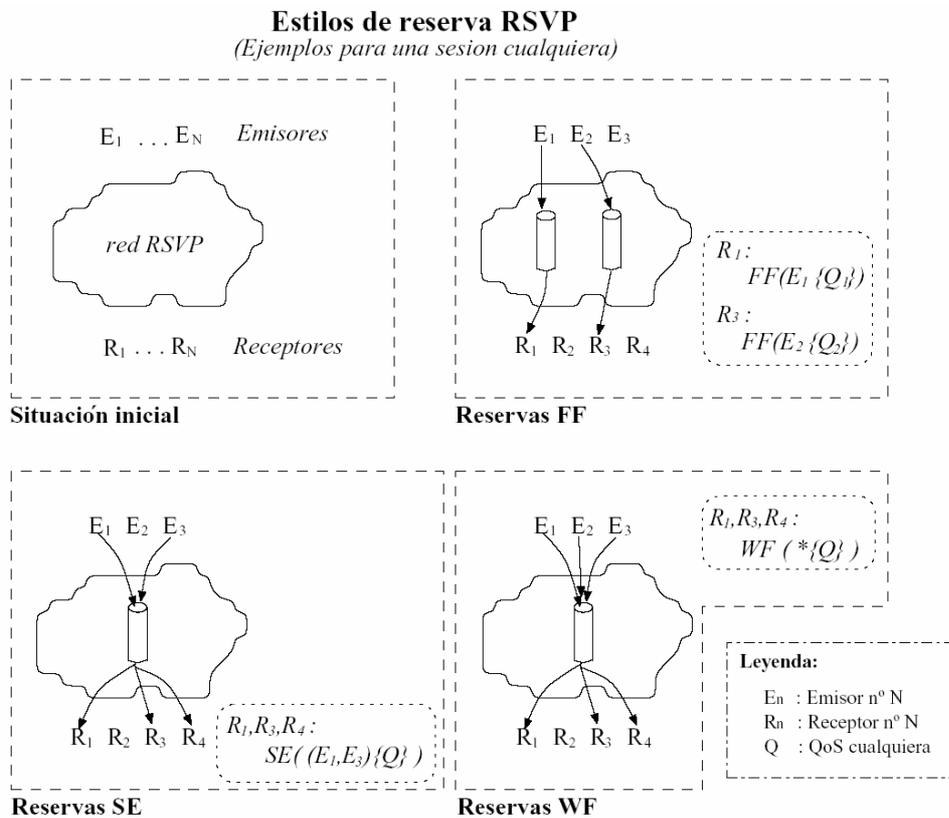
Y desde el punto de vista de cómo se especifican los emisores una reserva puede ser igualmente de uno de estos dos tipos:

- *Explicit* (explícita). Cada emisor debe especificarse explícitamente (dando su dirección IP y puerto)
- *Wildcard* (por patrón). Los emisores se seleccionan implícitamente mediante un patrón.

Mezclando estas características, se obtienen los estilos de reserva que ofrece RSVP. Existen tres estilos de reservas en RSVP (la combinación *distinct* y *wildcard* no tendría sentido), son los siguientes:

- FF: *Fixed-Filter* (filtro fijo). Los emisores se seleccionan de forma explícita y se hacen reservas separadas por cada uno de ellos. Se crean reservas diferentes para cada emisor.
- SE.- *Shared-Explicit* (explícito compartido). Los emisores se seleccionan de forma explícita pero las reservas se hacen de forma compartida. Conceptualmente es como una tubería (*pipe*) compartida por varios emisores los cuales son seleccionados uno a uno por los receptores.
- WF: *Wildcard-Filter* (por patrón fija). Los emisores se seleccionan de forma implícita (por patrón) y la reserva es compartida por todos los emisores seleccionados. Se crea una única reserva que es compartida por todos los

emisores, conceptualmente puede verse como una tubería (*pipe*) en la que pueden escribir todos los emisores y de la que pueden leer varios receptores.



La figura muestra conceptualmente los tres estilos de reserva disponibles. El estilo de reserva WF es el más voraz en cuanto a recursos necesarios y es automáticamente extendida en el momento que aparecen nuevos emisores. El estilo SE es una restricción del anterior en el cual los emisores se seleccionan explícitamente. El estilo FF, por último, es el más restrictivo de todos, ya que no permite que las reservas sean compartidas por varios emisores.

Las reservas compartidas (*shared*) son apropiadas para aplicaciones *multicast* donde es improbable que los diferentes emisores transmitan al mismo tiempo (por ejemplo una aplicación de audio). En cambio, las reservas diferenciadas (*distinct*) son más apropiadas

para servicios como el vídeo, donde cada receptor deseará recibir diferentes imágenes de forma simultánea.

### 3.8.5 CREACIÓN DEL CAMINO DE DATOS

El primer paso para la utilización del protocolo RSVP es la creación del camino de datos. Con este paso un emisor inicia el camino que deben seguir los datos hacia el/los receptor/es.

El mensaje RSVP que realiza este cometido es el mensaje *Path*. Cada emisor transmite mensajes *Path* en el sentido del flujo de datos por las rutas *uni/multicast* proporcionadas por el protocolo de encaminamiento. En cada nodo RSVP de la ruta que recibe un mensaje *Path* se guarda un "*path state*" (estado de camino), en el cual se guarda como mínimo la dirección IP del *PHOP* (*Previous Hop*, salto anterior), la cual servirá para encaminar por la ruta inversa los mensajes de reserva que enviarán los receptores.

Además del *PHOP*, el mensaje *Path* contendrá información para definir el tipo de tráfico que el emisor espera generar (guardada en un objeto *SENDER-TSPEC*). Esto servirá a los receptores a la hora de calcular las necesidades de QoS para recibir dicho tráfico de forma efectiva.

El formato concreto del mensaje *Path* aparece en la figura 4.6.

La información recogida en este paso (el '*path state*') se almacena en una estructura de datos llamada *Path State Block*, o PSB (definida en el [RFC 22091]), la cual guarda el estado *path* para una pareja (sesión, emisor) concreta.

Si durante el envío del mensaje *Path* se produjera algún error, se detendría el reenvío de dicho mensaje y se devolvería un error mediante el mensaje *PathErr*. De todas formas hay pocos motivos por los que se pueda producir un error en el envío de un *Path*.

### 3.8.6 REALIZACIÓN DE LA RESERVA

A la recepción de un mensaje *Path* por parte de un receptor, este conoce a un posible emisor para la sesión dada y conoce tanto las características del flujo de datos que generará el primero (mediante el objeto SENDER-TPEC), como las características del camino de datos (mediante el objeto ADSPEC). Con esta información el receptor tiene información suficiente para hacer una reserva.

Las reservas las pide el receptor mediante un mensaje de tipo *Resv*, el cual viaja hacia el/los emisor/es siguiendo el camino inverso al que siguió el correspondiente mensaje *Path*

El formato del mensaje *Resv* se muestra en la figura 4.6. El objeto que describe la reserva propiamente es el etiquetado como '*flow-descr*', el cual tiene formato diferente para cada tipo de estilo de reserva. Las siguientes especificaciones describen el formato de este objeto:

`<flow-descr> ::= <flow-descr-WF> | <flow-descr-SE>`

`<flow-descr-WF> ::= <FLOWSPEC>`

`<flow-descr-SE> ::= <FLOWSPEC> | <lista-filter-spec>`

`<lista-filter-spec> ::= <FILTER-SPEC> | <lista-filter-spec> <FILTER-SPEC>`

`<flow-descr> ::= <FLOWSPEC> <FILTER-SPEC> | <flow-descr> <flow-descr-FF>`

`<flow-descr-FF> ::= <FLOWSPEC> <FILTER-SPEC>`

### 3.8.6.1 Tratamiento en Cada Nodo

Cuando un mensaje *Resv* llega a uno de los nodos RSVP intermedios de la ruta, deben realizarse básicamente dos acciones, las cuales se describen a continuación:

1. Instalar la reserva en el nodo. Esto implica varios pasos, entre los que destacan los tres siguientes:
  - Validación administrativa. El módulo *Policy Control* del nodo identifica al usuario que realiza la petición y lo valida como usuario con permisos suficientes para instalar una reserva en el nodo. Si esta validación fallara se devolvería un mensaje de error (mensaje *ResvErr*)
  - Validación de admisión. El módulo de control de admisión consulta el estado del nodo para averiguar si se dispone de recursos suficientes para instalar la reserva.
  - Realización de la reserva. Si se dispone de recursos para la reserva en curso y la validación administrativa ha resultado válida, se procede a la instalación de la reserva, lo que consiste (entre otras operaciones a nivel local) en el paso del *filterspec* y del *flowspec* de la petición al módulo clasificador (*Classifier*) y al planificador de paquetes (*packet scheduler*) respectivamente.
2. Reenviar la reserva. Una vez instalada la reserva en el nodo local, la reserva debe continuar su camino hacia los emisores. Esto básicamente implica dos pasos:
  - *Merging* de reservas. Si el nodo actual ya tiene instalada una reserva igual o mayor a la solicitada, la petición no se reenvía, y decimos que las reservas han sido fusionadas (*merged*).

- Envío al PHOP. Si no es posible hacer una fusión de reservas, la petición actual debe reenviarse hacia los emisores apropiados. Para ello el nodo recupera la dirección IP (también podrían ser varias) del *Previous Hop*, o salto anterior, y envía la petición a esta dirección.

Los datos de una reserva se guardan en el nodo local mediante una estructura de datos llamada *Reservation State Block* (RSB, definida en el [RFC 22091]). Esta estructura de datos guarda una petición de reserva para la tripleta: (Sesión, *next hop*, *filter spec - list*). Donde "*filter\_spec\_list*" será una lista de *FILTER-SPECs* (en estilo SE), un único *FILTER-SPEC* (el] estilo FF) o una lista vacía (en estilo WF).

Si no puede instalarse la reserva en un nodo debe informarse del error al receptor que inició la reserva. Para ello se utiliza el mensaje *ResvErr*, el cual viaja desde el nodo que produjo el error hasta el nodo receptor que envió la petición.

### **3.8.7 LOS '*SOFT STATE*'**

Tanto el estado *path* como el estado de reserva en cada nodo del camino deben ser refrescados periódicamente dado que solo son válidos durante un cierto período de tiempo, por eso se dice que los estados RSVP en los nodos son '*soft*' (blandos) en el sentido de que deben refrescarse continuamente.

Asociado con cada estado *path* y con cada estado de reserva existen dos valores de tiempo:

- "*Refresh timeout*" Indica el intervalo de tiempo en el que se debe refrescar el estado *path* del salto siguiente (en el caso del estado *path*) o el estado de reserva del salto anterior (en el caso del estado de reserva).
- "*Cleanup timeout*" Indica el intervalo de tiempo durante el que el estado asociado es válido. En otras palabras, el estado *path* o de reserva dejará de ser válido si no recibe un refresco antes de que expire el *cleanup timeout*"

Esta elegante aproximación permite, por ejemplo, que si el camino cambiase a la hora del reenvío de un mensaje *Path* (por un cambio en el algoritmo de encaminamiento), el camino antiguo se destruiría automáticamente transcurrido el tiempo de *cleanup*, y todo ello sin afectar ni al envío de datos ni al envío de mensajes de reserva (los cuales seguirían el nuevo camino)

### 3.8.7.1 Funcionamiento de los Refrescos

Tanto el mensaje *Path* como el mensaje *Resv* contienen un parámetro que corresponde con el "*refresh timeout*" Este valor se guarda en cada nodo intermedio junto con el "*cleanup timeout*" el cual es calculado como un cierto número de veces el período de refresco (el motivo de lo cual se explicará más adelante)

Cada nodo intermedio del camino tiene la misión de refrescar al nodo siguiente tras la expiración del período de refresco. Es decir, supongamos que el período de refresco de un cierto estado *path* ha expirado en un nodo; en este caso dicho nodo debe refrescar al NHOP (salto siguiente en el sentido de los datos) mediante el envío de un mensaje *Path* (lo cual es equivalente para el estado de reserva y los mensajes *Resv*).

El emisor es el encargado de refrescar el estado *path* del primer nodo del camino. De forma análoga, el receptor es el encargado de refrescar el estado de reserva del último nodo del camino.

Si a la expiración del período de "*cleanup*" no se ha recibido un mensaje de refresco, se elimina el estado (path o de reserva), y se envía un mensaje de cancelación del estado (mediante un mensaje *PathTear* en el caso del estado *path*, o mediante un mensaje *ResvTear* en el caso del estado de reserva)

### 3.8.7.2 Procesado de los Mensajes de Refresco

Cuando un nodo recibe un mensaje *Path*, el procesado del mismo tiene en cuenta los siguientes criterios:

1. Si no existe un estado *path* relacionado, lo crea y reenvía el mensaje al NHOP según las tablas de *routing*.
2. Si existe un estado *path* relacionado (es un mensaje de refresco):
  - (a) Restablece el "*cleanup timeout*" (el estado *path* ha sido refrescado)
  - (b) El mensaje *Path* es reenviado al siguiente salto (NHOP) en alguno de los casos siguientes:
    - i. Constituye un cambio en el camino establecido. Esto se detecta consultando las tablas de *routing* del nodo.
    - ii. El estado *path* cambia con el nuevo mensaje *Path*.

Igualmente, en cada nodo del camino se envía un mensaje *Path* al NHOP en los siguientes casos:

1. El protocolo de *routing* informa de un cambio en los interfaces de salida.
2. El *timeout* de refresco de un estado *path* ha expirado.

El funcionamiento en el caso de la recepción de un mensaje *Resv* es equivalente:

1. Si no existe un estado de reserva relacionado, lo crea y reenvía el mensaje al PHOP según la información del estado *path* asociado.
2. Si existe un estado de reserva relacionado (es un mensaje de refresco):
  - (a) Restablece el "*cleanup timeout*" (el estado de reserva ha sido refrescado)
  - (b) El mensaje *Resv* es reenviado al salto previo (PHOP) en el caso siguiente:
    - i. Constituye un cambio en el estado actual (por ejemplo como consecuencia de una petición de aumento de la QoS de una reserva ya instalada)

En cada nodo del camino se envía automáticamente un mensaje *Resv* al PHOP en el siguiente caso:

1. El *timeout* de refresco del estado de reserva ha expirado.

### 3.8.7.3 Cancelación de los Estados Soft

A la expiración del "*cleanup timeout*" tanto para un estado *path* como para un estado de reserva deben hacerse los siguientes pasos:

3. Eliminar el estado responsable en el nodo local.
4. Destruir los estados relacionados en el camino mediante el envío de un mensaje de "*teardown*". Este mensaje será un *PathTear* en el caso de expiración del estado *path*, o será un *ResvTear* en el caso de expiración de un estado de reserva.

Un mensaje de cancelación de estado destruye el estado en el nodo local y debe ser reenviado al siguiente nodo salto a salto y sin retardo.

La expiración de este *timeout* puede ocurrir por varias circunstancias:

5. Se ha perdido un mensaje de refresco. Esto es posible dado que los mensajes RSVP son entregados sin garantías, por lo que es posible que se pierda alguno de ellos. Este caso, que en principio es un grave problema, puede aliviarse fácilmente con las siguientes actuaciones:
  - (a) Hacer que el "*cleanup timeout*" sea varias veces superior al "*refresh timeout*". Supongamos que el período de *cleanup* es 3 veces el de refresco, serán necesarias tres pérdidas de mensajes de refresco para que expire el *timeout* de cancelación (lo cual es posible pero poco probable)

- (b) Reservar una pequeña cantidad de ancho de banda en los enlaces para los mensajes RSVP. Esta reserva actúa como protección frente a las eventuales pérdidas por congestión.
6. Han cambiado las rutas en el algoritmo de *routing*. Es posible que al envío de un mensaje *Path* de refresco se detecte que el NHOP guardado en el estado *path* ya no concuerde con el algoritmo de *routing*, es decir, es posible que hayan cambiado las rutas. Bajo estas circunstancias ciertos nodos del camino original no recibirán nunca los refrescos, con lo que expirado el *timeout* cancelaran sus estados.
  7. Otros problemas como caída de nodos, etc. En estos casos tanto emisor como receptor serán informados del problema mediante la llegada de los correspondientes mensajes de cancelación.

### 3.8.8 ONE PASS WITH ADVERTISING (OPWA)

Los mensajes *Path*, que son los mensajes que inicializan el camino de datos, pueden contener un tipo de información opcional (materializada mediante el objeto ADSPEC). La información que contiene el *Adspec* recoge las características, en cuanto a QoS, del camino de datos, y es una información muy útil para los receptores, puesto que les informa de la QoS que puede ofrecerles el camino de datos.

Cuando los mensajes *Path* contienen este tipo de información, se dice que RSVP funciona en modo OPWA ("*One Pass with Advertising*")

El hecho de que se introduzca esta información en los mensajes *Path* es importante desde el punto de vista del *overhead* que genera el protocolo RSVP. Supongamos que los receptores no disponen de la información del *Adspec* (y, por tanto, no tienen idea de la calidad de servicio que el camino de datos les puede ofrecer) En esta situación cuando un receptor desee realizar una reserva tendrá que realizarla "a ciegas", es decir, ir probando hasta que una reserva le sea aceptada.

### 3.8.8.1 La Información del *Adspec*

La información que contiene el *Adspec* informa a los receptores de las características extremo a extremo del camino de datos, entendiendo como características la calidad de servicio soportada por el camino así como los tipos de servicio disponibles (véase sección 5 para una descripción de estos servicios)

- La información del *Adspec* contiene datos generales del camino y datos dependientes del servicio ofrecido (*carga controlada*, *garantizado*) Los datos generales incluyen:
- Latencia mínima del camino. Contiene la latencia del camino de datos en ausencia de retardos de espera en los nodos (es decir, básicamente la suma de las latencias individuales de cada enlace) En el servicio *garantizado* los receptores pueden sumar a este valor el límite del retardo en colas (valor relacionado con este tipo de servicio en concreto, véase sección 5.3 para más información), y de esta manera tener una cota máxima del retardo que se introducirá en los paquetes de datos.

- Ancho de banda del camino. Este valor indica el ancho de banda disponible para reservas en todos los enlaces del camino. Contiene el ancho de banda más pequeño de entre todos los enlaces individuales que componen el camino de datos.
- Global *Break bit*. Es un bit que indica si hay algún nodo en el camino que no soporta RSVP.
- Cuenta de los saltos IS (*Integrated Services*). En un valor que se incrementa por cada uno de los nodos del camino que soportan el protocolo RSVP y el modelo de Servicios Integrados.
- MTU del camino. Informa del MTU mínimo de entre todos los de los enlaces individuales del camino. Es importante dado que no debe permitirse la fragmentación de los paquetes de un flujo de datos con reserva RSVP.

Para un servicio *garantizado* el *Adspec* contiene además la siguiente información:

- *Ctot*. El valor del parámetro C compuesto de todo el camino. Los valores C y D son parámetros del servicio *garantizado* que ayudan a modelar el retardo extremo a extremo (véase la sección 5.3 para más información)
- *Dtot*. Valor del parámetro D compuesto de todo el camino.
- *Csum*. Valor compuesto de C desde el último punto de "*reshaping*".
- *Dsum*. Valor compuesto de D desde el último punto de "*reshaping*".

- *Guaranteed Service Break Bit*. Informa si hay algún salto en el camino de datos que no implemente el servicio garantizado.

Para un servicio *carga controlada* se incluye la siguiente información:

- *Controlled-Load Service Break Bit*. Informa si hay algún salto en el camino de datos que no implemente el servicio *carga controlada*.

### **3.8.9 ZONAS NO RSVP**

Un nuevo protocolo de red no puede instalarse en todos los nodos de la red al mismo tiempo, y menos en una red de las dimensiones de Internet. La implantación del nuevo protocolo, una vez estandarizado, se realizará por fases, desde una primera implantación en Intranets y redes de mediano tamaño e investigación, hasta la totalidad de los nodos de la red. En las fases intermedias de la implantación de RSVP, los nodos que implementen el protocolo deberán convivir con otros que no lo implementen.

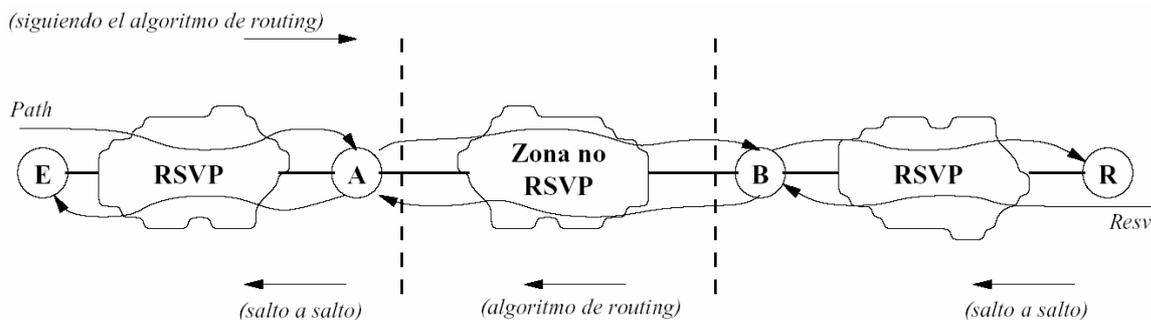
Esto, que en principio es un problema, se ha tenido en cuenta en el diseño del protocolo RSVP, de tal manera que el protocolo está preparado para detectar zonas intermedias que no implementen el protocolo, y para funcionar incluso en esas circunstancias.

Evidentemente una zona no RSVP en un camino de datos susceptible de hacer una reserva de recursos afecta a la calidad de servicio que puede ofrecerse. Esto es así porque en los nodos de dicha zona no RSVP no hay políticas de planificación del enlace que garanticen al flujo una calidad de servicio, y si las hay, no se implementa el protocolo para instalar las reservas.

El protocolo RSVP informa a un receptor de que en el camino de datos hay una zona no RSVP, información que el receptor utilizará de manera conveniente.

### 3.8.9.1 Gestión de las Zonas no RSVP por el Protocolo

La siguiente figura muestra una posible situación de zona no RSVP. En este ejemplo el camino de datos pasa por una zona no RSVP delimitada por los nodos A y B que sí implementan el protocolo.



Cuando el emisor envía un mensaje *Path* para inicializar el camino de datos, dicho mensaje no se ve afectado por la zona RSVP porque, de hecho, este mensaje no necesita del protocolo RSVP para ser encaminado (recuérdese que RSVP utiliza los algoritmos de encaminamiento propios de los nodos). Cuando el mensaje *Path* llega al nodo A se guarda el estado *path* (la dirección TP del nodo anterior) y se reenvía por los mecanismos de *forwarding* del nodo. Este mensaje se encamina hacia el receptor llegando al siguiente nodo B (habiendo pasado por N nodos no RSVP anteriores) El nodo B tiene constancia de que el último nodo RSVP fue A, con lo que en su estado *path* guarda la dirección IP de A como nodo anterior (o salto previo), y dado que no ha recibido el paquete directamente del mismo nodo A, deduce que se ha pasado por una zona no RSVP y activa el *flag NonRSVP* en la cabecera del mensaje. El mensaje llega al receptor de forma normal.

Cuando el receptor envía un mensaje de reserva, dicho mensaje llega a B salto a salto por la zona RSVP. A B le consta que el salto RSVP anterior fue A, con lo que reenvía el mensaje poniendo como dirección destino la dirección TP de A. El nuevo mensaje se encamina hasta A (nótese que, dado que se encamina mediante los algoritmos de encaminamiento de la red, es posible que no siga el mismo camino que el Path que le precedía). Desde A se encamina el mensaje salto a salto hacia el emisor.

### 3.8.9.2 El Impacto en la Calidad de Servicio

Es evidente que una zona no RSVP en un camino de datos con reserva tiene un impacto en la calidad de servicio obtenida por el flujo. ¿Cuánto afecta este hecho en la calidad de servicio obtenida? Bien, depende de varios factores:

- Del tipo de servicio requerido. De un servicio de carga controlada se espera que entregue los datos como en un servicio *best-effort* sin sobrecarga (véase el capítulo 5), mientras que de un servicio garantizado se espera, además, ciertas garantías sobre el retardo máximo que la red introducirá en la entrega de los paquetes. El servicio carga controlada exige menos de la red, es de esperar que este tipo de servicio se vea menos afectado por una zona no RSVP.
- Del tráfico de la zona RSVP. Una zona RSVP garantiza el servicio para los flujos reservados (mediante los correspondientes mecanismos de planificación de enlaces), incluso bajo situaciones de alta carga de tráfico. Pero esto no pasa en la zona no RSVP, por lo que los paquetes de los flujos reservados son tratados como el resto del tráfico, y sobretodo, no pasa en situaciones de alta carga. Por tanto, la carga de tráfico en la zona no RSVP es un parámetro muy importante que influirá en la pérdida de calidad de servicio global.

### 3.8.10 FORMATO DE LOS MENSAJES RSVP

Los mensajes RSVP son enviados encapsulados dentro de datagramas IP (aunque el estándar también prevé la posibilidad de encapsulación dentro del protocolo de transporte UDP, véase apéndice C del RFC 2205).

El formato general de un mensaje RSVP se divide en dos partes, una cabecera común de tamaño fijo (64 bits) y una secuencia variable de objetos RSVP (los cuales, a su vez, pueden tener también tamaño variable)

Existen diferentes tipos de objetos RSVP. Estos objetos contienen la información propiamente dicha del mensaje.

#### 3.8.10.1 La Cabecera Común

La cabecera común es el primer objeto de cualquier mensaje RSVP y contiene los campos estrictamente necesarios para identificar el tipo de mensaje. Estos campos son los siguientes:

- *Vers*: La versión del protocolo (actualmente 1)
- *Flag*: Actualmente no tiene uso definido.
- Tipo *Mens*: Tipo de mensaje RSVP. Actualmente se han definido los siguientes:

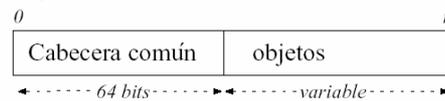
1. *Path*

2. *Resv*

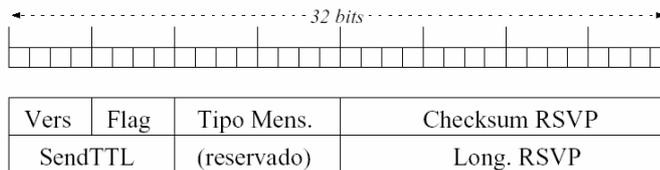
3. *PathErr*
4. *ResvErr*
5. *PathTear*
6. *ResvTear*
7. *ResvConf*

- *Checksum* RSVP: Suma de comprobación (puede valer 0 para indicar que no se transmite *checksum*)
- *SendTTL*: Valor inicial del campo TTL del datagrama IP.
- Longitud RSVP: Longitud total en bytes incluyendo la cabecera común.

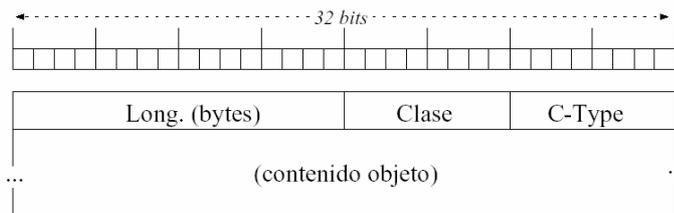
**Mensaje RSVP**



**Cabecera común**



**Objeto RSVP**



### 3.8.10.2 Resumen de los Mensajes RSVP

RSVP, en su versión 1, incorpora un total de 7 tipos de mensajes diferentes cuya utilización se resume a continuación:

- 1) Path. Lo genera los emisores de una sesión y se transmiten en el sentido del flujo de datos hacia los receptores. Este mensaje inicia la ruta que deberán seguir en el futuro el resto de mensajes RSVP.
- 2) Resv. Este mensaje lo genera un receptor que desea recibir datos de uno o varios emisores. RSVP transporta los mensajes Resv siguiendo la ruta que crearon los mensajes Path anteriormente enviados por los emisores, pero en sentido inverso.
- 3) PathErr. Informa de los errores que pueda provocar un mensaje Path. Se envía hacia el emisor que generó el error, y no cambia el estado path de los nodos por los que pasa.
- 4) ResvErr. Informa de que una petición de reserva ha fallado en alguno de los nodos del camino. El procesado de este mensaje es bastante complejo puesto que hay que asegurar que una reserva, que sería aceptada por el control de admisión una vez fusionada con otra mayor no sea rechazada (a este problema se le llama *Killer reservation*)
- 5) PathTear. Elimina los estados Path de todos los nodos del camino. Viaja de emisor a receptores en el sentido del flujo de datos.
- 6) ResvTear. Elimina el estado de reserva en todos los nodos del camino de datos, es enviado desde un receptor hacia todos los emisores, en el sentido contrario al flujo de datos.
- 7) ResvConf. Cuando se realiza una reserva (mediante el mensaje Resv) puede pedirse que se genere una confirmación. En este caso se enviará al receptor que lo solicitó, un mensaje ResvConf desde el último nodo que reciba el mensaje de reserva (puede

ser el nodo emisor, si no hay merging antes, o el nodo donde se hizo merging de la reserva enviada). Viaja en el sentido del flujo de datos.

### Mensajes RSVP

#### Path (1)

Cabecera común	INTEGRITY	SESSION	RSVP-HOP	TIME VALUES	POLICY DATA	...	POLICY DATA	SENDER TEMPLATE	SENDER TSPEC	ADSPEC
----------------	-----------	---------	----------	-------------	-------------	-----	-------------	-----------------	--------------	--------

#### Resv (2)

Cabecera común	INTEGRITY	SESSION	RSVP-HOP	TIME VALUES	RESV CONFIRM	SCOPE	POLICY DATA	...	POLICY DATA	STYLE	<i>flow-descr</i>
----------------	-----------	---------	----------	-------------	--------------	-------	-------------	-----	-------------	-------	-------------------

$$\text{flow-descr} : \begin{cases} \text{Estilo WF:} & \boxed{\text{FLOWSPEC}} \\ \text{Estilo SE:} & \boxed{\text{FLOWSPEC}} \mid \boxed{\text{FILTERSPEC 1}} \mid \dots \mid \boxed{\text{FILTERSPEC n}} \\ \text{Estilo FF:} & \boxed{\text{FLOWSPEC 1}} \mid \boxed{\text{FILTERSPEC 1}} \mid \dots \mid \boxed{\text{FLOWSPEC n}} \mid \boxed{\text{FILTERSPEC n}} \end{cases}$$

#### PathErr (3)

Cabecera común	INTEGRITY	SESSION	ERROR SPEC	POLICY DATA	...	POLICY DATA	SENDER TEMPLATE	SENDER TSPEC	ADSPEC
----------------	-----------	---------	------------	-------------	-----	-------------	-----------------	--------------	--------

#### ResvErr (4)

Cabecera común	INTEGRITY	SESSION	RSVP-HOP	ERROR SPEC	SCOPE	POLICY DATA	...	POLICY DATA	STYLE	<i>error-flow-descr</i>
----------------	-----------	---------	----------	------------	-------	-------------	-----	-------------	-------	-------------------------

$$\text{error-flow-descr} : \begin{cases} \text{Estilo WF:} & \boxed{\text{FLOWSPEC}} \\ \text{Estilo SE:} & \boxed{\text{FLOWSPEC}} \mid \boxed{\text{FILTERSPEC 1}} \mid \dots \mid \boxed{\text{FILTERSPEC n}} \\ \text{Estilo FF:} & \boxed{\text{FLOWSPEC}} \mid \boxed{\text{FILTERSPEC}} \end{cases}$$

#### PathTear (5)

Cabecera común	INTEGRITY	SESSION	RSVP-HOP	SENDER TEMPLATE	SENDER TSPEC	ADSPEC
----------------	-----------	---------	----------	-----------------	--------------	--------

#### ResvTear (6)

Cabecera común	INTEGRITY	SESSION	RSVP-HOP	SCOPE	STYLE	<i>flow-descr</i>
----------------	-----------	---------	----------	-------	-------	-------------------

#### ResvConf (7)

Cabecera común	INTEGRITY	SESSION	ERROR SPEC	RESV CONFIRM	STYLE	<i>flow-descr</i>
----------------	-----------	---------	------------	--------------	-------	-------------------

### 3.8.11 LOS OBJETOS RSVP

Un mensaje RSVP contiene una secuencia variable de objetos RSVP. Cada objeto RSVP contiene un tipo de información que tendrá un uso determinado para cada tipo de mensaje.

El formato de cualquier objeto RSVP es el mostrado en la figura de la página anterior. Contiene una parte fija (de 32 bits) la cual identifica al objeto, y una parte de tamaño variable que alberga el contenido del objeto. Los campos de un objeto RSVP son los siguientes:

- Long: Longitud total del objeto (en bytes). Debe ser múltiplo de 4.
- Clase: Clase del objeto (identifica al objeto)
- C-Type: Subtipo del objeto dentro de la clase.

Una implementación RSVP debería reconocer los siguientes objetos:

- NULL. El contenido de este objeto será ignorado. Podrá aparecer en cualquier parte del mensaje.
- SESSION. Identifica la sesión. Contiene una dirección IP, un identificador del protocolo IP utilizado y un puerto de destino. Todo mensaje RSVP deberá contenerlo.
- RSVP-HOP. Contiene la dirección IP del nodo RSVP que envía el mensaje.
- TIME-VALUES. Valor para el período de refresco usado por el generador del mensaje (R).
- STYLE. Indica el estilo de reserva usado (FF, SE o WF).
- FLOWSPEC. Define la QoS deseada. Tiene formatos diferentes para cada estilo de reserva.
- FILTER-SPEC. Define el subconjunto de paquetes de datos que recibirán la QoS definida por un FLOWSPEC.
- SENDER-TEMPLATE. Dirección IP del emisor y puerto de origen o etiqueta de flujo. Identifica al emisor.

- SENDER-TSPEC. Define las características del flujo de datos que generará el emisor. El formato depende del tipo de servicio que se implemente en la red.
- ADSPEC. Recoge estadísticas en los mensajes Path para informar a los receptores de las capacidades reales de la red.
- ERROR-SPEC. Especifica el tipo de error en mensajes PathErr o ResvErr, y una confirmación en el mensaje ResvConf.
- POLICY-DATA. Datos para decidir si una reserva es administrativamente permitida, es decir, si el usuario dispone de privilegios suficientes para instalar una reserva en el nodo local. Actualmente su formato no ha sido definido.
- INTEGRITY. Contiene datos criptográficos para autenticar al nodo origen y verificar la integridad del mensaje.
- SCOPE. Lista explícita de emisores hacia los que debe reenviarse el mensaje.
- RESV-CONFIRM. Dirección IP de un receptor que solicitó confirmación. En la sección 4.8.5 se discute el contenido exacto de cada uno de los objetos RSVP.

### 3.8.11.1 Composición de los Mensajes RSVP

Como anteriormente se describe, un mensaje RSVP está compuesto por la cabecera común más una secuencia de objetos.

Esta secuencia se define para cada tipo de mensaje RSVP. Dentro de un mensaje RSVP existen objetos que son de obligada aparición y objetos opcionales. La tabla anterior indica el contenido de cada uno de los mensajes RSVP, la figura siguiente muestra la ordenación recomendada de estos objetos dentro de cada mensaje (aunque este orden podría ser modificado sin repercutir en el funcionamiento del protocolo)

	<i>Path</i>	<i>Resv</i>	<i>PathErr</i>	<i>ResvErr</i>	<i>PathTear</i>	<i>ResvTear</i>	<i>ResvConf</i>
<i>NULL</i>	<i>opc.</i>	<i>opc.</i>	<i>opc.</i>	<i>opc.</i>	<i>opc.</i>	<i>opc.</i>	<i>opc.</i>
<i>SESSION</i>	✓	✓	✓	✓	✓	✓	✓
<i>RSVP-HOP</i>	✓	✓		✓	✓	✓	
<i>TIME-VALUES</i>	✓	✓					
<i>STYLE</i>		✓		✓		✓	✓
<i>FLOWSPEC</i>		✓		✓		✓	✓
<i>FILTER-SPEC</i>		✓		✓		✓	✓
<i>SENDER-TEMPLATE</i>	✓		<i>opc.</i>		<i>opc.</i>		
<i>SENDER-TSPEC</i>	✓		<i>opc.</i>		<i>opc.</i>		
<i>ADSPEC</i>	<i>opc.</i>		<i>opc.</i>		<i>opc.</i>		
<i>ERROR-SPEC</i>			✓	✓			✓
<i>POLICY-DATA</i>	<i>opc.</i>	<i>opc.</i>	<i>opc.</i>	<i>opc.</i>			
<i>INTEGRITY</i>	<i>opc.</i>	<i>opc.</i>	<i>opc.</i>	<i>opc.</i>	<i>opc.</i>	<i>opc.</i>	<i>opc.</i>
<i>SCOPE</i>		<i>opc.</i>		<i>opc.</i>		<i>opc.</i>	
<i>RESV-CONFIRM</i>		<i>opc.</i>					✓

### 3.8.11.2 Composición de los Objetos RSVP

Los mensajes RSVP se componen de una secuencia de objetos. Cada objeto RSVP tiene un cometido concreto y transporta parte de la información que precisa el mensaje. En esta sección se detalla la estructura interna de cada objeto RSVP.

Un objeto RSVP se compone de una cabecera que lo identifica y un cuerpo (véase la sección 4.8.3). La identificación del objeto se hace a través de los parámetros Clase y *C-Type* que aparecen en la cabecera (figura 4.6), cuyos valores se listan en la tabla 4.3.

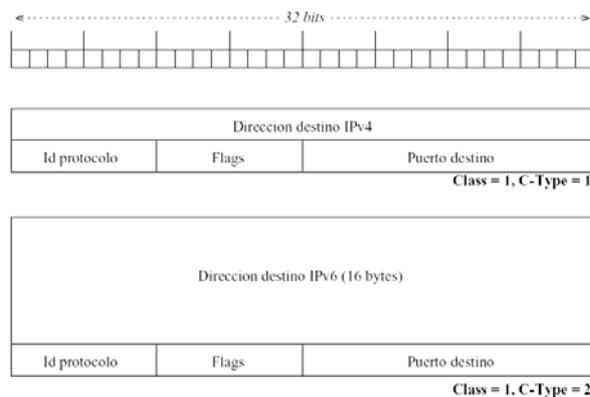
El parámetro *C-Type* se utiliza para diferenciar entre subtipos de objetos, y normalmente diferencia entre la versión del objeto para IPv4 y la versión para IPv6 (aunque en algún caso sirve para diferenciar entre el tipo de servicio requerido: de carga controlada o garantizado)

Objeto	Clase	C-Type
SESSION	1	1, 2
RSVP_HOP	3	1, 2
INTEGRITY	4	?
TIME_VALUES	5	1
ERROR_SPEC	6	1, 2
SCOPE	7	1, 2
STYLE	8	1
FLWSPEC	9	1, 2
FILTER_SPEC	10	1, 2, 3
SENDER_TEMPLATE	11	1, 2, 3
SENDER_TSPEC	12	2
ADSPEC	13	2
POLICY_DATA	14	1
RESV_CONFIRM	15	1, 2

Las siguientes secciones explican el contenido exacto de cada objeto. Nótese que las siguientes especificaciones solo se centran en el contenido concreto del objeto, dejando de lado la cabecera común (para todos los objetos tiene la misma estructura)

### SESSION

El objeto SESSION identifica a una sesión RSVP. Este objeto aparece en todos los mensajes RSVP necesariamente, dado que informa sobre la sesión a la que corresponde el mensaje. El formato de este objeto se muestra en la figura.

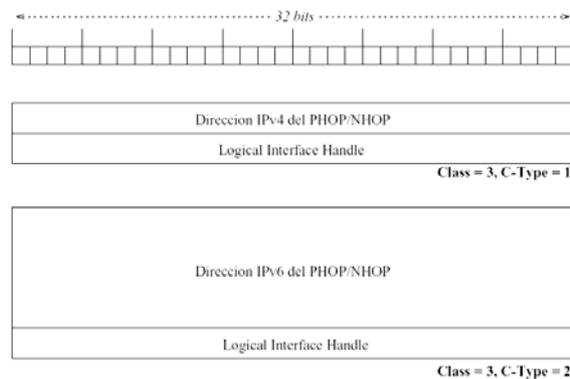


Los parámetros que contienen son los siguientes:

- Dirección destino: La dirección TP (IPv4 o IPv6 dependiendo del C-Type) unicast o multicast del destino de la sesión. Debe ser diferente de 0.
- Id protocolo: El identificador del protocolo TP para el flujo de datos. Debe ser diferente de 0.
- *Flags*: - 0x01 - *E-Police*
- Puerto destino: El puerto UDP/TCP destino para la sesión. Para indicar "ninguno" se dejará a cero.

### RSVP-HOP

Este objeto contiene la dirección IP y el identificador del interface del último salto RSVP conocido que envía el mensaje, depende del sentido del mensaje corresponderá al PHOP (*Previous Hop*) o al NHOP (*Next Hop*). Aparece en todos los mensajes RSVP excepto en el *PathErr* y en el *ResvConf*. Su estructura se muestra en la figura.



Los parámetros que contiene son los siguientes

- Dirección del PHOP / NHOP: Dirección TP del último salto RSVP conocido.

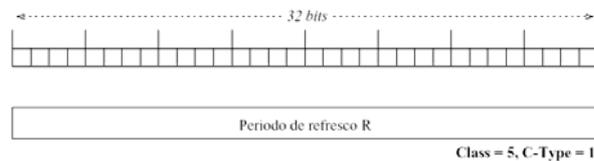
- *Logical Interface Handle* (LIH): Contiene un identificador lógico del interface para identificar el interface por el que se reciben los mensajes de esa sesión. Un valor 0 indica que no hay LIH.

## INTEGRITY

El objeto INTEGRITY no ha sido definido por el momento, es una de las partes de RSVP que se encuentran en estos momentos en estado de definición. Contendrá datos criptográficos para autenticar el nodo origen del mensaje y para validar la integridad del propio mensaje. Puede aparecer en cualquier mensaje RSVP aunque su uso es opcional

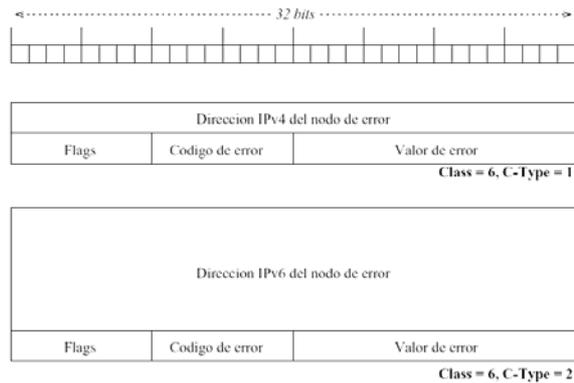
## TIME-VALUES

Este objeto contiene el periodo de refresco R (en milisegundos) usado para generar el mensaje que lo contiene. Aparece en los mensajes *Path* y *Resv*. Su formato es el mostrado en la figura.



## ERROR-SPEC

El objeto ERROR-SPEC especifica el tipo de error en mensajes *PathErr* o *ResvErr*, y la confirmación en el mensaje *ResvConf*. Su formato se muestra en la figura siguiente.

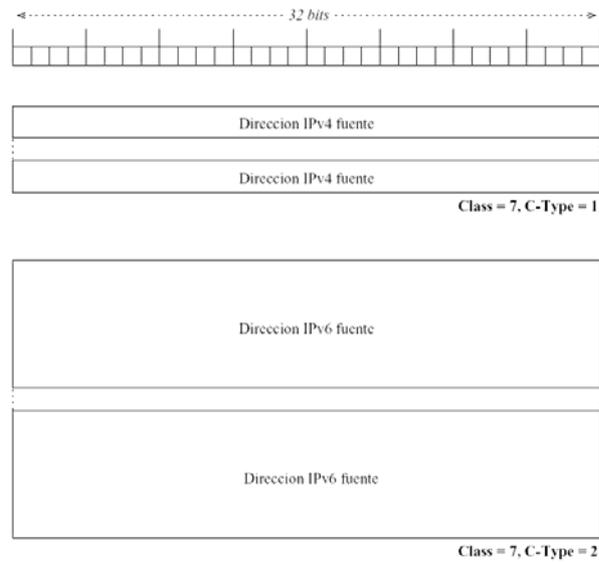


Los parámetros que contiene el objeto ERROR-SPEC son los siguientes:

- Dirección IP del nodo que causó el error.
- *Flags*:
- 0x01 - *InPlace*. Indica que todavía hay una reserva en el punto de fallo. (Solo en
- 0x02 - *NotGuilty*. Indica que el FLOWSPEC que ha fallado es estrictamente mayor que el FLOWSPEC pedido por el receptor. (Solo en mensajes *ResvErr*)
- Código de error: Un byte que describe el error.
- Valor de error: Dos bytes indicando información adicional sobre el error.

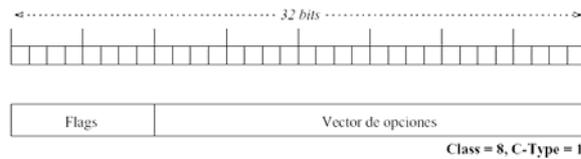
### SCOPE

El objeto SCOPE contiene una lista de direcciones TP, usadas para encaminar mensajes con ámbito *wildcard* sin iteraciones. Las direcciones deben listarse en orden numérico ascendente. Su formato se muestra en la figura:



## STYLE

Este tipo de objeto aparece en todos los mensajes de tipo *Resv* (los que se envían en el sentido inverso al de los datos), es decir, en los mensajes *Resv*, *ResvTear*, *ResvErr* y *ResvConf*. Informa del estilo de reservas que se utiliza en el siguiente objeto (FILTERSPEC), y su formato se muestra en la figura.



Los parámetros que contiene son los siguientes:

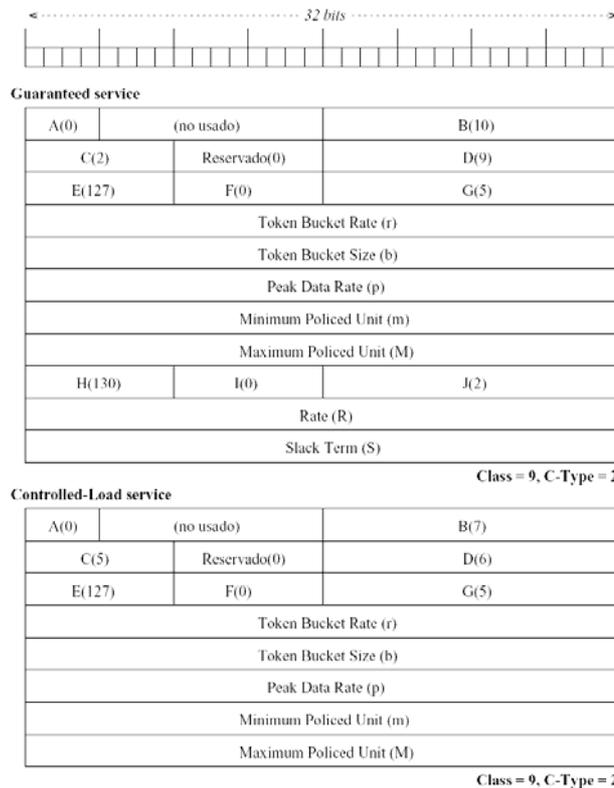
- Flags: (todavía no definidos)
- Vector de opciones: Es un campo de bits en el que se describe el tipo de reserva. Queda abierto para nuevas ampliaciones. Si un nodo no reconociera alguno de los bits, debería ignorarlos y solo interpretar aquellos que conozca. Por ahora se

han definido las siguientes combinaciones (en los bits menos representativos de la palabra de 24 bits):

- Estilo WF: 10001b
- Estilo FF: 01010b
- Estilo SE.- 10010b

### FLAWSPEC

El objeto FLOWSPEC contiene los parámetros que describen la calidad de servicio requerida. Existen formatos diferentes para cada tipo de servicio soportado (por ahora solo *garantizado y carga controlada*). El formato concreto se describe en la figura



El objeto FLOWSPEC tiene el número de clase 9. El FLOWSPEC con *C-Type* 1 es un FLOWSPEC que ha quedado obsoleto con la definición del Modelo de Servicios

Integrados (que engloba los servicios *garantizado* y *carga controlada*), el cual tiene C-Type 2 para los dos tipos de servicios soportados.

El formato de este objeto no es propio de RSVP, sino que se define en el [RFC 22101, el cual introduce el uso de los servicios del Modelo de Servicios Integrados con RSVP.

Los parámetros que aparecen en el FLOWSPEC del servicio garantizado son los siguientes:

- A: Número de versión (0)
- B: Longitud total, 9 palabras sin contar la cabecera.
- C: Cabecera de servicio, servicio número 2 (*garantizado*)
- D: Longitud de los datos del servicio, 9 palabras sin la cabecera por servicio.
- E: Id. del parámetro, parámetro 127 ("*Token Bucket TSpec*")
- F: Flags del parámetro, 127 (no activado)
- G: Longitud del parámetro 127, 5 palabras sin la cabecera del parámetro.
- *Token Bucket Rate* (r), *Token Bucket Size* (b), *Peak Data Rate* (p), *Mininum Policed Unit* (m), *Maximun Packet Size* (M) : Especificación del flujo de datos mediante la descripción de un Un bucket"
- H: Id. del parámetro, parámetro 130 ("*Guaranteed service RSpec*")
- I: *Flags* del parámetro 130, ninguno activado.
- J: Longitud del parámetro 130, 2 palabras sin la cabecera del parámetro.
- Rate (R), Slack Term (8) : Descripción de la calidad de servicio requerida mediante un ancho de banda (R) y un término de relajación del retardo (S). Para más información véase la sección 5.3.

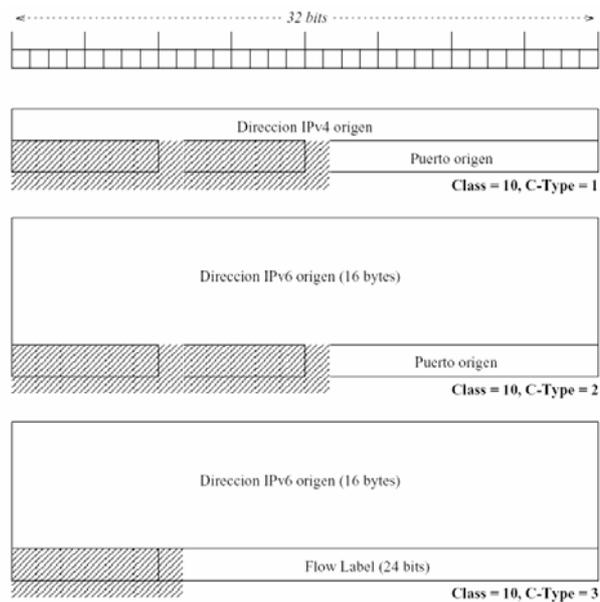
Los parámetros que aparecen en el FLOWSPEC del servicio carga controlada son los siguientes:

- A: Número de versión (0)

- B: Longitud total, 7 palabras sin contar la cabecera.
- C: Cabecera de servicio, servicio número 5 (carga controlada)
- D: Longitud de los datos carga controlada, 6 palabras sin la cabecera.
- E: Id. del parámetro, parámetro 127 ("Token Bucket TSpec")
- F: Flags del parámetro, 127 (no activado)
- G: Longitud del parámetro 127, 5 palabras sin la cabecera del parámetro.
- *Token Bucket Rate (r)*, *Token Bucket Size (b)*, *Peak Data Rate (p)*, *Minimum Policed Unit (m)*, *Maximum Packet Size (M)* : Especificación del flujo de datos mediante la descripción de un "token bucket"

### FILTER-SPEC

Este objeto describe los paquetes que deberán obtener la calidad de servicio descrita en el FLOWSPEC. Su formato es el mostrado en la figura.



Los parámetros que caracterizan a este objeto son:

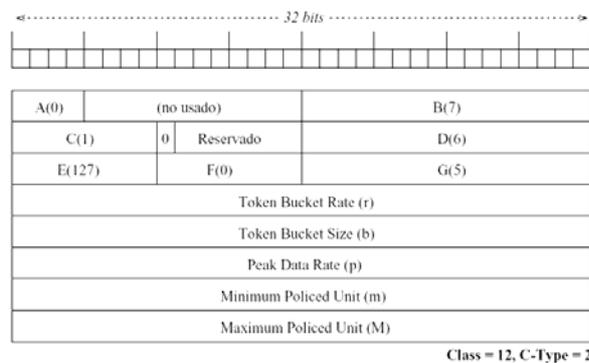
- Dirección IP de origen: La dirección IP de un host emisor. Debe ser diferente de 0.
- Puerto origen: El puerto UDP/TCP de un emisor, un valor cero indica "ninguno".
- Flow Label: Una etiqueta de flujo de 24 bits (definida en IPv6) que identifica a los paquetes pertenecientes a un flujo de datos.

### SENDER.-TEMPLATE

El objeto SENDER-TEMPLATE identifica a un nodo originador de tráfico. El formato del objeto es equivalente al del objeto FILTER-SPEC, aunque con número de clase 12

### SENDER-TSPEC

Este objeto describe las características del flujo de datos que generará el emisor lo cual se hace mediante un *token-bucket*. El formato es este:



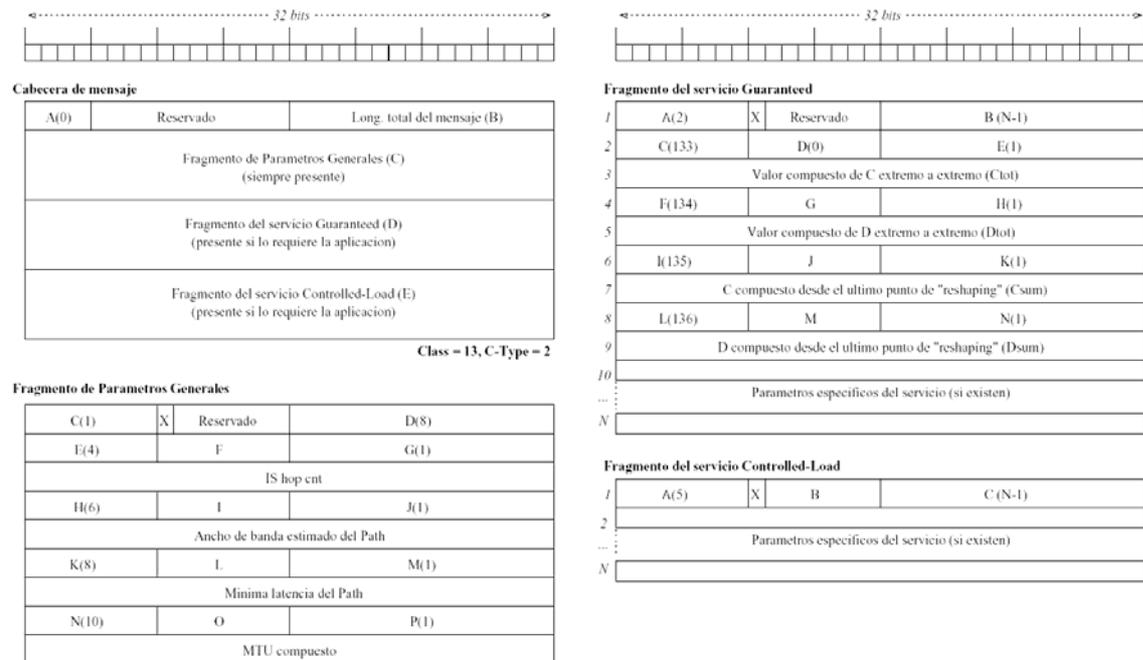
Y los parámetros son:

- A: Número de versión (0)
- B: Longitud total, 7 palabras sin contar la cabecera.
- C: Cabecera de servicio, servicio número 1 (información global por defecto)
- D: Longitud de los datos del servicio, 6 palabras sin la cabecera.
- E: Id. del parámetro, parámetro 127 ("*Token Bucket TSpec*")

- F: Flags del parámetro, 127 (no activado)
- G: Longitud del parámetro 127, 5 palabras sin la cabecera del parámetro.
- Token Bucket Rate (r), Token Bucket Size (b), Peak Data Rate (p), Minimum Policed Unit (m), Maximum Packet Size (M) : Especificación del flujo de datos mediante la descripción de un "token bucket"

**ADSPEC**

El objeto ADSPEC recoge las capacidades de reserva de los nodos del camino de datos. Se utiliza en el mensaje Path para informar a los receptores de las características reales y las posibilidades de reserva el camino de datos.



La estructura del objeto es un tanto compleja Consta de 3 partes:

- Cabecera de objeto global.
- Fragmento de parámetros globales por defecto.
- Secuencia de fragmentos, uno por cada servicio de QoS soportado en el camino.

Son obligatorias las dos primeras partes (cabecera y parámetros globales), mientras que la aparición del resto vendrá dada por las características del camino de datos.

Los parámetros de la cabecera del objeto ADSPEC son los siguientes:

- A: Número de versión (0)
- B: Longitud total del objeto sin contar la cabecera.
- C, D, E : Fragmentos de datos.
- G: Longitud del parámetro 4, 1 palabra sin la cabecera del parámetro.
- IS Hop count: Contabiliza cuantos routers con capacidades RSVP/IS existen en el camino de datos (IS - *Integrated Services*). Definido en el [RFC 2215].
- H: Id. del parámetro, parámetro 6 ("Path-BW")
- I: Flags del parámetro 6, ninguno activado.
- J: Longitud del parámetro 6, 1 palabra sin la cabecera del parámetro.
- Ancho de banda del path: Estimación del ancho de banda del camino de datos, corresponde con el mínimo ancho de banda de los enlaces individuales que componen el camino ([RFC 2215]).
- K: Id. del parámetro, parámetro 8 ("Minimum path latency")
- L: Flags del parámetro 8, ninguno activado.
- M: Longitud del parámetro 8, 1 palabra sin la cabecera del parámetro.
- Mínima latencia del path: Representa la latencia extremo a extremo sin tener en cuenta retardos de encolamiento, es decir, la suma de las latencias de los enlaces individuales que componen el camino. Definido en el [RFC 2215].
- N: Id. del parámetro, parámetro 10 ("Composed path MTU")
- O: Flags del parámetro 10, ninguno activado.
- P: Longitud del parámetro 10, 1 palabra sin la cabecera del parámetro.
- MTU compuesto: El mínimo de los MTUs de los enlaces individuales que componen el camino de datos. Definido en el [RFC 2215].

El fragmento del servicio garantizado contiene los siguientes parámetros:

- A: Cabecera del servicio, servicio número 2 ("garantizado")

- B: Guaranteed Service Break Bit y longitud de los datos del servicio sin incluir la caera.
- C: Id. del parámetro, parámetro 133 ("Ctot compuesto")
- D: Flags del parámetro 133 (no activado)
- E: Longitud del parámetro 133, 1 palabra sin la cabecera del parámetro.
- Valor compuesto de C extremo a extremo.
- F: Id. del parámetro, parámetro 134 ("Dtot compuesto")
- G: Flags del parámetro 134 (no activado)
- H: Longitud del parámetro 134, 1 palabra sin la cabecera del parámetro.
- Valor compuesto de D extremo a extremo.
- I: Id. del parámetro, parámetro 135 ("Csum compuesto")
- J: Flags del parámetro 135 (no activado)
- K: Longitud del parámetro 135, 1 palabra sin la cabecera del parámetro.
- Valor compuesto de C desde el último punto de "reshaping".
- L: Id. del parámetro, parámetro 136 ("Dsum compuesto")
- M: Flags del parámetro 136 (no activado)
- N: Longitud del parámetro 136, 1 palabra sin la cabecera del parámetro.
- Valor compuesto de D desde el último punto de "reshaping"

El fragmento del servicio garantizado contiene los siguientes parámetros:

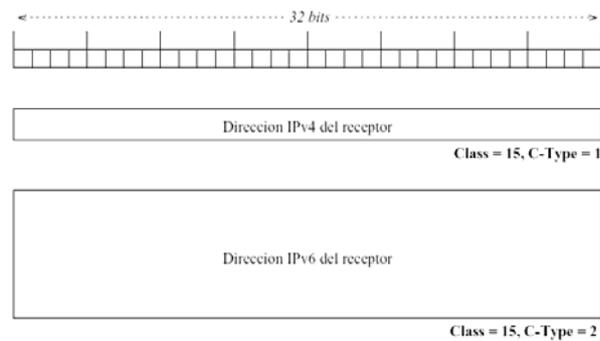
- A: Cabecera del servicio, servicio número 5 ("carga controlada")
- B: Controlled-Load Break bit.
- C: Longitud de los datos del servicio, sin contar la cabecera.

## POLICY-DATA

El objeto POLICY-DATA contiene los datos necesarios para decidir si una reserva es administrativamente permitida, es decir, si el usuario dispone de privilegios suficientes para instalar una reserva en el nodo local. Actualmente su formato no ha sido definido, es uno de los aspectos del estándar que se encuentra en desarrollo.

## RESV-CONFIRM

Contiene, dentro de un mensaje *ResvConf* o *Resv*, la identificación del nodo receptor que ha pedido una confirmación de reserva. Su formato es el siguiente:



## 3.9 EL INTERFAZ PACKET32

Nos va a permitir acceder directamente a los paquetes que llegan al adaptador, independientemente, incluso si no es el destinatario del mismo. Aunque el interfaz viene como ejemplo en [SDDKEx], fue el intento de proporcionar un interfaz similar al *pcap* de Linux [Viano01] el que desarrolló este interfaz. La documentación completa se puede encontrar en ambas citas bibliográficas. Aquí sólo detallaremos la parte del interfaz del que haremos uso sin entrar en un análisis exhaustivo de todas sus estructuras y funciones.

### 3.9.1 ESTRUCTURAS DE DATOS

Las estructuras de datos definidas en *packet32.h* son las siguientes:

- PACKET
- ADAPTER
- Bpf\_insn
- Bpf\_program
- Bpf\_hdr
- Bpf\_stat
- NetType

Las dos primeras son específicas del controlador mientras que el resto se definieron en un intento de proveer a Windows de una herramienta de captura de paquetes común en *Linux* como es el *libpcap*. El resto de estructuras son usadas para realizar operaciones como configurar filtros de captura o interpretar los datos que el controlador en modo kernel pasa a la aplicación que usa esta librería. `PACKET_IOD_DATA` es una estructura usada y cuya definición se encuentra en el fichero *ntddpack.h*. No se especificará toda la documentación de esta herramienta y nos ceñiremos únicamente a las estructuras de datos y funciones usadas.

#### 3.9.1.1 La Estructura PACKET

La estructura `PACKET` se compone de los siguientes campos:

- PVOID Buffer
- UINT Length

- PVOID Next
- UINT ulBytesReceived

*Buffer* es un puntero a una zona de memoria conteniendo datos válidos, *Length* indica el tamaño del buffer. *ulBytesReceived* indica la cantidad de bytes recibidos y que se encuentran en el buffer.

### 3.9.1.2 La Estructura ADAPTER

Esta estructura describe un adaptador de red. Tiene cuatro campos:

- HANDLE hFile
- TCHAR SymbolicLink
- Int NumWrites
- HANDLE ReadEvent

*hFile* es un puntero al controlador y sirve para comunicarlos directamente con él. Aunque la aplicación puede hacer uso de este puntero, el paquete de funciones aporta la funcionalidad necesaria para no tener que hacer uso de él.

*SymbolicLink* es una cadena de caracteres que identifica de manera unívoca el adaptador de red. Esta cadena de caracteres se forma con un prefijo más el identificador de red de Windows.

*NumWrites* es de uso interno del controlador Packet32 y debe considerarse opaco para el usuario.

*ReadEvent* es una notificación de lectura. Puede activarse debido a que se ha recibido un número de bytes mínimo o porque haya expirado el tiempo de espera de paquete. Este

campo tampoco será usado ya que las funciones proveen de la funcionalidad necesaria como para no tener que recurrir a ella.

### 3.9.1.3 La Estructura `bpf_hdr`

Esta estructura define la cabecera usada por el `packer32` para pasar un paquete a la aplicación. Los bytes de esta cabecera son encapsulados junto al paquete capturado y se usa para mantener información importante como la longitud o el tiempo de llegada con precisión de microsegundos. Se compone de los siguientes campos:

- `struct timeval bh_tstamp`
- `UINT bh_caplen`
- `UINT bh_datalen`
- `USHORT bh_hdrlen`

`bh_tstamp` guarda el tiempo de llegada y es a su vez una estructura formada por 2 campos:

- `tv_sec`: guarda los segundos pasados desde el 1-1-1970
- `tv_usec`: microsegundo de la captura

`bh_caplen` es longitud de la porción capturada

`bh_datalen` es la longitud original del paquete.

`bh_hdrlen` es la longitud de la cabecera que encapsula el paquete

## 3.9.2 FUNCIONES

`PACKET32.DLL` proporciona un conjunto completo de funciones para enviar y recibir paquetes, recibir información de la tarjeta de red, abrir y cerrar un adaptador, direccionar dinámicamente la memoria necesaria para almacenar los paquetes, activar y configurar el filtrado y recibir estadísticas de uso. Las funciones usadas en este proyecto son:

- PacketOpenAdapter
- PacketCloseAdapter
- PacketAllocatePacket
- PacketInitPacket
- PacketFreePacket
- PacketReceivePacket
- PacketSendPacket
- PacketSetHwFilter
- PacketSetBuff
- PacketSetReadTimeout
- PacketSetMinToCopy

### **3.9.2.1 LPADAPTER PacketOpenAdapter(LPTSTR AdapterName)**

Esta función recibe una cadena conteniendo el nombre del adaptador que se desea abrir y retorna un puntero a un objeto ADAPTER convenientemente inicializado. El nombre se puede generar concatenando la cadena `\\Device\\Packet_` al identificador de adaptador de Windows.

### **3.9.2.2 VOID PacketCloseAdapter(LPADAPTER lpAdapter)**

Esta función libera la estructura ADAPTER *lpAdapter* y cierra el uso por PACKER32 del adaptador apuntado por él.

### **3.9.2.3 LPPACKET PacketAllocatePacket(void)**

Direcciona una estructura PACKET y retorna un puntero a ella. La estructura retornada debe ser convenientemente inicializada mediante el uso de la función `PacketInitPacket` Que se describe a continuación.

Advertencia: La zona de memoria de la estructura PACKET no se reserva en esta función. El Buffer debe ser reservado por la aplicación y asociado a la estructura PACKET mediante el uso de la función `PacketInitPacket`.

### **3.9.2.4 VOID PacketInitPacket(LPPACKET lpPacket, PVOID Buffer, UINT Length)**

Inicializa la estructura PACKET. Hay 3 parámetros de entrada:

- Un puntero a la estructura a inicializar
- Un puntero a una zona reservada por el usuario que contiene los datos capturados
- Longitud del buffer. Es el tamaño máximo que se transfiere del controlador a la aplicación en una única lectura.

Como nosotros no vamos a esperar que se acumulen paquetes en la zona de memoria reservada al buffer, el tamaño de este buffer no va a ser un parámetro crítico de diseño. En el software de captura sí es importante aumentar este valor ya que se requiere menos uso de procesador y es más rápida la transferencia en bloque de muchos paquetes que muchas transferencias a paquete por transferencia.

### **3.9.2.5 VOID PacketFreePacket(LPPACKET lpPacket)**

Esta función libera la estructura PACKET apuntada por *lpPacket*. Sin embargo la zona de memoria tiene que ser liberada por el usuario.

### **3.9.2.6 BOOLEAN PacketReceivePacket(LPADAPTER AdapterObject, LPPACKET lpPacket,BOOLEAN Sync)**

Esta función realiza la captura de un conjunto de paquetes dado un adaptador, una estructura que recoja los paquetes de usuario y un booleano que indica si espera indefinidamente a que llegue algún paquete.

### **3.9.2.7 BOOLEAN PacketSendPacket(LPADAPTER AdapterObject, LPPACKET lpPacket,BOOLEAN Sync)**

Esta función se usa para mandar paquetes directamente al dispositivo de red indicado por *AdapterObject* desde una zona de memoria. No es necesario poner la cabecera *bpf\_hdr* ni el CRC del nivel MAC ya que es driver el que se encarga de hacer esta operación.

### **3.9.2.8 BOOLEAN PacketSetHwFilter(LPADAPTER AdapterObject, ULONG Filter)**

Esta función indica al driver el filtrado que debe hacer de los paquetes capturados. Hay 4 posibilidades:

- `NDIS_PACKET_TYPE_PROMISCUOUS`: Activa el modo promiscuo. Todos los paquetes que escuche el adaptador serán pasados a la aplicación.
- `NDIS_PACKET_TYPE_DIRECTED`: Sólo se aceptan los paquetes cuyo destino sea el propio ordenador.
- `NDIS_PACKET_TYPE_BROADCAST`: Solo se aceptan paquetes *broadcast*.

- NDIS\_PACKET\_TYPE\_MULTICAST: Sólo se aceptan paquetes *multicast* en el caso de que el host pertenezca a uno de los grupos multienvío.
- NDIS\_PACKET\_TYPE\_ALL\_MULTICAST: Se aceptan todos los paquetes *multicast*.

### **3.9.2.9 BOOLEAN PacketSetBuff(LPADAPTER AdapterObject, int dim)**

Cambia el tamaño del buffer del driver asociado al adaptador *AdapterObject* dando un nuevo tamaño. Devuelve TRUE en caso de que la operación se haya completado con éxito y con FALSE si no hay memoria disponible.

### **3.9.2.10 BOOLEAN PacketSetReadTimeout ( LPADAPTER AdapterObject , int timeout )**

Asigna el tiempo máximo que puede permanecer un proceso bloqueado en el driver en espera de que llegue un paquete. Si se pone 0 no retorna si no es con un paquete.

### **3.9.2.11 BOOLEAN PacketSetMinToCopy(LPADAPTER AdapterObject,int nbytes)**

Es una función exclusiva de Windows 2000/NT y configura el número de bytes que es necesario capturar para que el driver devuelva el control a la aplicación.

# **CAPÍTULO 4: Desarrollo del Encaminador IP Convencional.**

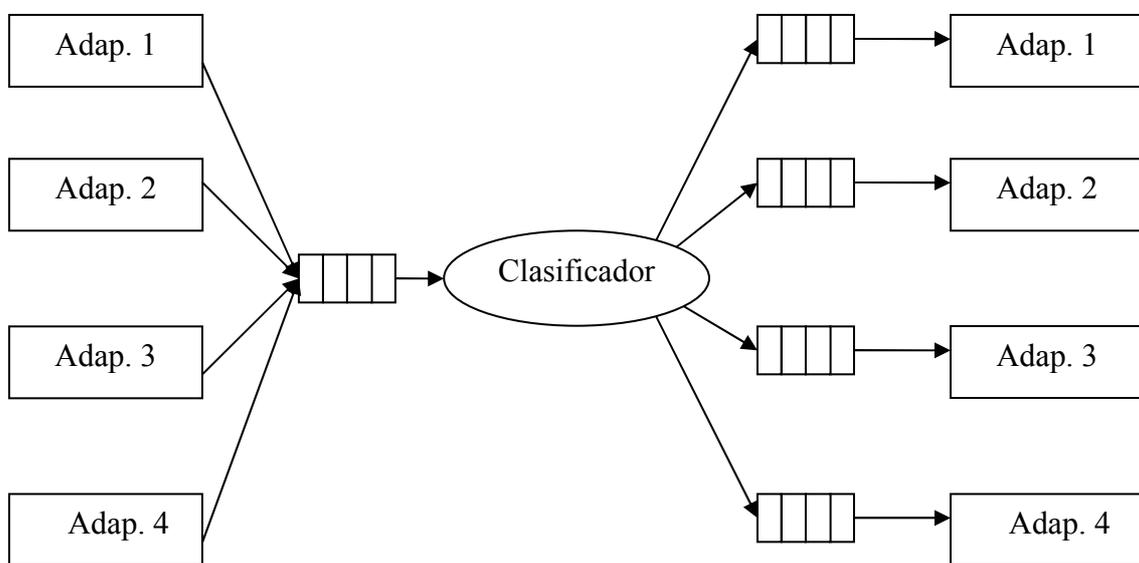
## **4.1 INTRODUCCIÓN**

Una vez realizado el estudio de los protocolos relacionados con la implementación a realizar, se aborda la estructura que se usará en este router. No obstante a priori el gran problema relacionado con este capítulo no fue el diseño del router, sino encontrar el medio para poder acceder directamente a la tarjeta de red ya que sin esta característica es imposible poder garantizar el envío de los paquetes directamente al medio físico sin pasar antes por la cola del sistema operativo. Una vez salvado este escollo se procedió a la creación del resto de los objetos que conformaron posteriormente el router básico.

## 4.2 ESTRUCTURA DEL ROUTER

### 4.2.1 EL CONCEPTO INICIAL

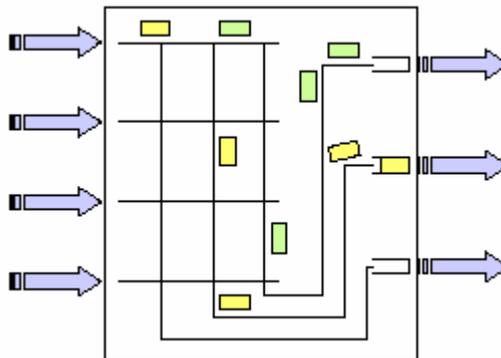
La estructura del router convencional, es derivada de cualquier encaminador o conmutador. Se compone de 3 elementos básicos:



- Interfaz de Entrada y salida de paquetes: Es el bloque funcional que interactúa directamente con el controlador en modo kernel del sistema operativo. Realiza 2 tareas básicas: Leer los paquetes que llegan del medio físico y así mismo escribir los paquetes de salida en el adaptador.
- Colas de paquetes: Se necesita una cola de espera a la unidad clasificadora de paquetes que tradicionalmente se denomina cola de entrada y otra en cada adaptador de salida que sirve como buffers de espera en el caso de ráfaga.

- Clasificador de paquetes: Se encargan de leer paquetes de la cola de espera de entrada y mandarlos a la cola que corresponda según el algoritmo de encaminamiento.

Se ve claramente que, como el clasificador tiene un único punto de entrada de paquetes, únicamente cuenta con una cola. Como se puede comprobar es un esquema muy similar al de un conmutador ATM por ejemplo, la única diferencia es que en un conmutador ATM por hardware los circuitos actúan concurrentemente y eso no es posible en un conmutador por software ya que únicamente contamos con un procesador. No obstante la emulación de ese trabajo concurrente va a ayudar a mantener esta estructura que asemeja más a un conmutador hardware que a un router software como se puede apreciar en la siguiente imagen:



## 4.2.2 IDENTIFICANDO PROCESOS

A las tareas concurrentes de un proceso se denomina hebra o en inglés *thread*. Estas hebras trabajan conjuntamente para llegar a un fin común, que en nuestro caso es que un paquete entrante salga por el adaptador correspondiente lo más rápidamente posible. Para identificar estas hebras usaremos el símil de una oficina de correos en la que hay varias líneas de salida por donde pueden enviarse los mensajes y varios buzones donde llegan los

sobres. Los mensajes son los datagramas, y los funcionarios de correos son las hebras del proceso. La justificación de este símil se puede encontrar en la sección 4.5.

En esta oficina hay en principio 3 tipos de funcionarios:

- cartero: hay uno en cada línea de salida esperando que le llegue a su mesa un mensaje para llevarlo a su destino.
- Funcionarios de recogida: atienden en el buzón y pasan el correo entrante a la cinta transportadora que los llevará al clasificador. Este trabajador podría llevar directamente el mensaje y entregarlo directamente al cartero del barrio correspondiente, pero entonces tendría que abandonar la cesta del buzón y hacer esperar a los siguientes mensajes. Es por ello por lo que existe otro tipo de funcionario.
- Funcionario Clasificador: Se encarga de leer la dirección del destinatario y ponerlas en la cola de correo pendiente del cartero que le corresponda llevar dicho mensaje. Como mínimo tiene que haber un de ellos aunque puede haber más.

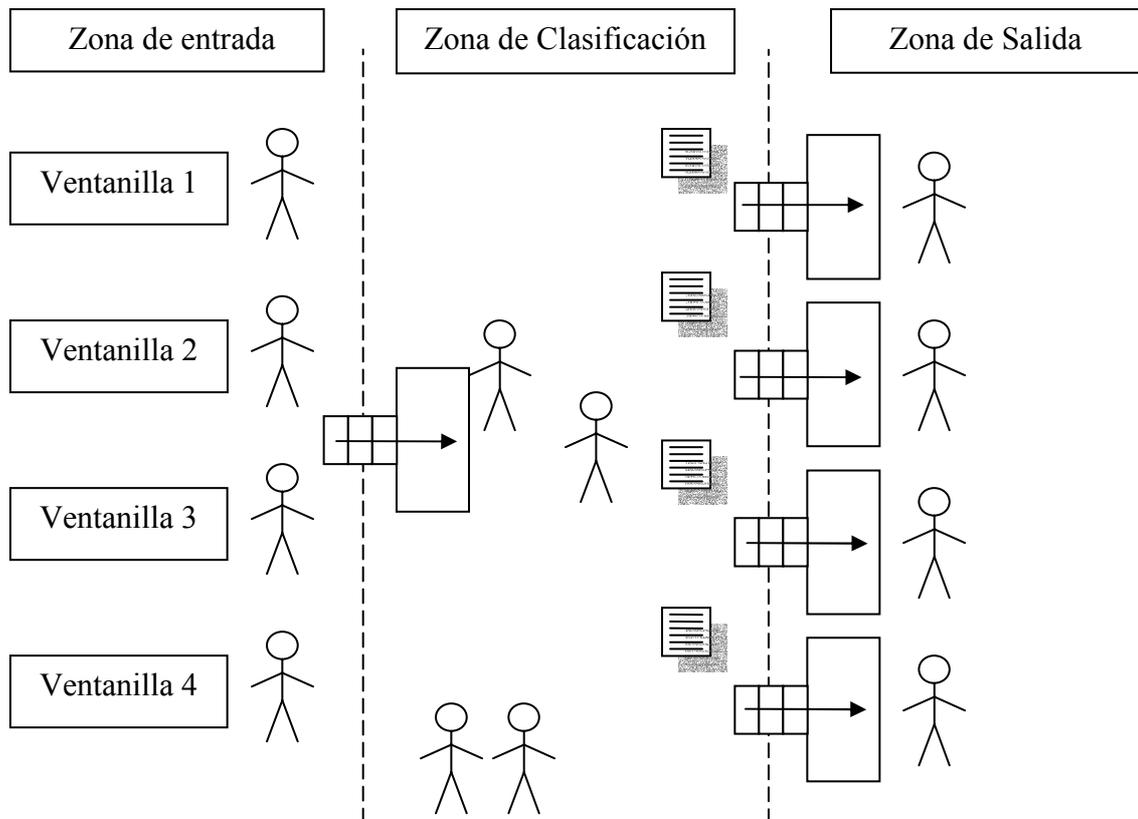
Sin embargo si llega un mensaje para un destinatario del que no se conoce su dirección, necesitarán a otro tipo más de funcionario, el cuarto y último tipo de trabajador:

- Funcionario Explorador: se encarga de buscar la dirección del destinatario de ese mensaje. Como es un proceso extremadamente lento se contrata a un explorador por cada mensaje del que no se conoce la dirección del destinatario y tanto ese mensaje como los que lleguen posteriormente antes de que retorne el explorador se guardan en un archivador donde además se encuentran anotadas las direcciones de los últimos destinatarios de mensajes. Esto es la caché ARP.

Así pues hay 4 tipos de trabajadores que desempeñan 4 tareas distintas con el fin de que los mensajes esperen lo menos posible en la oficina y se envíen lo antes posible sin dar más prioridad a un mensaje frente a los demás.

### 4.2.3 ESTRUCTURA DEFINITIVA

Incluyendo los 4 tipos de trabajadores, podemos ya determinar la estructura final del router que quedaría como en la figura anterior pero con los operarios trabajando en sus respectivas zonas. La siguiente ilustración es de elevada importancia ya que será la guía que nos sitúe los objetos usados a lo largo de este capítulo y el siguiente. Se recomienda al lector que reflexione sobre el símil e identifique en la ilustración cada elemento descrito en la sección anterior:



En este caso hay 4 adaptadores con 4 funcionarios esperando que llegue un mensaje para pasarlo a la zona de clasificación. Los clasificadores saben a qué destinatario va el mensaje pero no saben la dirección exacta y por ello miran en el archivador de la línea que va a ese destinatario para ver su dirección. Si la encuentran y no es de hace mucho tiempo la anotan en el mensaje y se encola en la mesa del cartero. Si por el contrario no hay una anotación en el archivador o la anotación es muy antigua, se contrata a un explorador para que salga a buscar dónde vive ese destinatario. El mensaje se guarda hasta que regresa de su exploración. Si ha encontrado una dirección válida se mandan todos los mensajes que hubiera recibido mientras el explorador realizaba su trabajo.

## 4.3 RECEPCIÓN Y ENVÍO DE DATOS

### 4.3.1 INTRODUCCIÓN

Para realizar esta actividad se necesitaba un método de recepción y envío de datos a la tarjeta de red de manera que los paquetes no pasaran por colas no controladas por el encaminador. Esto elimina la posibilidad de usar los Sockets de Windows en modo *raw*. Al final se usó una librería de enlace dinámico (DLL: *Dynamic Link Library*) y sus funciones que permiten acceder al adaptador E/S de red en contienda con la pila de protocolos de Windows. Este paquete de funciones procede del DDK (*Driver Development Kit*) de Windows NT y 2000, aunque igualmente funciona bajo Windows 9x. Haremos un repaso a las estructuras de datos y los servicios que proporciona.

### 4.3.2 EL INTERFAZ CON PACKET32

En el router se han aislado todas las operaciones que se tienen que realizar con PACKET32 ya que de esta manera se puede actualizar a futuras versiones o herramientas de captura. El router no va a necesitar escuchar paquetes con destinos que no sean la máquina propia o un *broadcast* a nivel de enlace. Es por ello por lo que el PACKET32 no va a funcionar en modo promiscuo, tal y como se suele usar este controlador en las aplicaciones de captura de paquetes.

Las cuatro funciones necesarias para el funcionamiento del router son las siguientes:

- Abrir el adaptador al interfaz Packet32
- Cerrar el adaptador del interfaz packet32
- Mandar un paquete
- Recibir el siguiente paquete

Estas cuatro funciones coinciden con el **constructor**, el **destructor** y dos métodos *Send* y *Receive* de la clase **TDriver**. De esta forma encapsulamos todo el acceso necesario al sistema de E/S.

La recepción es bloqueante, ya que la el funcionario que espera correo se quedará dormido un tiempo hasta que llegue un paquete. Si pasa un tiempo límite de 300ms despierta por si hay una notificación de jubilación del funcionario dormido en el driver.

Las instancias de escritura y lectura de un adaptador pueden ser completamente concurrentes ya que realizan llamadas al kernel del sistema y no interfieren entre ellas.

## 4.4 USO DE MULTITAREA

### 4.4.1 VENTAJAS

Un proceso en el que sólo hay un único elemento operativo, no puede pararse ya que podría haber otro punto del programa que necesitara su atención. Si esto ocurre, pueden perderse datos que necesitaban ser procesados, y lo que es más grave, que el proceso no se pararía nunca y tendría que repasar constantemente todos los puntos por si alguno de ellos requiere su atención, consumiendo todo el tiempo de CPU disponible en la máquina. La programación concurrente nos ofrece la posibilidad de tener un proceso en cada punto que necesite atención, estando el proceso *dormido* sin usar tiempo de CPU hasta que se solicita su acción. Con esto el uso de CPU se reduce considerablemente a niveles mínimos en caso de que el proceso no tenga mucha carga evolucionando según la misma.

Otra ventaja es que en un sistema con más de una CPU, sí que se puede realizar multiproceso paralelo simétrico, consiguiendo multiplicar la velocidad de la versión monotarea del programa.

### 4.4.2 INCONVENIENTES

Ya se ha comentado que recurrir a la multitarea es el mejor recurso si queremos emular un conmutador hardware y aprovechar al máximo los recursos de computación del PC. La programación concurrente no obstante no está carente de inconvenientes ya que se presentan problemas que sin las herramientas necesarias serían muy difíciles de solucionar. La multitarea en un proceso tiene tres grandes problemas:

- Sincronización entre procesos
- Acceso a zonas de memoria comunes a varios procesos.
- Creación y destrucción de hebras.

Sin embargo la herramienta de programación elegida nos proporciona las herramientas necesarias para solucionar estos problemas. En las dos siguientes secciones podemos observar las soluciones que ofrece el Borland C++ Builder 5 [TheBits] para sincronizar procesos y bloquear el acceso a memoria si esta está siendo usada.

### 4.4.3 SINCRONIZACIÓN ENTRE PROCESOS

La sincronización entre procesos es imprescindible para coordinar las acciones de 2 procesos. En nuestro router ocurren este tipo de situaciones en puntos donde la acción de un proceso activa a otro que permanece esperando en algún punto del código. Como ejemplo se puede citar el de las colas de paquetes:

Cuando una cola está vacía, los funcionarios clasificadores están esperando la llegada de algún paquete y se paran esperando que alguno de los funcionarios de ventanilla les mande algún paquete. Es en ese momento cuando se sincroniza la actividad de ambos trabajadores ya que cuando uno acaba justo de escribir un paquete en la cola, uno de los funcionarios clasificadores sale del estado de bloqueo para continuar con su actividad.

Este tipo de problemas se solucionan gracias a la clase *TSimpleEvent* del Borland C++ Builder. Con ella los funcionarios se pueden dormir en un punto determinado del código y ser despertados por otro trabajador en el momento en el que se le necesite. Se usará en todos los puntos donde la acción de un obrero necesite la respuesta de otro obrero, y la única zona donde esto ocurre será en las colas. Único sitio donde esperamos encontrar el uso de esta clase.

#### 4.4.4 ACCESO A ZONAS DE MEMORIA COMUNES

El otro gran problema consiste en la incoherencia de los datos de una zona de memoria accedida por más de un proceso en lectura/escritura. Para evitar que esto suceda se hace esperar a los demás procesos a que el primero en acceder a esa zona de memoria termine de usarla y ceda el derecho de uso al siguiente proceso. Para solucionar esto, el Borland C++ Builder 5 nos proporciona de las herramientas necesarias de las que sólo se han usado dos clases:

- ***TCriticalSection*** sólo deja entrar en una zona de memoria común a varios trabajadores a uno solo, haciendo esperar a los demás independientemente de si se va a realizar una lectura o escritura. Se usarán en zonas de datos compartidos por varios como por ejemplo la caché ARP en la que varios clasificadores o exploradores pueden estar consultando o actualizando si no se emplea un objeto de esta clase.
- ***TMultiReadExclusiveWriteSynchronizer*** es más avanzada que la clase anterior ya permite que varios procesos lean la misma zona de memoria concurrentemente y que cuando se realiza una escritura, no haya ningún otro proceso leyendo o escribiendo en la misma zona. Este tipo de objetos serán usados sobre todo en tablas de acceso común donde las consultas sean muy comunes y las escrituras o modificaciones no lo sean, incrementando enormemente la eficiencia de la multitarea.

#### **4.4.5 CREACIÓN Y DESTRUCCIÓN DE HEBRAS DE PROCESO**

Al igual que en las anteriores cuestiones, el Borland C++ Builder proporciona de una manera compacta y sencilla una solución a este problema. Comúnmente en Windows la creación de una nueva hebra de proceso conlleva una serie de llamadas al kernel del sistema. Este sistema está enormemente simplificado en la clase *TThread* del entorno de programación. Con ella podemos crear fácilmente una tarea y pasarle datos mediante los métodos y miembros que necesitemos añadir a la clase base *TThread*. La destrucción asimismo es fácilmente controlable mediante el método *Terminate*.

### **4.5 LOS TRABAJADORES (HEBRAS)**

El método de creación y destrucción de las hebras es muy similar al ciclo de vida de un trabajador ya que tanto unos como otros antes de ser activos pasan por un estado “latente” en el cual no se realiza ninguna actividad pero existen como hebras y trabajadores en cada caso. También se parecen en que para empezar a desarrollar una actividad la hebra necesita ser activada y el funcionario conseguir un puesto de trabajo. Otra de las similitudes estriba en el método de destrucción ya que ni la hebra ni el funcionario pueden volver a ese estado “latente” y el único cambio pueden afrontar es el cese de su actividad. Es por todas estas similitudes en su desarrollo por lo que se ha optado por usar la metáfora de la oficina de correos para representar las hebras de proceso y hacer más comprensible el funcionamiento interno del encaminador.

### 4.5.1 CICLO DE VIDA

Como se vio en el apartado 2 del presente capítulo, van a existir cuatro tipos de procesos. Así que se utilizará la clase *TTrabajador*, heredada de la *TThread*, para crear un obrero de cualquier tipo. Cada trabajador va a pasar siempre por dos estados:

1. **Latente**: Inicialmente se genera un elemento de proceso al que aún no se le ha asignado una tarea.
2. **activo**: se le asigna una tarea y seguirá desarrollándola hasta que se le ordene el cese de su actividad

Para asignar las tareas a los procesos latentes existen los siguientes métodos en la clase:

- **Receptor**: promociona un proceso en su estado latente a dedicarse a esperar que haya datos en la entrada de un adaptador.
- **Conmutador**: promociona un proceso de su estado latente a ser un de los clasificadores.
- **Transmisor**: promociona un proceso de su estado latente a estar esperando paquetes para llevarlos a la salida del adaptador.
- **Explorador**: promociona un proceso de su estado latente a ser uno de los que exploran en busca de la dirección de un destinatario desconocido.

Los procesos no se borran como otros objetos, en su lugar se les manda la orden de jubilación y abandonan su actividad voluntariamente. Si se encuentran dormidas (bloqueadas) en algún punto de su bucle, hay que esperar unos milisegundos a que salgan por expiración de tiempo.

## 4.5.2 PROCESO RECEPTOR

A la hora de madurar se le proporciona la referencia a un objeto de la clase *TDriver* asociado a la tarjeta de red que tiene que escuchar y la referencia a la cola común de entrada a la zona de clasificación.

Este proceso se dedica a esperar dormido a que llegue un paquete por el adaptador. Cuando llega comprueba que se trata de un paquete con el protocolo de red IP4 y lo mete en la cola de la zona de clasificación.

El sueño máximo de este proceso está limitado a 300ms después de los cuales despierta incluso si no ha llegado ningún paquete. Si ha llegado la orden de jubilación, proceso sale de su bucle y termina su ciclo de vida.

## 4.5.3 PROCESO TRANSMISOR

Tiene una funcionalidad completamente inversa al tipo de proceso anterior. A la hora de madurar se le proporciona la referencia a la cola de salida del adaptador y la referencia a un objeto de la clase *TDriver* asociado a la tarjeta de red donde tiene que escribir esos datos.

Este proceso se dedica a pasar paquetes de la cola al driver de E/S y en el caso de que no hubiera datos en esta cola espera dormida a que se encole un paquete. Una vez enviado el paquete a la red se libera la memoria usada para almacenar dicho paquete en la memoria.

El sueño máximo de este proceso está limitado a 300ms después de los cuales despierta incluso si no ha llegado ningún paquete. Si ha llegado la orden de jubilación, proceso sale de su bucle y termina su ciclo de vida.

### 4.5.3.1 Proceso Clasificador

Se encarga de filtrar y clasificar los paquetes que entran desde la zona de admisión. A la hora de madurar, se le debe pasar la referencia a la cola de entrada a la zona de clasificación. La tarea que realizan pasa por los las siguientes fases:

Comprueba que el destino del paquete no es el propio PC que alberga el router, ya que si el destino fuera local ya estaría en manos de la pila de protocolos del sistema operativo y se le entregaría a la aplicación pertinente. Si es ese el caso destruye el paquete y libera la zona de memoria ocupada.

Comprueba que el destino es accesible tanto directamente o a través de otro router. Es en este paso donde realmente se realiza la clasificación del paquete ya que se decide el adaptador por donde debe abandonar el router. Esta operación se hace usando un método estático de la clase *TAdaptador* denominado *BestIP4Adap* [Sección 4.9.2]

Si ha pasado por los anteriores filtrados tenemos un paquete con un destino alcanzable así que reemplazamos la cabecera de nivel de acceso al medio con las direcciones MAC de origen del adaptador de salida y la de destino de la cache ARP. Una vez realizados todos los pasos se manda a la cola de salida del adaptador correspondiente.

Si la dirección MAC destino no se encontrara en la cache ARP entonces se guarda ese paquete y todos los sucesivos con ese destino a la espera de que el explorador regrese y actualice la cache ARP con esa nueva entrada. Mientras, este clasificador puede continuar con su trabajo independientemente de lo que ocurra ya que cede la responsabilidad de entrega de esos paquetes al explorador.

El sueño máximo de este proceso está limitado a 300ms después de los cuales despierta incluso si no ha llegado ningún paquete. Si ha llegado la orden de jubilación, proceso sale de su bucle y termina su ciclo de vida.

### 4.5.3.2 Proceso Explorador

Es el único trabajador que se jubila el mismo y por lo tanto no realiza una tarea repetitiva como el resto sino que realiza una misión simple: Encontrar la dirección MAC de una dirección IP4. Para ello se le proporciona tanto la dirección IP como el adaptador donde supuestamente se encuentra dicho host. Lo que hace es lanzar una solicitud ARP a la red y esperar a que el hipotético destinatario responda. Mientras, el resto de procesos continúan trabajando.

Cuando retorna sólo hay 2 opciones: que se haya encontrado una dirección MAC válida con lo cual se vuelcan todos los paquetes de esa cola de paquetes temporal a la cola de salida del adaptador o que el destino sea inalcanzable con lo que se libera la memoria ocupada.

Una vez realizada una u otra acción el explorador termina sus obligaciones y se jubila a sí mismo.

## 4.6 LAS COLAS DE PAQUETES

La cola de paquetes es una cola FIFO (*First In First Out*) esto quiere decir que se van sirviendo estrictamente en el orden en el que llegaron. Se implementa en la clase *TColaPkt*. Esta clase tiene cuatro métodos importantes:

- **Constructor:** inicializa los datos.
- **Destructor:** libera la memoria ocupada.
- **Push:** inserta un paquete.
- **Pop:** extrae un paquete.

La cola es un objeto que es accedido por varios procesos así que debe hacer uso de dos técnicas de programación concurrente para evitar los dos posibles problemas que se pueden presentar:

- **Escritura concurrente:** Al leer se tiene que extraer un paquete y por lo tanto modificar la estructura de perchas encadenadas que sostienen los paquetes, al igual que al insertar un paquete. Esto puede provocar accesos de escritura concurrentes. Para evitar esto usaremos una zona crítica que proteja la cola.
- **Sincronización entre procesos:** El proceso que espera un paquete de una cola tendría que estar en un bucle que consumiría toda la CPU disponible hasta que llegara un paquete. Para evitar esto, se usará un evento de sincronización, que se señala cuando hay una inserción y bloquea el proceso que intenta sacar paquetes cuando no hay paquetes en la cola.

Este tipo de cola es la que se encuentra en la entrada de la zona de clasificación y a la salida a los adaptadores. Veamos cómo realiza su trabajo repasando los cuatro métodos principales de esta clase.

### **4.6.1 EL CONSTRUCTOR**

Es el método más simple ya que únicamente inicializa los parámetros y crea la sección crítica y el evento de sincronización. Nunca va a haber un acceso concurrente a este método por razones obvias.

### **4.6.2 LA INSERCIÓN DE PAQUETES**

Inicialmente se crea una estructura percha que haremos que sostenga el paquete de datos. Esta percha permitirá que el paquete pueda ser concatenado al resto de perchas formando la cola.

Pero antes de esta concatenación se solicita acceso exclusivo a la cola. Cuando se obtiene, se realiza la concatenación y se actualiza el contador de paquetes.

El último paso es activar el evento de sincronización por si hubiera algún proceso esperando un paquete a la salida de la cola.

### **4.6.3 LA EXTRACCIÓN DE PAQUETES**

Empezamos esperando un evento de sincronización, que será bloqueante en el caso de que no haya paquetes en su interior. Inmediatamente después solicitamos acceso exclusivo a la cola ya que seguro que vamos a tener que realizar alguna modificación en ella.

Una vez comprobado que sí hay paquetes en la cola, se desengancha la percha y se extrae el paquete datos, procediendo a la liberación de la zona de memoria ocupada por la percha.

En el caso de que este paquete fuera el único que estaba en la cola, se configura el evento de sincronización en modo bloqueante.

Una vez realizado esto se abandona la zona crítica permitiendo el acceso de otros procesos a la cola. El método termina retornando la referencia al paquete de datos.

#### **4.6.4 EL DESTRUCTOR**

Realiza el borrado de todas las perchas con sus respectivos paquetes y elimina tanto la zona crítica como el evento de sincronización que se habían contraído en la creación del objeto.

### **4.7 EL CLASIFICADOR DE PAQUETES**

El clasificador de paquetes está compuesto por un número mayor que cero de procesos clasificadores. Éstos realizan su trabajo concurrentemente y se relacionan directamente con la cola de entrada a esta sección y con cada uno de los adaptadores.

A esta sección también pertenecen los exploradores que acceden a los mismos objetos que sus compañeros clasificadores.

Esta sección donde se lleva cabo la mayor parte del procesamiento del router tradicional ya que tiene que hacer uso de la tabla de encaminamiento de Windows, la caché ARP y a dos colas durante el tránsito de cada paquete por esta sección.

Para saber cómo se realizan el paso de cada paquete por esta sección el lector puede consultar la sección que analiza los procesos citados. [Secciones 4.5.3.1 y 4.5.3.2.]

## 4.8 LA CACHE ARP

Este objeto se encuentra entre la etapa de clasificación y la de salida ya que es únicamente accedida por los procesos que trabajan en la sección de clasificación y sin embargo está relacionada con los adaptadores que claramente corresponden con las etapas de entrada y sobre todo con la de salida.

Este objeto proporciona 3 funcionalidades básicas para acoplarse perfectamente a los procesos que hacen uso de ellos:

- **Req.** Solicita a la tabla que le devuelva la dirección MAC de una determinada IP4.
- **Add.** Añade la asociación Dirección IP4 – Dirección MAC a la tabla.
- **Update.** Actualiza el tiempo de caducidad del asiento de una determinada dirección MAC.

Los datos en la caché tienen una vigencia por defecto de 1 minuto, a no se que se usen, por lo que el tiempo de caducidad se va prorrogando indefinidamente. No se usa el mismo método de actualización de la tabla de ARP que usa Windows así que los datos almacenados en esta tabla y la que en el mismo momento tiene la tabla de este router no tienen porqué coincidir.

### 4.8.1 PIDIENDO DATOS A LA CACHE ARP

Esta caché ARP es bastante especial ya que en cada entrada de la tabla hay presente una referencia a una cola que es donde se introducen los paquetes que están pendientes de determinar su dirección MAC, así que en la petición de dirección MAC se le pasa también un paquete. La cache nos puede retornar 3 tipos de mensajes:

- **ESTA:** quiere decir que hay una entrada MAC para esa IP y que se retorna con éxito por parámetro.
- **NO\_ESTA:** El paquete se ha insertado en la cola de la caché ARP y es necesario que el proceso que hace uso de esta función realice la creación de un proceso explorador para que busque si existe una dirección MAC de destino para esa IP mediante una consulta ARP a la propia red mediante *broadcast*. De lo contrario el resto de paquetes a esa dirección se seguirán encolando.
- **EN\_ESPERA:** Ya hay otros paquetes esperando esa resolución con lo que se encola en la caché y no es necesario crear otro proceso explorador para esta resolución.

### 4.8.2 INSERTANDO UNA ASOCIACIÓN EN LA CACHE ARP

La inserción de una asociación de direcciones IP – MAC siempre se realiza después de una consulta por broadcast así que posiblemente haya paquetes esperando en la cache. Es por este motivo por el cual al añadir información a la cache se nos devuelve la cola de paquetes que pudieran estar esperando para que el proceso usuario mueva los datos de la cola temporal a la cola de salida del adaptador asociado.

## 4.9 LOS ADAPTADORES

La clase *TAdaptador* engloba todo lo que tiene que ver unívocamente con un adaptador de red concreto. Esto comprende los siguientes elementos:

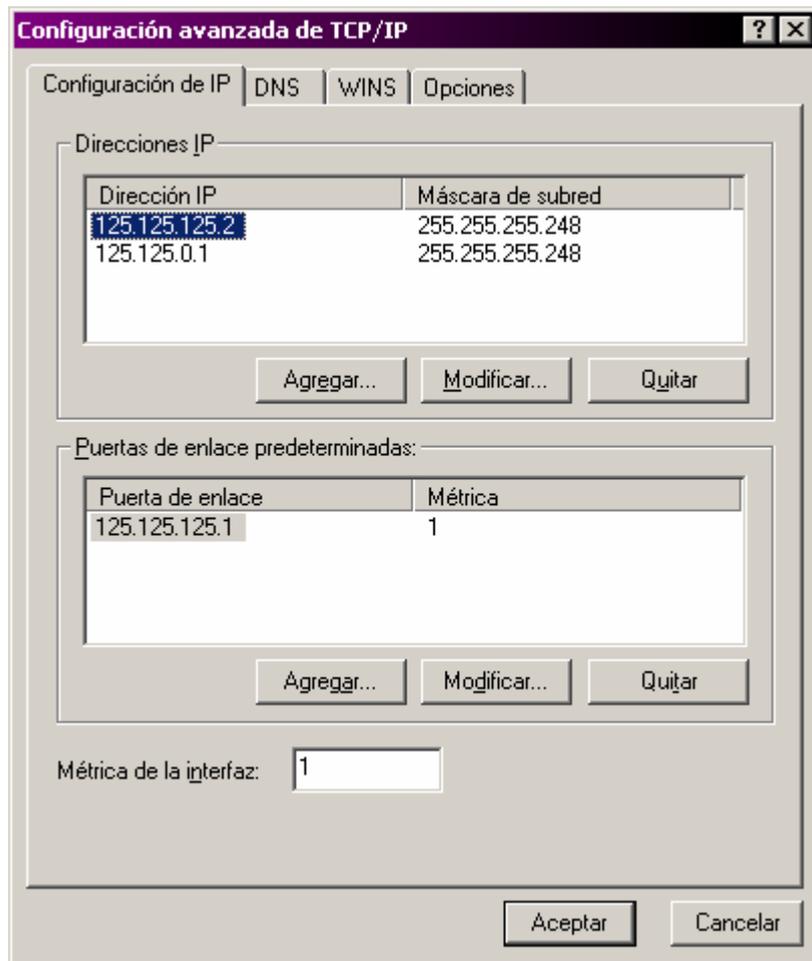
- **Una Cola de entrada:** Para cada tarjeta de red existe una cola de espera para los paquetes procedentes de la zona de clasificación
- **Una cache ARP** que se define por adaptador en vez de globalmente ya que de lo contrario se tardaría más en realizar una búsqueda en ella
- **Un driver de acceso a E/S** ya que para cada adaptador debe existir una instancia del driver para permitir leer y escribir del mismo.
- **Un proceso receptor:** que va leer los datos de entrada por el dispositivo de red.
- **Un proceso transmisor:** que escribirá los paquetes de salida al dispositivo de red.
- **Referencia a datos del adaptador:** datos que son proporcionados por Windows y que nos servirán para identificar los distintos interfaces asociados a este adaptador.

La creación de todos los objetos TAdaptador se realizan a la hora de iniciar el router y de ello se encarga un función estática que obtiene la información de los adaptadores de red de Windows y de esta manera recopilar toda la información necesaria para crear cada uno de los adaptadores.

### 4.9.1 INTERFACES RELACIONADOS

De los elementos que engloban los objetos de la clase TAdaptador hemos descrito todos con anterioridad excepto la información que Windows proporciona del adaptador.

La pila de protocolos de Windows que repasábamos en la sección 2.1.4 veíamos que el adaptador se situaba debajo de los protocolos de red. De esta manera no solamente se pueden tener varios protocolos de red diferentes sino que se pueden realizar distintas instancias de un mismo protocolo de red. Gracias a esto, la pila de protocolos TCP/IP nos permite configurar el acceso a distintas subredes IP mediante el mismo adaptador de red. En el cuadro de diálogo de configuración avanzada TCP/IP podemos ver lo siguiente:



En este ejemplo se puede ver como en las propiedades de este adaptador de red existen dos interfaces IP distintos. Gracias a esto, el router es capaz de funcionar como tal incluso con una única tarjeta de red.

## 4.9.2 OBTENIENDO EL MEJOR ADAPTADOR

Debido a la organización que usa Windows, se complicaría la elección del interfaz de red en el caso de que el propio router integrara la tabla de encaminamiento. Por otra parte, el duplicar la tabla de encaminamiento no serviría para nada.

Así que para buscar el mejor interfaz para una IP según la tabla de encaminamiento y la relación entre interfaz IP y adaptador se usarán los servicios proporcionados por el sistema operativo. Con esto, además de simplificar el código, mantiene la integridad del router con el PC que lo alberga. Otra ventaja que se obtiene con esto es que todos los datos de la tabla de encaminamiento se pueden consultar y modificar fácilmente mediante el comando *route*, descrito en la sección 3.3.8

Así pues, la búsqueda del nodo siguiente de una determinada IP se realiza mediante un miembro estático de la clase TAdaptador denominado *BestIP4Adap* y que devuelve la IP del siguiente salto y una referencia al adaptador por el que se tiene que encaminar el paquete.

## 4.10 OTRAS CLASES USADAS

Las clases comentadas anteriormente forman el esqueleto y explican el funcionamiento de todos los procesos que se llevan a cabo en el router. No obstante hay otras clases secundarias que también forman parte de este router básico.

### **4.10.1 LA CLASE PAQUETE**

Hasta ahora hemos estado hablando de paquetes que se pasan unos procesos a otros a través de colas. Estos objetos pertenecen a la clase *TPacket* que engloba todas las operaciones que se pueden realizar en el interior de un paquete.

En esta clase se implementa la capacidad de poder leer los datos necesarios de las cabeceras del nivel de acceso al medio y de red para identificar protocolos, leer direcciones IP4 y leer y escribir las direcciones MAC de origen y destino. Es una clase muy usada ya que prácticamente el resto de los objetos la necesitan.

### **4.10.2 LA CLASE TDIRMAC**

La dirección MAC se compone de 6 octetos. Este tamaño hace imposible usar un tipo básico para almacenar y manejar este tipo de información. Debido a esto se implementa esta clase que nos permitirá usar la información e incluso operaciones de comparación entre ellas, útiles a la hora de buscar una MAC en la caché ARP.

### **4.10.3 LA CLASE TRROUTER**

Sólo habrá un objeto de esta clase funcionando y se encarga de crear toda la estructura de objetos que necesita el router para funcionar correctamente.

El orden de creación de estos objetos es:

1. La cola de entrada a la etapa clasificadora
2. Inicializa todos los adaptadores, incluyendo en ellos las colas de salida, caches ARP y procesos receptores y transmisores.
3. Los procesos clasificadores, inicialmente sólo uno.

Una vez creada esta estructura ya puede funcionar y el objeto router se bloquea en un bucle hasta que le llega la señal de finalización con lo que destruye la estructura creada en orden inverso en el que fue creada.

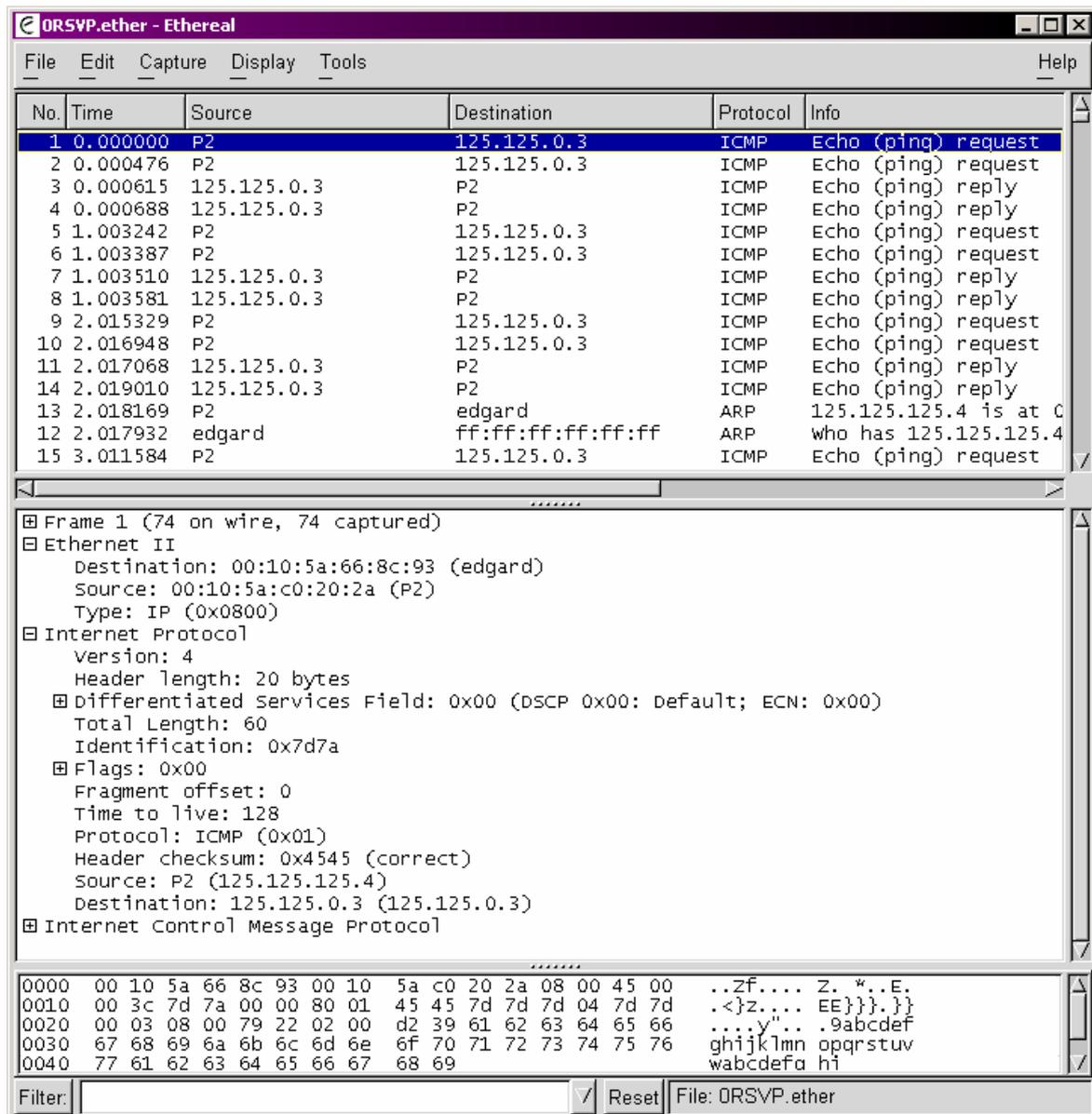
## 4.11 EL INTERFAZ DE USUARIO

En esta primera versión se presenta un interfaz de usuario sencillo, sin parámetros de configuración y que únicamente permitía lanzar y parar el router. En la figura siguiente podemos ver cómo se presenta esta aplicación al usuario:



## 4.12 PRUEBAS DE FUNCIONAMIENTO

Las primeras pruebas que se fueron realizando durante la elaboración parcial del router consistieron en el uso de un capturador de paquetes, gracias al cual se puede ver si realmente el router está capturando los paquetes y los está reenviando [Apparna99]. Para ello se usó el software *Ethereal* en su versión 0.8.18. Aquí podemos ver una de las capturas realizadas en una de las pruebas intermedias:

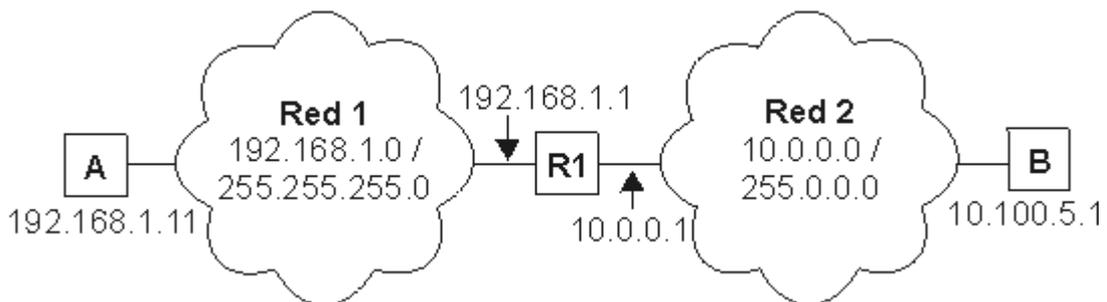


En su ventana principal se pueden ver 3 zonas:

- Lista de paquetes capturados con los datos más importantes.
- Detalle del contenido de todos los campos que integran las cabeceras de los paquetes
- Trascricpción en hexadecimal y ASCII del contenido del paquete.

En el ejemplo de la figura se puede ver como todos los mensajes aparecen repetidos, señal de que los paquetes pasaban de una subred a otra.

Las pruebas de funcionamiento definitivas se realizaron usando 3 ordenadores conectados a 2 subredes conectadas entre sí por un router según la siguiente ilustración:



Aunque el router usaba un único adaptador de red, se pudo configurar dos interfaces IP4 para realizar la prueba. De esta manera se comprobó primeramente con un mensaje de eco ICMP mediante el uso del comando ping como se describe en la sección **¡Error! No se encuentra el origen de la referencia.**

Adicionalmente se comprobó que tanto las transferencias FTP como la videoconferencia usando el software *NetMeeting* funcionaban correctamente en ambas direcciones y sin cargar en ningún momento la CPU del ordenador que hospedaba el router.

# **CAPÍTULO 5: Integración de la gestión RSVP**

## **5.1 INTRODUCCIÓN**

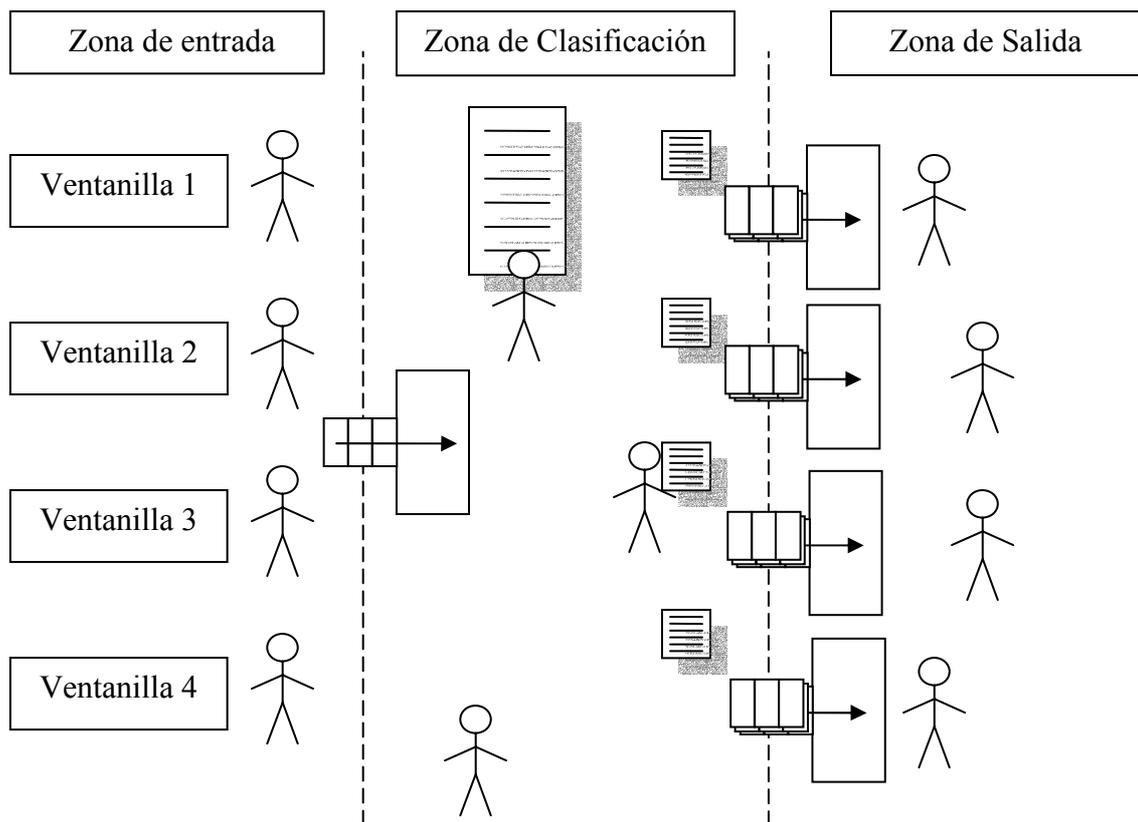
Una vez comprobado en correcto funcionamiento del router básico IP4 básico, se plantean pocos pero profundos desafíos. En este punto había que implementar toda la gestión RSVP, las colas con prioridad con distintas políticas de planificación, sistemas para controlar la carga, rutinas de monitorización y un entorno adecuado para modificar los parámetros de configuración y representar los datos que nos puede ofrecer el encaminador.

Esta fase es lo que realmente va a hacer que este software sea diferente al resto de Routers para Windows que se encuentran disponibles. Comenzaremos por los cambios

estructurales profundizando en cada uno de los aspectos. No se comentarán los elementos que hayan sido modificados levemente y se profundizará en aquellos otros de mayor importancia.

## 5.2 ESTRUCTURA BÁSICA

La estructura básica no varía en gran medida. Ahora aparece una nueva tabla de datos más grande en la etapa de clasificación y las colas de salida a los adaptadores ahora son mucho más complejas ya que no son simples colas de paquetes como continúa siendo la cola de entrada a la zona de clasificación, sino que son colas con gestión interna de prioridad que pueden contener en su interior distintas colas para cada flujo.



Con respecto al número y tipo de trabajadores, no va a existir ningún cambio en ninguna parte del router, sin embargo los clasificadores van a tener que trabajar ahora más, ya que tienen que mirar en la tabla RSVP (en la parte superior central de la figura anterior) por si el paquete que llega pertenece o no a un sesión RSVP antes de hacer cualquier cosa. Adicionalmente el proceso transmisor también va a estar más atareado ya que para extraer un paquete de una cola con política de prioridad se necesita más tiempo de proceso que si se extrae de la cola de paquetes que ya conocemos de la fase anterior.

## 5.3 EL GESTOR RSVP

El gestor RSVP se va a encargar de tres tareas básicas, que serán ejecutadas por los clasificadores. Estas tareas son:

- Interpretar los mensajes de señalización RSVP
- Gestionar las reservas según los mensajes de señalización
- Realizar el encaminamiento de los paquetes que pertenecen a una sesión.

Veremos cómo se integra el gestor RSVP en la secuencia de código de los procesos clasificadores y cómo realizan estas tareas. Todo el código relacionado con el gestor RSVP se encapsula en la clase *TRSVPPData* al igual que la tabla de sesiones RSVP.

### 5.3.1 PUNTO DE ENTRADA AL GESTOR RSVP

Las tareas a realizar por el gestor RSVP se realizan por los clasificadores. Recordando el funcionamiento del proceso clasificador del router tradicional en la sección 4.5.3.1, situamos el punto de entrada del gestor RSVP antes de comprobar si el destino es el propio

ordenador. Esto se hace ya que según se vio en la descripción del protocolo RSVP, algunos mensajes se envían salto a salto con lo cual algunos mensajes vendrán destinados al propio ordenador y estos deben ser gestionados correctamente.

El punto de entrada es la llamada al método *EncaminaRSVP* de la clase *TRSVPPData*. Es una función que retorna verdadero en el caso de que el paquete se haya clasificado gracias a la tabla de sesiones, con lo que se ahorraría el resto de código del clasificador y por lo tanto se ganaría en eficiencia.

Sin embargo si el método retorna falso, quiere decir que el paquete no pertenece a ninguna sesión y que debe ser encaminado por métodos tradicionales. Este es el peor caso ya que se ha invertido un tiempo extra en comprobar la tabla de reservas RSVP para nada, retrasando el envío del paquete y gastando tiempo de proceso inútilmente.

### **5.3.2 INTERPRETAR LOS MENSAJES DE SEÑALIZACIÓN RSVP**

Esta función se realiza internamente dentro del gestor. Recibe un paquete y lo que hace es extraer el buffer con la información del mensaje RSVP. Inicialmente lee la cabecera y va interpretando uno por uno todos los objetos que contenga el mensaje.

Devuelve una estructura de punteros que apuntará a los datos que contenga ese paquete. Esta estructura será usada por el gestor de mensajes para saber el tipo de acción que hay que realizar sobre la tabla de reservas.

### 5.3.3 EL GESTOR DE MENSAJES

Cuando el proceso clasificador entra en la gestión RSVP lo primero que se examina es el protocolo de la capa 4 del paquete que entra. Si se trata de un paquete RSVP entonces se pasa al gestor de mensajes RSVP que realizará las acciones oportunas según el mensaje que llegue.

#### 5.3.3.1 Llegada de un Mensaje PATH

Tal y como se puede comprobar en la documentación del protocolo RSVP en el capítulo 3, cuando llega un mensaje de PATH lo primero que se realiza es comprobar que están presentes todos los objetos requeridos para un mensaje de PATH. Posteriormente se realiza una búsqueda por si se trata de un mensaje de refresco del ‘*soft state*’ correspondiente a una sesión. Si es así actualiza el tiempo de caducidad de la sesión y reenvía el paquete al próximo salto.

Si no fuera un mensaje de refresco se introducen todos los datos que se especifican en los objetos en la tabla y se marca en estado de PATH, es decir que aún no se trata de una reserva en firme, entre otras cosas porque faltan datos como los parámetros de calidad de servicio. Lo que sí se guarda en este momento es la referencia al adaptador correspondiente al siguiente salto así como la del salto anterior.

#### 5.3.3.2 Llegada de un mensaje RESV

Se realiza la comprobación de que se encuentren todos los objetos necesarios para hacer la reserva. Aquí se pudo comprobar como *NetMeeting* no se ciñe al estándar RSVP ya que no aporta el objeto *Filter-Spec* en sus mensajes de reserva. No obstante como el protocolo

RSVP no es un estándar cerrado se relajaron las exigencias del router y el objeto *Filter-Spec* no se hizo necesario. En la siguiente ilustración se puede ver el contenido de un mensaje de reserva generado por *NetMeeting*:

No.	Time	Source	Destination	Protocol	Info
93	5.814393	P2	125.125.0.3	RSVP	PATH Message
94	5.814554	P2	125.125.0.3	RSVP	PATH Message
95	5.814995	125.125.0.3	edgard	RSVP	RESV Message
97	5.824030	edgard	P2	RSVP	RESV Message
418	7.811062	P2	125.125.0.3	RSVP	PATH Message
419	7.811186	P2	125.125.0.3	RSVP	PATH Message
420	7.812765	125.125.0.3	P2	RSVP	RESV Message
421	7.812856	edgard	P2	RSVP	RESV Message
422	7.813207	125.125.0.3	edgard	RSVP	RESV Message
423	7.813232	125.125.0.3	P2	RSVP	RESV Message
425	7.813367	edgard	P2	RSVP	RESV Message
426	7.813461	edgard	P2	RSVP	RESV Message
721	12.956310	P2	125.125.0.3	RSVP	PATH TEAR Message
722	12.956405	P2	125.125.0.3	RSVP	PATH TEAR Message
761	14.808980	125.125.0.3	edgard	RSVP	RESV TEAR Message

Frame 95 (118 on wire, 118 captured)  
 Ethernet II  
 Internet Protocol  
 Resource Reservation Protocol (RSVP)

- RSVP Header**
  - SESSION: IPv4, 125.125.0.3, 17, 49600
  - HOP: IPv4, 125.125.0.3
  - TIME VALUES: 30000 ms
  - STYLE: wildcard Filter (17)
  - FLOWSPEC: Controlled-Load, 57782 bytes/sec

```

0020  00 01 10 02 1e 6a 80 00 00 54 00 0c 01 01 7d 7d  .....T....}}
0030  00 03 11 00 c1 c0 00 0c 03 01 7d 7d 00 03 00 00  .....}}....
0040  00 00 00 08 05 01 00 00 75 30 00 08 08 01 00 00  .....u0.....
0050  00 11 00 24 09 02 00 00 00 07 05 00 00 06 7f 00  ...$. ....0.
    
```

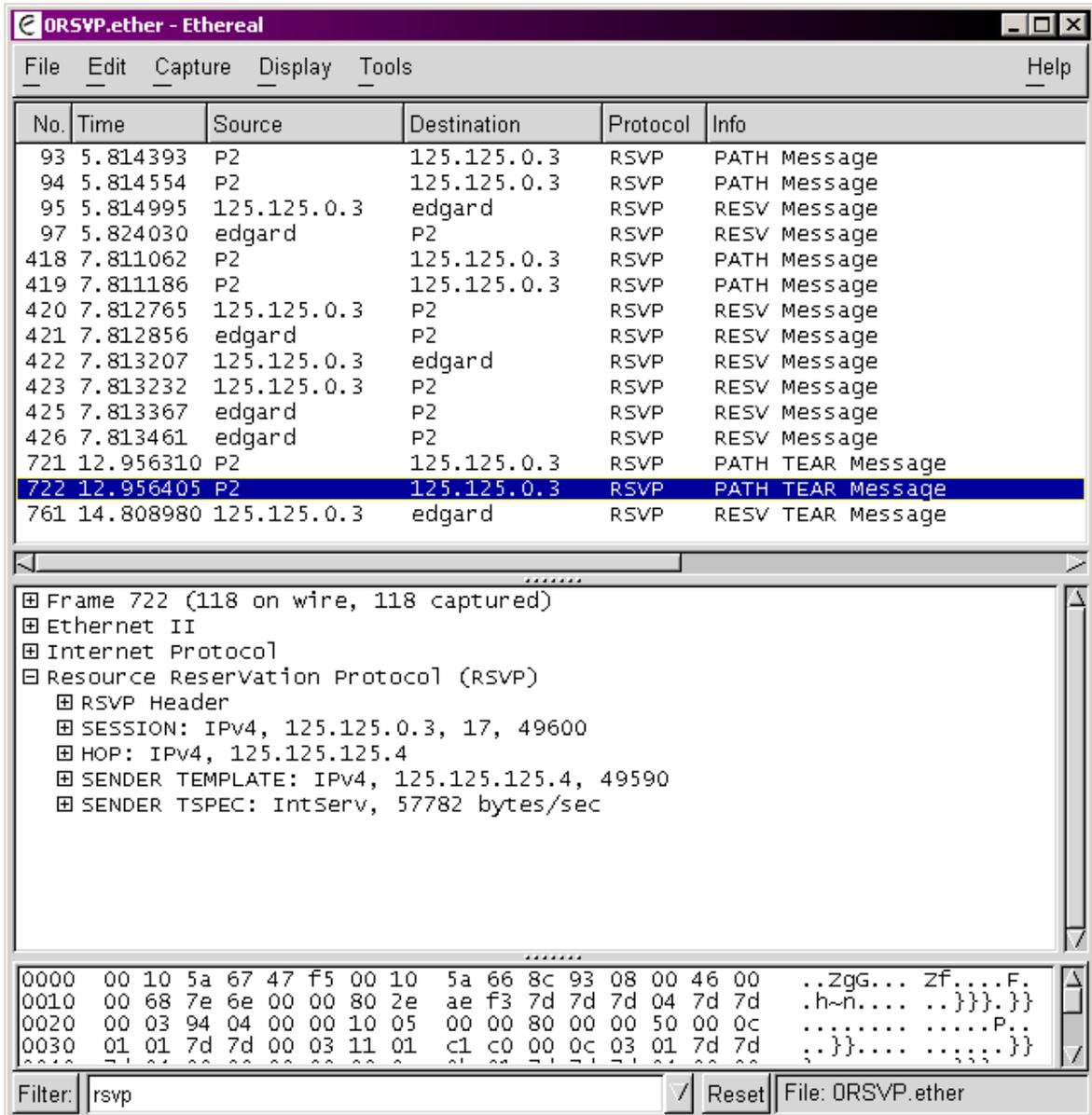
Filter: rsvp [Reset]

Si se encuentran el resto de objetos necesarios se busca en la tabla de sesiones. Si se encuentra la sesión se pasa al estado de reserva o en caso de que ya estuviera en estado de reserva se actualiza el tiempo de caducidad y demás parámetros configurables. Además se modifica el paquete antes de devolverlo al clasificador tradicional ya que el envío de mensajes de reserva se hace salto a salto.

Si la reserva estaba previamente en estado de *path*, entonces se manda la información de una nueva sesión a la cola de prioridad correspondiente que devolverá un identificador de sesión que se almacenará con el resto de los datos de la sesión. Además de esto, se inicializa el limitador de ancho de banda para esta sesión según los parámetros que se especifiquen en el objeto *FlowSpec*.

### **5.3.3.3 Llegada de un mensaje PATH\_TEAR o RESV\_TEAR**

En estos casos se desmonta la reserva y se elimina la entrada de la tabla ya que estos mensajes notifican de la finalización de una sesión o la cancelación de un estado de *path*. En cualquier caso significa finalizar la sesión. Posteriormente se reenvía el mensaje RSVP al siguiente salto en la línea.



### 5.3.3.4 Llegada de un mensaje de ERROR

Los mensajes de error son extremo a extremo, no obstante un mensaje de error RSVP significa que no se ha podido completar el protocolo de reserva en todo el camino así que

la acción a realizar es la misma que en el caso anterior, es decir se destruye la sesión y se reenvía el mensaje al salto que corresponda.

### 5.3.4 ENCAMINAMIENTO DE LOS PAQUETES DE UNA SESIÓN

Cuando el proceso clasificador entra en el gestor RSVP y ha comprobado que el paquete no es de señalización RSVP y es TCP o UDP, busca en la tabla si hay alguna reserva con el mismo patrón: Dirección IP, protocolo y puerto.

Si no hay coincidencias sale a buscar el siguiente salto por medios tradicionales, pero si pertenece a una sesión y no vulnera los parámetros especificados en la reserva, se encuentra con todo lo necesario para encaminar ese paquete inmediatamente.

- Adaptador de salida
- Dirección MAC del próximo salto
- Dirección MAC del adaptador de salida

El siguiente paso consiste en reemplazar la cabecera de nivel de acceso al medio y ponerlo en la cola de prioridad del adaptador de salida que hará todo lo posible para enviar el paquete lo antes posible.

### 5.3.5 PROBLEMAS DE ACCESOS CONCURRENTES AL GESTOR

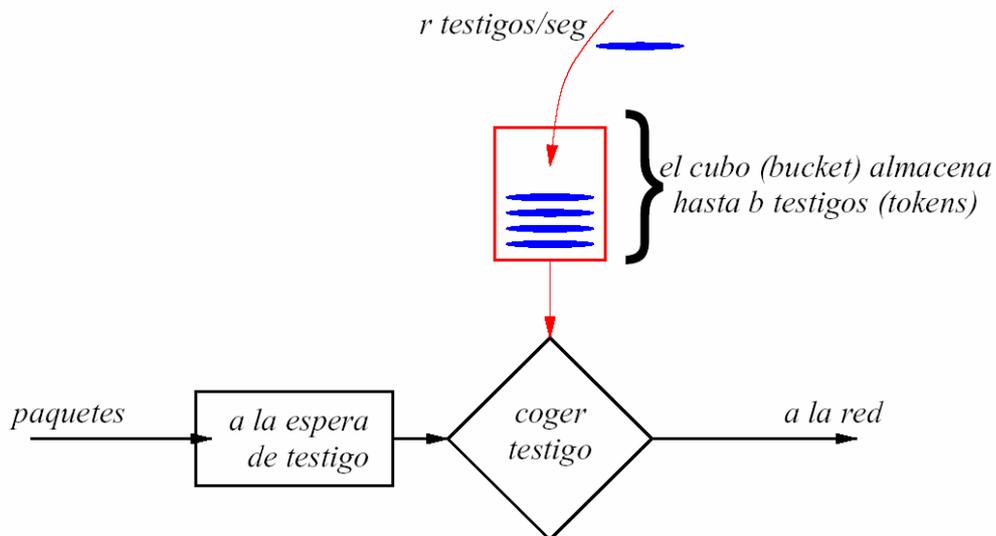
El gestor RSVP es accedido concurrentemente por varias unidades clasificadoras. Esto no es un problema en el caso de que todas sean lectoras, pero cuando hay que gestionar un mensaje de señalización RSVP es necesario modificar datos en la tabla de reservas. Por este motivo se hará uso de la clase *TMultiReadExclusiveWriteSynchronizer*. Con ella

es posible realizar múltiples lecturas concurrentes y hacer escrituras con acceso exclusivo. De esta manera se optimiza el código ya que en situaciones donde haya muchas reservas, las unidades clasificadoras van a pasar bastante tiempo consultando la tabla de reservas.

## 5.4 LIMITANDO EN ANCHO DE BANDA

Para limitar el uso del ancho de banda cedido, el router debe emplear mecanismos de control. Podemos considerar a este tipo de mecanismos como aquellos que regulan la frecuencia con la que una aplicación puede inyectar paquetes en la red.

El mecanismo de control más utilizado es el *leaky bucket*, también conocido como *token bucket* (que traducido a nuestro idioma es algo así como "cubo de testigos") Es un mecanismo que conceptualmente es muy fácil de explicar y es el utilizado en el protocolo RSVP para definir las características del flujo de datos.



De hecho, el cubo de testigos es una abstracción que ayuda a entender y caracterizar los límites del flujo, no es en sí mismo un algoritmo. La idea es que se dispone de un cubo (*bucket*) en el que se almacenan testigos (*tokens*), los cuales se generan a un ritmo de  $r$  testigos por segundo. El cubo puede almacenar un máximo de  $b$  testigos.

Antes de enviar un paquete a la red, este debe coger un testigo del cubo, Si no existieran testigos disponibles el paquete debería esperar (o se descartaría, dependiendo de las implementaciones).

Dado que como máximo hay  $b$  testigos, la ráfaga máxima que se podrá entregar a la red será de  $b$  paquetes. Dado que los nuevos testigos se generan a un ritmo de  $r$  por segundo, en un periodo de tiempo  $T$  podrán entregarse como máximo  $r.T + b$  paquetes.

Si, además, acotamos la longitud máxima de los paquetes (pongamos  $M$  bytes), el ritmo máximo de entrega de datos a la red queda acotado por la expresión  $M.(r + b)$  bytes/seg.

El algoritmo puede aproximarse también a nivel de bytes. Supongamos que un testigo equivale a un byte, es decir, un paquete de tamaño  $L$  bytes no podrá abandonar la cola de espera hasta que el bucket contenga al menos  $L$  testigos. Evidentemente, la tasa de creación de testigos se altera con esta nueva aproximación del algoritmo, pasando a ser de  $r$  bytes/seg en vez de  $r$  paquetes/seg, y como consecuencia, el tamaño del bucket se modifica a  $b$  bytes en vez de a  $b$  testigos.

Por lo tanto, en un intervalo de tiempo  $T$ , se podrán entregar a la red un máximo de  $r.x T + b$  bytes.

En el router este mecanismo está completamente encapsulado en una clase independiente denominada ***TTokenBucket***. Los parámetros se configuran en la creación

del objeto y no pueden modificarse. En tal caso es lo mismo que si se destruye y se vuelve a crear un nuevo objeto con otros parámetros.

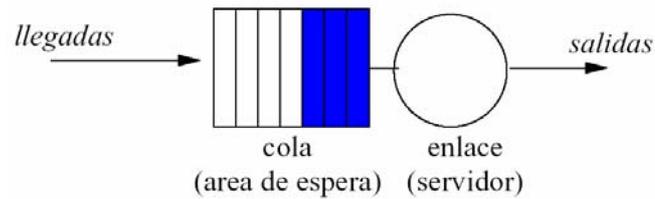
## 5.5 PLANIFICACIÓN DE COLAS

Las colas de salida son las que en situaciones normales se saturarán y generarán los problemas con el tráfico en tiempo real como se estudiaba en el capítulo 2. Para evitar que el tráfico en tiempo real se vea afectado por situaciones de congestión, es necesaria una planificación de colas mucho más avanzada ([Golestani94]) que una simple cola FIFO. Veamos las políticas más comunes y que se implementan en el router:

### 5.5.1 FIRST-IN-FIRST-OUT (FIFO)

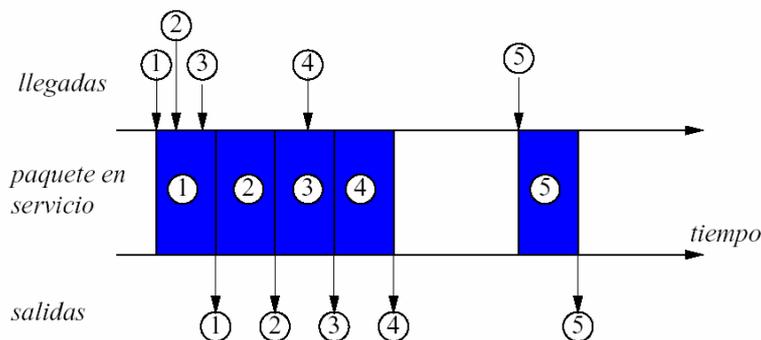
La figura muestra el modelo de funcionamiento de una política de planificación del enlace FIFO. Bajo esta política, si un paquete llega al nodo y en ese momento se está transmitiendo otro paquete, el nuevo se almacena en una cola, la cual se vacía por estricto orden de llegada.

Si no hubiera suficiente espacio libre en la cola entraría en juego la "política de descarte de paquetes" la cual decide si hay que hacer sitio para el nuevo paquete, y si esto es así qué paquete de los almacenados debe eliminarse.



Cuando un paquete ha sido transmitido completamente por el enlace se elimina de la cola y se escoge el siguiente (el que lleva más tiempo en la cola)

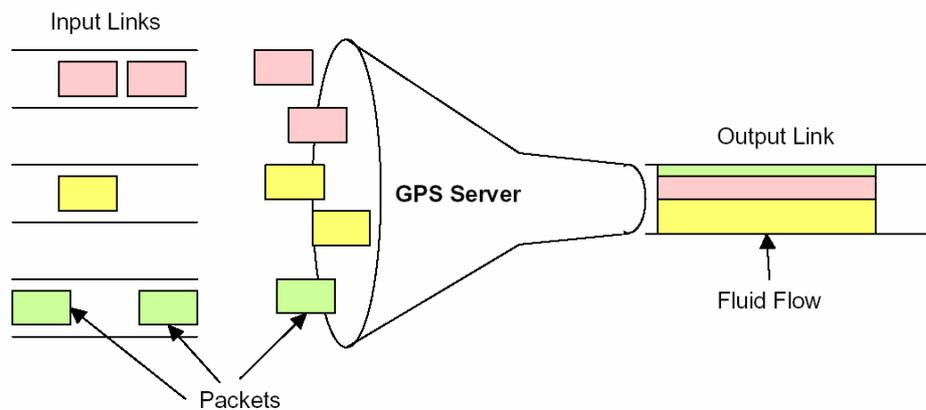
La siguiente ilustración muestra el funcionamiento de la política FIFO. Los paquetes que llegan al nodo, así como el tiempo en el que llegan, se muestran en la parte superior; los paquetes que salen de la cola, y cuando salen, se muestran en la parte inferior. En la zona central se muestra qué paquete se está transmitiendo por el enlace.



Esta política es la utilizada de forma general por los Routers de Internet sin soporte para reservas de calidad de servicio, es decir, aquellos que sólo ofrecen servicio best-effort.

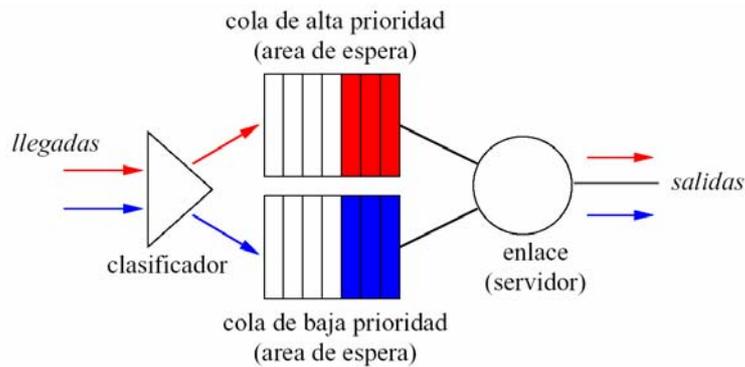
### 5.5.2 GPS (GENERALIZED PROCESSOR SHARING)

Todo el tráfico se envía simultáneamente mezclado de forma que todos obtienen un flujo constante. Esta técnica es completamente irrealizable ya que los paquetes tienen que enviarse enteros. No obstante el modelo de fluidos GPS [Parekh93] es la idealización y todas las políticas van a intentar emularlo de una u otra manera.



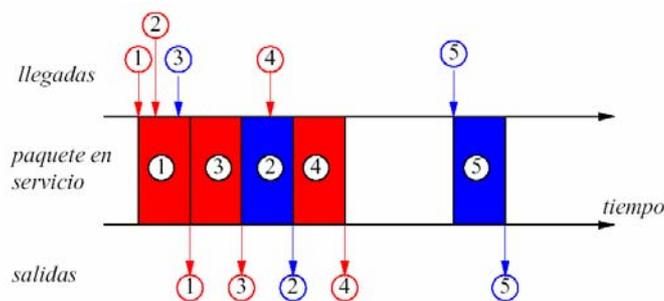
### 5.5.3 SQ (STATISTIC QUEUEING)

La idea es tener varias colas con prioridades diferentes. La cola de prioridad alta recibe todo el tráfico de carga controlada (“*controlled-load*”) y multiplexa todas las sesiones estadísticamente de ahí su nombre SQ (Statistic Queuing), mientras que la cola de prioridad baja recibe el resto de tráfico. Si la cola se dimensiona de manera correcta (hay que tener en cuenta para ello las diferentes reservas de todos los flujos *de* carga controlada) el tráfico carga controlada no se verá afectado por sobrecargas en el nodo [Zhang90] (ya que el resto de tráfico siempre se sirve con prioridad más baja)



La siguiente figura muestra el funcionamiento de la política de colas con prioridades bajo las mismas llegadas del ejemplo anterior. Los paquetes 1, 3 y 4 pertenecen a la clase de mayor prioridad, mientras que los paquetes 2 y 5 pertenecen a la clase de baja prioridad. El paquete 1 llega el primero y es transmitido inmediatamente dado que el enlace no está ocupado. Durante la transmisión del primero llegan los paquetes 2 y 3. Después de la transmisión del paquete 1 se selecciona el paquete 3 (que pertenece a la clase de prioridad alta) aunque llegó después del paquete 2.

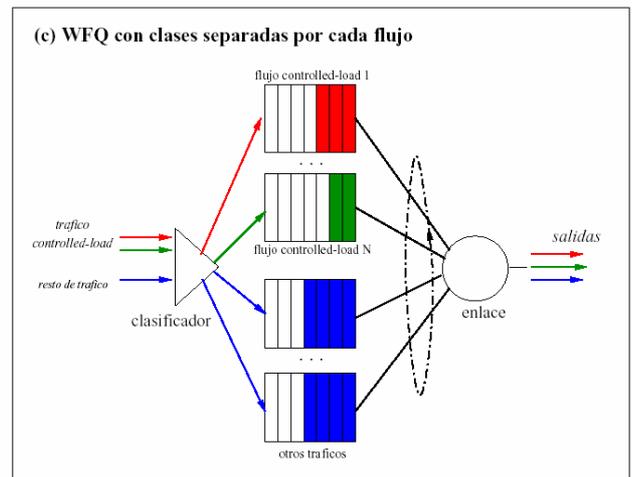
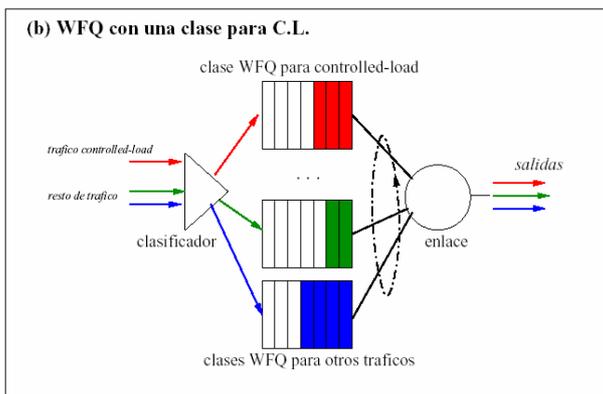
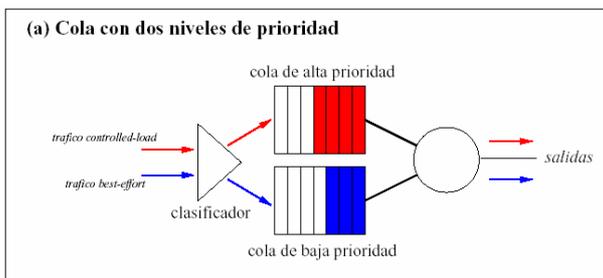
Durante la transmisión del paquete 2 llega un nuevo paquete de prioridad alta. Obsérvese que la llegada del nuevo paquete no interfiere en la transmisión del paquete 2 aunque sea de prioridad más alta, a este comportamiento se le denomina no preemptivo.



El resultado vuelve a ser el mismo: el resto de tráfico no interfiere en la calidad recibida por el tráfico *controlled-load*.

### 5.5.4 WFQ (WEIGHTED FAIR QUEUEING)

Una generalización de la política Round Robin que se ha mostrado muy importante en las arquitecturas con reserva de QoS es la política Weighted Fair Queueing (WFQ) [Demers89]. Esta política es la más utilizada, y entre otros, se utiliza en los routers Cisco. En esta tercera aproximación se asocia una clase de tráfico WFQ a cada uno de los flujos de datos *carga controlada*, es decir, si existen  $N$  flujos con reservas *carga controlada* instaladas, se usan  $N$  clases de tráfico WFQ con sus correspondientes pesos. Esta implementación ofrece la mayor independencia de las tres descritas, ya que ningún flujo (sea o no *carga controlada*) interfiere a otro, dado que cada uno ve un "enlace virtual" con ancho de banda igual a su reserva.



### 5.5.5 WF<sup>2</sup>Q (WORST-CASE FAIR WEIGHTED FAIR QUEUEING)

La característica que debe cumplir una implementación del servicio garantizado es la de aproximarse lo máximo posible al "modelo fluido" (GPS). Existen varios algoritmos propuestos en la bibliografía relacionada que aproximan este modelo, por ejemplo:

- Weighted Fair Queueing (WFQ)
- Worst-case Fair Weighted Fair Queueing (WF<sup>2</sup>Q) [Bennet96]
- Stop And Go Queueing
- Delay EDD
- Jitter EDD
- Hierarchical Round Robin
- Rate Control Priority Queueing
- Self Clocked Fair Queueing (SCFQ)
- Virtual Clock

Todos estos algoritmos aproximan el modelo fluido a un entorno basado en paquetes. Las diferencias entre ellos no es el hecho de que sean o no válidos en términos de retardo máximo (única restricción que introduce el servicio garantizado), sino en las diferencias entre ellos respecto otros parámetros como el retardo medio.

Como ejemplo entre las diferencias que presentan dos de estos algoritmos, la siguiente figura muestra los tiempos en los que se sirven varios paquetes bajo los algoritmos WFQ y WF<sup>2</sup>Q. En esta figura se muestran cuatro diagramas; el primero de ellos muestra los tiempos de negada de los paquetes para las cuatro reservas realizadas (S1..S4). El segundo diagrama muestra el "*Virtual Time*" para cada paquete, el cual es un cálculo del tiempo que se tardaría en servir el paquete en un modelo fluido (también denominado GPS). El virtual

time de cada paquete es el parámetro que utilizan estos dos algoritmos para determinar qué paquete servir en un momento determinado. Los otros dos diagramas de la figura muestran cuándo son servidos cada uno de los paquetes usando los dos algoritmos.

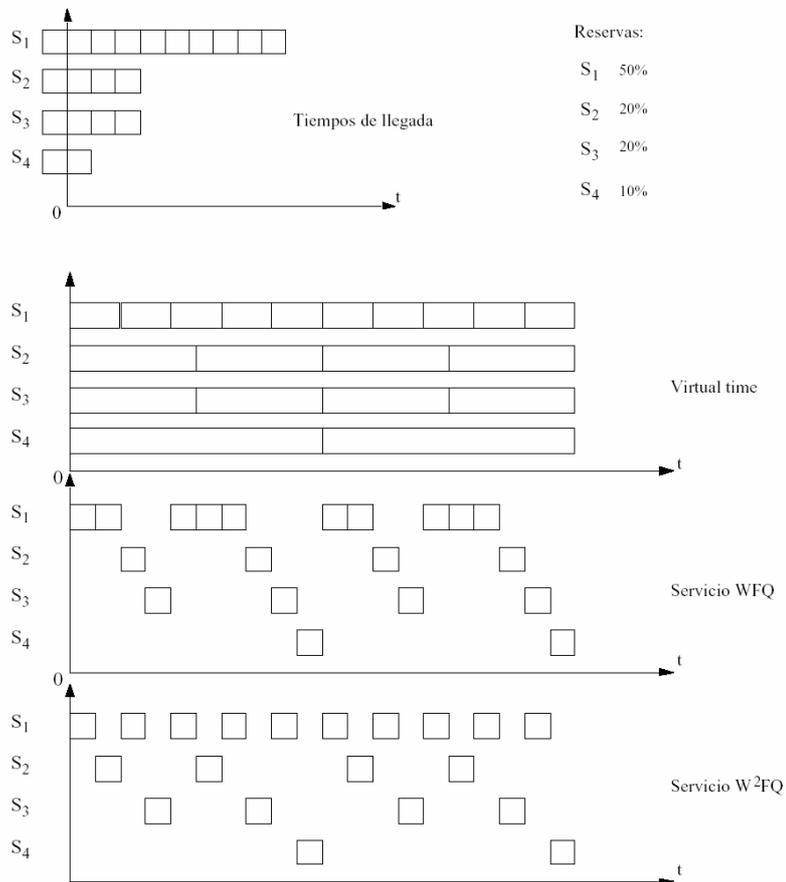
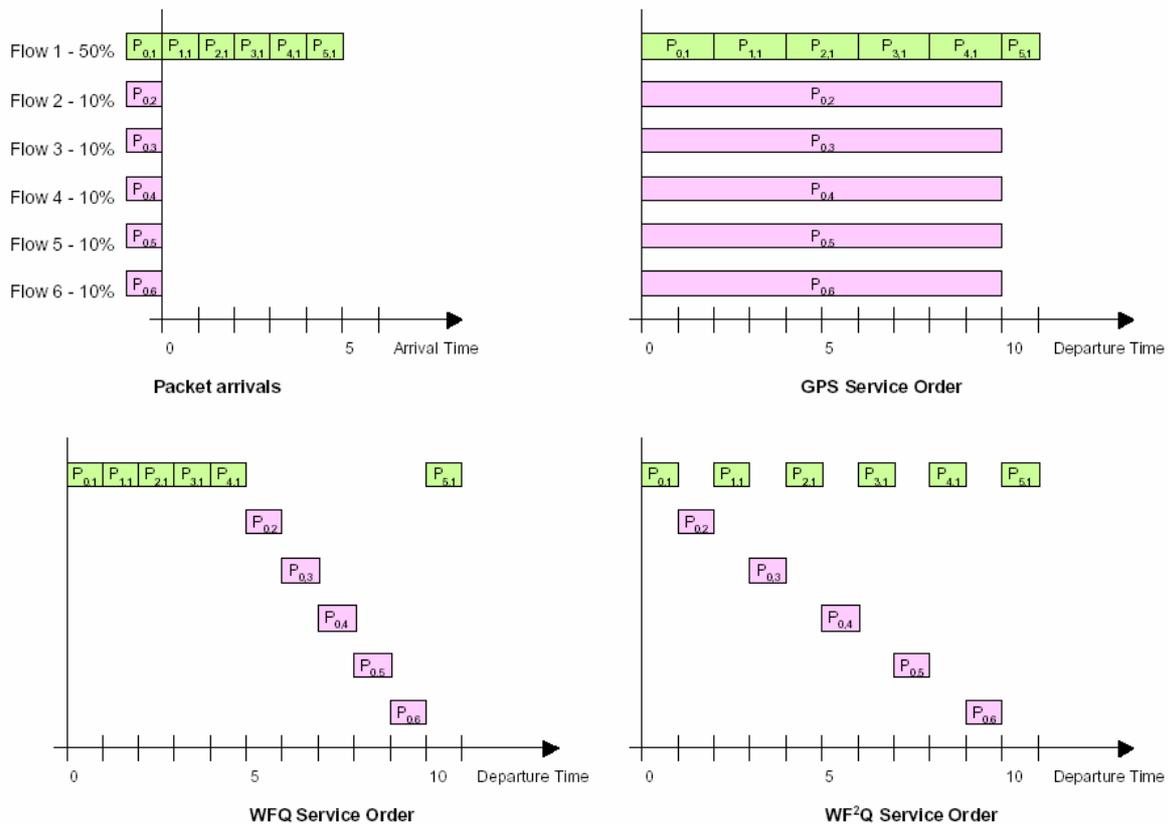


Figura 5.12: Diferencias entre WFQ y WF<sup>2</sup>Q.

Como se aprecia en la siguiente, tanto el algoritmo WFQ como el WF<sup>2</sup>Q ofrecen el mismo servicio, pero mientras que WFQ sirve cada clase a ráfagas de paquetes, WF<sup>2</sup>Q sirve los paquetes más repartidos en el tiempo, lo que redundaría en unos mejores retardos medios. En la siguiente simulación se ve más claramente la diferencia entre estas 2 últimas políticas.



## 5.6 ESTIMACIÓN DEL ANCHO DE BANDA

### 5.6.1 ETHERNET ES UN MEDIO DE ACCESO EN CONTIENDA

Para el correcto funcionamiento de todo este sistema de control de ancho de banda se hacen necesarios mecanismos de control como el limitador de ancho de banda visto en la sección anterior. Pero sin embargo el router no sabe cual es el ancho de banda disponible en cada momento y sin este dato no podría acotar la cantidad de recursos asignados al tráfico QoS. Aunque el sistema operativo nos informa de la velocidad del medio usado,

Ethernet es un medio en contienda. Esto quiere decir que el acceso a él se gana disputándose entre todos los emisores. Esto es así ya que el medio es común y no está gestionado por ningún mecanismo que asegure un medio para asegurar *slots* de tiempo.

La solución es comprobar el tiempo que tarda el transmisor en mandar un paquete. Con ese tiempo y el tamaño de paquete se puede calcular fácilmente el ratio de bits por segundo alcanzados dividiendo el tamaño de los datos por el tiempo tardado en realizar el trabajo.

Como el ancho de banda puede variar según las condiciones de carga no solo de la red sino también de la propia CPU donde se ejecuta el encaminador. Esta técnica se empleará en los elementos de clasificación igual que en los en los adaptadores de salida para estimar no solamente el ancho de banda alcanzable de cada tarjeta de red y así poder medir la velocidad con la que son capaces de trabajar estas unidades clasificadoras.

Para ello se ha creado una clase específica que vemos a continuación.

### 5.6.2 CLASE TBENCHMARK

Esta clase controla los tiempos de trabajo y bloqueo tanto de las unidades clasificadoras como de las transmisoras. Lo que hace es guardar los tiempos de trabajo y descanso totales, así como los bits que han pasado en todo ese tiempo. La clase consta de 2 métodos principales:

- **Entra en Bloqueo:** Asume que ha estado trabajando y actualiza el contador de tiempo trabajado y bytes gestionados.

- **Sale de Bloqueo:** Asume que ha estado en bloqueo y que empieza una nueva sesión se trabajo. Actualiza el acumulador de tiempo de bloqueo.

Mediante el cociente entre la cantidad de información y el tiempo trabajado se puede hallar la estimación del ancho de banda.

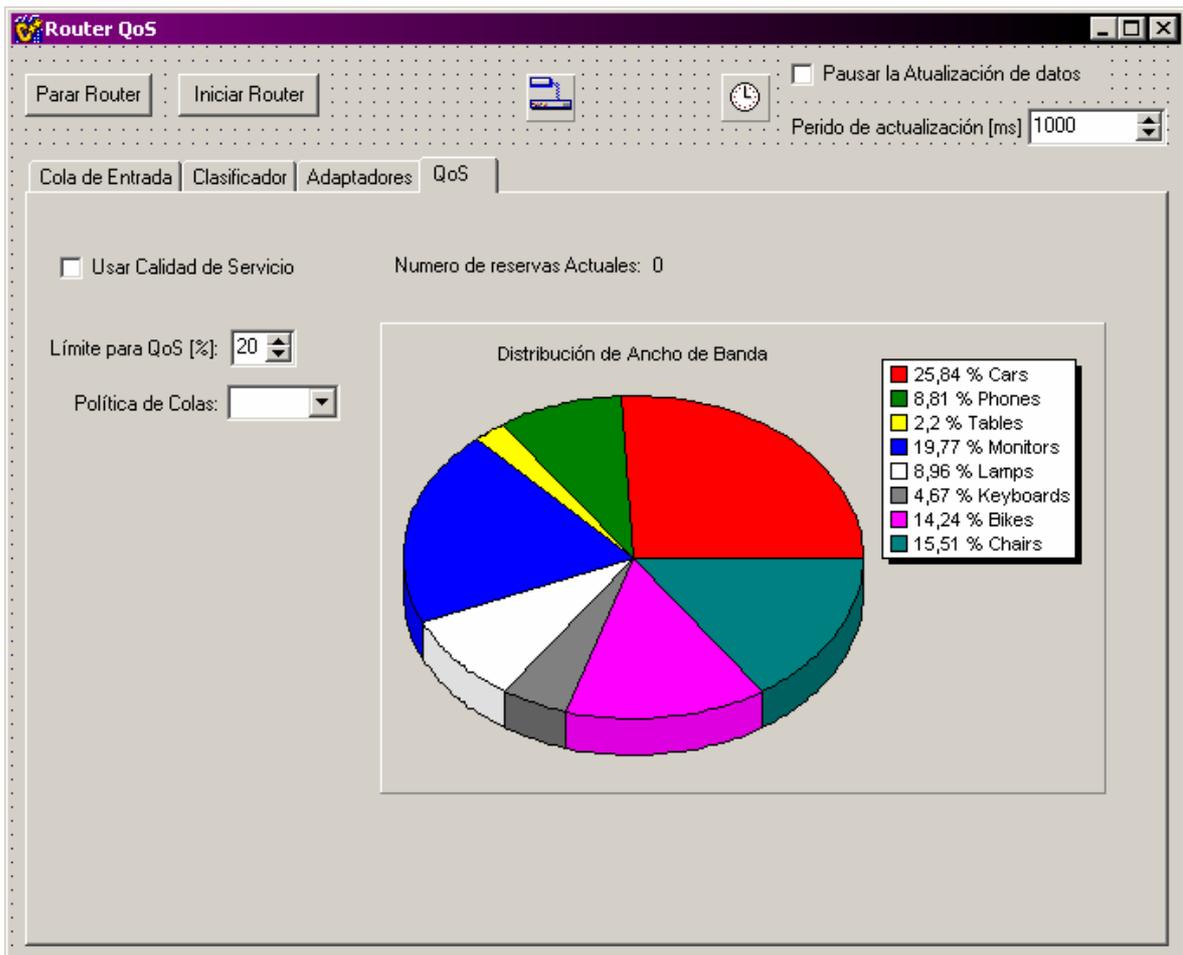
## 5.7 CREACIÓN DEL ENTORNO DE USUARIO

La clase TRouter no solamente va a encargarse de crear el resto de la estructura sino que también va a proveer de capacidad de configuración de ciertos parámetros e información al usuario.

Esta información se presenta de forma gráfica en un entorno fácil de usar. Repasemos cada una de las ventanas tal y como se presentan en el entorno de programación. De esta manera se podrán ver todos los controles usados, incluso aquellos que no tienen representación gráfica en tiempo de ejecución.

### 5.7.1 VENTANA PRINCIPAL

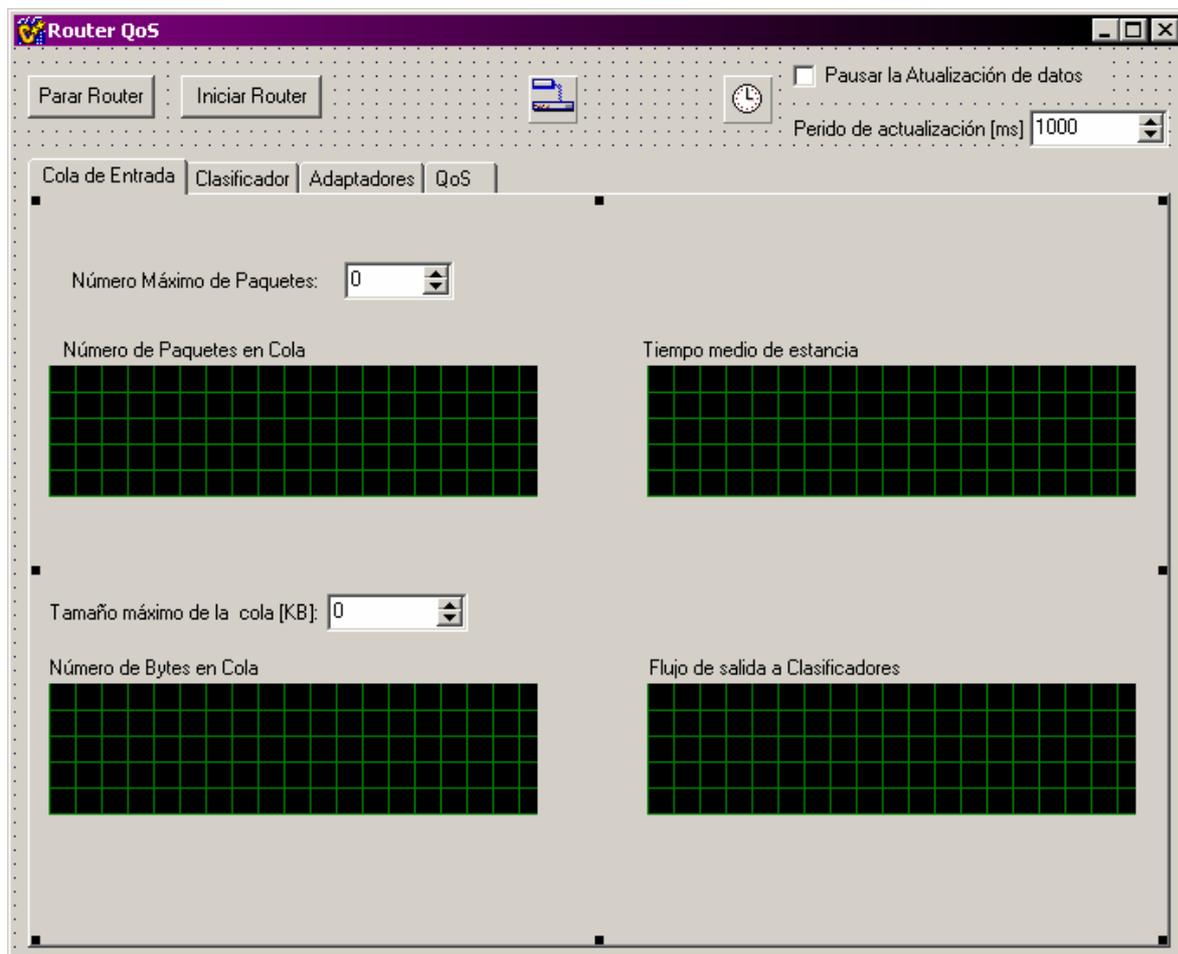
En la ventana principal se encuentran 7 elementos que podemos ver en la siguiente figura:



- Botón para parar el router.
- Botón para iniciar el router.
- Un componente de Borland C++ Builder que nos va a permitir minimizar el programa a la barra de tareas de sistema de Windows, junto al reloj. Este componente se encuentra en la parte superior central.
- Un temporizador que va disparar la actualización periódica de los datos estadísticos.
- Una casilla de marcación que va a detener los disparos del temporizador.

- Un campo donde vamos a poder determinar el periodo de disparo del temporizador.
- Una carpeta con 4 pestañas donde se van a poder configurar los parámetros de cada elemento del router así como visualizar los datos estadísticos proporcionados por el mismo.

### 5.7.2 PESTAÑA COLA DE ENTRADA

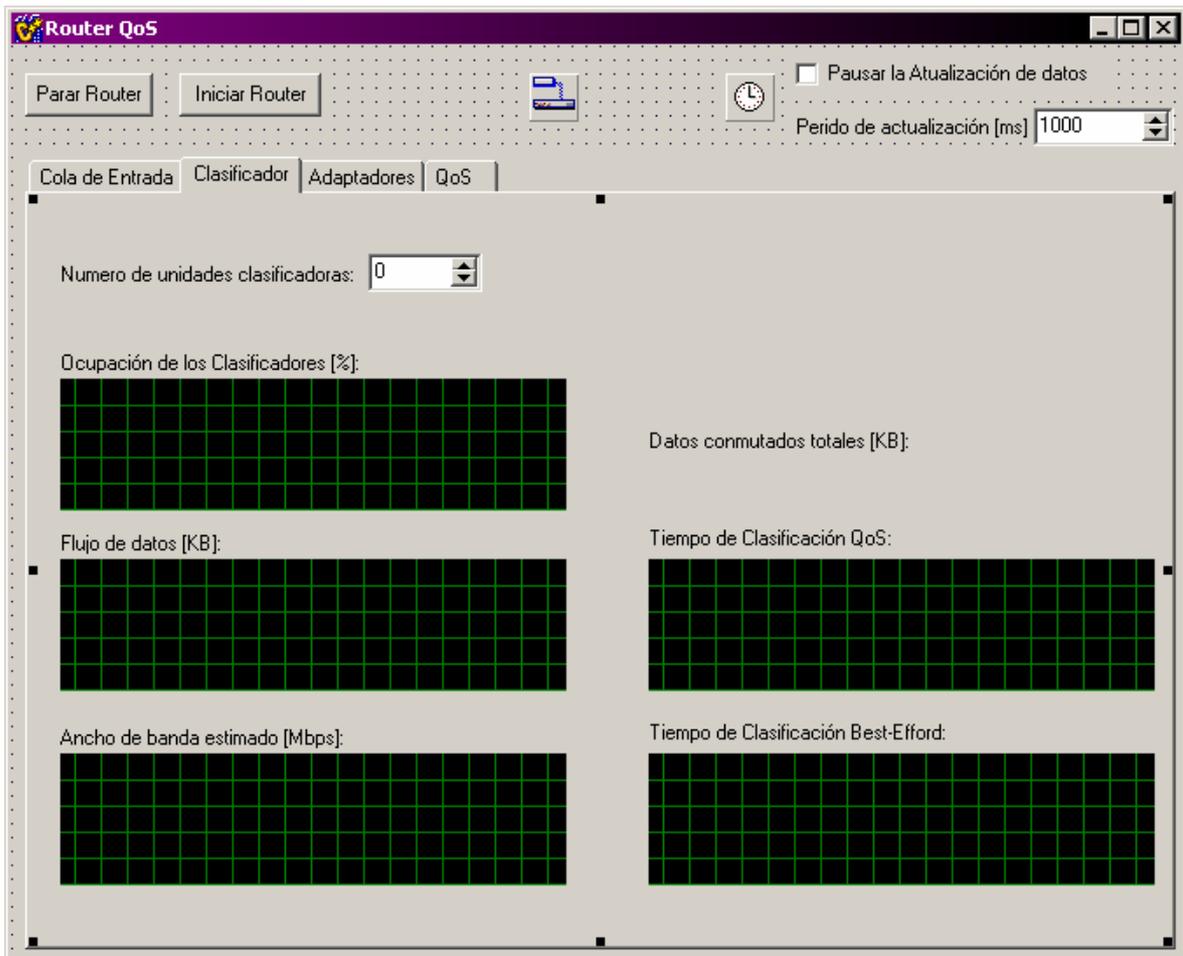


En esta pestaña podemos configurar los límites de la cola FIFO de entrada a la zona de clasificación. Esta limitación puede ser tanto en número de paquetes como en cantidad de memoria usada. En el router se aplicará siempre la más restrictiva.

Junto a estos controles se encuentran cuatro gráficas que nos muestran la evolución de cuatro parámetros:

- Número de paquetes en la cola
- Cantidad de memoria usada
- Tiempo medio de estancia en la cola
- Flujo de salida

### 5.7.3 PESTAÑA CLASIFICADOR



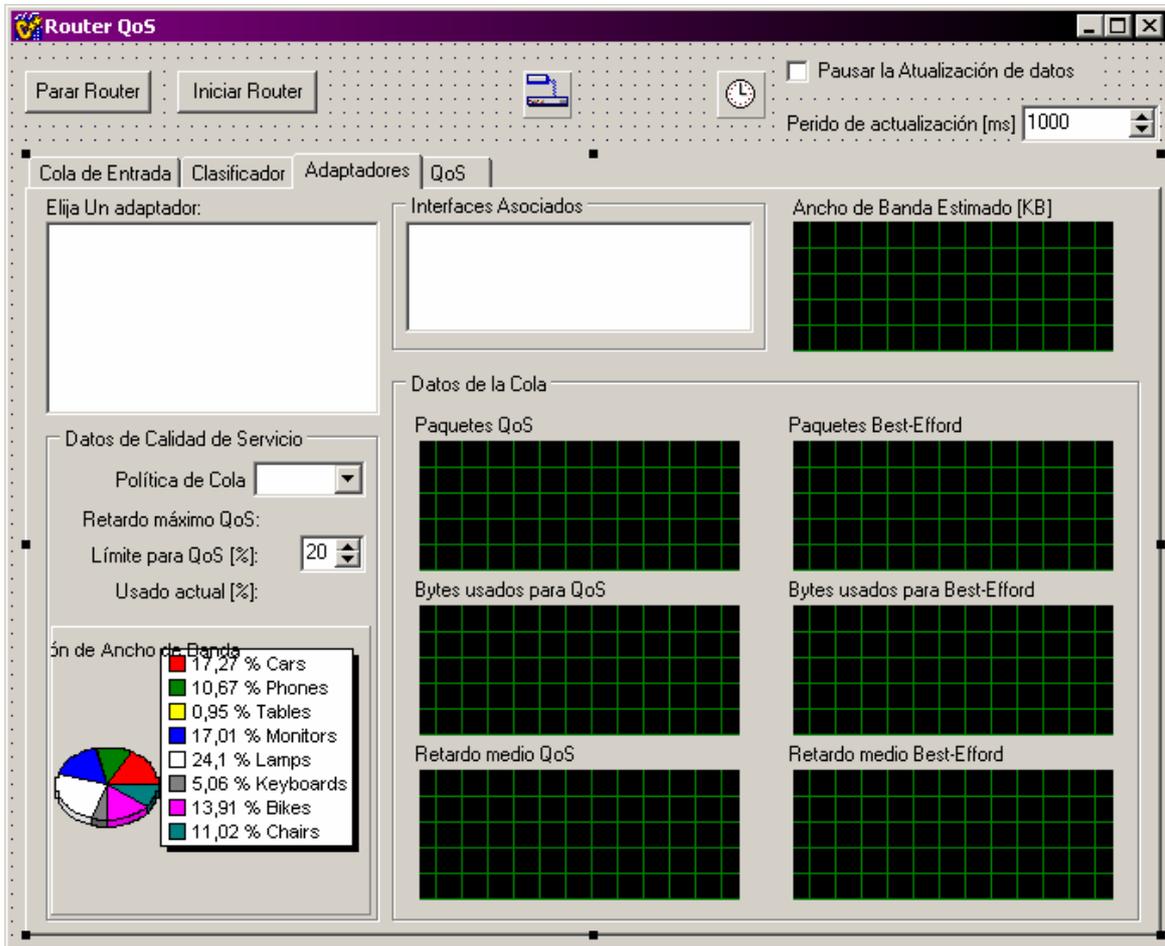
En la sección de clasificación se puede configurar un único parámetro que dirá al router la cantidad de unidades clasificadoras que se quieren ejecutar. El mínimo para que el router funcione es de una unidad.

Las cinco gráficas mostradas nos informan de la evolución de cinco variables relativas al clasificador de paquetes.

- Ocupación de los Clasificadores: Muestra el promedio de la proporción de tiempo que las unidades clasificadoras están trabajando con un paquete. El resto del tiempo estarán bloqueadas en la cola de entrada de paquetes.
- Flujo de datos: Indica la suma del flujo de salida de paquetes de todas las unidades clasificadoras.
- Ancho de banda estimado: Teniendo en cuenta el flujo actual y la carga promedio de las tareas clasificadoras se estima el ancho de banda total dividiendo el flujo entre la carga promedio en tanto por uno.
- Tiempo de clasificación QoS: nos da el tiempo promedio que tarda un paquete QoS en ser clasificado y enviado a la cola correspondiente.
- Tiempo de clasificación *Best-Effort* nos da el tiempo promedio que tarda un paquete sin calidad de servicio en ser clasificado y enviado a la cola del adaptador correspondiente.

Además de estas gráficas se aporta el total de información clasificada por el router.

### 5.7.4 PESTAÑA ADAPTADORES



Sin duda esta es la pestaña más cargada de información ya que ofrece todo tipo de datos del adaptador que elijamos. Repasemos los controles uno por uno:

Lista de adaptadores: Etiquetado como “*Elija un Adaptador*” presenta una lista con los nombres de todos los adaptadores. Cuando seleccionamos uno de ellos el resto de controles ofrecerán datos referentes a ese adaptador.

En la parte superior central se encuentra la lista de interfaces IP asociados a este adaptador, presentando la dirección IP y la máscara de subred.

En la parte superior derecha se encuentra la gráfica que muestra la evolución del ancho de banda estimado. Este valor es fluctuante ya que depende del uso de la Ethernet y del tamaño de los paquetes que se envían, y tenderá a bajar en el caso de que los paquetes sean pequeños o exista carga en la red

El grupo calidad de Servicio. Permite configurar 2 parámetros QoS de la cola asociada al adaptador:

- Política de cola: Se puede elegir entre SQ, SFQ, WFQ o WF<sup>2</sup>Q, algoritmos descritos en la sección 5.5.
- Límite de reservas para QoS en %: Define el límite de ancho de banda que se destina a tráfico garantizado en ese adaptador.

Y ofrece información sobre el retardo máximo de los paquetes con QoS en la cola y el tanto por ciento del ancho de banda total usado por QoS.

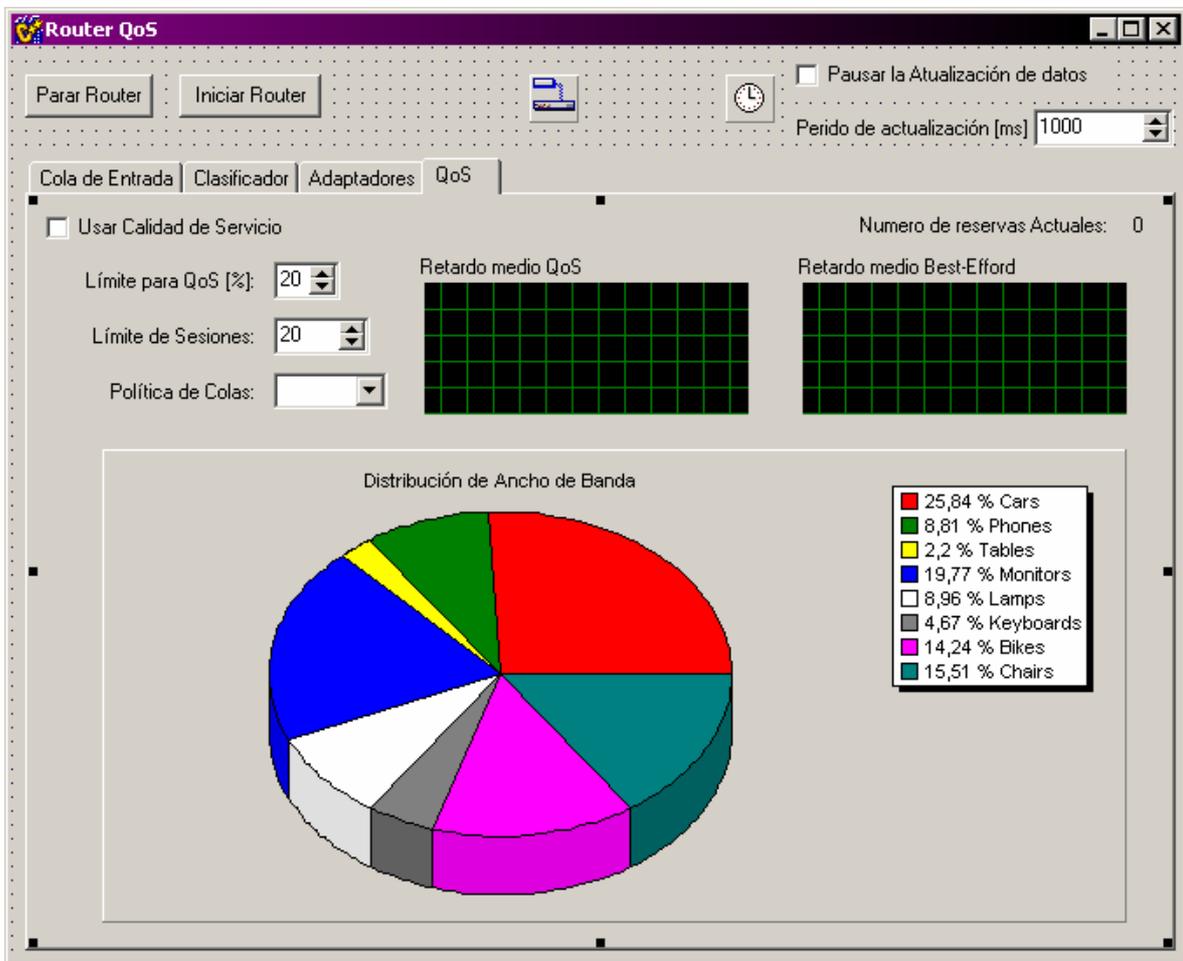
Adicionalmente se incluye una gráfica de tarta donde se puede ver la distribución actual del ancho de banda.

Adicionalmente se ofrecen 6 gráficas que muestran la evolución de 6 parámetros referentes a la cola:

- Paquetes QoS: representa la evolución del número de paquetes QoS que esperan en la cola del adaptador.
- Paquetes *Best-Effort*: representa la evolución del número de paquetes sin QoS que esperan en la cola del adaptador.

- Bytes usados para QoS: representa la evolución de la cantidad de memoria usada para almacenar los paquetes QoS que esperan en la cola
- Bytes usados para Best-Effort: representa la evolución de la cantidad de memoria usada para almacenar los paquetes sin QoS que esperan en la cola
- Retardo medio QoS: Muestra la evolución del tiempo de espera de los paquetes QoS en la cola.
- Retardo medio Best Effort: Muestra la evolución del tiempo de espera de los paquetes sin QoS en la cola.

### 5.7.5 PESTAÑA QoS



Permite configurar y visualizar datos globales de calidad de servicio.

Ofrece cuatro parámetros de configuración:

- Usar Calidad de servicio: Al desactivarlo deshabilita la búsqueda de los datos en la tabla de sesiones RSVP y la planificación de colas, lo que convierte el router en un router convencional.
- Límite de reservas para QoS en %: Define el límite de ancho de banda que se destina a tráfico garantizado en cada uno de los adaptadores.
- Límite de sesiones RSVP autorizadas. Permite limitar el tamaño de la tabla de sesiones RSVP.
- Política de colas: Se puede elegir entre SQ, SFQ, WFQ o WF<sup>2</sup>Q, algoritmos descritos en la sección 5.5. Permite elegir el tipo de planificación de todas las colas de una sola vez.

Adicionalmente ofrece los siguientes datos globales:

- Número de sesiones RSVP que se encuentran activas.
- Tiempo medio de tránsito de los paquetes QoS a través del encaminador.
- Tiempo medio de tránsito de los paquetes sin QoS a través del encaminador.
- Distribución en gráfico de tarta del tráfico total que atraviesa el router.

## 5.8 PRUEBAS DE FUNCIONAMIENTO

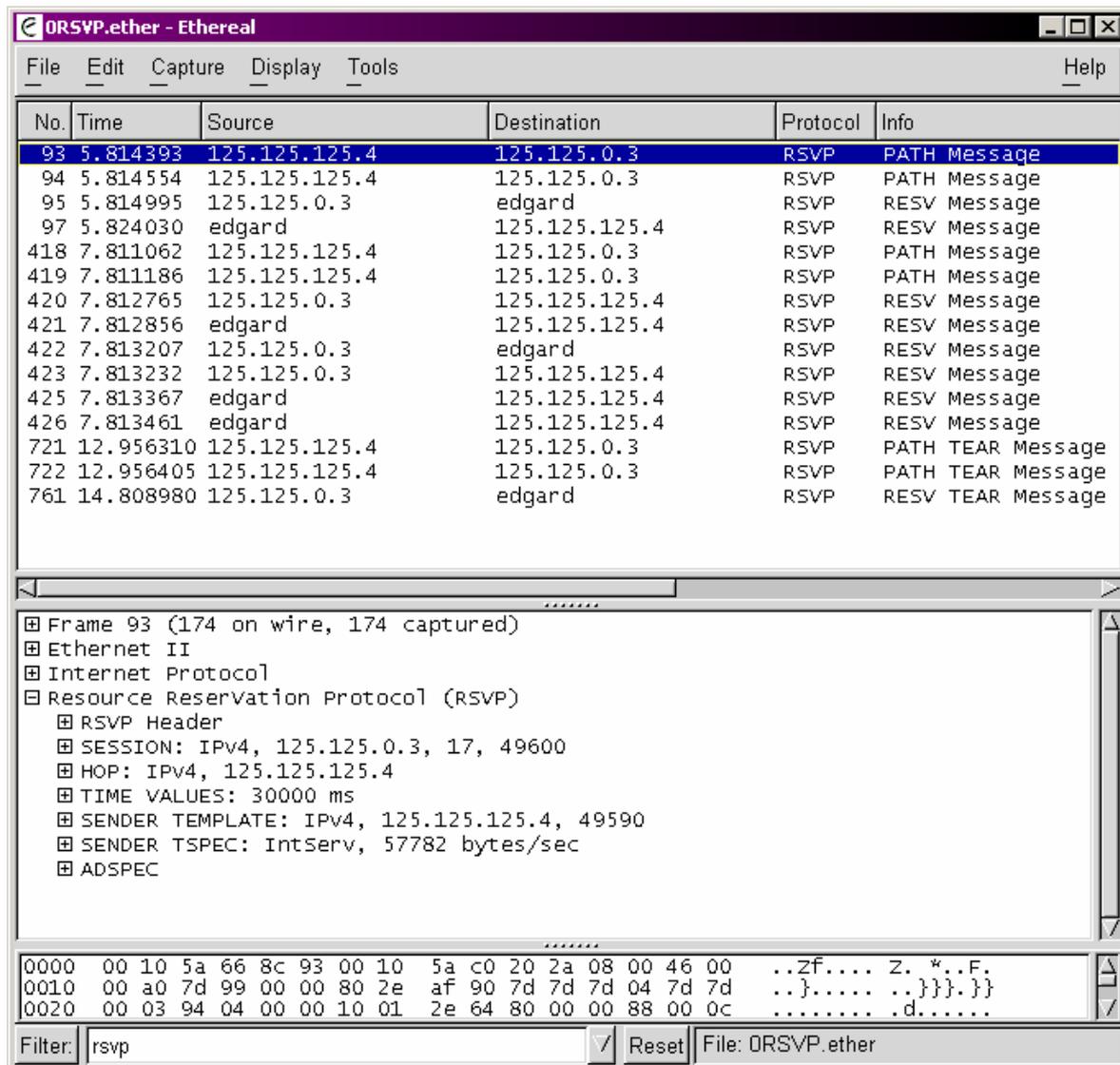
Una vez terminado se comprobó que el router realizaba todas las tareas de clasificación de paquetes, y efectuaba todos los mecanismos de reserva RSVP usando el depurador de código integrado del Borland C++ Builder 5.

Adicionalmente se realizó la misma simulación que para probar el router básico [Sección 4.12].

Aunque el router usaba un único adaptador de red, se pudo configurar dos interfaces IP4 para realizar la prueba. De esta manera se comprobó primeramente con un mensaje de eco ICMP mediante el uso del comando ping como se describe en la sección 3.4.10.

Adicionalmente se comprobó que tanto las transferencias FTP como la videoconferencia usando el software *NetMeeting* funcionaban correctamente en ambas direcciones y sin cargar en ningún momento la CPU del ordenador que hospedaba el router.

La diferencia que hay entre esta simulación y la anterior es que en esta hay una conversación entre los gestores QoS de los *hosts* comunicados por *NetMeeting* y en la captura de paquetes con *Ethereal* se puede ver el flujo de paquetes del protocolo RSVP que pasan por el router. En la siguiente ilustración podemos ver la captura de una comunicación *Netmeeting* de un solo *stream* de video a la que le aplicamos el filtro 'rsvp' para seleccionar únicamente los paquetes relativos a este protocolo.



Podemos ver como los paquetes *path* son enviados de extremo a extremo y los de reserva salto a salto.

La prueba comienza con la solicitud de una ruta para iniciar una sesión, posteriormente se devuelve un mensaje de reserva salto a salto. Después hay algunos paquetes de refresco

del *'soft-state'* y NetMeeting además sabe que está en una conversación porque termina con sendos mensajes de cancelación de ruta y reserva.

Con esta prueba se da por concluida la implementación del código.

## **CAPÍTULO 6: Manual de Usuario**

### **6.1 INTRODUCCIÓN**

Este software añade la funcionalidad de encaminamiento con calidad de servicio RSVP a un sistema basado en Windows 9x, Windows NT o Windows 2000. La configuración es extremadamente sencilla ya que todos los datos de red que usan son los que usa el propio sistema operativo. Este router va a permitir a software que use RSVP como NetMeeting o MSN Messenger, mantener su tráfico en condiciones de sobrecarga del nodo de red.

## 6.2 REQUERIMIENTOS

Este software no tiene muchos requerimientos y es por ello por lo que resulta muy atractivo a la hora de incorporar un router a una red sin necesidad de comprar hardware específico. Los requerimientos son los siguientes:

- Un PC compatible Pentium o superior.
- Al menos una tarjeta de red.
- Microsoft Windows 9x, Windows NT *Service Pack 6* o Windows 2000 instalado.

*Nota:* Si se desea usar este programa bajo Windows 2000 o Windows NT es necesario que el usuario tenga privilegios de administrador

## 6.3 INSTALACIÓN

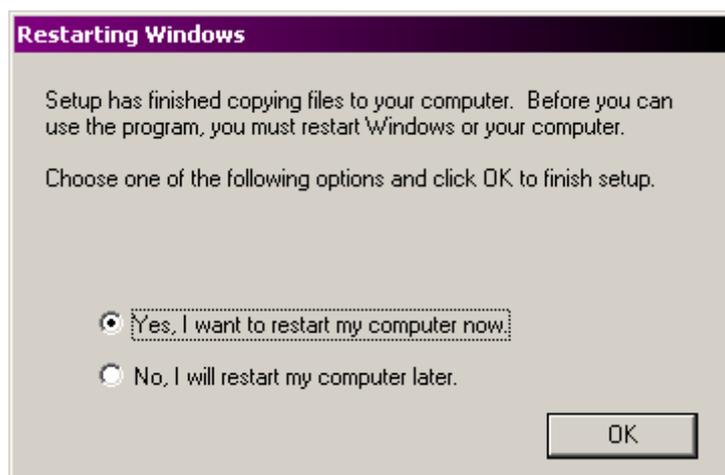
La instalación es muy simple. El único proceso que hay que realizar para poder usar el software es la instalación del driver de acceso a la tarjeta de red. Para ello ejecutaremos el instalador del controlador *WinPCap.exe* con lo que nos saldrá la siguiente pantalla:



Pulsamos el botón *Next* y realizará los pasos necesarios para instalar el controlador correspondiente al sistema operativo que estemos usando. Cuando termina nos muestra el siguiente mensaje de instalación realizada con éxito:



Al pulsar aceptar nos preguntará si deseamos reiniciar el sistema para que los cambios surtan efecto como se muestra en la siguiente ilustración:

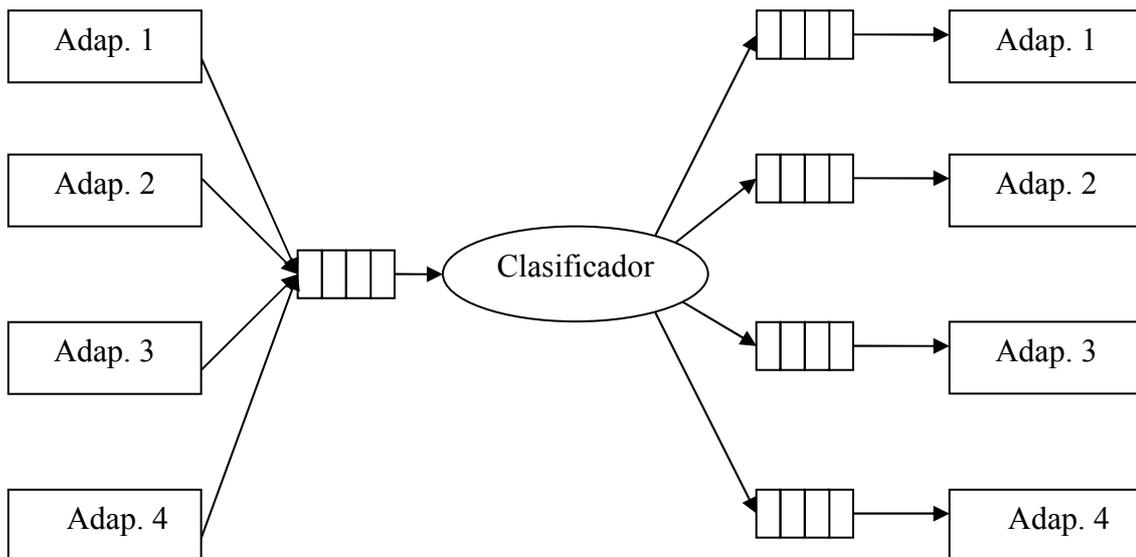


Se recomienda que se elija la primera opción y se reinicie Windows, de lo contrario podría no funcionar. Después de volver a entrar en el sistema ya estamos en condiciones de ejecutar el fichero *RouterQoS.exe* que arranca la aplicación.

**Nota:** Si la instalación se realiza en Windows 2000 o NT es necesario que el usuario tenga privilegios de administrador en esa máquina.

## 6.4 FUNCIONAMIENTO INTERNO

La estructura interna del encaminador sigue el siguiente esquema:



Los adaptadores cuando reciben un paquete lo depositan en una cola de entrada al módulo clasificador. El módulo clasificador decidirá cual es el mejor adaptador para ese paquete y lo introducirá en la cola exclusiva de ese adaptador. Estas colas tienen distintas políticas para dar preferencia a los paquetes garantizados frente al resto. Cuando un paquete sale de la cola se envía a través de la tarjeta de red del adaptador.

El interfaz de usuario nos va a permitir configurar algunos parámetros generales así como opciones específicas para la Cola de entrada, El módulo clasificador y las colas de salida.

## 6.5 INTERFAZ DE USUARIO

Una vez que entramos en el interfaz de usuario nos encontramos con varios elementos. En la ventana principal se encuentran siete elementos. En la siguiente ilustración podemos ver la parte superior donde se encuentran los controles de acceso rápido



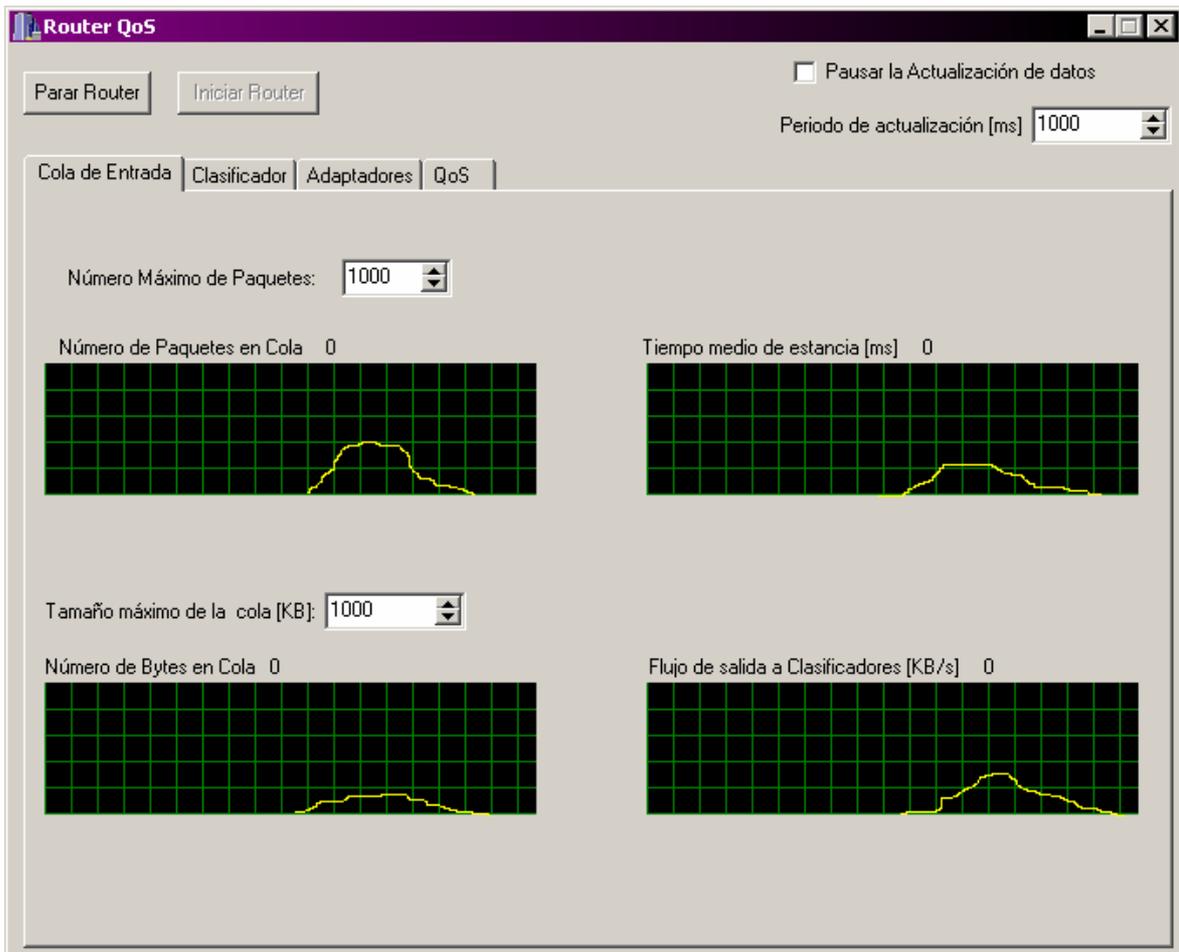
- Botón para llevar la aplicación a un icono situado en la barra de tareas junto al reloj.
- Botón para cerrar la ventana y con ello el encaminador.
- Botón para parar el router.
- Botón para iniciar el router.
- Una casilla de marcación que va a detener la actualización de los datos en pantalla aunque el router seguirá funcionando.
- Un campo donde vamos a poder determinar el periodo de actualización del interfaz.
- Una carpeta con 4 pestañas donde se van a poder configurar los parámetros de cada elemento del router así como visualizar los datos estadísticos proporcionados por el mismo.

Las cuatro pestañas corresponden a:

- La cola de entrada
- El módulo clasificador
- Los adaptadores y sus colas de salida
- Parámetros generales de QoS

Las siguientes secciones se dedicarán a repasar cada pestaña comentando cada uno de los datos de configuración y visualización después de haber realizado una transferencia.

### 6.5.1 PESTAÑA COLA DE ENTRADA

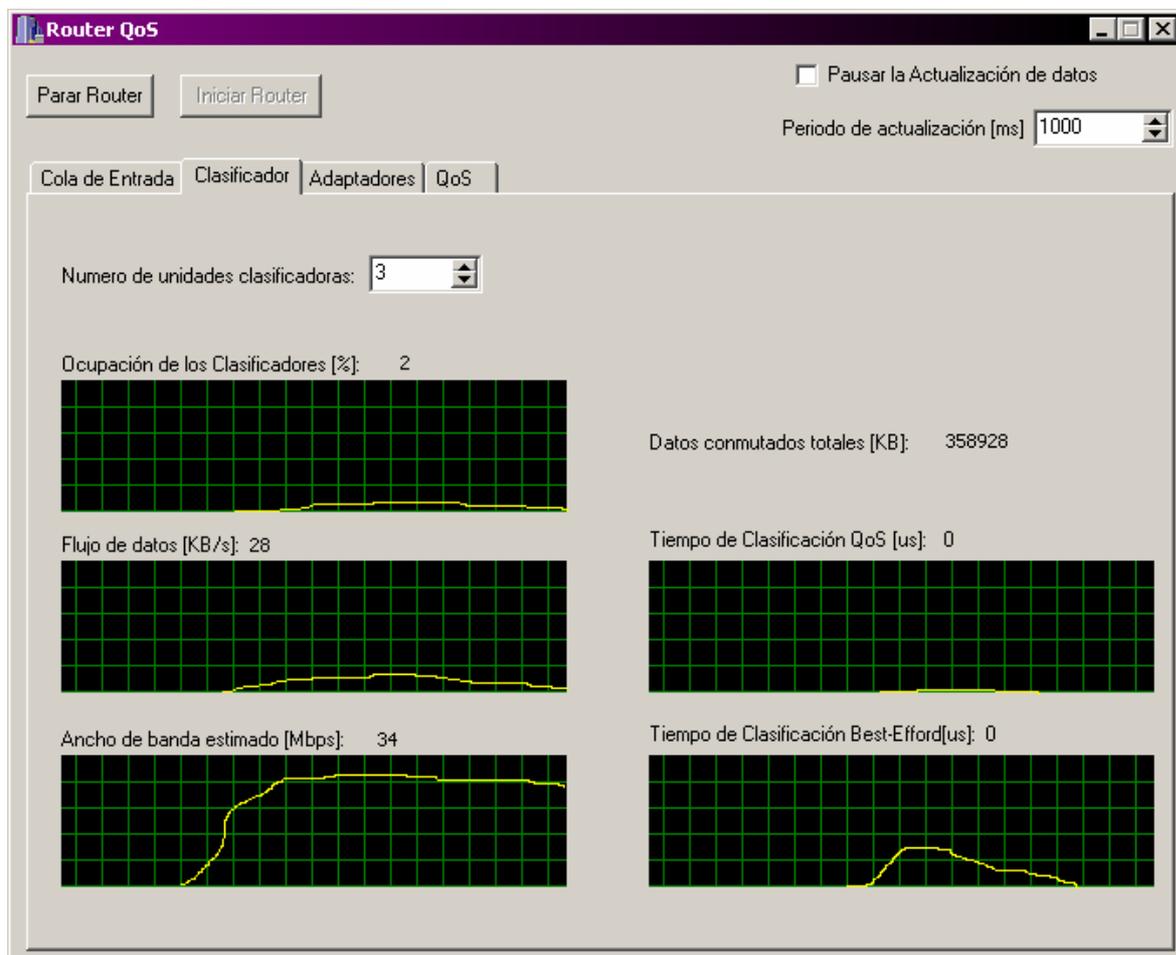


En esta pestaña podemos configurar las limitaciones de la cola de entrada a la zona de clasificación. Esta limitación puede ser tanto en número de paquetes como en cantidad de memoria usada. En el router se aplicará siempre la más restrictiva de las 2.

Junto a estos controles se encuentran cuatro gráficas que nos muestran la evolución de cuatro parámetros:

- Número de paquetes en la cola
- Cantidad de memoria usada por la cola
- Tiempo medio de estancia en la cola de un paquete
- Flujo de salida a la unidad clasificadora

## 6.5.2 PESTAÑA CLASIFICADOR



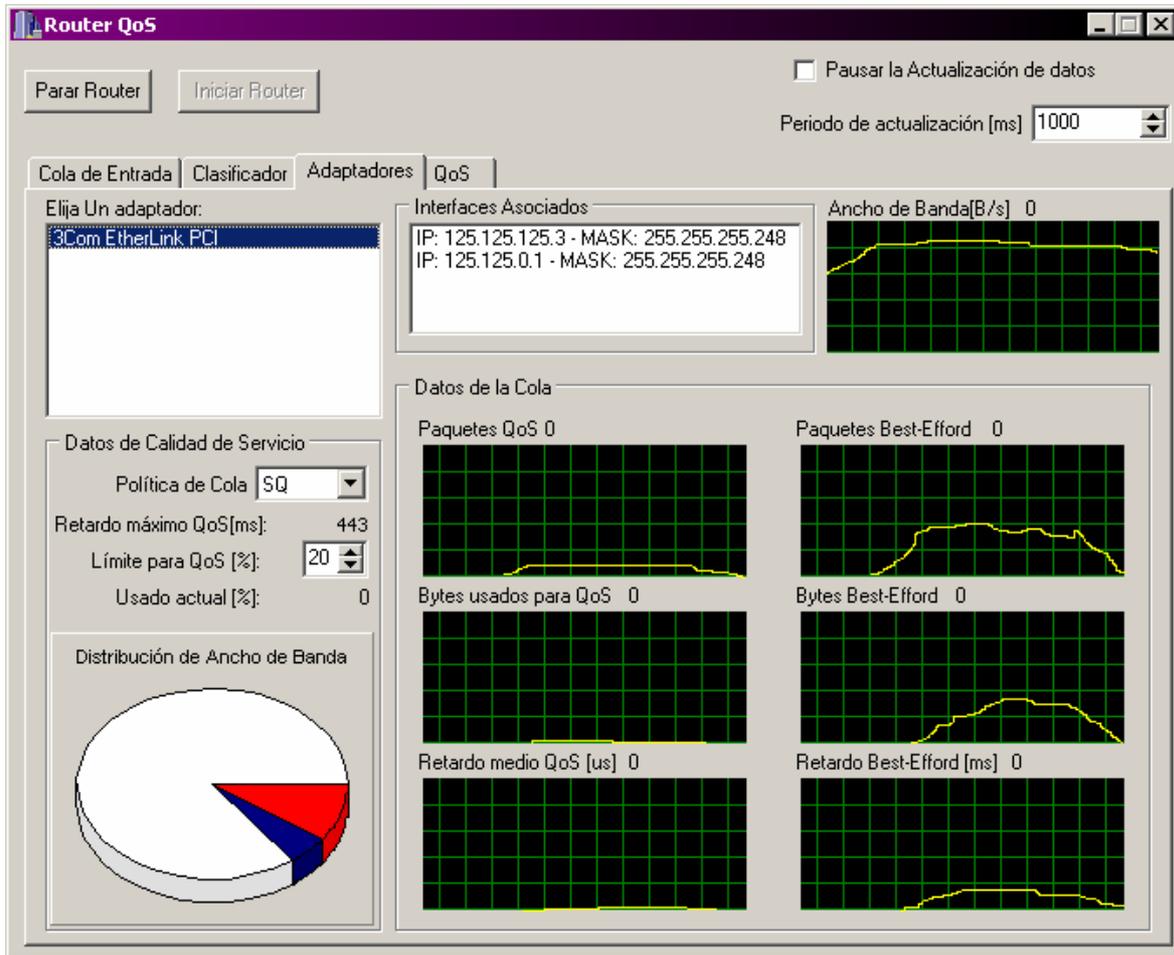
En la sección de clasificación se puede configurar un único parámetro que dirá al router la cantidad de unidades clasificadoras que se quieren ejecutar y que trabajarán en paralelo. El mínimo para que el router funcione es de una unidad. Aunque no hay un máximo para este valor, incrementar mucho este número puede reducir las prestaciones.

Las cinco gráficas mostradas nos informan de la evolución de cinco variables relativas al clasificador de paquetes.

- Ocupación de los Clasificadores: Muestra el promedio de la proporción de tiempo que las unidades clasificadoras están trabajando con un paquete. El resto del tiempo estarán bloqueadas en la cola de entrada de paquetes.
- Flujo de datos: Indica la suma del flujo de salida de paquetes de todas las unidades clasificadoras.
- Ancho de banda estimado: Teniendo en cuenta el flujo actual y la carga promedio de las tareas clasificadoras se estima el ancho de banda total dividiendo el flujo entre la carga promedio en tanto por uno.
- Tiempo de clasificación QoS: nos da el tiempo promedio que tarda un paquete QoS en ser clasificado y enviado a la cola correspondiente.
- Tiempo de clasificación *Best-Efford* nos da el tiempo promedio que tarda un paquete sin calidad de servicio en ser clasificado y enviado a la cola del adaptador correspondiente.

Además de estas gráficas se aporta el total de información clasificada por el router.

### 6.5.3 PESTAÑA ADAPTADORES



Sin duda esta es la pestaña más cargada de información ya que ofrece todo tipo de datos del adaptador que elijamos. Repasemos los controles uno por uno:

Lista de adaptadores: Etiquetado como “*Elija un Adaptador*” presenta una lista con los nombres de todos los adaptadores. Cuando seleccionamos uno de ellos el resto de controles ofrecerán datos referentes a ese adaptador.

En la parte superior central se encuentra la lista de interfaces IP asociados a este adaptador, presentando la dirección IP y la máscara de subred.

En la parte superior derecha se encuentra la gráfica que muestra la evolución del ancho de banda estimado. Este valor es fluctuante ya que depende del uso de la Ethernet y del tamaño de los paquetes que se envían, y tenderá a bajar en el caso de que los paquetes sean pequeños o exista carga en la red

El grupo calidad de Servicio. Permite configurar 2 parámetros QoS de la cola asociada al adaptador:

- Política de cola: Se puede elegir entre SQ, SFQ, WFQ o WF<sup>2</sup>Q, algoritmos descritos en la sección 5.5.
- Límite de reservas para QoS en %: Define el límite de ancho de banda que se destina a tráfico garantizado en ese adaptador.

Y ofrece información sobre el retardo máximo de los paquetes con QoS en la cola y el tanto por ciento del ancho de banda total usado por QoS.

En este grupo también podemos ver una gráfica de tarta donde se puede ver la distribución actual del ancho de banda.

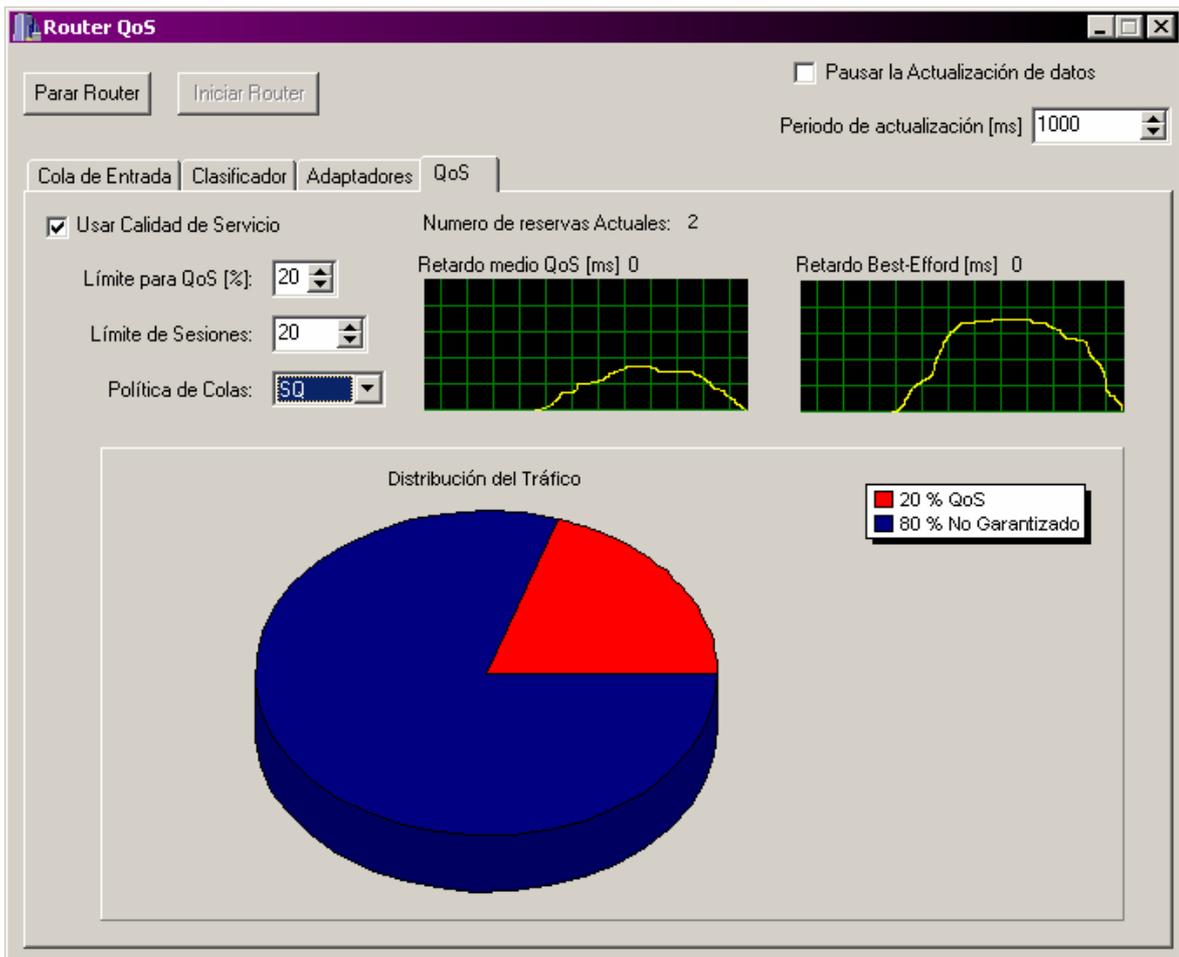
- Tráfico QoS: En rojo
- Tráfico sin QoS: en Azul
- Ancho de banda disponible sin usar: En blanco

Adicionalmente se ofrecen 6 gráficas que muestran la evolución de 6 parámetros referentes a la cola:

- Paquetes QoS: representa la evolución del número de paquetes QoS que esperan actualmente en la cola del adaptador.
- Paquetes *Best-Efford*: representa la evolución del número de paquetes sin QoS que esperan actualmente en la cola del adaptador.
- Bytes usados para QoS: representa la evolución de la cantidad de memoria usada para almacenar los paquetes QoS que esperan actualmente en la cola
- Bytes usados para Best-Efford: representa la evolución de la cantidad de memoria usada para almacenar los paquetes sin QoS que esperan actualmente en la cola
- Retardo medio QoS: Muestra la evolución del tiempo de estancia de los paquetes QoS en la cola.
- Retardo medio Best Efford: Muestra la evolución del tiempo de estancia de los paquetes sin QoS en la cola.

#### **6.5.4 PESTAÑA QOS**

Permite configurar y visualizar datos globales de calidad de servicio.



Ofrece cuatro parámetros de configuración:

- Usar Calidad de servicio: Al desactivarlo deshabilita la búsqueda de los datos en la tabla de sesiones activas RSVP y la planificación de colas de salida, lo que convierte el router en un router convencional.
- Límite de reservas para QoS en %: Define el límite de ancho de banda que se destina a tráfico garantizado en cada uno de los adaptadores. Este control

configura la reserva en todas las colas. Se puede configurar este parámetro independientemente para cada dispositivo de red.

- Límite de sesiones RSVP autorizadas. Permite limitar el número de sesiones RSVP que se pueden reservar en el nodo.
- Política de colas: Se puede elegir entre SQ, SFQ, WFQ o WF<sup>2</sup>Q, algoritmos descritos en la sección 5.5. Permite elegir el tipo de planificación de todas las colas de una sola vez. Este control configura todas las colas. Se puede configurar este parámetro independientemente para cada dispositivo de red.

Adicionalmente ofrece los siguientes datos globales:

- Número de sesiones RSVP que se encuentran activas.
- Tiempo medio de tránsito de los paquetes QoS a través del encaminador.
- Tiempo medio de tránsito de los paquetes sin QoS a través del encaminador.
- Distribución del tráfico total que atraviesa el router.

# **CAPÍTULO 7: Conclusiones y líneas futuras**

## **7.1 CONCLUSIONES FINALES**

Durante el desarrollo de este proyecto se pueden sacar las siguientes conclusiones:

1. Este proyecto es un paso más hacia la estandarización del protocolo RSVP ya que va a permitir estudiar el comportamiento de tráfico real en un entorno tan usual como son las redes locales con medio físico compartido. Hasta ahora gran parte el estudio de las ventajas y problemas del encaminamiento con calidad de servicio se ha venido

realizando con simuladores en entornos no tan extendidos como Microsoft Windows. Así mismo puede usarse para comunicar subredes en una red corporativa de implantación rápida y económica, ayudando a las comunicaciones en tiempo real tales como telemetría, videoconferencia, etc...

2. La elaboración de este software no sólo ayudará a extender este protocolo sino que en su desarrollo hemos profundizado en otros importantes aspectos relacionados con el protocolo como:

- Funcionamiento de colas con distintas políticas de planificación.
- Limitar el ancho de banda mediante el cubo de testigos (*token-bucket*).
- Realizar una estimación de la capacidad de medios tan impredecibles como lo es Ethernet.
- Identificar una sintaxis de comunicación.

3. Durante el mismo se ha comprobado que el protocolo RSVP necesita de una estandarización más cerrada ya que no todas las implementaciones siguen el protocolo estrictamente. Esto ha obligado a relajar los parámetros de exigencia del router para poder ser compatible con un mayor número de aplicaciones.

4. Igualmente hemos profundizado en otros temas que poco tienen que ver con RSVP pero sí que han sido de vital importancia para la elaboración de este software. Destacamos las cuestiones más importantes que se han tenido que afrontar y a las que se ha aportado solución:

- Programación orientada a Objetos en Borland C++ Builder.
- Organización de adaptadores en Windows y acceso directo al hardware de red.
- Implementar la multitarea.

- Solucionar los problemas relativos con la multitarea como acceso a zonas críticas o sincronización de tareas.
  - Programación visual
5. Por otra parte se ha enfatizado en la importancia de elegir bien la herramienta de programación a la hora de crear una aplicación ya que en este caso era de extrema importancia la velocidad del código generado. Además hay que tener en cuenta las facilidades que nos ofrece la herramienta para construir los objetos que necesitemos y gestionar los problemas que puedan producirse.
  6. El uso de la multitarea tiene muchas ventajas y merece la pena ser usada ya que los problemas que se nos pueden presentar tienen fácil solución. Ayuda a mantener controlada la carga de CPU, optimiza el uso de recursos y aumenta la eficiencia del código.

## 7.2 LÍNEAS FUTURAS

Este proyecto deja el campo abierto a varios frentes. Los más interesantes.

1. Usar el conmutador para realizar estudios con tráfico QoS real bajo condiciones de carga en una red económica basada en PCs bajo el sistema operativo Windows. Esta línea de desarrollo es importante ya que bajo Windows funcionan la mayoría de aplicaciones usadas por el usuario típico y el futuro de la comunicación de datos en tiempo real parece que va a estar muy concentrado a este sistema operativo.

2. Añadir funcionalidad IPv6 al router, con lo que se tendría una herramienta preparada para el futuro estándar en protocolos de red.
3. Añadir funcionalidad NAPT (*Network Address and Port Translation*) al router. Con lo que se aliviaría el creciente problema de agotamiento de direcciones IP4 mediante la conexión de red privada a una única dirección pública. Sería interesante un reconocedor de protocolos de aplicación importantes de videoconferencia que hoy en día son impracticables a través de Routers NAPT debido a que introducen la dirección IP4 en capas superiores al igual que hace RSVP.
4. Añadir funcionalidad de un router promiscuo que pueda encaminar y recibir paquetes a otros Routers sin necesidad de configurar los encaminadores adyacentes respondiendo las peticiones ARP de los Routers adyacentes cuando intentan ubicar uno de los ordenadores aislados por el router.

## Bibliografía

- [Wright] R. Wright, W. Richard Stevens, “*TCP-IP illustrated*” Volumen 2, capítulo 31. Addison-Wesley professional computing series.
- [Charte99] F. Charte, “*Programación con C++ Builder 4*”. Anaya Multimedia 1999
- [TheBits] The Bits, “*A discusión about Synchronization*”. The C++ Builder Information & Tutorial Site. Documento en formato HTML sin acceso directo, disponible en: <http://Thebits.org/>
- [Casado00] M. Casado, A. Diaz, “*Estudio de protocolos para la garantía de calidad de servicios en redes IP*”. PFC Dpto. DTE de la Universidad de Málaga, 2000

- [Adan99] D. Adán, J.J. Lopez “*Simulador de Protocolo de Reserva de Recursos (RSVP)*” PFC Dpto. FIB de la Uiv. Politécnica de Calatuña. 1999
- [Apparna99] R. Apparna, B. Premkumar. “*Monitoring Ethernet Network Activity with NDIS driver*”. California Software Laboratories, 4 de Enero de 1999.
- [Viano01] P. Viano, L. Degioanni. “*Packet Driver API: Programmer’s Manual*” Netgroup del Politécnico de Torino 2001.
- [SDDKEx] Microsoft “*Software Development Kit and Driver Development Kit Examples*”, Microsoft Corporation.
- [MS3com] Microsoft Corporation, 3Com Corporation, “*NDIS, Network Driver Interface Specification*”, May 1988
- [MSDDK] Microsoft “*Windows 95, Windows 98, Windows NT and Windows 2000 Driver Development Kit documentation*”, Microsoft Corporation.
- [MSDN] Microsoft “*MSDN Library*”, Microsoft Corporation, agosto 2000.
- [RFC791] J. Postel, Request for Comments RFC791, “*Internet Protocol specification*”, Septiembre 1981.
- [RFC792] M. Borden, Request for Comments RFC1821, “*ICMP : Internet Control Message Protocol*”, Septiembre 1981.
- [RFC768] Postel, J., “*User Datagram Protocol*”, STD 7 RFC768, Agosto 1980.

- 
- [RFC793] Postel, J., "*Transmission Control Protocol*", STD 7 RFC793, Septiembre 1981.
- [RFC 2205] R.Branden, L.Zhang, S.Berson, S.Herzog, S.Jamin. "*Resource ReSerVation (RSVP) Version 1 Functional Specification*" RFC 2205, Septiembre 1997.
- [RFC 2209] R. Braden, L. Zhang. "*Resource ReSerVation Protocol (RSVP) - Version 1 Message Processing Rules*" RFC 2209, Septiembre 1997.
- [RFC 2210] J.Wroclawski. "*The Use of RSVP with IETF Integrated Services*" RFC 2210, Septiembre 1997.
- [RFC 2211] J.Wroclawski. "*Specifications of the Controlled-Load Network Element Service*" RFC 2211, Septiembre 1997.
- [RFC 2212] S.Shenker, C.Partridge, R.Guerin. "*Specifications of Guaranteed Quality oo Service*". RFC 2212, Septiembre 1997.
- [RFC 2215] S.Shenker, J. Wroclawski. "*General Characterization Pararneters oo Integrated Service*". Network Elements" RFC 2215, Septiembre 1997.
- [Demers89] A.Demers, S.Keshav, S.Shenker. "*Analysis and Simulation of a Fair Queueing Algorithm*". Journal of Internetworking: Research and Experience, Vol.1, No.1, 1990, tambien en Proceedings of ACM SIGCOMM89.
- [Bennet96] J.C.R.Bennet, H.Zhang. "*WF2Q: Worst-case Fair Weighted Fair Queueing*". IEEE INFOCOM96, San Francisco, 1996.

## Bibliografia

---

- [Zhang90] L.Zhang. "*Virtual Cloc: A new trafficControl Algorith For Packet Switching Networks*". ACM SIGCOMM, p19-29, Agosto 1990
- [Golestani94] S.J. Golestani. "*A Self-Clocked Fair Queueing Scheme for High Speed Applications*". INFOCOMM, 1994
- [Parekh93] A.Parekh, R.Gallager, "*A generalized processor sharing approach to flow control in integrated services networks: the single node case*" IEEE/ACM Transactions on Networking, Vol.1, No.3, 1993.