TESIS DEFENDIDA POR

Álvaro Armenta Ramade

Y APROBADA POR EL SIGUIENTE COMITÉ

M.C. Jorge Enrique Preciado Velasco

Director del Comité

Dr. José Luis Medina Monroy

Miembro del Comité

Dra. Caridad Emma Anias Calderón

Miembro del Comité

M.C Carlos Roberto Duarte Muñoz

Miembro del Comité

M.C. Raúl Rivera Rodríguez

Miembro del Comité

Dr. José Luis Medina Monroy

Jefe del Departamento de

Electrónica y Telecomunicaciones

Dr. Luis Alberto Delgado Argote

Director de Estudios de Posgrado

15 de agosto de 2001

Contenido

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA

División de Física Aplicada

Departamento de
Electrónica y Telecomunicaciones

Análisis de desempeño de esquemas de control de tráfico en una red de cobertura amplia utilizando el protocolo IP

Tesis

Que para cubrir parcialmente los requisitos necesarios para obtener
el grado de MAESTRO EN CIENCIAS presenta

Álvaro Armenta Ramade

Ensenada, Baja California, Septiembre de 2001.

RESUMEN de la tesis presentada por **Álvaro Armenta Ramade** como requisito parcial para la obtención del grado de **MAESTRO EN CIENCIAS** en **ELECTRÓNICA Y TELECOMUNICACIONES**, Ensenada, Baja California, México. Septiembre de 2001.

Análisis de desempeño de esquemas de control de tráfico en una red de cobertura amplia utilizando el protocolo IP

Resumen aprobado por:

M.C. Jorge E. Preciado Velasco
Director de tesis

En la actualidad es posible encontrar una gran cantidad de tipos de tráfico en las redes que se basan en el protocolo IP: aplicaciones de correo electrónico, acceso al web, aplicaciones multimedia (incluyendo aquéllas que se reproducen en tiempo real) y comercio electrónico entre otras. La capacidad de los enlaces no es infinita, por lo que es necesario diferenciar las diferentes aplicaciones y beneficiar con más recursos a aquéllas (en especial a las aplicaciones de tiempo real) que requieran un desempeño mayor de la red. El objetivo es disminuir los efectos que las condiciones de congestión en la red puedan provocar sobre las aplicaciones.

Para solucionar estos retos, se han implementado diversos mecanismos de calidad de servicio (QoS) que abordan este problema utilizando una estrategia diferente. Sin embargo, sin importar que mecanismo de QoS se utilice, los mecanismos de control de tráfico (calendarizadores de paquetes) son los encargados de transmitir los paquetes de los nodos.

En este trabajo se realizó una comparación de los cuatro mecanismos de control de tráfico más comunes: FIFO, PQ, WFQ y CBQ. La comparación se llevó a cabo con cuatro tipos de tráficos, entre los que se incluye una aplicación de reproducción en tiempo real. Se compararon cuatro parámetros de desempeño los cuales afectan principalmente a las aplicaciones de tiempo real: jitter, retardo de principio a fin, tasa de pérdida de paquetes y el caudal eficaz.

Los resultados obtenidos permitieron determinar el desempeño de cada uno de los mecanismos de control de tráfico así como su correlación con los Acuerdos de Nivel de Servicio (SLA). Los mecanismos que permiten ofrecer varios niveles de servicio son los que obtienen un mejor desempeño, sin embargo, la determinación de cuál mecanismo es el mejor, dependerá de los niveles de servicio y de la confiabilidad que se requieren ofrecer en la red.

Palabras claves: QoS, SLA, calendarizadores, modelos de servicios.

ABSTRAC of the thesis presented by **Álvaro Armenta Ramade** as a partial requirement to obtain the **MASTER** of **SCIENCE** degree en **ELECTRONIC** and **TELECOMMUNICATIONS.** , Ensenada, Baja California, México. August of 2001.

Perfomance analysis of packet schedulers in a Wide Area Network using IP

In today networks, it's possible to find a great amount of applications types: email, web access, multimedia (including real time) and e-commerce applications are all sharing the same media. The capacity of links is not infinite, reason why it is necessary to differentiate between applications and give more network resources to those applications (specially real time applications) that require better performance from the network. The objective is to trim down the negative effects of network congestion to the applications.

In order to solve these challenges, diverse quality of service (QoS) mechanisms have been implemented that approach this problem using a different strategy. Nevertheless, without mattering what mechanism of QoS is used, packet schedulers are responsible of transmitting packages of the nodes.

A comparison of the four more common packet schedulers (FIFO, PQ, WFQ and CBQ) was made. The comparison was carried out with four types of traffic, between which a real time application is included. Four parameters of performance were compared which mainly affect real time applications: jitter, end-to-end delay, loss packet rate and throughput.

Results obtained from this work allowed to determine the performance of each one of the packet schedulers as well as its correlation with the Service Level Agreements (SLA). Packet schedulers that offer several levels of service, give a better performance, nevertheless, the determination of which scheduler is the best one, will depend on the levels of service and the trustworthiness that are required in the network

Keywords: QoS, SLA, packet schedulers, service models.

DEDICATORIA

A mis padres

Filiberto y Evelia

Gracias por todo el apoyo para llegar hasta donde estoy ahora.

A mi esposa Liliana y a mi hija Leslie,

Con todo mi amor. Gracias por su paciencia y comprensión.
Sin ellas, este trabajo no hubiera sido posible.

AGRADECIMIENTOS

A mi comité de tesis: Jorge Preciado, Caridad Anias, José Luis Medina, Raúl Rivera Rodríguez y Carlos Duarte. Gracias por todo su apoyo.

Al personal de la Dirección de Estudios de Posgrado: Federico, Margarita, Citlali, Ivonne, Ana María. Gracias por los innumerables apoyos que recibí de ustedes, sin ellos, no hubiera sido posible este trabajo.

A todos mis compañeros de la Dirección de Vinculación: Marisa, Norma, Marcela, Julieta, Elizabeth, Marina, Enrique, Ulises y Carlos. Gracias por su amistad y apoyo.

A Ulises Cruz y Carlos Duarte, gracias por otorgarme todas las facilidades para seguir estudiando.

Al CICESE por dejarme ser parte de él

Al Consejo Nacional de Ciencia y Tecnología
por el apoyo económico para realizar mis estudios.

Índice de figuras

Índice de tablas

Análisis de desempeño de esquemas de control de tráfico en una red de cobertura amplia utilizando el protocolo IP

I Introducción

I.1 Antecedentes

La generalización del empleo de computadoras de los empresarios y el acelerado crecimiento en el uso doméstico de ellas, junto con la reducción en los costos por el acceso a la Internet, ha propiciado que muchas de las aplicaciones multimedios, las cuales eran del dominio de los CD-ROMs y de los enlaces dedicados que contratan las empresas, se extiendan a la Internet.

La cobertura casi mundial de la *red de redes* (como se le conoce a la Internet), permite que aplicaciones como la voz sobre IP se convierta en una opción viable sobre la telefonía convencional para la comunicación de larga distancia. Aquel dispositivo de ciencia-ficción que aparecía en las revistas de Mecánica Popular, donde era posible ver a la persona con la que se hablaba, el videoteléfono, se ha convertido en una realidad gracias a las aplicaciones de videoconferencia y a la Internet. Las visiones futurísticas de comprar desde nuestra casa se llevan a cabo hoy en día en la Internet gracias a las aplicaciones de comercio electrónico.

Las redes actuales han aumentado en complejidad y en capacidad. A finales de los años ochenta y principio de los noventa, las redes locales de datos (LAN[1] por sus siglas en inglés) eran utilizadas para el correo electrónico y compartir recursos de cómputo como el almacenamiento secundario e impresoras.

En los inicios de la Internet, la mayoría de las aplicaciones como el correo electrónico, la transferencia de archivos (FTP[2]) y el acceso remoto (TELNET) [Braun, H. and Claffy, K. 1994], no exigían muchos recursos por parte de la red. El ancho de banda de las LAN se limitaba a los 10 Mbps mientras que el ancho de banda de los enlaces que conectaban a la Internet, no pasaba de los 64kbps.

En la actualidad, este escenario ha cambiado notablemente. Las LAN han aumentado en número de usuarios y las velocidades de transmisión han llegado a gigabits por segundo (Gbps). La fibra óptica, los sistemas de radiofrecuencia para las telecomunicaciones inalámbricas y el desarrollo en el área de la microelectrónica han incrementado la capacidad de transmisión y velocidad de los conmutadores. Las redes de cobertura amplia (WAN[3] por sus siglas en inglés) también se han visto beneficiados con estos desarrollos, y hoy en día, la mayoría de los enlaces que son utilizados en las dorsales de las WAN son de fibra óptica. La capacidad de estos enlaces, se cuantifica en cientos de megabits por segundo (Mbps) e incluso en Gbps.

Las aplicaciones también han sufrido un cambio. Por ejemplo, Internet se ha convertido en un instrumento ideal para las empresas, el gobierno y la educación. La industria del entretenimiento está realizando esfuerzos para utilizar la Internet en la

---

[1] Local Area Network
[2] File Transfer Protocol
[3] Wide Area Networks

distribución masiva de obras musicales e incluso, de obras cinematográficas (Napster[4] es un ejemplo de estos esfuerzos).

En las LAN, se ha incrementado el uso de aplicaciones de videoconferencia, herramientas colaborativas, transferencia de imágenes de alta definición y la utilización de las redes de datos para la transmisión de voz (por ejemplo VoIP).

Debido a estas circunstancias, la calidad de servicio (QoS[5] por sus siglas en inglés), ha cobrado especial atención en los últimos años en los servicios de red. La incorporación de estas nuevas aplicaciones multimedios y comercio electrónico a las redes basadas en IP, ha sobrepasado la capacidad de los esquemas tradicionales de manejo del tráfico en los enrutadores de las redes y ha sido necesario implementar mecanismos que hagan frente a estos problemas.

La manera tradicional de manejar el tráfico, utilizando el esquema de servicio de *mejor esfuerzo*, no permite que estas aplicaciones puedan transportarse dentro de la red IP manteniendo un nivel de calidad aceptable. Este esquema, trata a todos los paquetes de la misma manera, y no hace ninguna diferenciación en cuanto al tipo de paquete de que se trate. El objetivo de este esquema, es entregar los paquetes a su destino, sin importar el tiempo que los paquetes tarden en llegar al nodo destino ni el orden en que éstos son entregados. Para lograr este objetivo, este esquema se auxilia del protocolo TCP[6] [Postel, J., 1981] el cual es un protocolo de la capa de transporte y se orienta a la conexión.

---

[4] http://www.napster.com
[5] Quality of Service
[6] Transport Control Protocol

Para soportar las nuevas aplicaciones, es necesario establecer diferentes niveles de servicio, los cuales garanticen al usuario, un desempeño adecuado de la aplicación. Los proveedores de servicio de red se pueden auxiliar de los acuerdos de nivel de servicio (SLA[7] por sus siglas en inglés), los cuales especifican límites en los parámetros de desempeño de la red. Los SLA se apoyan en las arquitecturas de QoS para mantener los niveles de desempeño dentro del rango permitido.

El objetivo de QoS es distribuir adecuadamente el ancho de banda disponible en los enlaces digitales. Por ejemplo, algunas aplicaciones como la videoconferencia, necesitan tener una tasa de arribo de paquetes con muy poca variación (jitter pequeño). Para garantizar ésto, las arquitecturas de QoS se basan en los mecanismos de *calendarización de paquetes* para dar un trato preferencial a estos paquetes.

I.2 Presentación

El desarrollo de la Internet y su utilización en nuestra vida cotidiana, ha impuesto nuevos retos para los proveedores de servicios de red. Voz sobre IP (VoIP), videoconferencias, transacciones comerciales y bancarias, música, imágenes de alta resolución, son ejemplos de las nuevas aplicaciones que ahora comparten los enlaces junto con el tráfico tradicional de correo electrónico, transferencia de archivos y acceso a páginas web (ver figura 1) .

Estas nuevas aplicaciones tienen diferentes necesidades de ancho de banda, retardo, jitter y tasa de descarte de paquetes. La manera tradicional de transportar los paquetes es sin hacer ninguna diferenciación de los paquetes (servicio de mejor esfuerzo),

---

[7] Service Level Agreements

con lo que no es posible garantizarle a las aplicaciones multimedia valores máximos de retardo y jitter, así como el ancho de banda que éstas requieran.

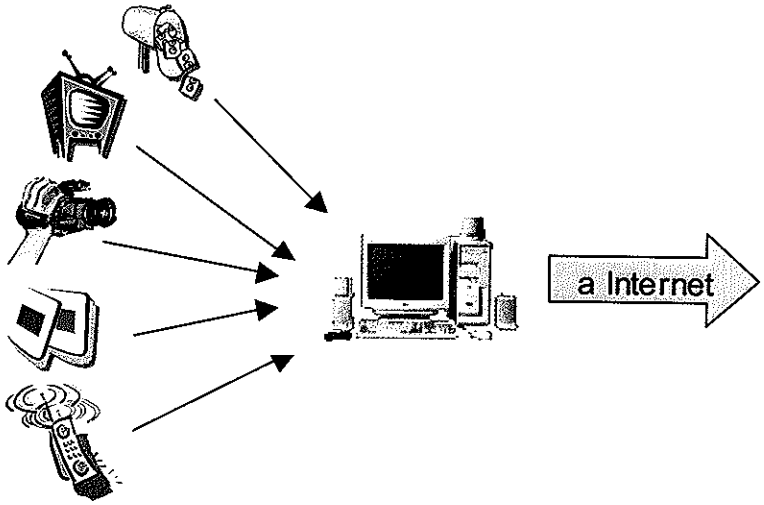

Figura 1. Convergencia de tráfico multimedios en las redes IP

Como el ancho de banda es un recurso limitado, en muchas ocasiones ocurren situaciones en que la cantidad de información que se necesita transmitir es mayor que la capacidad del canal. Esta situación genera que los nodos de conmutación o enrutadores se congestionen. La condición de congestión en un nodo se presenta cuando el nodo procesa paquetes a una tasa mayor que el ancho de banda del enlace de salida (ver figura 2a) y cuando la capacidad del nodo para procesar paquetes sea menor a la tasa de llegada de paquetes (figura 2b) (siempre y cuando la tasa de llegada de paquetes sea menor o igual al ancho de banda del enlace de salida).

Cuando se presenta congestión en un nodo, el enrutador puede elegir entre dos opciones: descartar todos los paquetes que lleguen al nodo, o almacenarlos en un espacio de memoria llamado "cola" hasta que el canal de salida se encuentre disponible para

transmitir paquetes. Si la congestión es severa, las colas se pueden llenar, y por ende, todos los paquetes que lleguen posteriormente tendrán que ser descartados.

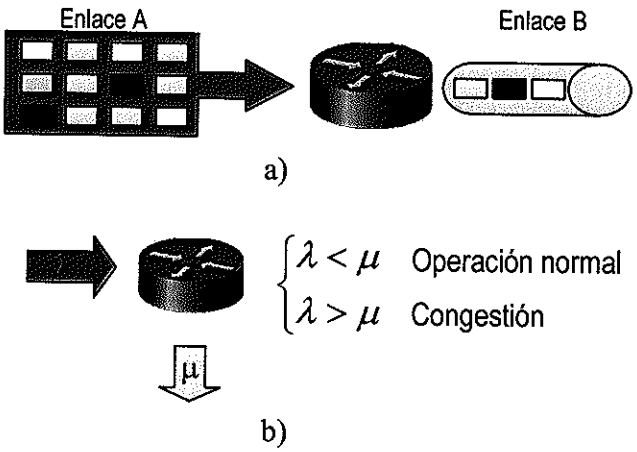

Figura 2. Condiciones que pueden causar congestión

Aunque almacenar temporalmente los paquetes es una mejor opción que descartarlos, algunas aplicaciones de tiempo real como la videoconferencia y la voz sobre IP, pueden verse perjudicadas al incrementarse el jitter debido al tiempo que tienen que pasar los paquetes en las colas.

Para el usuario final, los efectos de la congestión son muy notorios. En aplicaciones de tiempo real, como la videoconferencia y la VoIP, los efectos se observan en la calidad de la imagen y del sonido respectivamente. En el caso del video, los efectos van desde pequeños mosaicos de pixeles en la pantalla, reducción de la velocidad del video[8], hasta la pérdida total de la conexión (ver figura 3). Para aplicaciones tales como la transferencia de archivos o la navegación en páginas web, el usuario experimenta retardos considerables, los cuales pueden llegar a la frustración y hasta la pérdida de la conexión.

[8] En el sistema de televisión NTSC, el número de cuadros por segundo es de 30.

El reto para los administradores de red es distribuir de una forma más "justa" el ancho de banda de los canales de transmisión permitiendo que las aplicaciones que requieren mayores servicios por parte de la red, lo obtengan sin detrimento de aquéllas que no requieren tantos recursos.

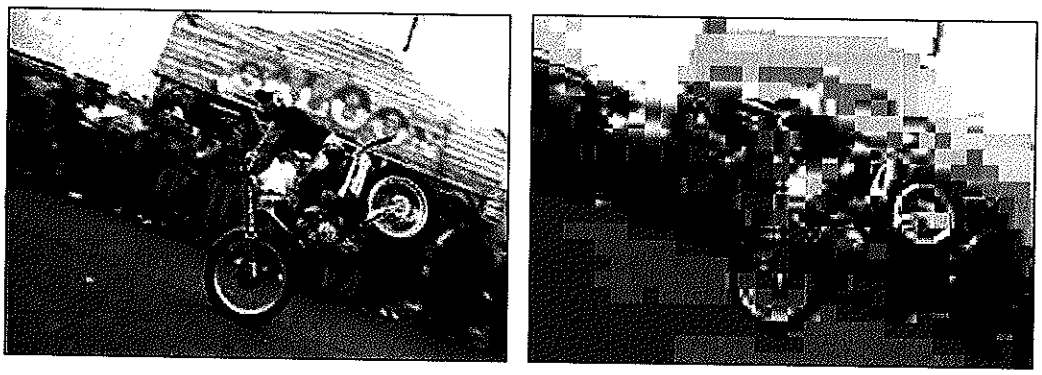

Figura 3. Imagen de video transmitida sin congestión (izquierda) y con congestión severa en la red (derecha)

I.3 Objetivo

Simular los esquemas de control de tráfico en una red de cobertura amplia con protocolo IP que lleven a definir de acuerdo a su desempeño, el mejor mecanismo de control de tráfico y correlacionarlo con los acuerdos de nivel de servicio que se pueden ofrecer.

Para lograr este objetivo se establecieron las siguientes metas:

- Establecer lineamientos para clasificar los diferentes niveles de servicio
- Selección del método de control de tráfico de acuerdo al modelo de línea de espera (FIFO, PQ, WFQ y CBQ).

- Realización de los modelos de control de tráfico utilizando el simulador de redes *Network Simulator* v2.1b8a

I.4 Infraestructura

La infraestructura utilizada para llevar a cabo el proyecto fue:

- Una computadora Pentium III @ 450Mhz, 256Mb en RAM y 43 GB en disco duro
- Conexión a Internet
- Biblioteca del CICESE
- Biblioteca de la Universidad de California en San Diego

Programas utilizados:

- Sistema operativo Windows 2000
- Sistema Operativo Linux Mandrake versión 7.1
- Network Simulator versión 2.8
- Programa para graficar Microcal Origin 6.1 (versión de evaluación)
- Microsoft Word 2000

I.5 Organización del trabajo

La organización de este trabajo fue la siguiente:

En el Capítulo II se hace una introducción a los niveles de servicio. Se estudian en forma breve los modelos de servicio más utilizados para ofrecer calidad de servicio en una red de datos, los parámetros de desempeño que más afectan a las aplicaciones de tiempo real así como los mecanismos de control de tráfico los cuales se encargan de determinar el nivel de servicio que obtengan los flujos de la red. En el Capítulo III se estudia el modelo que se implementó para este trabajo así como los diferentes tipos de fuentes de tráfico y mecanismos de control de tráfico que se usaron. En el Capítulo IV se hace una descripción de la implementación del modelo en el simulador "*Network Simulator*". En el Capítulo V se hace un análisis de los resultados obtenidos, en específico, sobre la tasa de transmisión, retardo, jitter y tasa de pérdida de paquetes para cada uno de los mecanismos de control de tráfico estudiados (FIFO, PQ, WFQ y CBQ). En el Capítulo VI se hace una introducción a los Acuerdos de Nivel de Servicio y su relación con los mecanismos de control de tráfico. El Capítulo VII incluye las conclusiones sobre los mecanismos de control de tráfico y recomendaciones sobre nuevas implementaciones para futuros trabajos. Finalmente, se anexa un apéndice el cal contiene algunas rutinas de generación y monitoreo de tráfico para el simulador "*Network Simulator*".

II Niveles de servicio

II.1 Introducción

Para poder tratar lo relacionado con los niveles de servicio, es necesario definir primero sobre lo que es servicio, y en específico, lo que es *calidad de servicio*. El concepto de calidad en el contexto de las redes, está muy relacionado con la percepción que el usuario tiene de la red. No existe una forma de medir si ésta es buena, regular o mala, todo depende de las necesidades del usuario. Para algunos usuarios, un mensaje de correo electrónico puede tardar un poco más de una hora para ser entregado y no le representa ningún inconveniente, sin embargo, para un departamento que da asistencia técnica a clientes de una empresa, probablemente requerirá que el correo llegue en menos de cinco minutos. La aplicación es la misma, sin embargo, las necesidades de ambos usuarios son diferentes y cada uno de ellos definirá su estándar de calidad.

II.2 Calidad de servicio

La calidad de servicio consiste en ofrecerle al usuario de acuerdo a la aplicación, los recursos de red necesarios y suficientes para cumplir con los requerimientos establecidos por el cliente. Todas las aplicaciones requieren de un ancho de banda determinado, y si el ancho de banda no es suficiente, la red tendrá que retardar o descartar paquetes disminuyendo el desempeño de las aplicaciones.

Algunos de los parámetros que afectan a las aplicaciones son el retardo, el jitter y el ancho de banda, los cuales se describirán más adelante.

Cuando se aplica la QoS en una red, se administra el ancho de banda disponible en los enlaces entre todos los flujos que se encuentran en la red. Esta administración se realiza en función de los recursos que los flujos requieren y de las políticas de servicio que estén definidas en la red.

Un término muy utilizado cuando se habla de QoS son los *flujos*. En este trabajo, un flujo se definirá como una secuencia de paquetes que tienen la mismas direcciones (fuente y destino) IP y puertos TCP o UDP.

En la actualidad, el ancho de banda es un recurso limitado y costoso. Tecnologías como la fibra óptica y la multiplexión por longitud de onda (WDM[9] por sus siglas en inglés) prometen aumentar considerablemente la capacidad de transmisión de los enlaces [Karasan, E. y Ayanoglu, E., 1998] y al mismo tiempo abaratar los costos, sin embargo, mientras esto llega, es necesario implementar mecanismos que puedan compartir el ancho de banda actual de la manera más eficaz posible.

Ofrecer QoS en una red permite utilizar de una manera más eficaz de ancho de banda de los enlaces. A través de los mecanismos de QoS se pueden ofrecer diferentes niveles de servicio dentro de la red, donde algunos de estos niveles beneficiaran a aquellos flujos que sean sensitivos a parámetros temporales como la voz o la videoconferencia en tiempo real. Algunos tipos de tráfico se verán beneficiados, mientras que otros, serán perjudicados en su desempeño. Un objetivo importante de la QoS es que los tráficos de baja prioridad o que no requieren recursos garantizados, obtengan solo una porción del ancho de banda total.

---

[9] Wave Division Multiplexing

Es importante mencionar que para que en una red se obtengan beneficios al aplicar mecanismos de QoS, es necesario que algunos flujos se traten con baja prioridad y otros con prioridad alta, ya que si a todos los flujos se les brindara una prioridad alta, el beneficio resultante sería nulo.

Los administradores que se propongan ofrecer QoS en sus redes, deben de ser capaces de diferenciar entre los tipos de tráfico (videoconferencia, voz sobre IP, web, email, telnet, ftp, etc) así como al usuario al que pertenecen. Al identificar los paquetes, es posible tratarlos de manera diferente de acuerdo a las necesidades del cliente y a las políticas de calidad implementadas en la red. De esta manera, es posible ofrecer al cliente un nivel de calidad de acuerdo a sus expectativas.

II.2.1 Parámetros de desempeño

Entre los diferentes parámetros de desempeño que permiten medir el nivel de servicio que reciben las aplicaciones, existen cuatro que afectan directamente el nivel de calidad de las aplicaciones (principalmente a las aplicaciones en tiempo real):

- Caudal eficaz de la aplicación
- Retardo de principio a fin de los paquetes
- Variación del retardo (*jitter*)
- Tasa de pérdida de paquetes (PLR[10] por sus siglas en inglés)

---

[10] Packet Loss Rate

II.2.1.1 Caudal eficaz

Es un parámetro que mide la cantidad de información que se transmite en una unidad de tiempo y se mide en bits por segundo (bps). Algunos flujos requieren un mayor caudal eficaz que otros. La videoconferencia por ejemplo, requiere de un caudal eficaz mayor que la utilizada para enviar correos electrónicos. A mayor cantidad de información que se necesite enviar por unidad de tiempo, mayor será el caudal eficaz necesario.

Si la capacidad de los enlaces es menor al caudal eficaz requerido por las aplicaciones, implementar mecanismos de QoS no resolverá el problema. Los enlaces deben tener la capacidad suficiente para transportar el tráfico comprometido.

II.2.1.2 Retardo de principio a fin

Es el tiempo que transcurre desde que se genera el paquete en el nodo emisor, hasta que son recuperados por el nodo final. Existen dos componentes esenciales en la determinación de este retardo: a) el retardo constante el cual lo podemos determinar fácilmente y b) el retardo variable, el cual es difícil de determinar debido a que varía de acuerdo a las condiciones de operación de la red.

El retardo constante se debe a los procesamientos diferentes que sufre el paquete desde que se genera hasta que se procesa por el nodo receptor. La figura 4 muestra los diferentes componentes que afectan el retardo de principio a fin. Los retardos en los nodos transmisor y receptor incluyen los causados por la digitalización, codificación y compresión, así como el procesamiento de las capas 4, 3, 2 y 1 del modelo OSI [Day, J. y Zimmerman, H. 1983] para ensamblar y transmitir los paquetes por el enlace físico. El retardo por propagación, el cual está en función de la velocidad de la luz y el canal de

comunicación que se utilice; el retardo por el procesamiento de las capas 4, 3, 2 y 1 en los nodos intermedios y el retardo de serialización, el cual considera el tiempo que el enrutador se tarda en transmitir el paquete hacia el enlace.

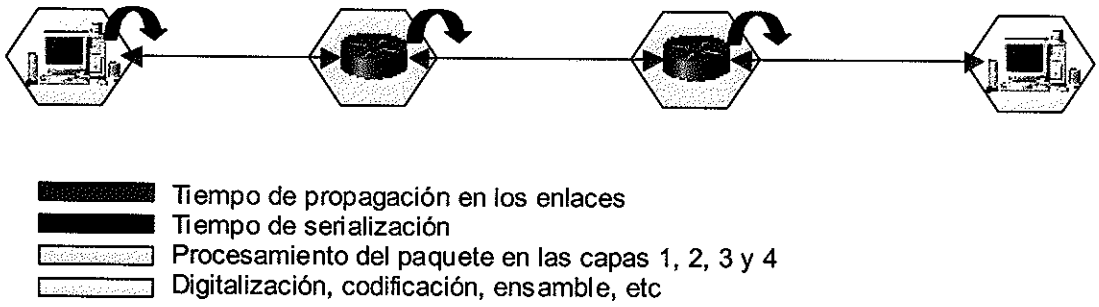

Figura 4. Componentes del retardo constante

El segundo tipo de retardo que se induce en la red es variable y es ocasionado por tres factores principalmente: el tamaño del paquete, el mecanismo de calendarización de paquetes y los reforzadores (buffers) para eliminar el jitter (ver figura 5). Este retardo aumenta cuando existe congestión y por tal razón, es muy difícil predecirlo con exactitud. Los diferentes mecanismos de calendarización tratan de disminuir este retardo para aplicaciones que son sensibles al retardo.

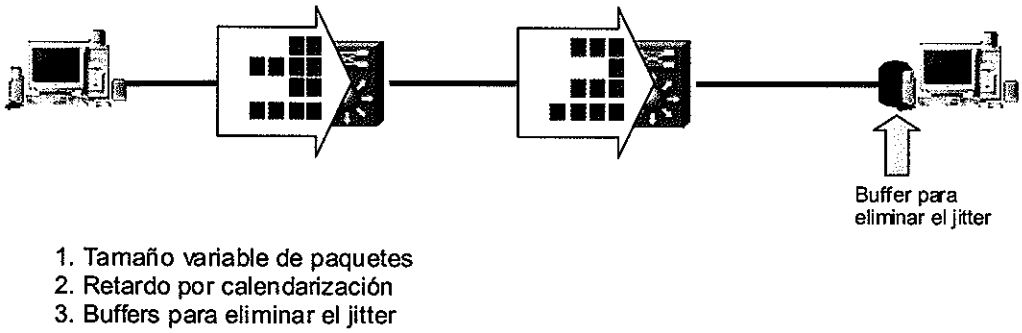

Figura 5. Componentes del retardo variable

II.2.1.3 Variación del retardo (jitter)

Cuando existen condiciones de congestión en un nodo, los paquetes tienen que ser almacenados temporalmente en una cola hasta que el enlace esté disponible. Este escenario se puede presentar en alguno o todos los nodos de la red, por lo que el tiempo que los paquetes deben permanecer en las colas es variable.

Una de las aplicaciones que son muy sensibles al jitter son las de reproducción en tiempo real (*playback*). Las aplicaciones en tiempo real son aquellas que presentan al usuario la información casi en el mismo instante en que los paquetes se reciben. Ejemplo de estas aplicaciones son la telefonía sobre IP o las aplicaciones de videoconferencias.

Como la operación de las aplicaciones *playback* requiere de una tasa de llegada constante de paquetes, los paquetes que van llegando al receptor tienen que almacenarse en un buffer para amortiguar las diferencias en los tiempos de interarribo de los paquetes. La capacidad de almacenamiento de este buffer está en función del retardo máximo que exista entre la llegada de dos paquetes consecutivos y la variación del retardo máximo que la aplicación pueda tolerar.

La variación máxima del retardo está determinada por factores como la longitud de las colas, el tamaño de los paquetes y el número de flujos.

II.2.1.4 Tasa de pérdida de paquetes (PLR)

Este parámetro mide la cantidad de paquetes que son descartados. El descarte de paquetes se debe a las condiciones de congestión en los nodos, lo que causa que las colas se llenen y los paquetes que lleguen posteriormente tengan que ser descartados. Las

aplicaciones de tiempo real son muy sensibles al descarte de paquetes, ya que utilizan UDP[11] como protocolo de transporte, y éste, no cuenta con un mecanismo para la retransmisión de paquetes en caso de que los paquetes sean descartados dentro de la red [Postel J. 1980]. En el caso del audio y video es posible tolerar un número máximo de paquetes perdidos (el cual es muy pequeño) gracias a los algoritmos de compresión y codificación. Sin embargo, cuando el número de paquetes perdidos sobrepasa este umbral, la calidad de la señal se degrada considerablemente.

Aquellas aplicaciones que están basadas en TCP, no se ven muy afectadas por la pérdida de paquetes, ya que el funcionamiento propio del protocolo permite que los paquetes descartados vuelvan a ser retransmitidos [Allman M. *et al.* 1999].

II.2.2 Modelos de servicios

Desde que se desarrolló la Internet, el objetivo primordial era la confiabilidad en la entrega de paquetes al nodo destino. Para las aplicaciones militares esto era muy importante ya que requerían de una red que siguiera funcionando aún cuando algunos de los nodos fueran destruidos en algún ataque. De igual manera, era importante que en los casos en que se presentará congestión en la red, la información no se perdiera. El primer modelo de entrega de paquetes se denominó *servicio de mejor esfuerzo*. Posteriormente, con la integración de nuevas aplicaciones multimedios a la red, se desarrollaron nuevos modelos de servicio para permitir que estas nuevas aplicaciones utilicen las redes de IP. Los dos modelos más conocidos en la actualidad son *servicios integrados* y *servicios diferenciados*. A continuación se describirán brevemente estos modelos de servicio.

---

[11] User Datagram Protocol

II.2.2.1 Mejor esfuerzo

La forma de operar de la mayoría de las redes actuales basadas en IP, proveen el servicio de *mejor esfuerzo*, es decir, los paquetes se procesan lo más rápido posible si existen los recursos para transmitirlos, y se descartan si no se pueden transmitir [Postel, J. 1981]. En otras palabras, la red hace su mejor esfuerzo para entregar los paquetes, sin tomar en consideración el tiempo que los paquetes tarden en llegar y el ancho de banda utilizado por el flujo. Los enrutadores o conmutadores pueden descartar paquetes en caso de que exista una congestión severa en la red sin notificar al nodo emisor del paquete. En los casos en que la aplicación utilice TCP, se provee de un mecanismo de retransmisión de paquetes, mientras que las aplicaciones basadas en UDP no cuentan con un mecanismo similar.

Esta manera de operar, no garantiza un ancho de banda mínimo para el flujo, y por consiguiente, tampoco garantiza límites en los parámetros de retardo, jitter y pérdida de paquetes.

Utilizando este esquema de servicio, sólo es posible ofrecer un nivel de servicio. Este nivel de servicio garantiza que la red hará todo lo posible por entregar paquetes, sin garantizar el caudal eficaz que la aplicación obtenga. Cuando la aplicación está basada en TCP, los paquetes que son descartados son retransmitidos, lo que causa una disminución del caudal eficaz y un aumento del tiempo en la entrega de paquetes. Al contrario de las aplicaciones TCP, las aplicaciones basadas en UDP, no retransmiten los paquetes, por lo que una condición de congestión en la red, afectará considerablemente la calidad en la reproducción de la aplicación. Mientras la red no se encuentre congestionada, los flujos obtendrán el ancho de banda que necesiten, el retardo será mínimo y la tasa de descarte de

paquetes será pequeña si no es que nula. Sin embargo, cuando se presenten condiciones de congestión en la red, no se podrán garantizar recursos de la red para ningún flujo.

Este esquema de servicio funciona muy bien cuando el tipo de tráfico que se transporta en la red, no es sensitivo a los retardos y no requiere de un caudal eficaz mínimo. El esquema de mejor esfuerzo se basa en la capacidad de manejo de la congestión del protocolo TCP [Allman M. *et al.*, 1999], el cual disminuye la tasa de transmisión de paquetes cuando detecta que existe una congestión en la red. De igual forma, cuando existe descarte de paquetes en un nodo debido a la congestión, este protocolo retransmite los paquetes perdidos, evitando que se pierda información del flujo.

Ejemplos de aplicaciones que se pueden transportar utilizando este esquema son el correo electrónico, transferencia de archivos (FTP), acceso remoto (TELNET) y el tráfico de http los cuales no son sensitivos al tiempo o no requieren de un ancho de banda mínimo.

II.2.2.2 Servicios integrados (IntServ)

Este modelo de servicio permite ofrecer la QoS a flujos de audio, video y datos en tiempo real. Se fundamenta principalmente en la reservación de recursos. Funciona de manera análoga al sistema de conmutación de circuitos de la red telefónica convencional (ver figura 6); antes de iniciar una transmisión, es necesario reservar un circuito lógico a través de la red. Cada uno de los nodos debe mantener variables de estado para cada uno de los flujos que tengan recursos reservados [Braden, R. *et al.*, 1994].

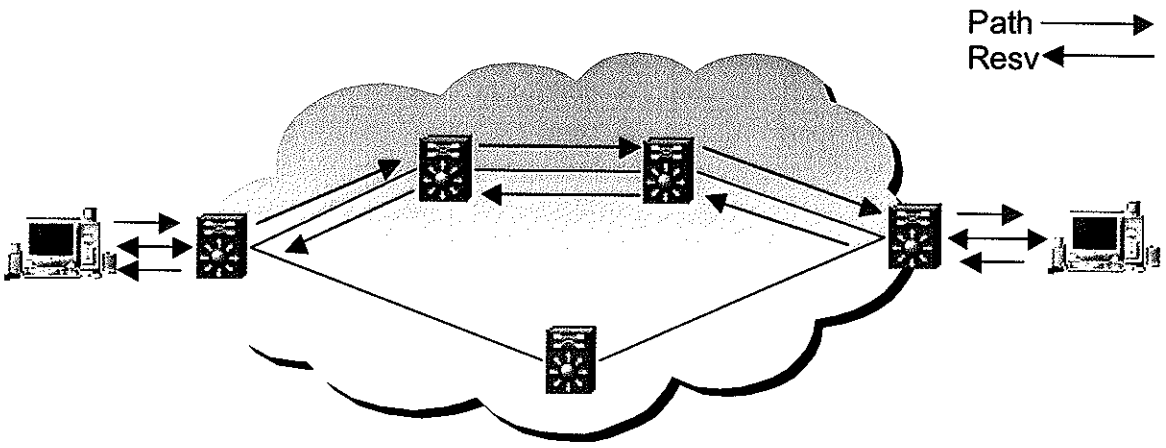

Figura 6. Modelo de Servicios Integrados

El protocolo para señalización que utiliza IntServ, es el RSVP[12] el cual fue desarrollado como un protocolo de señalización con el fin de que las aplicaciones reserven recursos de la red [Braden R. *et al.*, 1997]. La forma de operación de este protocolo consiste en que cuando una terminal desea establecer una comunicación con otra terminal reservando recursos, ésta envía una petición de reservación (paquete PATH) en el cual se especifica el tipo de tráfico y los requerimientos. Cuando el receptor recibe un paquete de petición PATH, éste responde con un mensaje RESV para solicitar la reservación en cada uno de los nodos intermedios. Cuando este paquete llega a un nodo intermedio, éste puede aceptar o rechazar la solicitud. La decisión de si acepta o rechaza la solicitud, depende de si existen los recursos disponibles en el nodo para satisfacer la demanda del flujo. Si la solicitud es aceptada, el paquete RESV se pasa al siguiente nodo en la trayectoria. En caso de que el nodo rechace la petición de reservación, se enviará un mensaje de error a la

[12] Resource reSerVation Protocol

terminal que solicitó la reservación y el proceso de señalización termina. Una vez que la reservación de recursos es aceptada por todos los nodos intermedios, una parte del ancho de banda de los enlaces, así como espacio en los buffers de los enrutadores, se reservan para el flujo entre ambas terminales.

Este modelo de servicio puede brindar tres clases de servicio:

- Mejor esfuerzo
- Servicios garantizados
- Carga controlada

Mejor esfuerzo. La clase de servicio de mejor esfuerzo fue descrita anteriormente; en ella caen todos aquellos flujos que no requieren realizar una reservación de recursos. Aunque la función primordial de IntServ es garantizar recursos a los flujos que lo soliciten, los flujos que utilizan la clase de *mejor esfuerzo* deben obtener un servicio adecuado por parte de la red. A esta clase también se le debe de garantizar al menos una porción del ancho de banda del enlace.

Servicio garantizado. Este servicio provee el mejor nivel de calidad de servicio en IntServ. Provee a las aplicaciones de tiempos de retardos máximos y garantiza un ancho de banda mínimo a las aplicaciones [Shenker, S. y Wroclawski, J. 1997].

Carga controlada. Este servicio permite ofrecer un nivel de QoS semejante al que un flujo recibiría en un nodo no congestionado. Este servicio no garantiza un ancho de banda mínimo ni un tiempo de retardo máximo, sin embargo, el retardo experimentado

por un paquete en un nodo congestionado no podrá ser mayor que el retardo que experimente en condiciones normales de operación. La admisión de paquetes al nodo es controlada (de ahí el nombre de la clase) por el mecanismo de "Token Bucket" [McDysan, D. 2000]; [Shenker, S. *et al.*, 1997]. Cualitativamente, esta clase no es tan buena como el servicio garantizado, pero es mucho mejor que la clase de mejor esfuerzo [Wroclawski, 1997].

Una desventaja de IntServ es el hecho de cada nodo debe mantener información de cada uno de los flujos que lo atraviesan. En una red con cientos de flujos, sería muy complicado el mantener la información correspondiente a cada uno de los flujos. Es por esta razón que este servicio no es muy utilizado en redes de cobertura amplia.

Los RFC 1633, 2211 y 2212 describen a mayor detalle el modelo de servicios integrados así como las clases de servicio que ofrecen.

II.2.2.3 Servicios diferenciados (DiffServ)

Este modelo de servicio busca simplificar el proceso de ofrecer QoS dentro de una red. A diferencia del modelo de IntServ en donde el proceso de reservación de recursos requiere que los nodos internos participen activamente y mantengan información del estado de cada uno de los flujos que tienen asignados recursos, DiffServ delimita el proceso de clasificación a los puntos de acceso a la red (PAR) [Blake, S. *et al.*, 1998]. De esta manera, los nodos internos únicamente realizan el proceso de calendarización de paquetes de acuerdo al nivel de QoS correspondiente.

Como medio de señalización, este modelo utiliza el campo de tipo de servicio (TOS[13] por sus siglas en inglés) (ver figura 7) del encabezado del paquete IP para indicar el nivel de servicio que se le debe ofrecer al paquete. Este modelo rescata el uso original del campo TOS [Postel, J. 1981], el cual permitía ofrecer un nivel de QoS al paquete cuando éste pasaba por un nodo. Sin embargo, la mayoría de las redes no utilizaron este campo, y su significado se perdió.

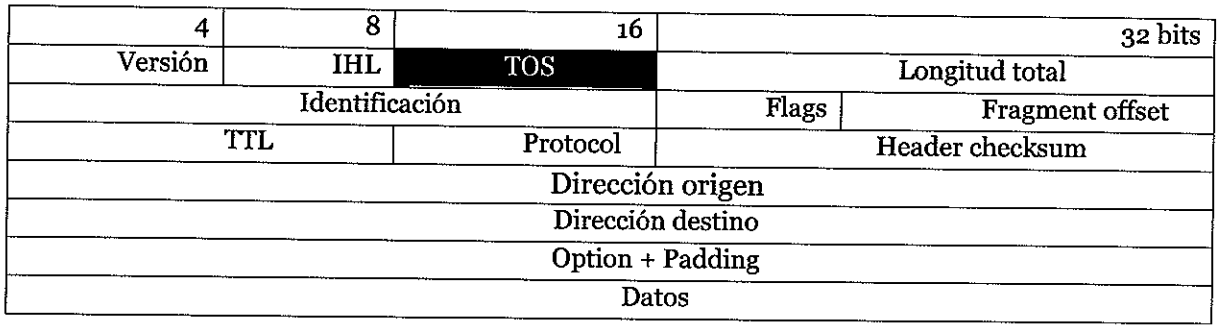

Figura 7. Encabezado del paquete IP y la posición del campo TOS

Al código que se utiliza para definir la clase de servicio que el paquete debe recibir se llama código de servicios diferenciados (DSCP por sus siglas en inglés). Este código utiliza los primeros seis bits del campo TOS para mantener compatibilidad con aquellas redes que si utilizan las funciones de precedencia del campo TOS [Black, D., *et al.*, 2001].

Cuando un paquete entra a una red que implementa DiffServ (ver figura 8), los nodos de ingreso a la red se encargan de clasificar y marcar el paquete con el DSCP correspondiente al nivel de servicio que se le debe aplicar al paquete [Nichols, K. y Carpenter, B., 2001]. Los nodos internos de la red, revisan el contenido del campo DSCP y procesan al paquete de acuerdo al nivel de QoS que le corresponda.

---

[13] Type Of Service

Cada código DSCP representa una clase de servicio, lo que permite ofrecer diferentes niveles de servicio. [Kilkki, K. 1999]

Debido a que la señalización en este esquema es intrínseca, el ancho de banda disponible para los flujos es mayor en comparación con IntServ.

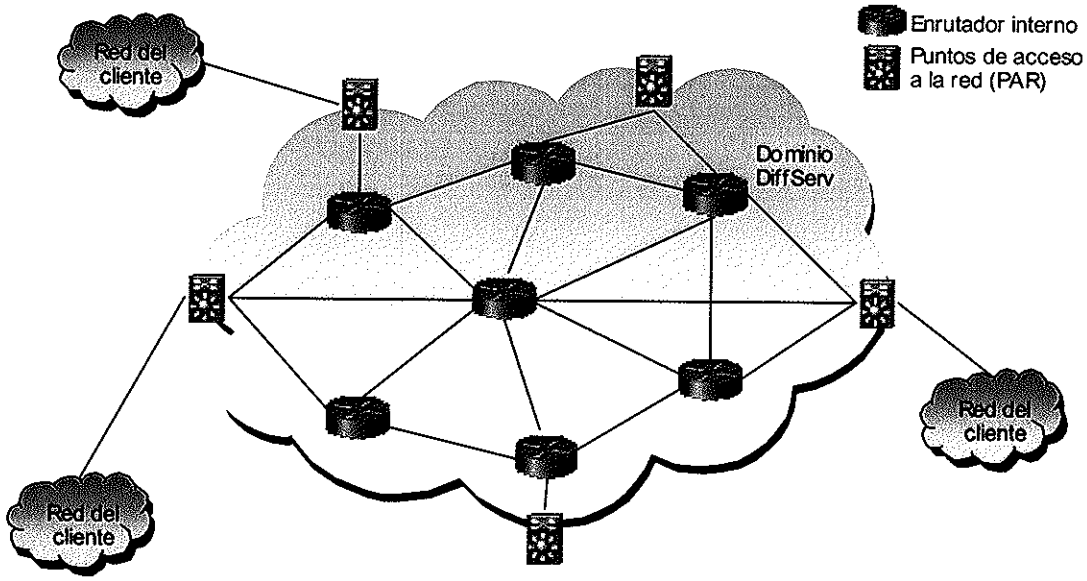

Figura 8. Modelo de Servicios Diferenciados

II.3 Clasificación de las aplicaciones

El retardo y el jitter no afectan por igual a todas las aplicaciones. Algunas de ellas se verán más afectadas que otras, incluso, afectarán de manera directa el nivel de calidad que es observado o escuchado por el usuario.

El tipo de aplicaciones que existen en la actualidad se pueden clasificar en dos grupos principalmente: aplicaciones de tiempo real y aplicaciones elásticas. Las

aplicaciones de tiempo real son aquéllas en que los paquetes deben ser entregados o reproducidos casi en el momento en que éstos llegan.

Dentro de las aplicaciones de tiempo real, existen diferentes requerimientos de nivel de servicio. Aplicaciones interactivas como por ejemplo, la telefonía, son sensitivas tanto al retardo como al jitter; mientras que aplicaciones como el video en demanda, son sensibles al jitter únicamente.

Las aplicaciones elásticas son aquellas en que el retardo o el jitter no afectan en forma directa la calidad percibida por el usuario, pero que sin embargo, un retardo alto o un jitter grande, podrían afectar la calidad de la aplicación. Ejemplos de estas aplicaciones son TELNET, FTP, WEB y por lo general todas aquellas aplicaciones que estén basadas en TCP.

Uno de los objetivos de este trabajo, es estudiar los efectos que los calendarizadores tienen sobre las aplicaciones de tiempo real.

II.3.1 Requerimientos de calidad de las aplicaciones

Todas las aplicaciones requieren un nivel de servicio específico de la red. Los niveles de servicio están definidos por los valores máximos de los paquetes descartados, por el retardo máximo y por el jitter máximo que puede soportar una aplicación.

II.3.1.1 Tasa de pérdida de paquetes

La tasa de pérdida de paquetes (PLR[14] por sus siglas en inglés) es un parámetro que sirve para medir el desempeño de la red en forma cuantitativa. La tasa se obtiene de la relación entre el número de paquetes descartados y el número de paquetes enviados.

Cada aplicación puede soportar hasta un máximo PLR. La tabla I muestra los valores máximos para cada una de las aplicaciones que se van a utilizar en el modelo propuesto.

Tabla I. Tasa pérdida de paquetes recomendada

Servicio	PLR	Referencia
Video	10^{-6}	Felix Cota, H. 1998
Datos	10^{-6}	Felix Cota, H. 1998

II.3.1.2 Retardo y jitter

Dos parámetros que también ayudan a determinar el desempeño de la red es el retardo y el jitter que sufren los paquetes al atravesar la red. El retardo incluye los componentes de retardo constante así como el retardo generado por los mecanismos de calendarización de paquetes. El jitter es la variación máxima en los tiempos de interarribo de los paquetes.

[14] Packet Loss Rate

II.4 Calendarizadores de paquetes

Cuando un paquete llega a un nodo (se utilizará la palabra nodo para referirse a un dispositivo de conmutación como lo puede ser un conmutador o un enrutador), el software controlador del nodo tiene solamente tres opciones para procesar el paquete: en caso de que la interfaz de salida se encuentre disponible, transmitirá el paquete inmediatamente; si la interfaz de salida está ocupada puede descartar el paquete o almacenarlo para ser transmitido posteriormente.

Cuando existe una condición de congestión en un nodo, los nodos que arriban se almacena temporalmente en una *cola* hasta que el enlace de salida se encuentre disponible.

Las colas son espacios de memoria en los enrutadores (ver figura 9) donde se almacenan temporalmente los paquetes que no pueden ser transmitidos por el enrutador debido a una congestión en el nodo. Almacenar los paquetes en una cola es una mejor opción que descartarlos, sobre todo cuando se trata de tráfico interactivo o de tiempo real. Una vez que la congestión desaparece, se empiezan a despachar los paquetes que se encuentren en las colas. La tasa de transmisión con la cual los paquetes salen es menor o igual al ancho de banda del enlace.

Con el modelo de servicio del mejor esfuerzo, se necesita una sola cola por cada uno de los enlaces que tenga conectado el enrutador. Sin embargo, con la introducción de nuevos modelos de servicios, es posible tener dos o más colas por cada enlace. Cuando se tiene más de una cola, es necesario clasificar los paquetes para determinar la cola a la que pertenecen los paquetes. Esta clasificación se realiza basándose en la información de los encabezados del paquete.

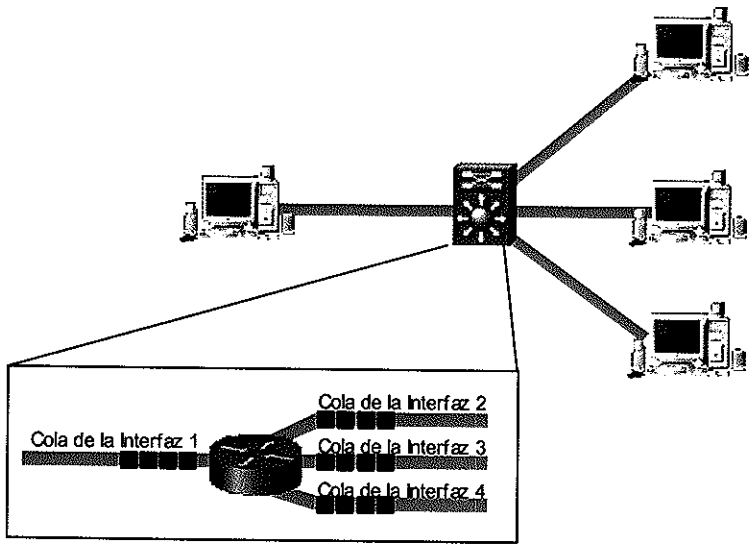

Figura 9. Los enrutadores utilizan espacio de memoria (buffers) para almacenar paquetes en caso de congestión.

Aunque el uso de colas para almacenar paquetes es una mejor opción que descartarlos en caso de congestión, se introducen nuevas variables al funcionamiento de la red, las cuales pueden perjudicar el desempeño de algunas aplicaciones.

En una red ideal, el tiempo que debería de tardar un paquete en ir de un extremo a otro extremo, es la suma del tiempo de propagación de cada enlace por el que circule mas el tiempo de serialización. Sin embargo, debido al empleo de las colas, este tiempo se incrementa por la estancia de los paquetes en las colas. Por otro lado, el uso de las colas permite redistribuir el ancho de banda disponible en los enlaces entre los diferentes flujos presentes, y de esta manera, ofrecer diferentes niveles de servicio.

Es importante mencionar que el uso de las colas, únicamente se presenta en caso de congestión en los enlaces de salida. Cuando los enlaces cuentan con el ancho de banda suficiente para transmitir paquetes, no tiene sentido almacenar los paquetes en las colas.

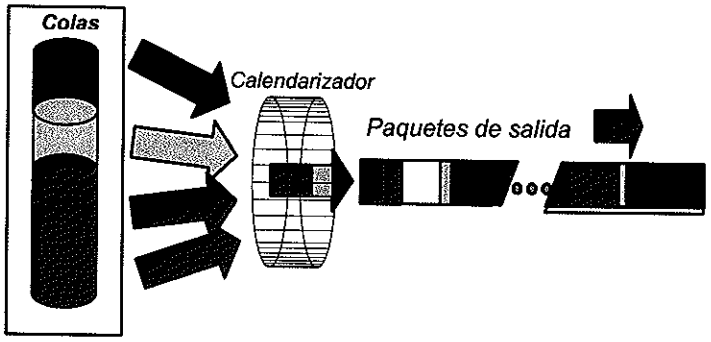

Figura 10. Modelo de un calendarizador de paquetes

Los mecanismos que se encargan de decidir cuál paquete se transmite primero son los *calendarizadores de paquetes.* La figura 10 muestra el modelo de operación de los calendarizadores. Cada una de las colas representa un nivel de servicio diferente. El calendarizador, dependiendo del algoritmo que se tenga implementado, determinará la cola y el número de paquetes de la cuál se deben de extraer paquetes. Por otro lado, el orden en que los paquetes van siendo extraídos de las colas, depende del algoritmo del calendarizador y de los SLA presentes en la red. Algunos ejemplos de calendarizadores de paquetes son FIFO, PQ, WFQ y CBQ.

II.4.1 FIFO (First in, first out)

El primero que entra, es el primero que sale, ha sido el manejador de colas que se ha utilizado desde que se desarrolló la Internet. Su función primordial es almacenar paquetes hasta que el enlace de salida se encuentre disponible. Es el más sencillo de todos los mecanismos, ya que no realiza ningún procesamiento a los paquetes. La manera en que los paquetes se colocan en el enlace, va de acuerdo al orden en que éstos llegaron. La figura 11 muestra el funcionamiento de este mecanismo de control de colas.

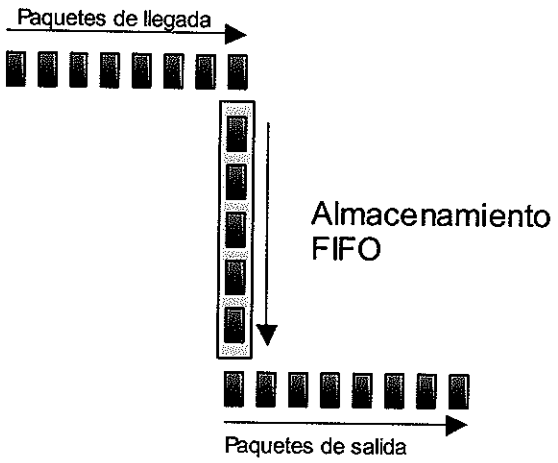

Figura 11. Modelo de operación de una cola FIFO

Este mecanismo es muy eficiente cuando la red no se encuentra congestionada. De hecho, en la actualidad, los enrutadores utilizan este mecanismo cuando no existe congestión en los enlaces de salida.

Si la tasa de llegada es mayor que la tasa de servicio, la capacidad de almacenamiento de la cola se acabará y los paquetes que lleguen subsecuentemente serán descartados (no habrá donde almacenarlos).

Los efectos que este mecanismo puede tener sobre aplicaciones de tiempo real, se reflejan en una disminución de la tasa de transmisión, aumento del retardo y jitter así como un aumento en la tasa de pérdida de paquetes.

II.4.2 Calendarización por prioridad (PQ[15])

Una primera aproximación para resolver los problemas que se generan con los mecanismos FIFO es darle prioridad a los paquetes que requieren ser atendidos de una manera expedita para mantener un jitter o retraso más constante.

El funcionamiento de este mecanismo consiste en atender primero los paquetes de mayor prioridad y cuando no haya paquetes de mayor prioridad para transmitir, los paquetes de baja prioridad serán atendidos.

La clasificación de los paquetes puede ser determinada por la interfaz de llegada del paquete, los puertos origen y destino de TCP o UDP, el protocolo de transporte (TCP, UPD, IPX, etc) y las direcciones de IP de origen y destino.

La figura 12 muestra el modelo de operación de este mecanismo. Se incluye un clasificador de paquetes, cuya función consiste en determinar de acuerdo a la información del encabezado del paquete la prioridad que le corresponde. La prioridad la determina el nivel de servicio que requiera el paquete. El número de colas es configurable; y cada una tendrá asignado un nivel de prioridad diferente.

Este mecanismo presenta una mejora considerable con respecto al mecanismo FIFO, al ofrecer un mejor servicio a los paquetes con mayor prioridad. En consecuencia, los paquetes de baja prioridad pueden sufrir aumentos significativos en el retraso, jitter y tasa de pérdida de paquetes. Aún más, se pueden llegar a presentar situaciones en que los tráficos de alta prioridad monopolicen los enlaces.

---

[15] Priority Queueing

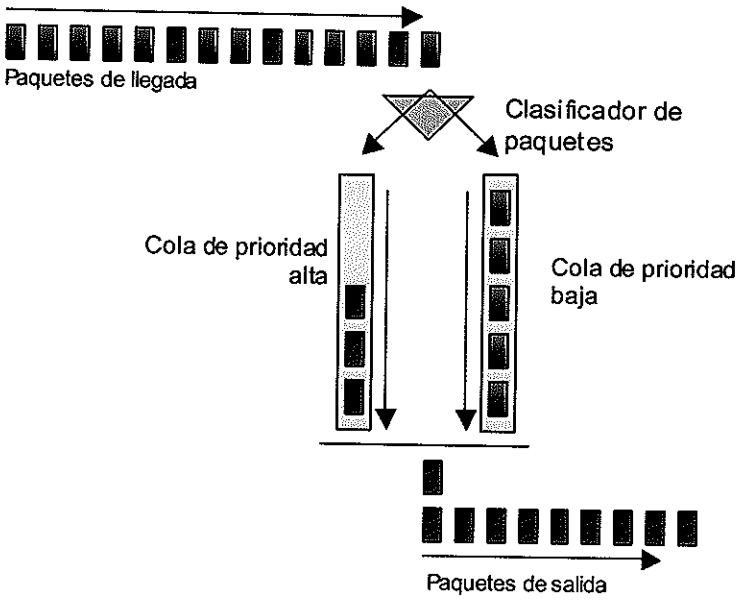

Figura 12. Modelo de operación de la cola PQ

Esto se debe a que el algoritmo únicamente transmite paquetes de baja prioridad cuando la cola de alta prioridad se encuentra vacía. Si la tasa de llegada del tráfico de alta prioridad es mayor o igual al tráfico de baja prioridad, la cola de baja prioridad no será atendida hasta que el emisor de tráfico de alta prioridad deje de transmitir paquetes.

Cuando el tiempo que pasan los paquetes de baja prioridad en las colas es mayor que el tiempo RTT[16] del protocolo TCP, los paquetes son retransmitidos (se asume que estos paquetes han sido descartados), lo que ocasiona una disminución del caudal eficaz y una sobre utilización de los enlaces al reenviar paquetes que aún no se han descartado.

Otro caso extremo que puede llegar a suceder, es cuando los paquetes de baja prioridad son más grandes que los de alta prioridad. Cuando no existen paquetes de alta prioridad, el manejador de la cola comienza a enviar a los paquetes de baja prioridad, y

[16] Round Trip Time

cuando llegan paquetes de alta prioridad de un tamaño pequeño, éstos tendrán que esperar a que se termine de transmitir el paquete de baja prioridad, lo que provocará un incremento en el retardo fin a fin para los paquetes de alta prioridad.

Cuando no existe una congestión en los enlaces de salida, los paquetes se envían utilizando el mecanismo FIFO, sin importar la prioridad de los paquetes.

Algunos enrutadores pueden manejar más de dos colas para las prioridades, reservando una cola para cada una de las prioridades.

II.4.3 Calendarización por un factor de ponderación (WFQ[17])

Este mecanismo evita que tráficos de alta prioridad monopolicen los enlaces. El mecanismo consiste en darle un servicio justo a cada una de las colas de acuerdo a un factor de ponderación. El factor se especifica en proporción al nivel de prioridad del flujo.

Este mecanismo es una modificación del mecanismo FQ[18] [Demers, A. *et al.*, 1989] el cual distribuye el ancho de banda disponible en el enlace de una manera equitativa entre los diferentes flujos. FQ funciona muy bien cuando los paquetes son del mismo tamaño, pero debido a la naturaleza de la Internet, la longitud de los paquetes puede variar desde 40 hasta 1500 bytes. Esta variación impide que FQ funcione adecuadamente. WFQ es una variación de FQ con la adición de factores de ponderación asociados a cada una de las colas, los que determina el número de paquetes que pueden ser extraídos de una cola en cada turno (ver figura 13). Ponderando cada cola, es posible contrarrestar los efectos de las variaciones en el tamaño de los paquetes.

---

[17] Weighted Fair Queuing
[18] Fair Queueing

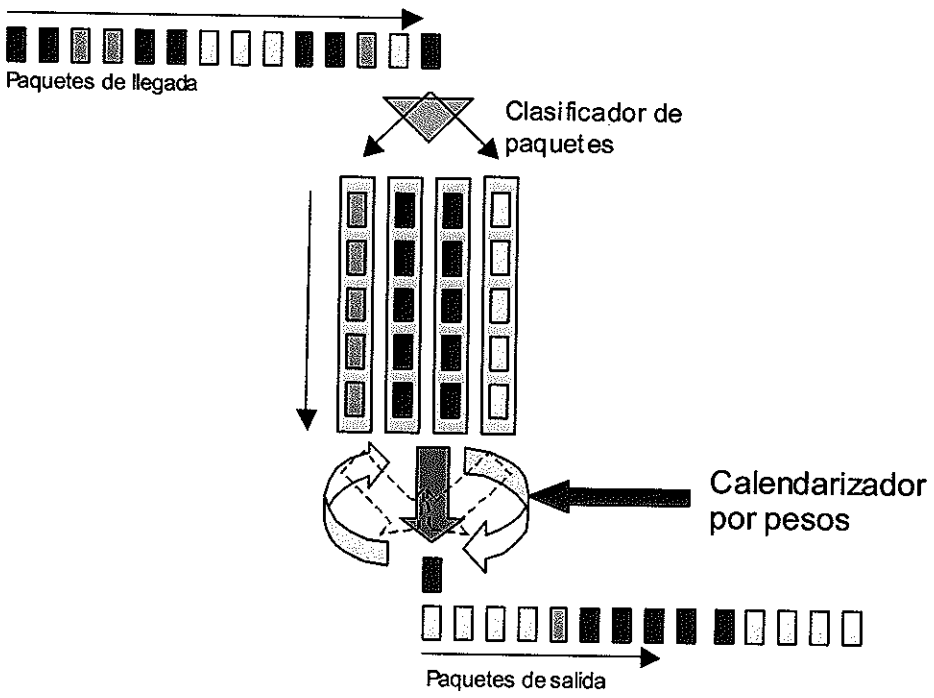

Figura 13. Modelo de operación de WFQ

II.4.3.1 Algoritmo de WFQ

El algoritmo básicamente consiste en asignar un tiempo virtual a cada paquete [Parekh, A., Gallager, R, 1993]. Este tiempo virtual esta en función del número de sesiones activas en un intervalo pequeño de tiempo así como de la suma de los pesos para todas las sesiones:

$$V(t_{j-1}+\tau) = V(t_{j-1}) + \frac{\tau}{\sum_{i \in B_j} \phi_i} \qquad (1)$$
$$\tau \leq t_j - t_{j-1}, j = 2,3,....$$

donde $V(t_{j-1} + \tau)$ representa el tiempo virtual en un intervalo de tiempo donde se tienen B_j sesiones activas y $\sum_{i \in B_j} \phi_i$ representa la suma de los pesos de las sesiones activas en el

intervalo de tiempo $t_{j-1} + \tau$. Cuando no existen sesiones activas, se define $V(t) = 0$ [Parekh, A., Gallager, R, 1993].

Una vez que se asigna el tiempo virtual para el paquete, se necesita obtener los tiempos virtuales en los cuales el paquete inicia y termina de ser procesado. Sean S_i^k y F_i^k los tiempos de inicio y terminación de servicio respectivamente. Si se define $F_i^0 = 0$ para toda *i*:

$$S_i^k = \max\left\{F_i^{k-1}, V(a_i^k)\right\} \qquad (2)$$

$$F_i^k = S_i^k + \frac{L_i^k}{\phi_i} \qquad (3)$$

donde L_i^k es la longitud del paquete *i* de la sesión *k*. El orden en que los paquetes se van a transmitir está determinado por el tiempo de terminación de servicio (F_i^k). El paquete que tenga el F_i^k más pequeño, será el siguiente en transmitirse. Donde *N(t)* es el tiempo real en que el siguiente paquete se tiene que transmitir y F_{min} el tiempo virtual de terminación más pequeño en el tiempo *t*. Entonces, de (1) se tiene que:

$$F_{\min} = V(t) + \frac{N(t) - t}{\sum_{i \in B_j} \phi_i} \qquad (4)$$

$$\Rightarrow N(t) = t + (F_{\min} - V(t)) \sum_{i \in B_j} \phi_i \qquad (5)$$

N(t) contiene el tiempo en el que el siguiente paquete será transmitido. Este proceso se puede resumir de la siguiente manera: cuando un paquete llega al nodo, el tiempo virtual se actualiza y el paquete se etiqueta con el tiempo virtual que le

corresponda. El calendarizador de paquetes procesa los paquetes de acuerdo al tiempo especificado en $N(t)$ en orden ascendente.

II.4.4 Calendarización por clases (CBQ[19])

CBQ fue desarrollado como una herramienta que permite compartir el ancho de banda de los enlaces entre organizaciones o por medio de una técnica llamada "*enlace compartido jerárquico*" [Floyd, S., Jacobson, V. 1995] en forma controlada.

Este esquema consiste en reservar una porción del ancho de banda del enlace a cada una de las organizaciones que estén conectadas al enlace. Dentro de las organizaciones mismas, el tráfico se clasifica en clases, ya sea por el tipo de tráfico, por el protocolo o por las direcciones y se asigna un porcentaje del ancho de banda que tenga asignada la organización a cada una de las clases.

La figura 14 ejemplifica el uso de una estructura jerárquica para compartir el enlace entre tres organizaciones y además entre clases en cada una de las organizaciones. A cada una de las organizaciones se le asigna una porción del ancho de banda del enlace. A su vez, cada organización define de acuerdo a sus necesidades, cómo distribuir el ancho de banda entre sus aplicaciones.

Esta distribución del ancho de banda permite que todas las clases obtengan al menos el ancho de banda asignado en condiciones de congestión.

---

[19] Class-Based Queueing

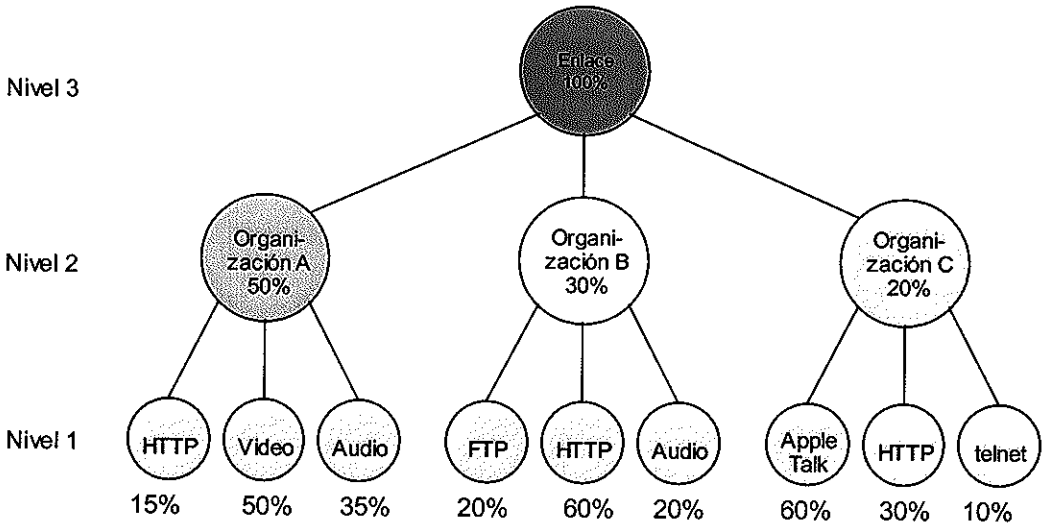

Figura 14. Estructura jerárquica para enlaces compartidos

II.4.4.1 Elementos de CBQ

Los componentes esenciales de CBQ se pueden observar en la figura 15.

Clasificador. Clasifica los paquetes que llegan al enrutador a la clase correspondiente. La clasificación se puede realizar de acuerdo a la información del encabezado del paquete.

Estimador. Este dispositivo se encarga de medir el ancho de banda utilizado por cada clase en un intervalo de tiempo, para determinar si la clase está obteniendo el ancho de banda comprometido. CBQ no describe cómo debe de realizarse el estimador, por lo que se puede seguir cualquier estrategia para implementarlo [Floyd, S. y Jacobson, V. 1995].

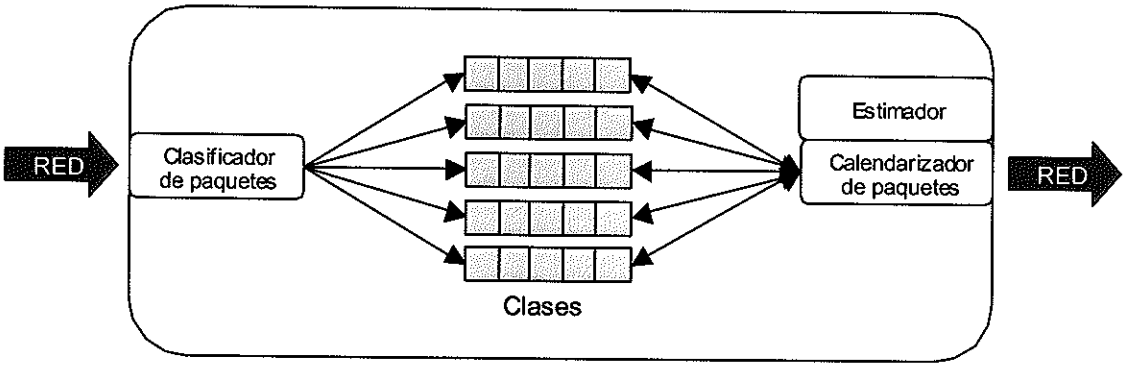

Figura 15. Esquema funcional de CBQ

Calendarizador. El calendarizador se encarga de seleccionar la clase del siguiente paquete que se transmitirá. Una de las funciones del calendarizador es garantizar el nivel de QoS a las clases que así lo requieran. CBQ no define ningún calendarizador en especial, por lo que se puede utilizar cualquier calendarizador que satisfaga las necesidades de QoS.

II.4.4.2 Definiciones importantes

Algunas otras definiciones importantes para entender el funcionamiento de CBQ son las siguientes:

Clases. Todo los paquetes que ingresan al enrutador deben clasificarse en una clase sin excepción. Cada clase tiene asignada una cola dentro del enrutador y es totalmente independiente de las otras clases.

Niveles. Todas las clases en una estructura jerárquica tienen un nivel 1 [Floyd, S., 1995], y sus clases interiores tienen un nivel superior a su predecesor. La figura 14 muestra un ejemplo de la distribución de los niveles en la estructura jerárquica.

Sobre-limitada. Se dice que una clase se encuentra sobre-limitada cuando el ancho de banda promedio en un intervalo de tiempo es mayor que el ancho de banda asignado a la clase.

Sub-limitada. Si no ha utilizado completamente el ancho de banda asignado.

En el límite. Cuando la clase está utilizando todo el ancho de banda asignado sin excederse.

Satisfecha/insatisfecha: Una clase se considera insatisfecha cuando se encuentra en una condición de sub-limitada y cuando todavía tiene paquetes en la cola que no se han transmitido.

III Descripción del modelo

III.1 Introducción

Uno de los elementos que más afectan directamente los parámetros de calidad son los calendarizadores de paquetes. En este trabajo, se hace una comparación entre cuatro de los calendarizadores más comunes que se utilizan en la actualidad: FIFO, PQ, WFQ y CBQ. Para la evaluación de estos mecanismos, se utilizó el modelo que se muestra en la figura 16.

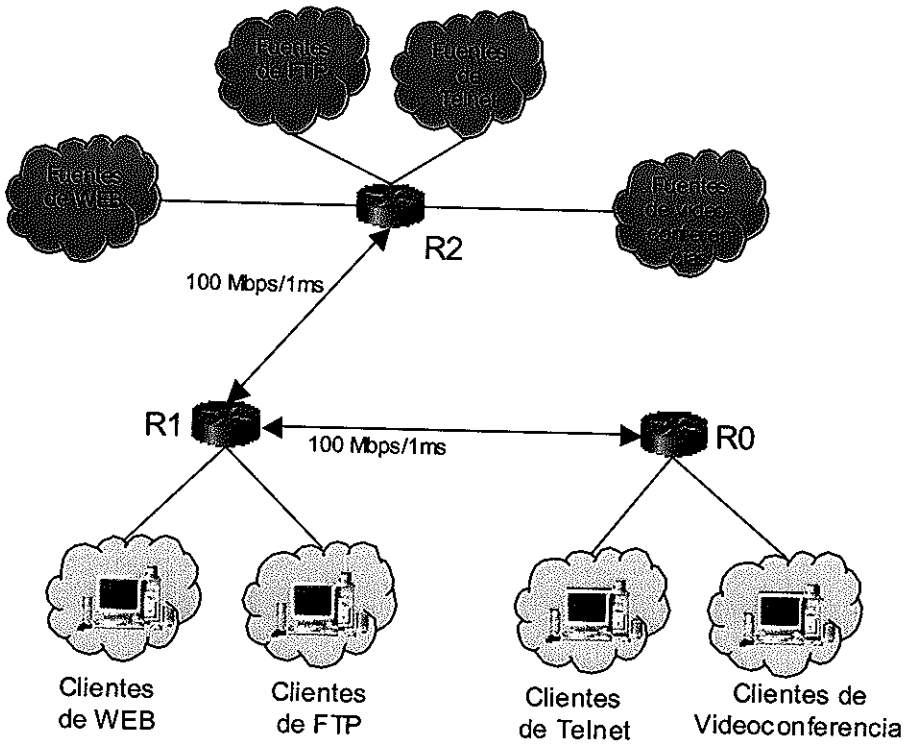

Figura 16. Modelo utilizado para las simulaciones

El modelo incluye fuentes de tráfico web, telnet, FTP[20] y de videoconferencia. Es de especial interés observar los efectos que los mecanismos calendarizadores tienen sobre el tráfico de videoconferencia, ya que este tipo de tráfico es muy sensible al jitter y requiere de un ancho de banda constante.

El modelo consta de tres enrutadores R0, R1 y R2 sobre los cuales se implementaron los diferentes mecanismos de colas. Todos los enrutadores se encuentran conectados con enlaces fast-ethernet (100Mbps) en modo *full-duplex*. Las nubes que se encuentran en la parte inferior de la figura 16, representan a los clientes de las diferentes aplicaciones mientras que las superiores representan a los generadores de tráfico para cada tipo de aplicación. La cantidad de tráfico que se genera es superior a los 100Mbps para congestionar el enrutador R2.

Para los análisis de desempeño se asumieron los siguientes criterios:

- El tráfico de entrada a la red es por el nodo R2.
- Las terminales de los clientes no generan tráfico salvo el requerido para la señalización de los protocolos y las solicitudes del usuario

III.2 Fuentes de tráfico

Cada de uno de los generadores que se utilizaron en el modelo de la simulación siguieron el patrón de la figura 17.

[20] File Transfer Protocol

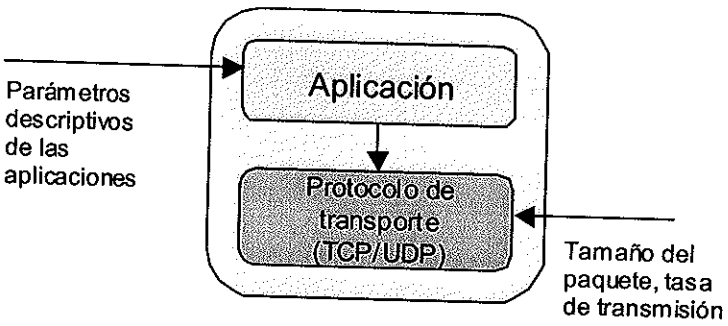

Figura 17. Modelo de los generadores de tráfico

Protocolo de transporte. Este bloque realiza todas las funciones implícitas de los protocolos de transporte. Se encarga de realizar el control del flujo y de la congestión (para el caso de TCP) y de las funciones de comunicación entre nodo fuente y destino. La información contenida en el encabezado de la capa de transporte se utiliza por los nodos receptores para la elaboración de reportes estadísticos.

Aplicación. Este bloque simula el comportamiento de las diferentes aplicaciones que se utilizaron en el modelo. Cada una de las sesiones creadas por este bloque, se identificó por un número de flujo único.

Las fuentes de tráfico utilizadas en el modelo correspondieron a tres aplicaciones basadas en TCP (telnet, FTP y WEB) y una aplicación basada en UDP (videoconferencia).

Cada una de las fuentes de tráfico se conecta con su propio cliente de donde se obtienen estadísticas sobre ancho de banda utilizado, retardo, jitter y pérdida de paquetes. El tiempo simulado fueron 30 segundos.

III.2.1 Parámetros de las fuentes de tráfico

A continuación se mencionan los diferentes parámetros que se utilizaron para describir los cuatro tipos de tráfico.

III.2.1.1 Tráfico web

Para simular el tráfico web se tomaron en cuenta las variaciones que este tipo de tráfico tiene en una red real. Una descripción de los parámetros utilizados [Feldmann, A. *et al.*, 1999] para generar tráfico web se enlista a continuación:

Número de sesiones. El número de sesiones web durante toda la simulación.

Inter-session time (tiempo entre sesiones). Es el tiempo entre las diferentes sesiones de diferentes usuarios

Pages per session (páginas por sesión). El número de páginas accesadas por cada usuario en una sesión

Inter-page time (tiempo entre páginas). Es el tiempo que pasa entre dos páginas consecutivas que un usuario baja del servidor web en la misma sesión.

Objects per page (número de objetos por página). Número de objetos dispuestos en una página web (archivos jpg, gif, au, etc.)

Tamaño de las páginas. Tamaño de las páginas en bytes (aquí se incluyen elementos gráficos así como de texto)

Inter-object time (tiempo entre cada objeto). El tiempo que pasa entre cada objeto de una misma página.

Object size (tamaño de los objetos). Tamaño de los objetos en KB

La tabla II muestra los valores que se propusieron para este tipo de tráfico. Estas condiciones de operación permitieron que se generara tráfico web a una tasa mayor a los 100 Mbps durante los 30 segundos que duró la simulación.

Tabla II. Parámetros para el generador de tráfico web

Parámetro	Distribución	Valor
Número de sesiones	Constante	400
Tiempo entre sesiones	Exponencial	1seg
Número de páginas/sesión	Exponencial	250
Tiempo entre páginas	Exponencial	2.5 seg
Tamaño de las páginas	Exponencial	50Kb
Tiempo entre objetos	Exponencial	1ms
Tamaño de los objetos	Pareto I	24K
Número de servidores	--	5
Número de clientes	--	50

III.2.1.2 Tráfico telnet

Este tráfico se basa en el protocolo TCP. Los parámetros que se definen son el tamaño del paquete así como el intervalo en que llegan solicitudes al servidor telnet (ver tabla III).

Tabla III. Parámetros para la aplicación Telnet

Concepto	Valor
Tamaño del paquete	64 bytes
Intervalo de petición	1ms
Número de nodos	4

III.2.1.3 Tráfico FTP

La aplicación FTP está basada en el protocolo de transporte TCP. En élla se simula la transferencia de archivos a través de la red. Los parámetros especificados para la aplicación es el tamaño del paquete (ver tabla IV).

Tabla IV. Parámetros de la aplicación FTP

Concepto	Valor
Tamaño del paquete	1000 bytes
Número de servidores	12

La tasa de transmisión de paquetes la determina el algoritmo de control de flujos y congestión del protocolo TCP.

III.2.1.4 Tráfico de videoconferencia

El tráfico de videoconferencia se modeló como una fuente que produce paquetes a una tasa constante de bits (CBR[21] por sus siglas en inglés). El tráfico tipo CBR emite paquetes con un tiempo de inter-paquete constante. Los parámetros utilizados para las videoconferencia son los que se muestran en la tabla V:

Tabla V. Parámetros para el tráfico de videoconferencia

Concepto	Valor
Tamaño del paquete	64bytes
Intervalo de salida	341μs (1.5Mbps)
Número de servidores/clientes	de 10 a 50

[21] Constant Bit Rate

III.3 Calendarizadores de paquetes

Cada uno de los calendarizadores que se estudiaron anteriormente en el capítulo II.4, se implementaron en los nodos R0, R1 y R2 de la figura 16. La configuración de los calendarizadores se realizó de tal manera que el tráfico de videoconferencia tuviera un trato preferencial sobre los otros tres tipos de tráfico.

IV Implementación en el simulador

Para realizar este modelo, se utilizó la versión 2.1b8a del "*Network Simulator 2 (ns)*" el cual permite simular redes IP, ATM, LANS, inalámbricas y satelitales. Una de las principales ventajas de este simulador comparado con otros simuladores comerciales, es que su código fuente es abierto, lo que permite realizar modificaciones o adecuaciones al código original según las necesidades del usuario. Lo que no es posible en aplicaciones comerciales como OPNET, el cual presenta una capacidad limitada para realizar modificaciones al código.

La distribución original se encuentra disponible para compilarlo en varias versiones de UNÍX (FreeBSD, LUNIX, SunOS y Solaris) así como para MS Windows (95/98/Me/2000). Existe una distribución binaria para Windows para aquellos usuarios que no cuentan con MS Visual C++.

Para la realización de esta tesis, el código fuente se compiló en LINUX, en una computadora Pentium III a 450 Mhz con 256 Mb en RAM 43 GB de espacio en disco duro.

IV.1 Introducción al *ns*

El *ns* fue desarrollado en el Lawrence Berkeley National Laboratory (LBNL) en la Universidad de California en Berkeley (UCB). Actualmente, el sitio oficial[22] del *ns* se encuentra en el Information Sciences Institute (ISI) de la Universidad del Sur de California (USC).

[22] *http://www.isi.edu/nsnam/ns*

El *ns* está basado en un simulador de eventos discretos. Su principal aplicación es en el área de las telecomunicaciones ya que permite trabajar con redes locales así como redes de cobertura amplia, con enlaces físicos e inalámbricos (rf, satelitales y celulares).

Gran parte de la programación está realizada en C++ y utiliza Otcl (una versión orientada a objetos de Tcl) como la interfaz entre el usuario y la configuración del simulador. La utilización de dos lenguajes obedece a que el *ns* debe de realizar dos funciones diferentes: a) la implementación de protocolos, manejadores de colas, y calendarizadores de paquetes requieren de un lenguaje que pueda manipular eficientemente octetos, encabezados de paquetes y se puedan implementar algoritmos para procesar grandes cantidades de datos y b) el diseño de escenarios así como la necesidad de variar parámetros o la configuración de escenarios, requieren de un lenguaje que permita efectuar cambios y explorar diferentes escenarios de una manera rápida y sencilla. C++ y Otcl satisfacen ambos requerimientos.

C++ es un lenguaje de ejecución rápida pero es lento para modificarlo, lo que lo hace ideal para la implementación de algoritmos y procesadores de paquetes, como por ejemplo, los calendarizadores de paquetes, los cuales no se modifican constantemente. Por otro lado, Otcl es más lento para ejecutarse, pero es mucho más rápido para modificarse ya que no requiere de compilarse cada vez que se modifica.

El procedimiento que se siguió para armar el escenario fue:

1. Definir la topología
    - Definir los nodos
    - Unir los nodos por medio de enlaces

2. Definir los tipos de calendarizadores para cada enlace
3. Asociar agentes de la capa de transporte a los nodos fuentes y destino
4. Asociar generadores de tráfico a los agentes de transporte en los nodos fuente
5. Definir mecanismos para obtener estadísticas
6. Especificar los tiempos de inicio y fin de eventos

Las simulaciones generan archivos de registro de todos los eventos que ocurren en cada uno de los nodos de la red. Además se dispusieron de agentes monitores de flujos y colas que permiten obtener estadísticas para determinar los parámetros de desempeño de la red.

IV.2 Topología

La topología utilizada se basó en el modelo de la figura 16. Todos los enlaces se configuraron con un ancho de banda de 100Mbps y un tiempo de propagación de 1ms. El cuello de botella se localiza en el enlace entre los nodos R2 y R1 en dirección R2 a R1. En el nodo R2 se instalaron los diferentes mecanismos de control de tráfico mientras que los nodos R0 y R1 utilizaron el mecanismo FIFO.

IV.3 Fuentes de tráfico

Para la simulación de las aplicaciones de tráfico se utilizaron los módulos proporcionados por el simulador. A continuación se describen los módulos utilizados así como los parámetros de configuración para cada uno de ellos

Tráfico de FTP: Con este tráfico se simula la transferencia de archivos de gran tamaño como programas, imágenes, etc. El objeto utilizado para simular este tipo de aplicación, fue Application/FTP. Este objeto se asoció a un agente de transporte TCP. La manera de implementarlo fue:

```
# Define los agentes en la capa de tranporte fuente y destino
set src [new Agent/TCP]
set sink [new Agent/TCPSink]
$src set packetSize_ 1000 ; paquetes de 1000 bytes

# Asocia los agentes de transporte a los nodos fuente y destino
$ns attach-agent $nodo_servidor $src
$ns attach-agent $nodo_cliente $sink

# Realiza un enlace lógico entre los agentes de transporte fuente y destino
$ns connect $src $sink

# Asocia la aplicación FTP al agente TCP fuente
set ftp [new Application/FTP]
$ftp attach-agent $src
```

Tráfico TELNET: La manera de implementar esta aplicación en el simulador es utilizando el objeto Application/Telnet.

```
# Define los agentes en la capa de transporte fuente y destino
set src [new Agent/TCP]
set sink [new Agent/TCPSink]
$src set packetSize_ 64 ; Paquetes de 64 bytes

# Asocia los agentes de transporte a los nodos fuente y destino
$ns attach-agent $nodo_servidor $src
$ns attach-agent $nodo_cliente $sink

# Realiza un enlace lógico entre los agentes de transporte fuente y destino
$ns connect $src $sink

# Asocia la aplicación TELNET al agente TCP fuente
set telnet [new Application/Telnet]
$telnet attach-agent $src
$telnet set interval_ 0 ; El intervalo de salida se obtiene de una distribución
exponencial
```

Tráfico de videoconferencia: Para simular el tráfico de videoconferencia se utilizó una fuente de tráfico de tasa de transmisión constante (CBR). En el *ns* ésta se obtiene utilizando el objeto Application/Traffic/CBR. La implementación se realizó de la siguiente forma.

```
# Define los agentes en la capa de tranporte fuente y destino
set src [new Agent/UDP]
set sink [new Agent/LossMonitor]
$src set packetSize_ 64 ; Paquetes de 64 bytes

# Asocia los agentes de transporte a los nodos fuente y destino
$ns attach-agent $nodo_servidor $src
$ns attach-agent $nodo_cliente $sink

# Realiza un enlace lógico entre los agentes de transporte fuente y destino
$ns connect $src $sink

# Asocia la aplicación Traffic/CBR al agente UDP fuente
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $src
$cbr set rate_ 1Mb ; El ancho de banda de cada videoconferencia
```

Tráfico web. Este módulo está basado en Feldmann [1999]. La forma de implementarlo se muestra a continuación.

```
# Número de sesiones activas
set numSession 400

# Intervalo de tiempo entre sesiones
set interSession [new RandomVariable/Exponential]
$interSession set avg_ .5

# Número de páginas por sesión
set sessionSize [new RandomVariable/Exponential]
$sessionSize set avg_ 250

# Inicializa el generador de números aleatorios
global defaultRNG
$defaultRNG seed 0

# Rutina para crear las 400 sesiones
$pool set-num-session $numSession
```

```
set launchTime 0
for {set i 0} {$i < $numSession} {incr i} {
        set numPage [$sessionSize value]
        set interPage [new RandomVariable/Exponential]
        $interPage set avg_ 2.5
        set pageSize [new RandomVariable/Exponential]
        $pageSize set avg_ 50kb
        set interObj [new RandomVariable/Exponential]
        $interObj set avg_ 0.001
        set objSize [new RandomVariable/Exponential]
        $objSize set avg_ 24
        $pool create-session $i $numPage [expr $launchTime + 0.1] \
                $interPage $pageSize $interObj $objSize
        set launchTime [expr $launchTime + [$interSession value]]
}
```

El archivo *traficos.tcl* contiene el listado completo de los generadores de tráfico. Los parámetros utilizados para las fuentes de tráfico se detallaron anteriormente.

IV.3.1 Identificador de flujos

En el simulador, cada flujo que es creado por el generador de tráfico, se le asigna un número de flujo único (*flow_id*[23]), el cual se estampa en el encabezado IP de todos los paquetes. Este número de flujo, se utiliza en el procedimiento de clasificación de paquetes en los mecanismos PQ, WFQ y CBQ así como para el procesamiento de los resultados.

La tabla VI muestra los intervalos de flujos que se asignaron para cada una de las aplicaciones. Los rangos se dispusieron de tal manera que se facilitará el proceso de análisis de los archivos de registros.

---

[23] Campo del identificador del flujo en IPv6

Tabla VI. Distribución del rango de número de flujos por tráfico

Tipo de tráfico	Rango de flujos
WEB	0 -> 29999
Video	30000 -> 39999
FTP	40000 -> 49999
Telnet	50000 -> 50040

IV.4 Calendarizadores de paquetes

Los modelos de calendarizadores que se implementaron fueron: FIFO, PQ, WFQ y CBQ. Los cuatro calendarizadores se instalaron en los tres nodos enrutadores de la Figura 16 (R0, R1 y R2).

IV.4.1 Mecanismo FIFO

La implementación de este mecanismo es muy sencilla. Debido a que solo existe una sola cola, todos los paquetes que ingresan al nodo se colocan en la única pila de paquetes. Los paquetes van siendo extraídos del inicio de la pila.

IV.4.2 Mecanismo PQ

El mecanismo PQ introduce el empleo de prioridades para cada una de las colas. En este modelo, se implementaron cuatro colas, una para cada tipo de tráfico. El nivel de prioridad para cada tráfico se muestra en la tabla VII.

Este mecanismo incluye un proceso de clasificación, el cual se encarga de decidir en cuál cola deben de ir los paquetes que llegan al nodo. La figura 18a muestra el algoritmo que se siguió para clasificar los paquetes. Cuando un paquete llega al nodo, el clasificador

de paquetes revisa el encabezado y obtiene el flow_id. De acuerdo al valor de flow_id, el paquete se coloca en la cola correspondiente.

Tabla VII. Niveles de prioridad en PQ para los diferentes tipos de tráfico

Tipo de tráfico	Cola asignada	Prioridad
Videoconferencia	1	0
Telnet	2	1
FTP	3	2
WEB	4	3

El proceso de extracción de paquetes de las colas se puede apreciar en la figura 18. Primero se extraen todos los paquetes de la cola con prioridad cero. Cuando no existen paquetes en la cola con prioridad cero, se empiezan a extraer los paquetes de la cola con prioridad uno, y cuando no se tienen paquetes en las colas con prioridad cero y uno, se empiezan a extraer paquetes de la cola con prioridad dos y así sucesivamente hasta llegar a la cola con prioridad tres.

Para implementarlo en el simulador, se utilizó el modelo CBQ utilizando las opciones de prioridades. El archivo *topología.tcl* y *pq_cbq*.h contiene la configuración para este mecanismo.

IV.4.3 Mecanismo WFQ

La implementación de este mecanismo incluye cuatro colas: una cola para cada tipo de tráfico. El mecanismo de clasificación se lleva a cabo de manera similar a la figura 18.

Para determinar cuál paquete se debe de extraer primero de las colas, se siguió el procedimiento descrito en la sección II.4.3.

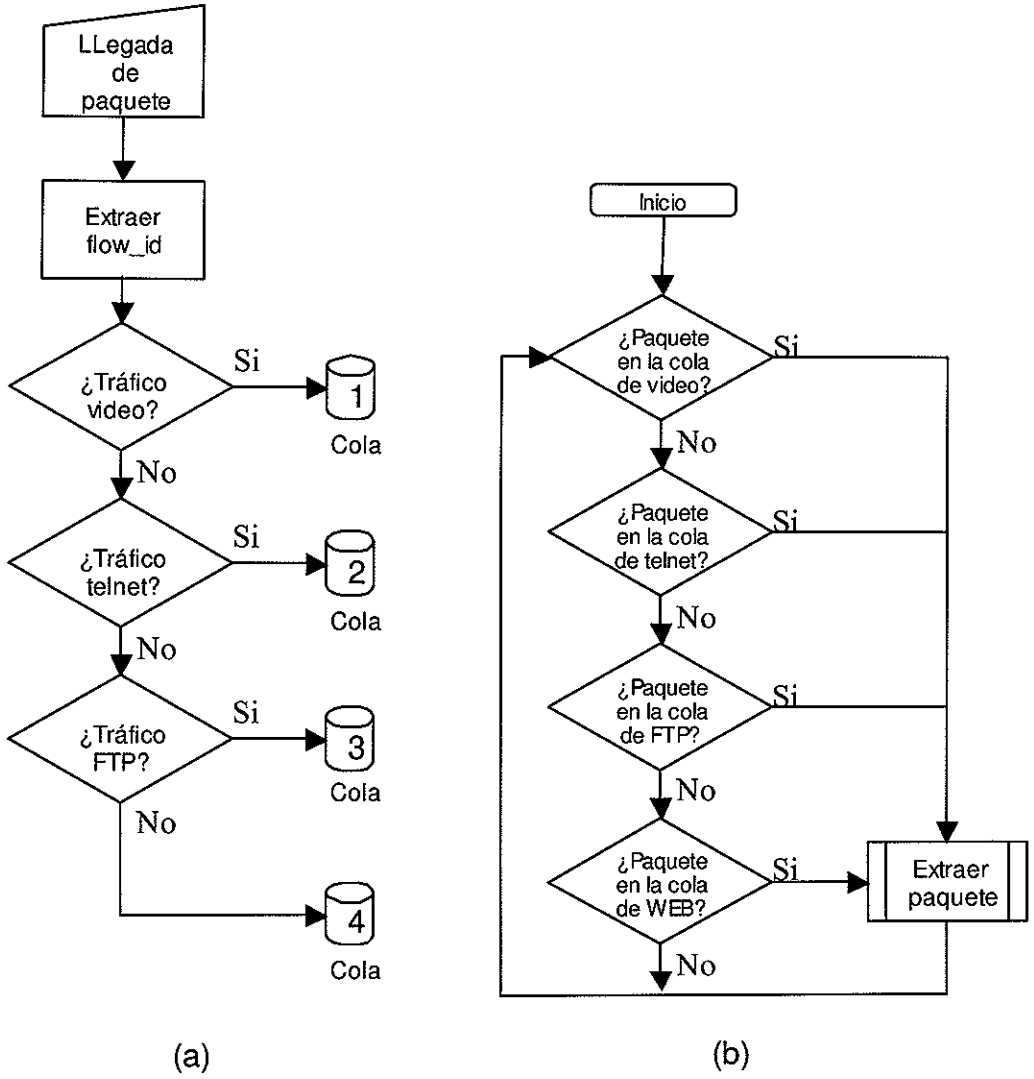

Figura 18. (a) Proceso de clasificación de paquetes.
(b) proceso de extracción de paquetes de las colas

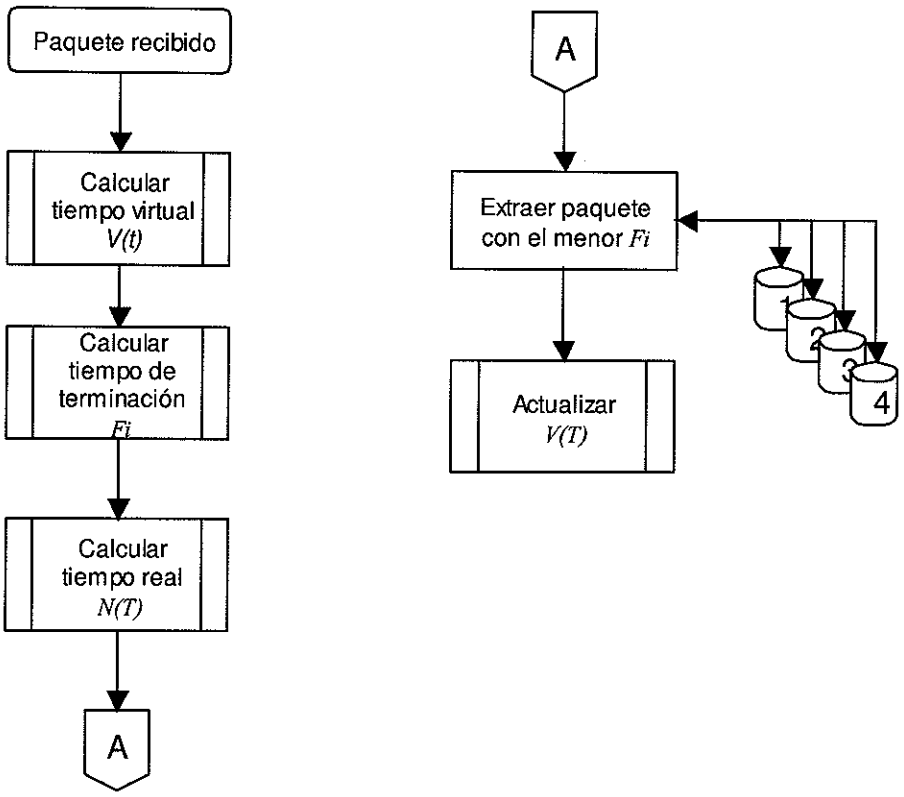

Figura 19.- Procedimiento de extracción de paquetes de una cola WFQ

Cuando un paquete llega al nodo, se calcula el tiempo virtual *V(T)* correspondiente. Con *V(T)* se calcula el tiempo de terminación F_i en función del peso de la cola, el tamaño del paquete y el tiempo virtual. El paquete que es extraído de las colas es el que tiene el F_i más pequeño. Cuando el paquete es extraído se actualiza el tiempo virtual con el valor de F_i.

La tabla VIII muestra los pesos que se utilizaron para cada una de las colas. La distribución de los pesos se realizó de manera experimental manteniendo una relación de dos a uno entre cada una de las colas.

Tabla VIII. Distribución de los pesos para cada una de las colas en WFQ

# de cola	Tipo de tráfico	Peso
1	Videoconferencia	2
2	Telnet	1
3	FTP	.5
4	WEB	.5

IV.4.4 Mecanismo CBQ

Los componentes más importantes de CBQ son el clasificador de paquetes, el esquema de enlace compartido, el calendarizador de paquetes y el estimador. El clasificador de paquetes funciona de manera similar al mostrado en la figura 18a, el cual se explicó anteriormente. Las funciones de enlace compartido no se utilizaron en este modelo. Sólo se utilizó un solo nivel de jerarquía. El orden de extracción de paquetes en CBQ es primero por prioridad (al menos que la clase esté catalogada como sobre limitada) y posteriormente por la reservación del ancho de banda. Cuando existen colas con el mismo nivel de prioridad, las colas son atendidas utilizando el algoritmo WRR[24] en función del ancho de banda reservado para cada cola.

El estimador mide el ancho de banda utilizado para cada una de las clases. Cuando alguna de las clases está por encima del límite especificado, la clase deja de enviar paquetes por un tiempo determinado.

---

[24] Weighted Round Robin

Se definieron 33 clases, donde tres de estas clases pertenecen a los tráficos de telnet, FTP y WEB. Las 30 colas restantes pertenecen a 30 flujos de videoconferencia.

La tabla IX resume los parámetros utilizados para la configuración de CBQ:

Tabla IX. Parámetros de configuración para CBQ

Cola	Tráfico	Ancho de banda	Prioridad
1	WEB	0.25	1
2	FTP	0.25	1
3	Telnet	0.05	1
4-33	Videoconferencia	0.015 c/u	1

IV.5 Procedimiento

Para evaluar el desempeño de los diferentes mecanismos calendarizadores y su capacidad para mantener un nivel de calidad adecuado para el tráfico que es sensitivo al retardo, al jitter y al ancho de banda, se realizaron simulaciones con 10, 20, 30, 40 y 50 videoconferencias para cada uno de los mecanismos. En total, se realizaron 20 corridas, lo que generó más de 25Gb de información.

El archivo *simula.bat* contiene la secuencia de todas las simulaciones así como del procesamiento de los archivos de registro producidos.

V Análisis de resultados

En este capítulo se presentan los resultados que se obtuvieron con la implementación de los diferentes mecanismos de control de tráfico. Los parámetros de desempeño que se midieron para cada uno de los mecanismos fueron: jitter, retardo extremo a extremo, la tasa de pérdida de paquetes (PLR) y el ancho de banda. Las gráficas que se presentan a continuación, fueron generadas a partir de los archivos de registros generados durante las diferentes simulaciones.

Todo el procesamiento se realizó en LINUX utilizando las siguientes herramientas:

- AWK (procesador de líneas en archivos)
- Cat (lista archivos)
- Grep (busca por patrones en archivos)
- Column.pl (Extrae columnas de los archivos de registro)

Los tres primeros programas se encuentran en la distribución UNIX, mientras que *column.pl* es un programa en lenguaje PERL.

En el CD que se anexa, dentro del directorio /analiza, se encuentran todos los programas que se utilizaron para procesar los archivos de registro. Las gráficas fueron generadas por medio del programa Origin Lab para Windows 98. En el disco que se anexa se puede encontrar una versión de evaluación con duración de 20 días.

V.1 Jitter

El primer parámetro que se midió fue la variación del retardo de los paquetes que llegan al nodo destino. Al ser el jitter un parámetro que afecta en mayor grado a las aplicaciones en tiempo real, se midió únicamente para la videoconferencia.

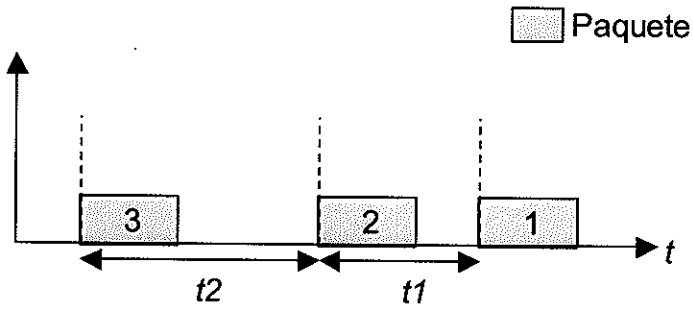

Figura 20. Medición del jitter

La figura 20 muestra la forma en que se realizó la medición del jitter. Un paquete se considera que ha arribado cuando el último bit del paquete es recibido por el nodo. El jitter es la diferencia entre el tiempo en que el último bit de un paquete es recibido por el nodo y el último bit del siguiente paquete. La medición se realizó entre paquetes que pertenecen al mismo flujo.

Las figuras 21, 22, 23 y 24 muestran los resultados obtenidos al implementar los mecanismos FIFO, PQ, WFQ y CBQ respectivamente. Cada figura incluye los resultados para 10, 20, 30, 40 y 50 videoconferencias excepto CBQ en el cual sólo se graficaron los casos para 10, 20 y 30 videoconferencias (45Mbps). En el caso de FIFO, el jitter varió desde 13ms (para 10 videoconferencias) hasta 90ms (para 50 videoconferencias). Si se toma en cuenta que cada 341μs se genera un paquete de un mismo flujo, los valores que se obtienen con FIFO se alejan considerablemente del tiempo esperado.

Por otro lado, los esquemas PQ, WFQ y CBQ mantuvieron variaciones menores a un milisegundo.

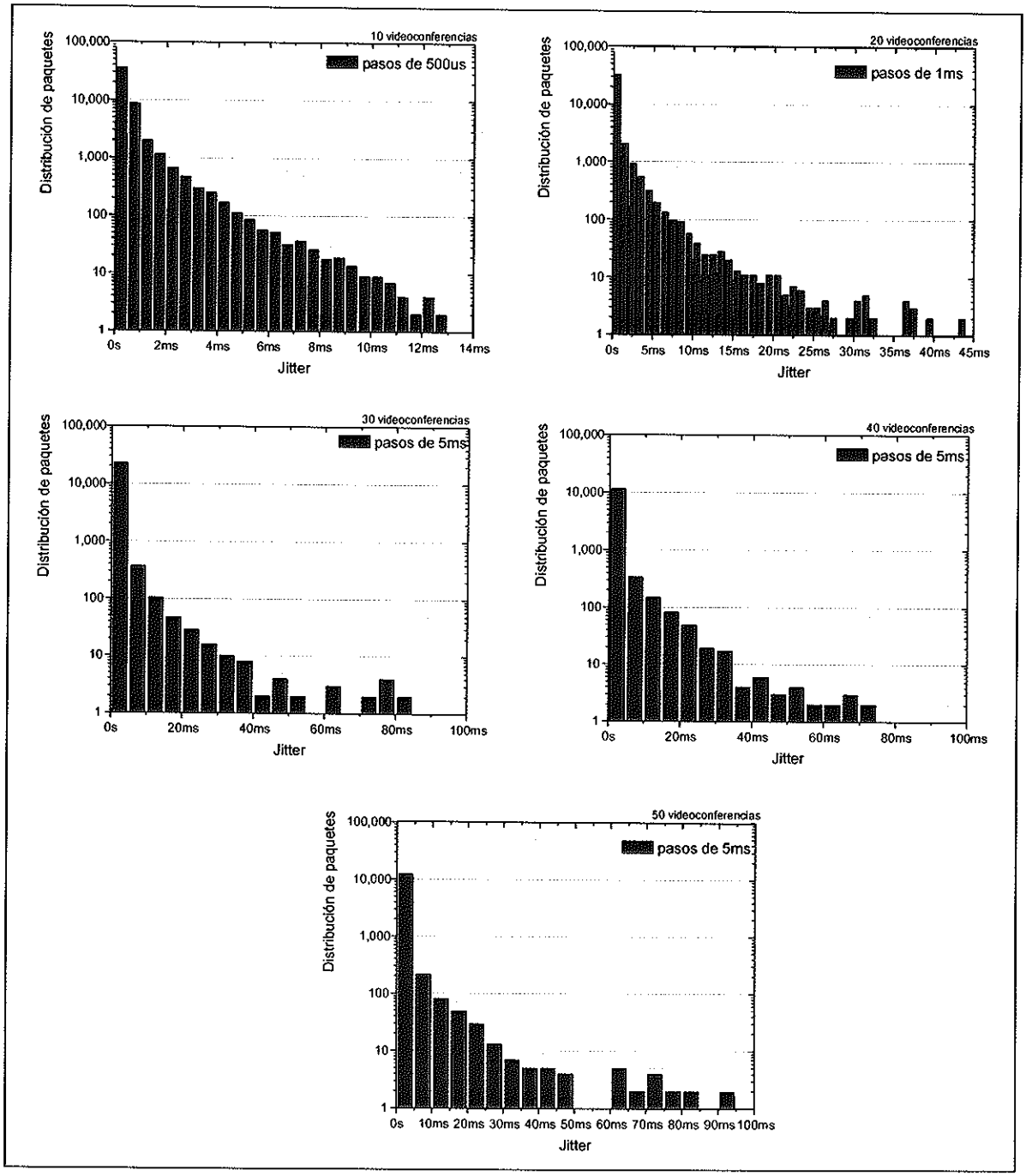

Figura 21. Jitter obtenido utilizando FIFO

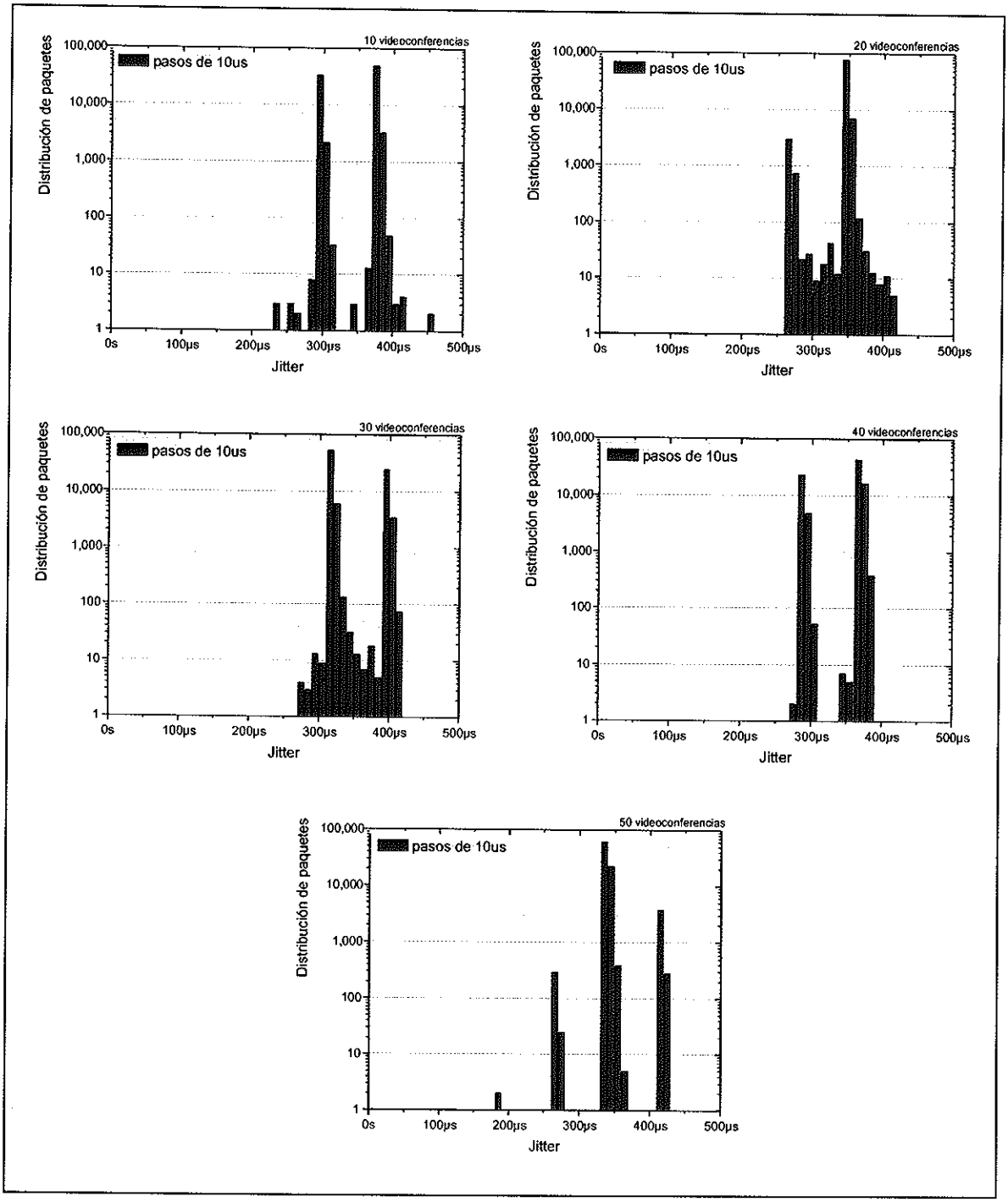

Figura 22. Jitter obtenido utilizando PQ

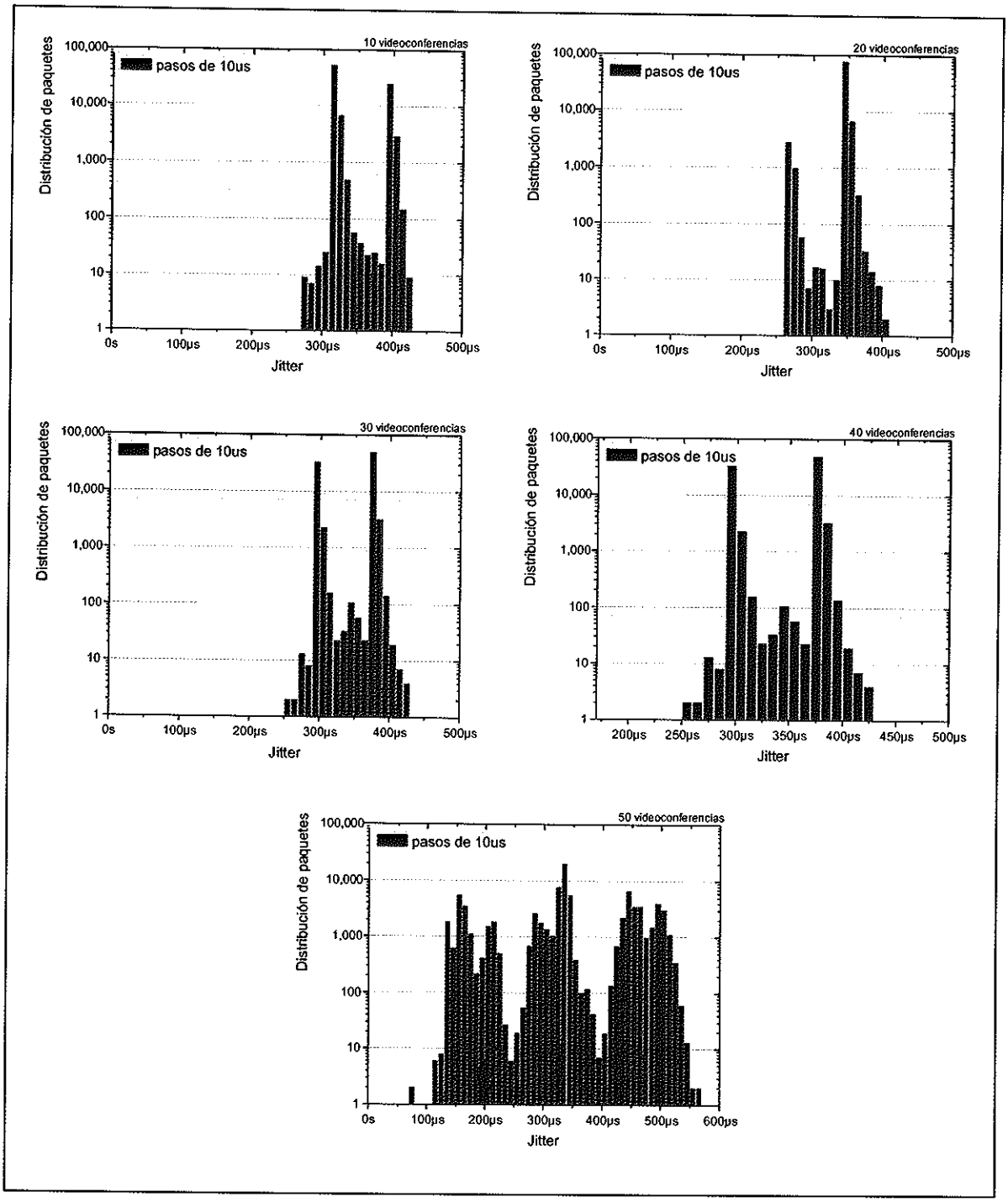

Figura 23. Jitter obtenido utilizando WFQ

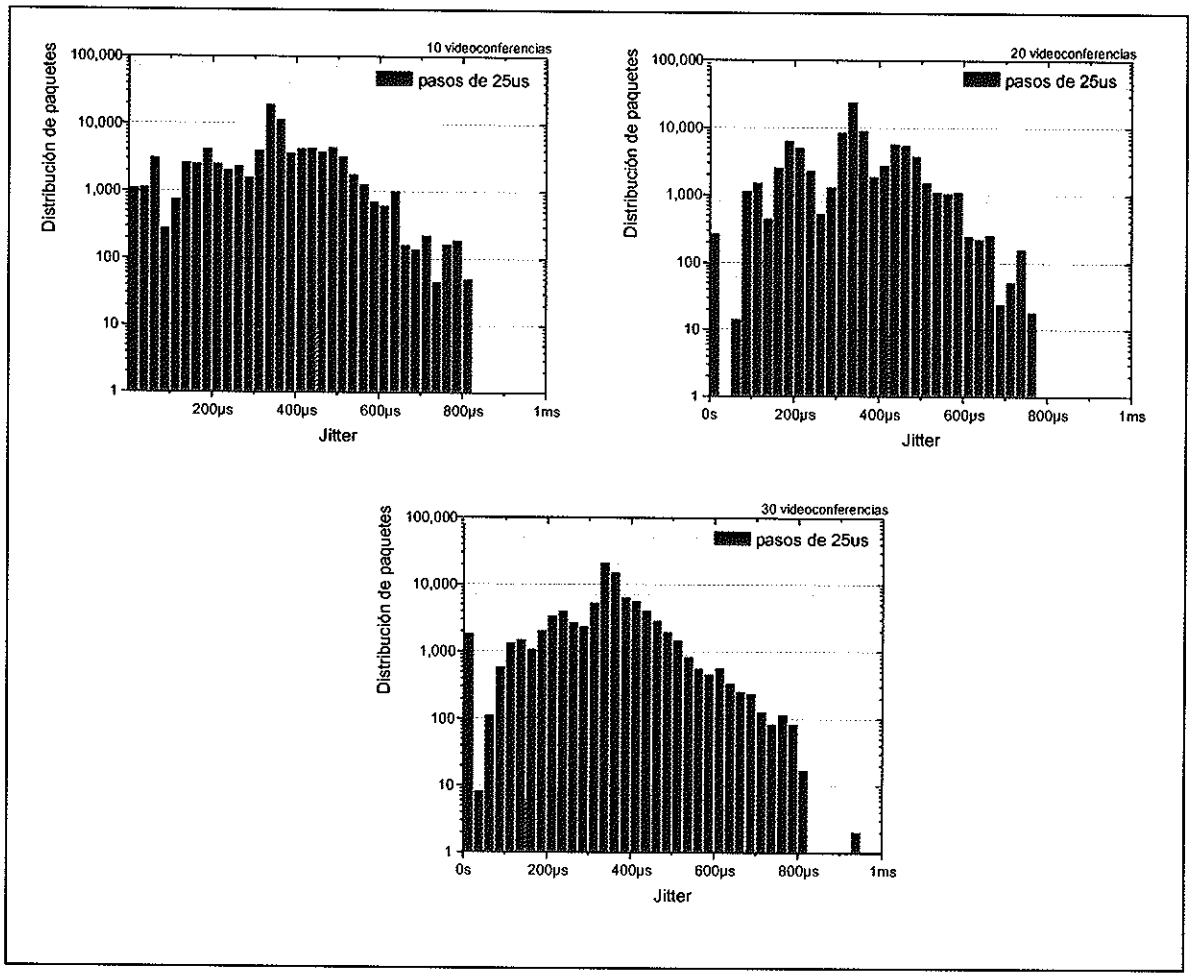

Figura 24. Jitter obtenido utilizando CBQ

De estos tres últimos casos, el que obtuvo una menor variación fue PQ (menor a 450µs para 50 videoconferencias), lo cual era de esperarse, debido a que siempre atiende primero a los paquetes de videoconferencia y permite que estos paquetes pasen poco tiempo en la cola. En el caso de WFQ se presenta un ligero incremento en la variación (hasta 560µs), debido a que WFQ trata de ofrecer el servicio a todas las colas, lo que genera una estancia un poco mayor de los paquetes de videoconferencia mientras los paquetes de otros tráficos son atendidos. CBQ por su parte presenta un incremento mayor que WFQ, y esto se debe a que CBQ garantiza un ancho de banda mínimo a cada una de las clases, lo

que genera un incremento en el tiempo de estadía de los paquetes de videoconferencia en la cola, mientras los flujos de las otras clases obtienen el ancho de banda comprometido, lo que no sucede con PQ ni WFQ.

V.2 Caudal eficaz

La figura 25 muestra el caudal eficaz obtenido por las aplicaciones de video, telnet, ftp y web utilizando los diferentes mecanismos de control de tráfico.

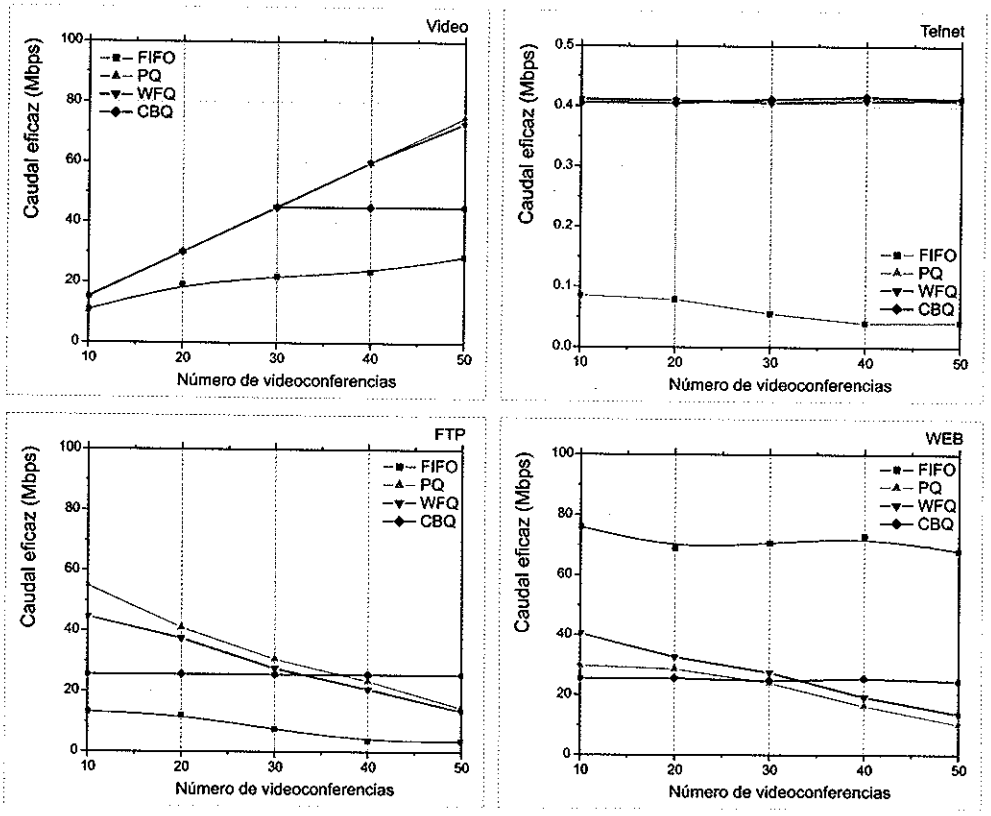

Figura 25. Caudal eficaz

V.2.1 Videoconferencia

En el caso del video, PQ y WFQ presentaron un comportamiento similar ofreciendo el 100% del caudal eficaz requerido a la aplicación hasta las 40 videoconferencias. A partir

de este punto, WFQ comenzó a disminuir el caudal eficaz ofrecido con respecto a PQ debido a que WFQ reparte el ancho de banda del enlace de manera obligatoria entre todos los flujos, mientras que PQ, únicamente envía paquetes de otros flujos mientras no existan paquetes en las colas con prioridad superior. En el caso de CBQ, debido a la política que se especificó de reservar un caudal eficaz de 45Mbps para el flujo de videoconferencias, se puede observar cómo el caudal eficaz se mantiene constante en los 45 Mbps después de que se llegan a las 30 videoconferencias. FIFO no ofreció el ancho de banda requerido por todas las videoconferencias. En todos los casos, el ancho de banda ofrecido fue menor al requerido.

V.2.2 Telnet

Los mecanismos que ofrecen diferenciación de paquetes, mantuvieron un caudal eficaz similar para este tipo de tráfico (alrededor de los 400Kbps). Sin embargo, FIFO ofreció un caudal eficaz muy por debajo del requerido por la aplicación, el cual disminuyó a medida que aumentó el número de flujos de videoconferencia.

V.2.3 FTP

CBQ mantuvo el caudal eficaz comprometido para los flujos de FTP (20Mbps) independientemente de los otros flujos presentes. En cambio, las curvas de PQ y WFQ decrecen a medida que el número de videoconferencias aumenta. Este comportamiento es función del incremento en el número de flujos de videoconferencia. La curva correspondiente a FIFO muestra como el caudal eficaz ofrecido a los flujos de FTP empieza en niveles bajos (alrededor de los 13Mbps) y desciende hasta cerca de 5 Mbps con 50 flujos de videoconferencia. Este comportamiento se debe a que los paquetes de FTP se

encuentran con la cola llena al llegar al nodo y son descartados, lo que obliga a las fuentes de FTP a disminuir su tasa de transmisión.

V.2.4 Web

El tráfico WEB se vio beneficiado con FIFO al ofrecerle cerca de 70 Mbps de caudal eficaz. Uno de los factores que determinaron este comportamiento es la cantidad de flujos que se generaron de los servidores WEB. Cuando la cola se encuentra llena, el mecanismo de manejo de congestión de TCP entra en funcionamiento. Sin embargo, como la cantidad de flujos del tráfico WEB es muy grande, no todos los flujos bajan su tasa de transmisión (porque no todos llegan a detectarla), lo que permite que sus paquetes ocupen primero los espacios disponibles en la cola cuando ésta comienza a desocuparse. PQ y WFQ tienen un comportamiento similar, ambos decrecen a medida que aumenta el número de videoconferencias. PQ ofrece menos caudal eficaz que el tráfico FTP, lo cual era de esperarse, ya que el tráfico de WEB tiene asignada la prioridad más baja.

V.3 Tasa de pérdida de paquetes (PLR)

La figura 26 muestra los valores de PLR obtenidos utilizando los diferentes mecanismos. El PLR fue obtenido del cociente de los paquetes recibidos entre el número de paquetes enviados.

V.3.1 Video

En el caso de FIFO, con sólo diez videoconferencias, la cantidad de paquetes descartados sobrepasó el 40%. Este valor tan alto impide que se pueda mantener una transmisión con una calidad aceptable.

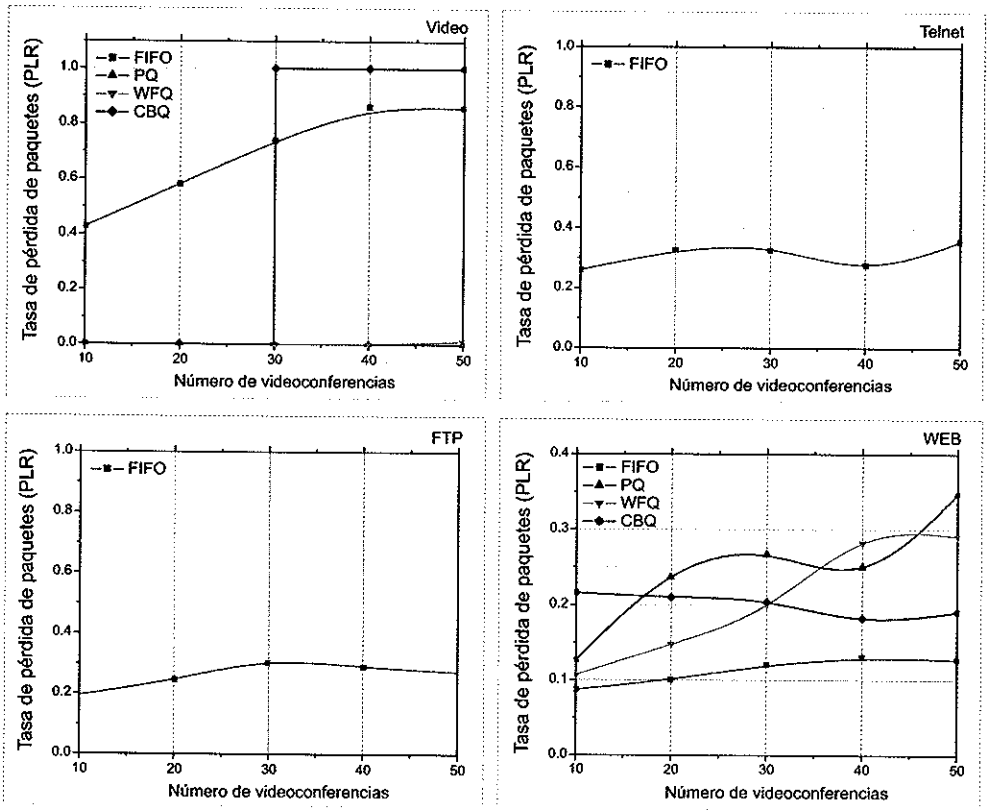

Figura 26. Tasa de pérdida de paquetes

A medida que el número de videoconferencias aumenta, la cantidad de paquetes que son descartados también aumenta considerablemente hasta llegar a un PLR de casi 88%. Este descarte de paquetes se ve reflejado en el caudal eficaz ofrecido a la aplicación, lo que se puede apreciar en la figura 25.

CBQ mantiene un PLR de cero (no se pierde ningún paquete) hasta las 30 videoconferencias. Después de las 30 videoconferencias, todos los paquetes son descartados de acuerdo a los parámetros definidos en la configuración.

Los mecanismos PQ y WFQ que privilegian al tráfico de videoconferencia, mantienen un PLR de cero con todas las videoconferencias, excepto por WFQ que después de las 40 videoconferencias el PLR aumenta ligeramente. Este aumento del PLR en WFQ

se puede ver reflejado en la disminución del ancho de banda ofrecido por WFQ como se puede observar en la figura 25.

V.3.2 Telnet

Los mecanismos PQ, WFQ y CBQ mantuvieron un PLR de cero durante toda la simulación. Debido a que TELNET es una aplicación que requiere muy poco ancho de banda, estos mecanismos no tuvieron problemas para retransmitir los paquetes. En cambio FIFO, descartó alrededor del 30% de los paquetes debido a la gran cantidad de paquetes de WEB que mantiene en la cola.

V.3.3 FTP

Los resultados obtenidos en FTP fueron similares al de TELNET. El PLR para PQ, WFQ y CBQ fue de cero, lo que se debió al poco número de flujos de FTP existentes en la red. De igual manera, el PLR en FIFO llega al 30%, lo que nuevamente se debe a la gran cantidad de paquetes de WEB que se encuentran en la cola.

V.3.4 WEB

FIFO ofreció el menor PLR para el tráfico WEB (cerca del 15%). Este resultado influyó en el desempeño que este mecanismo ofreció a los otros tres tipos de tráfico. A medida que el número de videoconferencias, el PLR para PQ y WFQ aumentó debido a la necesidad de estos dos mecanismos de ofrecer al tráfico de videoconferencia un desempeño adecuado. Una de las condiciones para lograr esto, fue a expensas de descartar paquetes del tráfico WEB. Por otro lado, CBQ mantiene un PLR casi constante durante toda la simulación, ya que le garantiza al tráfico WEB un porcentaje del ancho de banda.

V.4 Retardo de principio a fin

Los resultados obtenidos para el retardo se muestran en la figura 27. La forma de medir el retardo fue desde el momento en que el paquete fue generado en el nodo fuente hasta que fue recibido por el nodo destino.

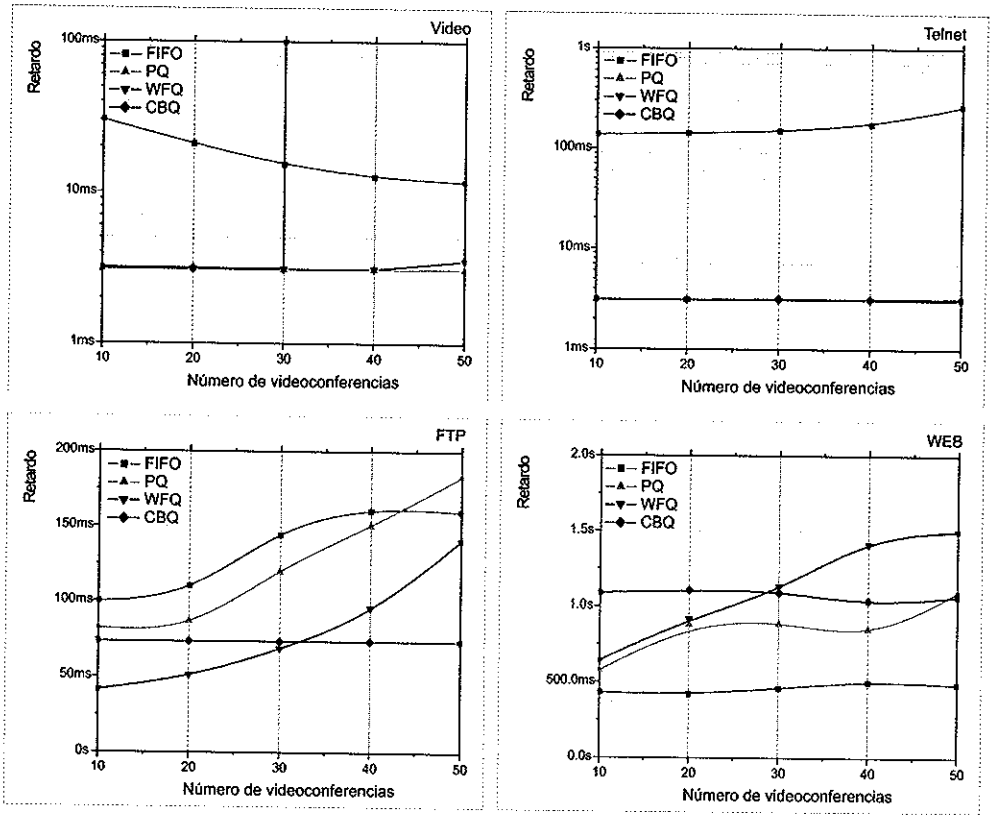

Figura 27. Retardo de principio a fin

V.4.1 Videoconferencia

El mecanismo FIFO mantuvo un retardo mayor a los 10ms durante la simulación. Sin embargo, se puede observar que el retardo disminuye a medida que se aumenta el número de videoconferencias. Esto se puede atribuir principalmente a lo siguiente: a medida que se aumenta el número de videoconferencias, el número de paquetes que

encuentran la cola llena también aumenta (los cuales son descartados). Por lo tanto el número de paquetes que es posible encontrar dentro de la cola es pequeño. El retardo de estos paquetes es el que se grafica, dando la impresión de que el retardo de la red disminuye también.

PQ mantiene un retardo constante de 3.3ms a lo largo de toda la simulación, lo cual era de esperarse (PQ atiende primero a todos los paquetes de videoconferencia). Por otro lado, WFQ también mantiene constante el retardo a 3.3ms excepto cuando se exceden las 40 videoconferencias, situación en la que empieza a aumentar ligeramente el retardo, motivado por la necesidad de ofrecer servicio a los otros tipos de tráfico, lo que obliga a mantener por un poco más de tiempo a los paquetes de videoconferencia en la cola.

CBQ por su parte, también mantiene constante el retardo en los 3.3ms hasta las 30 videoconferencias, posteriormente y como se observó en la figura 26, todos los paquetes son descartados por lo que teóricamente, el retardo aumenta hasta el infinito.

V.4.2 Telnet

La curva correspondiente a FIFO muestra un retardo de un poco más de 150ms y aumenta a medida que se aumenta el número de videoconferencias. Aunque el número de paquetes de telnet son pocos, el mecanismo de control de congestión de TCP contribuye a aumentar el retardo de los paquetes en caso de congestión. Los mecanismos PQ, WFQ y CBQ mantuvieron un retardo constante de 3.3ms durante toda la simulación.

V.4.3 FTP

Como era de esperarse, las curvas de los mecanismos PQ y WFQ crecen al aumentar el número de videoconferencias. El compromiso de ofrecer los recursos a las aplicaciones de videoconferencia y TELNET influyen en el tiempo en que los paquetes permanecen en

las colas. CBQ mantuvo constante el retardo alrededor de los 75ms. Por otra parte, la curva de FIFO se incrementa y se mantiene casi constante después de las 35 videoconferencias. Este comportamiento se puede deducir del resultado del PLR para el tráfico de FTP en donde la tasa se mantiene constante a partir de las 35 videoconferencias.

V.4.4 WEB

El tráfico WEB fue el que obtuvo los tiempos de retardos mayores. En general, el retardo para todos los mecanismos de control de tráfico superó los 500ms. En el caso de CBQ el retardo promedio fue superior a un segundo, lo que se puede considerar alto. PQ y WFQ iniciaron cerca de los 700ms hasta llegar a los 1.5 segundos.

VI Acuerdos de nivel de servicio y los mecanismos de control de tráfico

Los acuerdos de nivel de servicio (SLA por sus siglas en inglés) permiten definir la relación que existe entre el proveedor de servicio y el usuario. Es un documento que especifica el nivel de servicio que el usuario va a obtener por parte de la red en cuanto al ancho de banda, retardo, jitter y tasa de pérdida de paquetes.

El contenido de los SLA se puede dividir en dos partes: la relacionada con aspectos legales y de logística con el usuario y la relacionada con los aspectos técnicos de la red. En el primer punto, se definen el tipo de servicio que el usuario desea contratar, la forma en que el usuario va a interactuar con el personal del proveedor, requisitos legales, formas de pago, etc. El segundo punto se enfoca al nivel de servicio que el usuario espera recibir por parte del proveedor. En este punto, es donde se definen los valores de frontera para los parámetros de desempeño de la red así como las condiciones en las cuales se puede cumplir con lo estipulado en el acuerdo.

Para el proveedor, la definición de estos parámetros es muy importante, ya que dependiendo de los parámetros elegidos, y del mecanismo implementado para cumplirlos, dependerá el éxito o fracaso de la aplicación del cliente.

VI.1 Métricas

Además de definir los parámetros de desempeño, los SLA describen la manera en que se van a realizar las mediciones [Murray, M. y Claffy, K.

2001] para verificar que se están cumpliendo con los objetivos del acuerdo. Estas mediciones incluyen métricas de disponibilidad y de respuesta de la red [Verma, D. 1999].

VI.1.1 Métricas de disponibilidad

Las métricas de disponibilidad incluyen aquéllas relacionadas con la conectividad de la red como por ejemplo:

Conectividad de la red. La conectividad de la red es el porcentaje del tiempo en el cual un punto de acceso a la red es "visto" por otro punto de acceso. Una manera de medir este parámetro es utilizando paquetes ping de un dispositivo a otro, para verificar que los paquetes no se pierdan.

Cuenta de interrupciones. Las interrupciones en la conectividad se pueden monitorear a través de programas de administración de redes. Por lo general, los nodos dentro de la red están configurados para enviar mensajes de alerta cuando alguna de las interfaces del nodo falle. Las bitácoras de estos nodos se pueden analizar para obtener estadísticas sobre las interrupciones en el dominio de la red.

Las interrupciones en la conectividad se pueden clasificar en dos tipos principalmente:

a) aquéllas que no son advertidas por los usuarios,

b) aquéllas en las cuales los usuarios de la red, sí se dan cuenta de la interrupción.

Para propósitos prácticos de los SLA sólo se toman en cuenta aquellas interrupciones que sí son advertidas por los usuarios.

Tiempo de corrección de las interrupciones: Este es un parámetro importante en los SLA ya que indica el tiempo que transcurrirá desde que se notifica al

operador (ya sea por el cliente o por una notificación de un nodo en forma automática) de una interrupción del servicio hasta el momento en que el problema esta resuelto. Muchos tipos de interrupciones por lo general se corrigen de forma automática y en un tiempo relativamente pequeño (del orden de segundos hasta minutos). Otras interrupciones que son más severas, requerirán de mucho mayor tiempo para corregirlas del orden de algunas horas. Este parámetro es muy importante para el usuario que se encuentre especificado en SLA.

Tasa de error: Los errores en la red y en las aplicaciones se pueden medir de diferentes formas. Algunas de las más comunes son:

- Midiendo el número de transacciones fallidas
- Midiendo el número de paquetes descartados en un intervalo de tiempo
- Midiendo el número de paquetes con error en un intervalo de tiempo

El número de paquetes descartados por congestión o por errores en los encabezados dentro de la red, se almacenan en los enrutadores como parte de la base de información de administración (MIB por sus siglas en inglés). Una vez que están en la MIB, pueden ser revisados y procesados por cualquier programa de administración de redes. También es importante la medición de las transacciones fallidas al nivel de TCP. Como estas transacciones no se almacenan en la MIB del enrutador, es necesario colocar un agente en el enrutador que analice los paquetes correspondientes a TCP para obtener un estimado del número de conexiones que hayan sido abortadas.

Tasa de descarte de paquetes: La tasa de error de paquetes se mide como la fracción de paquetes que no llegan a su destino. Un conteo de los paquetes perdidos o con error se mantiene en cada enrutador dentro de la red.

VI.1.2 Métricas de respuesta

Algunos de los parámetros que más comúnmente se miden para determinar la respuesta de una red incluyen el retardo de paquetes, el retardo de paquetes ida-vuelta así como el jitter de la red.

Retardo en un solo sentido. Este parámetro representa el tiempo que tardan los paquetes en ir de un lugar a otro. Existen varias posibilidades para medir este parámetro dependiendo de la estrategia [Verma, 1999] de SLA qué se esté utilizando. Cuando se utiliza una estrategia tipo túnel, este retardo se mide entre dos puntos de intercambio específicos. Es importante destacar la dificultad que existe para medir el retardo en un solo sentido [Murray, M. y Claffy, K. 2001], ya que es necesario sincronizar los relojes de todos los dispositivos dentro de la red. Existen métodos por ejemplo, que se valen de posicionadores globales por satélite (GPS[25] por sus siglas en inglés) para lograr una sincronización exacta de los relojes de los enrutadores, pero estos métodos aún se encuentran en aplicaciones académicas.

Retardo del viaje de ida y vuelta (roundtrip). Este parámetro se mide en algunas ocasiones para poder solventar las dificultades que se presentan cuando se quiere medir el retardo en un solo sentido. Las mediciones del retardo se pueden realizar entre puntos de acceso dependiendo del enfoque del SLA que se haya especificado.

---

[25] Global Positioning Services

Uno de los problemas que se presentan con esta medición es que la ruta que sigue un paquete de ida, es muy probablemente que no sea la misma cuando el paquete viaje de regreso. Estudios que se han realizado sobre la Internet, han demostrado que el 40% del tiempo, los paquetes de regreso no utilizan la misma ruta.

Jitter. El jitter mide la variación que existe en el tiempo de interarribo de paquetes consecutivos. Este parámetro es crítico para aplicaciones de audio y video en tiempo real, en donde la reproducción se realiza a medida que van llegando los paquetes. Ya que la reproducción se realiza a una tasa constante de bits, es necesario realizar un almacenamiento temporal para minimizar los efectos de esta variación y entregar los bits en los intervalos adecuados.

Ancho de banda permitido. Dependiendo de lo estipulado en el SLA, el usuario tiene permitido introducir información a la red a una tasa específica con un retardo y jitter asegurado. Algunas veces sólo se especifica que el usuario puede transmitir hasta una cierta cantidad de tráfico a la red sin que su información sea descartada en los nodos de ingreso a la red. Una forma común que emplean los proveedores del servicio para limitar el uso de ancho de banda al acordado en el SLA, es utilizando un enlace físico con una tasa límite de transmisión igual a la estipulada en el SLA. De esta manera, se asegura que el tráfico originado por el cliente, siempre va a cumplir con el SLA.

VI.2 Los mecanismos de control de tráfico

Dependiendo del mecanismo de control de tráfico que se seleccione, serán los niveles de servicio que se puedan ofrecer dentro de una red.

VI.2.1 FIFO

Es el mecanismo más sencillo. Cuenta con una sola cola, por lo que no se realiza ninguna diferenciación de paquetes. No es posible garantizar ni ancho de banda ni tiempos máximos de retardo o jitter para cualquier flujo.

VI.2.2 PQ

Por lo general este mecanismo permite ofrecer dos niveles de servicio aunque es posible ofrecer más de dos. Este mecanismo consta de dos o más colas donde la cola de mayor prioridad obtendrá un servicio garantizado siempre y cuando la tasa de llegada de paquetes esté limitada. Para las colas de menor prioridad, el nivel de servicio que obtengan dependerá de la cantidad de tráfico que manejen las colas de prioridad superior. Cuando únicamente se requieren dos clases de servicio, por ejemplo, datos y video en tiempo real, PQ permite garantizar al tráfico de video los recursos necesarios de la red.

VI.2.3 WFQ

Este mecanismo permite manejar varias colas. Es posible garantizar el retardo y la tasa de pérdida de paquetes y el ancho de banda siempre y cuando el tráfico de entrada se limite en su tasa de transmisión. Funciona adecuadamente con aplicaciones de tiempo real como el video o la voz manteniendo bajo el retardo y las variaciones de éste.

VI.2.4 CBQ

Éste es un esquema que permite distribuir el ancho de banda disponible en un enlace entre muchos usuarios o aplicaciones. Cada usuario o aplicación obtendrá al menos el caudal eficaz comprometido cuando la red se encuentre en condiciones de congestión. Con este esquema se obtiene un retardo/jitter mayor que WFQ, sin embargo, es posible distribuir de una manera más eficaz el ancho de banda de los enlaces.

VI.3 Implementación

La implementación de los SLA dentro de una red incluye varias etapas. A continuación se enlistan tres de ellas:

1. Estudio de mercado
    - Es necesario realizar un estudio de mercado que ayude a identificar las necesidades de servicios así como dimensionar el mercado.
2. Determinar la infraestructura necesaria
    - De acuerdo a las necesidades del mercado, será necesario hacer modificaciones o definir una nueva topología y arquitectura de la red que permita cubrir el mercado potencial. En este punto es importante definir el modelo de calidad de servicio y los niveles de servicio que satisfagan las necesidades del mercado.
3. Operación de la red con SLA
    - Una vez que la red se encuentre en operación, será necesario disponer de dispositivos que realicen las métricas así como de mecanismos que hagan cumplir los SLA contratados.

La figura 28 muestra la forma en que los SLA se implementan dentro de una red. Los enrutadores y las terminales de extremo a extremo envían información sobre los paquetes a un verificador de SLA, el cual determina si se están cumpliendo los SLA especificados. Si no se estuviera cumpliendo con los SLA, el verificador modificará la configuración de todos los dispositivos de la red de acuerdo a un conjunto de reglas que describen el nivel de servicio de cada flujo.

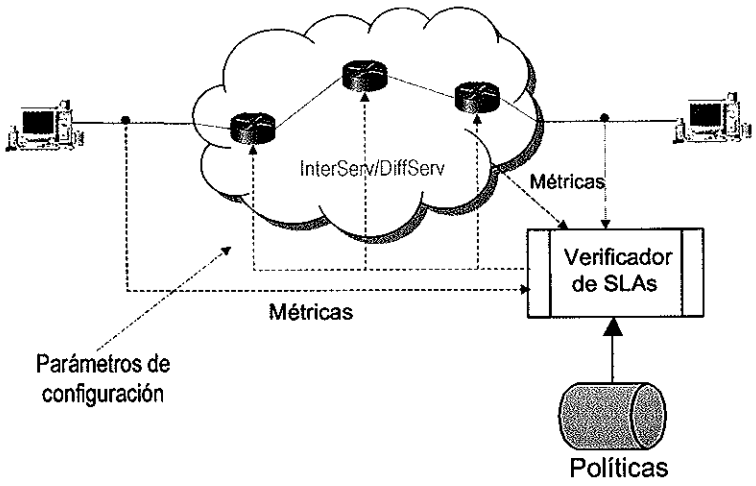

Figura 28. Implementación de los SLA en una red IP

VII Conclusiones y recomendaciones

VII.1 Conclusiones

De acuerdo a los objetivos y metas propuestas para este trabajo se cumplió con lo siguiente:

Se describieron dos modelos de servicio que permiten ofrecer QoS dentro de una red IP: el modelo de Servicios Integrados (IntServ) y el modelo de Servicios Diferenciados (DiffServ). Ambos servicios permiten soportar aplicaciones de tiempo real conviviendo con aplicaciones de datos. Ambos modelos presentan una estrategia diferente para ofrecer QoS. IntServ, por ser un modelo orientado a conexión es más recomendado para redes con pocos nodos, mientras que DiffServ es más eficaz en redes con un gran número de nodos.

Se simularon cuatro de los mecanismos más comunes en la actualidad: FIFO, PQ, WFQ y CBQ. Los elementos indispensables de los modelos de servicio, son los mecanismos de control de tráfico. Por su importancia, han sido implementados en la mayoría de los enrutadores y conmutadores comerciales actuales. El simulador utilizado fue el Network Simulator 2, un simulador de redes de datos de distribución gratuita. Es una excelente opción para la simulación de mecanismos de control de tráfico ya que permite modificar el código original y adaptarlo a las necesidades del usuario. Su distribución incluye los mecanismos más comunes. Su código abierto lo hace más útil para propósitos académicos que comerciales.

Cada uno de los mecanismos de control de tráfico presenta ventajas y desventajas. En condiciones de congestión, el mecanismo FIFO no permite que aplicaciones de tiempo real puedan ser transmitidas con una calidad aceptable. Es necesario aplicar mecanismos que incluyan una clasificación de paquetes y que cuenten con dos o más colas.

Los mecanismos PQ, WFQ y CBQ permiten transmitir aplicaciones de tiempo real en una red en condiciones de congestión. Sin embargo, pueden presentar efectos negativos sobre otros tipos de tráfico. PQ favorece a los flujos de más alta prioridad, garantizando el ancho de banda que los flujos requieran negando recursos a los otros tráficos que compartan el enlace.

WFQ ofrece ancho de banda a todos los flujos presentes, pero no puede garantizar un ancho de banda mínimo a ninguno de los flujos. Por otro lado, WFQ ofrece tiempos de retardo y jitter reducidos para las colas con mayor peso.

CBQ por su parte, garantiza el ancho de banda comprometido a todas las clases en caso de congestión. Esta característica causa que el retardo y el jitter aumenten ligeramente en comparación con WFQ.

Determinar cuál mecanismo es el más adecuado dependerá en gran medida de las condiciones de operación de la red así como de las necesidades de los usuarios. Por ejemplo, en redes en las que solamente se quieren implementar dos niveles de servicio (un servicio básico y un servicio "*premium*"), el mecanismo PQ es una buena opción. Si por el contrario, se quiere repartir el ancho de banda disponible en forma equitativa entre todos los usuarios así como mantener al mínimo el retardo y el jitter, WFQ es la opción correcta. Para el escenario planteado en este trabajo, donde se busca que el tráfico de videoconferencia obtenga un nivel de servicio adecuado (en función de los parámetros de

caudal eficaz, retardo, jitter y pérdida de paquetes) y manteniendo un nivel de servicio tolerable para los demás flujos de la red, el esquema WFQ ofreció un desempeño sobresaliente. La distribución del jitter fue más uniforme entre todos los paquetes, mientras que la tasa de descarte de paquetes se mantuvo en niveles mínimos. De igual forma, las otras aplicaciones no fueron del todo eliminadas, ofreciendo un caudal eficaz que les permitió obtener un desempeño tolerable.

Los mecanismos de control de tráfico son los elementos de la red que determinan el nivel de servicio que reciben los flujos. Existe una relación directa entre los mecanismos de control de tráfico y los SLA. Los mecanismos de control de tráfico implementan la calidad de servicio mientras que los SLA son los encargados de administrar la QoS. Los mecanismos de control de tráfico se pueden implementar sin los SLA, pero su manejo y administración puede ser complicada.

VII.2 Aportaciones

La implementación de mecanismos de control de tráfico que manejen más de una cola puede mejorar considerablemente el desempeño de una red, en especial, cuando existen aplicaciones de tiempo real como es el caso de la videoconferencia. La correcta elección de los parámetros de configuración es un factor importante en el desempeño de los mecanismos de control de tráfico.

Los datos obtenidos de este estudio permiten determinar los niveles de servicio que se pueden aplicar con los diferentes mecanismos de control de tráfico. Estos resultados pueden servir como punto de partida para otros estudios sobre la implementación de modelos de servicios y para la determinación del mejor mecanismo de control de tráfico que se puede aplicar para una necesidad de comunicación específica.

Durante el desarrollo de la tesis, se evaluó el desempeño de un simulador de redes de distribución gratuita y código abierto: el Network Simulador Versión 2. Los resultados obtenidos por medio de este simulador fueron satisfactorios debido a su versatilidad para programarse y a su respaldo por parte de un amplio número de académicos de diferentes instituciones del mundo. Su utilización dentro del CICESE permitirá a los estudiantes del área de redes, obtener un mejor conocimiento sobre las redes de datos y en específico sobre los propios protocolos de comunicaciones.

Los resultados de este trabajo se presentaron en la **12a Reunión de Otoño de Comunicaciones, Computación y Exposición Industrial (ROC&C)** la cual fue organizada por la IEEE en la ciudad de Acapulco, Guerrero en el mes de septiembre de 2001.

VII.3 Recomendaciones

La aplicación de tiempo real que se simuló fue la de videoconferencia transmitiendo a una tasa constante (CBR). Simular videoconferencias con tasa de transmisión variable (VBR) permitirá obtener una mejor percepción del comportamiento de los mecanismos con tráficos de tiempo real más dinámicos. Además, incluir otras aplicaciones de tiempo real como la voz sobre IP (VoIP) creará un escenario más general de las redes actuales.

Un papel importante en los mecanismos de control de tráfico, y que pueden determinar la tasa de pérdida de paquetes, el retardo y el jitter son las colas. Determinar la longitud correcta de las colas no es fácil. Es necesario trabajar en esta dirección para establecer criterios que permitan hacer un uso eficiente de los mecanismos.

Es de especial interés determinar cuál modelo de servicio es el que mejor desempeño puede tener en una red con características específicas. Es importante llevar a cabo simulaciones de los modelos DiffServ, IntServ y MPLS que permitan conocer su comportamiento en diferentes condiciones de operación. Para esta tarea se puede utilizar el simulador Network Simulator el cual incluye los tres modelos de servicios.

Referencias

Allman, M., Paxson, V. y Stevens, W. 1999. "TCP Congestion Control", RFC 2581, 14 pp.

Black, D., Brim, S., Carpenter, B., Le Faucheur, F., 2001. "Per Hop Behavior Identification Codes", RFC3140, 8 pp.

Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W., 1998. "An Architecture for Differentiated Services", RFC2475. 36 pp.

Braden, R., Clark, D. y Shenker, S., 1994. "Integrated Services in the Internet Architecture: an Overview", RFC1633, 33 pp.

Braden, R., Zhang, L., Berson, S., Herzog, S., y Jamin, S., 1997. "Resource ReSerVation protocol (RSVP)". RFC 2205. 112 pp.

Braun, H. y Claffy, K. 1994. "Web traffic characterization: an assessment of the impact of caching documents from the NCSA's web server". Proceedings of Second International World Wide Web (WWW) Conference '94. Chicago, IL. 15-17 p.

Day, J. D. y Zimmerman, H. 1983. The OSI reference model. Proceedings of the IEEE, 71(12): 1334—1340 p.

Demers, A., Keshav, S., Shenker, S., 1989. "Analysis and simulation of a fair queueing algorithm", Proceedings of ACM/SIGCOMM. 3-12 p.

Felix Cota, H. 1998. "Estudio del canal en el transporte de información utilizando SDH vía satélite Solidaridad". Tesis de maestría en ciencias. CICESE, Ensenada, B.C. 93 pp.

Floyd, S., y Jacobson, V. 1995. "Link-sharing and Resource Management Models for Packet Networks". IEEE/ACM Transactions on Networking, 3(4): 365-386 p.

Feldmann, A., Gilbert, A., Huang, P. y Willinger, W. 1999. "Dynamics of IP traffic: A study of the role of variability and the impact of control". In Proceedings of the ACM SIGCOMM, pp 301-313.

Karasan, E. y Ayanoglu, E. , 1998. "Performance of **WDM** Transport Networks". IEEE Journal on Selected Areas in Communications. 1081-1096 p.

Kilkki, K. 1999. "Differentiated Services for the Internet". Macmillan Technical Publishing, Indianapolis. 356 pp.

McDysan, D. 2000. "QoS & Traffic Management in IP & ATM Networks". McGraw Hill, New York. 456 pp.

Murray, M. y Claffy, K. 2001. "Measuring the Immeasurable: Global Internet Measurement Infrastructure." Proceedings of PAM2001 - A workshop on Passive and Active Measurements. Amsterdam, Netherlands.

Nichols, K. y Carpenter, B., 2001. "Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification", RFC3086. 24 pp.

Parekh, A., Gallager, R, 1993. "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case", IEEE/ACM Trans. on Networking. 1(3):344-357 p.

Postel, J., 1981. "Internet Protocol", RFC 791/STD0005. 45 pp.

Postel J,. 1981. "Transmission Control Protocol", RFC793. 85 pp.

Postel, J. 1980. "User datagram protocol". RFC 768. 3 pp.

Shenker, S., Partridge, C. y Guerin, R., 1997. "Specification of Guaranteed Quality of Service", RFC2212. 20 pp.

Shenker, S. y Wroclawski, J. 1997. "General characterization parameters for integrated service network elements". RFC 2215, 16 pp.

Wroclawski, J., 1997. "Specification of the Controlled-Load Network Element Service", RFC2211. 19 pp.

Verma, D. 1999. "Supporting Service Level Agreements on IP Networks". Macmillan Technical Publishing, Indianapolis. 274 pp.

Anexo A. Tutorial corto sobre el ns

Introducción

El Network Simulator (ns) es un simulador de eventos discretos orientado a objetos, el cual esta basado en el "REAL network simulator", un simulador desarrollado en 1989 para estudiar los flujos y la congestión en redes de conmutación de paquetes.

La principal aplicación del simulador son las redes locales de datos y las redes de cobertura amplia, aunque incluye aplicaciones para la simulación de redes inalámbricas, celulares y satelitales.

Su código fuente está escrito en dos lenguajes orientados a objetos: C++ y oTCL (una versión orientada a objetos de TCL). En C++ se codifican todas las funciones de procesamiento y ensamble de paquetes, mientras que en oTCL se codifican todas las funciones de configuración y ejecución.

En este anexo se muestran algunas funciones útiles para la creación de fuentes de tráfico así como funciones para monitorear y obtener estadísticas de las simulaciones. La construcción de escenarios e instrucciones de ejecución se programan en TCL.

Organización de un programa

Todos los programas deben de llevar la siguiente estructura:

1. *Inicialización de un objeto del simulador*

```
set ns [new Simulator]
```

2. *Inicialización del monitor de eventos (opcional)*

```
set nf [open out.nam w]
$ns namtrace-all $nf
```

3. *Rutina de finalización*

```
proc finalizar {} {
        global ns nf
        $ns flush-trace
        close $nf
        # Ejecuta el visualizador de simulaciones
        exec nam out.nam &
        exit 0
}
```

4. *Código del usuario*

```
....
```

5. *Instrucción para mandar llamar la rutina de finalización en el tiempo t:*

```
$ns at t "finalizar"
```

6. *Instrucción para ejecutar el archivo:*

```
$ns run
```

Ejemplo:

```
# Crea un objeto del simulador
set ns [new Simulator]

#Crea un archivo para guardar estadísticas
set nf [open out.nam w]
$ns namtrace-all $nf

#Define una rutina para terminar la simulación
proc final {} {
        global ns nf
        $ns flush-trace
        close $nf
        # Ejecuta el visualizador de simulaciones
```

```
        exec nam out.nam &
        exit 0
}

$ns at 0.1 "final"
$ns run
```

Creación de nodos y enlaces

Los nodos y los enlaces son las elementos básicos en una simulación. Para crear nodos en TCL se utiliza la siguiente instrucción:

```
$ns node
```

La creación de enlaces se realiza con la siguiente instrucción:

```
$ns duplex-link [n1] [n2] [BW] [retardo] [Calendarizador]
$ns simples-link [n1] [n2] [BW] [retardo] [Calendarizador]
```

Descripción de parámetros:

duplex-link: se utiliza para crear un enlace con las mismas características en ambos sentidos.

simples-link: crea un enlace en un solo sentido.

n1: nodo inicial

n2: nodo final

BW: El ancho de banda del enlace en Mb o Kb.

Retardo: Tiempo de propagación en ms o μs.

Calendarizador: mecanismo de control de tráfico para la cola de salida. La distribución original del ns incluye a DropTail (fifo), SFQ y CBQ. Existen otros módulos como por ejemplo WFQ que se pueden instalar a parte.

Ejemplo:

El siguiente código de programa muestra como crear un escenario simple con dos nodos y un enlace:

```
# Crea un objeto del simulador
set ns [new Simulator]

#Crea un archivo para guardar estadísticas
set nf [open out.nam w]
$ns namtrace-all $nf

#Define una rutina para terminar la simulación
proc final {} {
        global ns nf
        $ns flush-trace
        close $nf
        # Ejecuta el visualizador de simulaciones
        exec nam out.nam &
        exit 0
}

# Creación de dos nodos
set n0 [$ns node]
set n1 [$ns node]

#Crea un enlace entre los dos nodos
$ns duplex-link $n0 $n1 2Mb 10ms DropTail

$ns at 0.1 "final"
$ns run
```

Creación de Fuentes de tráfico sobre TCP

Una red sin aplicaciones no tiene sentido. El ns permite crear aplicaciones basadas en TCP y ligarlas a un nodo. La manera en que se crean aplicaciones sobre TCP es creando

primero el nodo, posteriormente debemos asociar al nodo un agente del protocolo de transporte (TCP o UDP) y finalmente asociamos una aplicación al protocolo de transporte. Una característica fundamental de TCP es que es un protocolo orientado a conexión, por lo que necesita una contraparte en el nodo receptor que reciba los paquetes y envíe las confirmaciones necesarias. La instrucción para crear un agente de TCP en el nodo emisor es:

```
set [identificador] [new Agent/TCP]
$ns attach-agent [nodo_emisor] [$identificador]
```

en el nodo receptor:

```
set [identificador] [new Agent/TCPSink]
$ns attach-agent [nodo_receptor] [$identificador]
```

La instrucción "[new Agent/TCP]" crea un agente de TCP y la función "attach-agent" asocia el agente creado al nodo deseado.

Descripción de los parámetros:

identificador: nombre asociado al protocolo creado. Este nombre se usará para modificar parámetros del protocolo. Cuando se haga referencia a este identificador, se debe de anteponer el símbolo $.

nodo_emisor/receptor: en el nodo emisor se instala el generador de paquetes TCP mientras que en el nodo recepetor debemos instalar un agente que procese los paquetes de TCP recibidos.

Variaciones del protocolo TCP

NS incluye diferentes versiones del protocolo TCP que se pueden utilizar tanto en el emisor como en el receptor:

Variaciones de TCP para los nodos emisor

- **Agent/TCP** - Tahoe
- **Agent/TCP/Reno** – Versión Reno
- **Agent/TCP/Newreno** – Versión Reno modificado
- **Agent/TCP/Sack1** – Repetición selectiva (RFC2018)
- **Agent/TCP/Vegas** - Vegas
- **Agent/TCP/Fack** - Reno con "forward acknowledgment"

Variaciones de TCP para los nodos receptores

- **Agent/TCPSink** – Receptor TCP que envía un ACK por paquete recibido
- **Agent/TCPSink/DelAck** – Receptor con retardo configurable para el ACK
- **Agent/TCPSink/Sack1** – Receptor con ACK selectivo (RFC2018)
- **Agent/TCPSink/Sack1/DelAck** - Combinación

Éstas variaciones deben incluirse al momento de crear el agente "[new Agent/XXX]".

Opciones de TCP

Loa agentes de TCP contienen varios parámetros y variables que se pueden modificar y acceder respectivamente. A continuación se enlistan algunos de ellos (los valores mostrados son definidos por omisión. Para ver su significado ver RFC TCP):

```
Agent/TCP set window_ 20                 # Tamaño máximo de ventana;
Agent/TCP set windowInit_ 1              # Tamaño inicial de la ventana
Agent/TCP set windowOption_ 1            # Algoritmo para congestión (1: standard);
Agent/TCP set windowConstant_ 4          # Usado cuando windowOption != 1;
Agent/TCP set windowThresh_ 0.002        #  Para calcular el promedio de la ventana;
Agent/TCP set overhead_ 0                # !=0 Agrega tiempos aleatorios entre envíos;
```

```
Agent/TCP set ecn_ 0                  # TCP debe responder al bit de ECN ;
Agent/TCP set packetSize_ 1000        # Tamaño del paquete (bytes);
Agent/TCP set maxrto_ 64              # Límite en el  RTO (seconds);
Agent/TCP set dupacks_ 0              # Contador de ACK duplicados;
Agent/TCP set ack_ 0                  # El mayor ACK recibido;
Agent/TCP set cwnd_ 0                 # Ventana de congestión (paquetes);
Agent/TCP set ssthresh_ 0             # Límite para slow-stat (paquetes);
Agent/TCP set rtt_ 0                  # muestra de rtt;
Agent/TCP set srtt_ 0                 # rtt promediado rtt;
Agent/TCP set rttvar_ 0               # desviación estándar de muestras rtt;
Agent/TCP set backoff_ 0              # Factor actual del  RTO de retroceso;
Agent/TCP set maxseq_ 0               # número de secuencia máximo enviado;
```

Para modificar o ver los valores de estas variables se debe de usar la siguiente instrucción:

$identificador set [variable] [valor]

variable: algunas de las listadas anteriormente como por ejemplo packetSize_, ack_, etc.

valor: se especifica el valor que debe de tomar la variable. Si este parámetro se omite entones la instrucción regresa el valor actual de la variable.

Si se desea modificar el valor de alguna de las variables para todos los agentes TCP que se definan en la simulación se puede usar la siguiente instrucción:

Agent/TCP set [variable] [valor]

Ejemplo

El siguiente ejemplo muestra la creación de un escenario con dos nodos y la instalación de un agente de TCP en el nodo emisor y el agente TCPSink en el nodo receptor.

```
# Crea un objeto del simulador
set ns [new Simulator]

#Crea un archivo para guardar estadísticas
set nf [open out.nam w]
$ns namtrace-all $nf

#Define una rutina para terminar la simulación
proc final {} {
        global ns nf
        $ns flush-trace
        close $nf
        # Ejecuta el visualizador de simulaciones
        exec nam out.nam &
        exit 0
}

# Creación de dos nodos
set n0 [$ns node]
set n1 [$ns node]

#Crea un enlace entre los dos nodos
$ns duplex-link $n0 $n1 2Mb 10ms DropTail

set tcp [new Agent/TCP]                 # Crea un agente emisor TCP
set sink [new Agent/TCPSink]            # Crea un agente receptor TCPSink
$tcp set packetSize_ 500                # Paquetes de 500 bytes
$tcp set window_ 50                     # Ventana de 50 paquetes
$ns attach-agent $n0 $tcp               # Asocia el emisor al nodo n0
$ns attach-agent $n1  $sink             # Asocia el receptor al nodo n1
$ns connect $tcp $sink                   # Establece la conexión TCP;

$ns at 0.1 "final"
$ns run
```

Fuentes de tráfico sobre TCP

El NS incluye dos aplicaciones simuladas para trabajar sobre TCP: FTP y Telnet.

Tráfico FTP

Produce paquetes en forma masiva para que un agente TCP los envíe. Las instrucciones para crearlo son:

```
set id [new Application/FTP]        # Crea la fuente FTP
$id attach-agent $agente            # Asocia id con el agente TCP
```

donde

id: el nombre que se le asignará a la fuente de FTP

agente: el nombre del agente de TCP que se creó para la fuente

Tráfico TELNET

Produce paquetes individuales con tiempos de interarribo variables.

```
Set id [new Application/Telnet]
$id attach-agent $agente                    # Asocia FTP con el agente  TCP
```

donde

id: el nombre que se le asignará a la fuente de TELNET

agente: el nombre del agente de TCP que se creó para la fuente

Ejemplo:

```
# Crea un objeto del simulador
set ns [new Simulator]

#Crea un archivo para guardar estadísticas
set nf [open out.nam w]
```

```
$ns namtrace-all $nf

#Define una rutina para terminar la simulación
proc final {} {
        global ns nf
        $ns flush-trace
        close $nf
        # Ejecuta el visualizador de simulaciones
        exec nam out.nam &
        exit 0
}

# Creación de dos nodos
set n0 [$ns node]
set n1 [$ns node]

#Crea un enlace entre los dos nodos
$ns duplex-link $n0 $n1 2Mb 10ms DropTail

set tcp [new Agent/TCP]          # Crea un agente emisor TCP
set sink [new Agent/TCPSink]     # Crea un agente receptor TCPSink
$tcp set packetSize_ 500         # Paquetes de 500 bytes
$tcp set window_ 50              # Ventana de 50 paquetes
$ns attach-agent $n0 $tcp        # Asocia el emisor al nodo n0
$ns attach-agent $n1  $sink      # Asocia el receptor al nodo n1
$ns connect $tcp $sink           # Establece la conexión TCP;

set ftp [new Application/FTP]    # Crea la fuente FTP
$ftp attach-agent $tcp           # Asocia la fuente FTP con el agente TCP

$ns at 0.1 "$ftp start"          #Envía paquetes a partir del 0.1 segundo
$ns at 0.9 "$ftp stop"           #Deja de enviar paquetes en el 0.9 segundo

$ns at 1.0 "final"
$ns run
```

Creación de Fuentes de tráfico sobre UDP

Para enviar información sobre el protocolo UDP es necesario primero crear un agente de UDP y asociarlo al nodo transmisor. La sintaxis es similar a la creación de agentes TCP:

set *nombre_agente* [new Agent/UDP]
$ns attach-agent $*nodo* $*nombre_agente*

donde

nombre_agente: el nombre con el cuál se identifica al agente

nodo: el nodo al que se le asocia el agente receptor

Cuando un paquete llega al nodo destino, el agente receptor puede tratar los paquetes de dos formas: recibirlo y descartarlo sin guardar información sobre el paquete o recibirlo y guardar información sobre cuantos paquetes han llegado así como la cantidad de bytes que han sido recibidos.

Únicamente descartando paquetes
set *nombre_agente* [new Agent/Null]
$ns attach-agent $*nodo* $*nombre_agente*

Procesando información sobre los paquetes.
set *nombre_agente* [new Agent/LossMonitor]
$ns attach-agent $*nodo* $*nombre_agente*

donde

nombre_agente: el nombre con el cuál se identifica al agente

nodo: el nodo al que se le asocia el agente receptor

Es necesario relacionar ambos agentes (el agente de UDP con su agente receptor) para que se pueda realizar la transmisión de paquetes:

$ns connect $*agente_udp* $*agente_receptor*

donde

agente udp: el nombre del agente de UDP del nodo transmisor

agente_receptor: nombre del agente que recibirá los paquetes del nodo transmisor.

Una vez que se haya configurado el protocolo de transporte es necesario asociarle una fuente de tráfico. Un ejemplo de fuente de tráfico que se puede transmitir sobre UDP son las aplicaciones con una tasa de transmisión constante (CBR):

set *nombre_agente* [new Application/Traffic/CBR]
$*nombre_agente_* attach-agent $*nombre_agente_UDP*

donde

nombre_agente: el nombre con el cuál se identifica al agente de tráfico

nombre_agente_UDP: el nombre del agente de UDP al cuál se va a asociar la fuente de tráfico

A las fuentes CBR se les puede modificar los siguientes parámetros:

#Tamaño del paquete
$cbr set packetSize_ 64

#Tasa de transmisión
$cbr set rate_ 1.5Mb

Ejemplo:

```
# Crea un objeto del simulador
set ns [new Simulator]

#Crea un archivo para guardar estadísticas
set nf [open out.nam w]
$ns namtrace-all $nf
```

```
#Define una rutina para terminar la simulación
proc final {} {
        global ns nf
        $ns flush-trace
        close $nf
        # Ejecuta el visualizador de simulaciones
        exec nam out.nam &
        exit 0
}

# Creación de dos nodos
set n0 [$ns node]
set n1 [$ns node]

#Crea un enlace entre los dos nodos
$ns duplex-link $n0 $n1 2Mb 10ms DropTail

set udp [new Agent/UDP]                 # Crea un agente emisor UDP
set null [new Agent/Null]               # Crea un agente receptor Null
$ns attach-agent $n0 $udp               # Asocia el emisor al nodo n0
$ns attach-agent $n1  $null             # Asocia el receptor al nodo n1
$ns connect $udp $null                  # Establece la conexión UDP

set cbr [new Application/Traffic/CBR]   # Crea una fuente de tráfico CBR
$cbr attach-agent $udp
$cbr set packet_size_ 64                # Paquetes de 64 bytes
$cbr set rate_ 1Mb                      # Tasa de 1Mbps


$ns at 0.1 "$cbr start"             #Envía paquetes a partir del 0.1 segundo
$ns at 0.9 "$cbr stop"              #Deja de enviar paquetes en el 0.9 segundo

$ns at 1.0 "final"
$ns run
```