COMPORTAMIENTO DE LOS SERVICIOS DIFERENCIADOS (DIFFSERV) Y LOS SERVICIOS INTEGRADOS (INTSERV) EN REDES IP PEQUEÑAS

THE BEHAVIOR OF DIFFSERV AND INTSER IN SMALL-SCALE IP NETWORKS

Abstract
This article shows the results of implementing Diffserv and IntServ models upon SUSE Operating System using Network Simulator NS-2.33 in order to compare the performance of the models regarding performance variables such as throughput, packet loss, jitter and delay.

Keywords: Differentiated Services, integrated Services, quality of Service, throughput, delay, jitter, RSVP.

Resumen
En el presente artículo se muestran los resultados obtenidos de la implementación de los modelos Diffserv e IntServ utilizando Network Simulator NS-2.33 sobre plataformas Linuxcon el fin de comparar el desempeño de los modelos, en factores como el throughput, la pérdida de paquetes, el jitter y el retardo.

Palabras clave: Servicios diferenciados, servicios integrados, calidad de servicio, rendimiento, retardo, variación del retardo, RSVP.

Diego A. Segura G.
Estudiante Ingeniería Electrónica de la Universidad Distrital "Francisco José de Caldas"
Diegoalejo001@hotmail.com

Francisco J. González A.
Estudiante Ingeniería Electrónica de la Universidad Distrital "Francisco José de Caldas"
frankyaz@gmail.com

Danilo A. López Sarmiento
Ingeniero Electrónico, MSc. en Teleinformática, docente planta de la Universidad Distrital "Francisco José de Caldas", Coordinador de la Maestría en Ciencias de la Información y las Comunicaciones.
dalopezs@udistrital.edu.co

Tipo: Artículo reporte de caso

Fecha de Recepción: Nov. 15 de 2010
Fecha de Aceptación: Marzo 15 de 2011

1. INTRODUCCIÓN

El protocolo de Internet (IP) es uno de los protocolos más utilizados para el envío de paquetes de bits desde un origen hasta un destino, mediante un sistema de redes interconectadas. Este protocolo no está ligado a una conexión específica ya que no es orientado a la conexión. Ofrece un servicio (Best Effort)[1], sin ningún tipo de calidad que permita asegurar que los paquetes enviados lleguen correctamente a su destino.

Debido al uso masivo de Internet y a la demanda de los usuarios por un protocolo confiable que garantice que la información que se desea transmitir por internet pueda ser enviada y a la vez recibida con unos mínimos de calidad, hace que las nuevas aplicaciones y servicios requieran estándares de facto que permitan que los administradores de la red den prioridad a la información que se desea enviar o al servicio que se requiere priorizar.

A nivel de QoS se tienen principalmente

cuatro parámetros asociados: throughput, retardo end-to-end [3], jitter [2] y cantidad de paquetes perdidos que determinan el funcionamiento óptimo de una red.

En el estudio de modelos de protocolos base que permiten el mejoramiento del estandar IP, se encuentran los servicios diferenciados (DiffServ) y los servicios integrados (IntServ).

DiffServ implementa un código que se le agrega al datagrama, para así definir un tipo de prioridad. Los demás nodos de la red deben conocer e identificar ese código para así poder dar trato especial a los paquetes que incluyan el código marcado con la mayor prioridad. Por su parte, IntServ, implementa una reserva del canal y control de admisión de los paquetes a través de los nodos que conforman la red.

La herramienta utilizada para el modelamiento de DiffServ e IntServ es Network Simulator 2. NS-2 es un simulador de eventos dirigidos a la investigación en redes y pruebas a protocolos, aplicaciones y servicios [5]. Permite además obtener datos que ayudan en el desarrollo de los protocolos de calidad de servicio y en el análisis de su funcionamiento en cuanto a la red a la se apliquen [4]. Todos estos datos se visualizan utilizando herramientas como Xgraph y Nam [5]. Xgraph permite graficar variables en función del tiempo para evaluar cambios y comportamientos ya sea en ancho de banda, pérdida de paquetes etc. Nam hace posible la visualización de la topología de la red (nodos, enrutadores, conexiones, etc) y su comportamiento a través del tiempo permitiendo observar tráfico, paquetes perdidos, encolamientos, etc.

En este artículo se pretende comparar el desempeño del modelo de servicios diferenciados con el modelo de servicios integrados (utilizando RSVP), analizando variables de red como ancho de banda y pérdida de paquetes. Para lograrlo se ha reproducido el comportamiento del modelo de servicios diferenciados e integrados sobre NS-2 usando una misma topología de red la cual está formada por pocos nodos.

Los resultados obtenidos se visualizan en gráficas a partir de las variables de interés, que permitirán abstraer el comportamiento de cada uno de los modelos. Específicamente se evaluará IP para el caso en que la estructura ofrece un tráfico de "mejor esfuerzo", otra para el caso de aplicación de QoS con Diffserv y una última que oferta QoS con Intserv. A partir de los resultados obtenidos se hará una comparación del comportamiento de las variables de interés (throughput, pérdida de paquetes, jitter, retardo).

2. IMPLEMENTACIÓN DE LOS MODELOS SOBRE NS-2

Con el fin de comparar el desempeño de los modelos de servicios diferenciados e integrados se propone la implementación de la red de la Fig. 1.

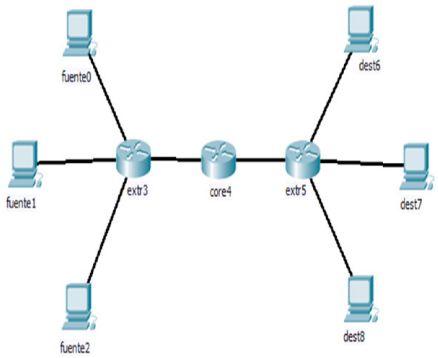

Fig. 1. Topología de Red a implementar.

Se observa una topología básica (ya que lo que se pretende es estudiar el comportamiento de las tecnologías en redes pequeñas) que cuenta con 3 host emisores (fuente0, fuente1, fuente2) y 3 receptores (dest6, dest7, dest8), además de 2 nodos de borde (extr3, extr5) y un enrutador de núcleo (core4).

El modelo de servicios diferenciados se puede implementar con las librerías que viene incluidas con el software [5], sin embargo, el modelo de servicios integrados

requiere de la instalación de librerías adicionales mostradas en [6].

2.1 Modelo "Best effort"

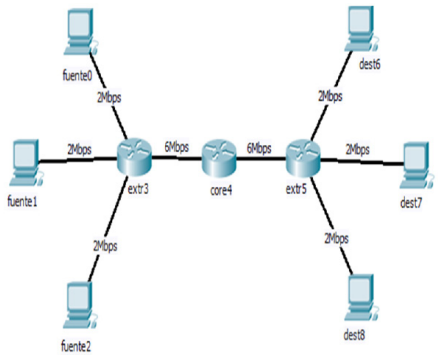

Fig. 2. Topología de red que se implementará con modelo IDEAL, en el que no se presentan pérdidas de paquetes.

A continuación se describirá el script en lenguaje Tcl, con el que se logró implementar la topología propuesta en la Fig. 2.

Los comandos básicos de un script Tcl son explicados con claridad en la documentación encontrada en línea [7], razón por la que se hará énfasis en los comandos que diferencien el comportamiento entre uno y otro modelo.

Se definen los enlaces entre los nodos, su capacidad de canal, el retardo asociado a cada medio de transmisión y el tipo de cola. En este caso la cola seleccionada es DropTail, en la que los paquetes que lleguen después de haberse superado el tamaño máximo de encolamiento se desechan.

```
$ns duplex-link $fuente0 $extr3 2Mb 10ms DropTail
$ns duplex-link $fuente1 $extr3 2Mb 10ms DropTail
$ns duplex-link $fuente2 $extr3 2Mb 10ms DropTail
$ns duplex-link $extr3 $core4 6Mb 10ms DropTail
$ns duplex-link $core4 $extr5 6Mb 10ms DropTail
$ns duplex-link $dest6 $extr5 2Mb 10ms DropTail
$ns duplex-link $dest7 $extr5 2Mb 10ms DropTail
$ns duplex-link $dest8 $extr5 2Mb 10ms DropTail
```

Como el interés es observar lo que sucede con los paquetes durante el proceso de encolamiento es necesario monitorear el comportamiento de la cola en un punto determinado. Para nuestro caso se monitoreará el comportamiento en el enrutador de núcleo (core_router4).

```
$ns duplex-link-op $core4 $extr5 queuePos 0.5
```

A continuación se crean las Fuentes que generan el tráfico y se asocian a los agentes udp0, udp1 y udp2.

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.002
$cbr0 attach-agent $udp0
```

La fuente0 es una fuente de tasa constante de bits (CBR), con una tasa de 2Mbps.

```
set pareto1 [new Application/Traffic/Pareto]
$pareto1 set packetSize_ 500
$pareto1 set burst_time_ 1ms
$pareto1 set idle_time_ 1ms
$pareto1 set rate_ 2000k
$pareto1 attach-agent $udp1
```

La fuente1 está asociada a un generador de tráfico de tipo Pareto. Pareto On/Off (POO) genera tráfico de acuerdo a una distribución On/Off. Los paquetes son enviados a cierto valor de tasa durante los periodos de encendido (burst_time) y durante los periodos de apagado (idle_ time) no se envían paquetes.

```
set exp2 [new Application/Traffic/Exponential]
$exp2 set packetSize_ 500
$exp2 set burst_time_ 2ms
$exp2 set idle_time_ 2ms
$exp2 set rate_ 2000k
$exp2 attach-agent $udp2
```

La fuente2 se asocia con un generador de tráfico exponencial. Éste transmite durante 2ms y deja de transmitir otros 2ms.

Con el fin de obtener una estadística de las variables de interés se implementa el agente LossMonitor, que entrega información de bytes recibidos y paquetes perdidos.

```
set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]

$ns attach-agent $dest6 $sink0
$ns attach-agent $dest7 $sink1
$nsattach-agent $dest8 $sink2
```

A fin de almacenar las variables estimadas se crea un proceso que almacena los datos en archivos con extensión .tr que posteriormente serán abiertos con XGRAPH. Estos archivos son datos organizados en dos columnas, de manera que en la primera columna se almacena el tiempo y en la segunda la variable de interés.

Lo último que realiza el script es ejecutar NAM y XGRAPH. Los parámetros con los que se inician son los archivos .tr que previamente se han guardado.

2.2 Modelo "Best effort"

Se presentará a continuación la parte del script TCL con el que se implementa el modelo de servicios diferenciados sobre NS-2. La topología de red es la mostrada en la Fig. 3.

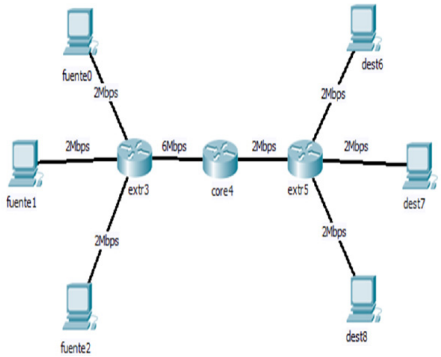

Fig. 3. Topología de red que se implementará con modelo Diffserv, para dar calidad de servicio mediante prioridad de paquetes.

Puesto que algunos comandos, como el de creación de los nodos, son similares en la implementación de los modelos, se mostrarán las variaciones relevantes y se obviarán los que se han presentado previamente.

Los enlaces entre los nodos se crean, pero entre los enrutadores de borde y de núcleo las colas son de tipo RED.

```
$ns duplex-link $fuente0 $extr3 2Mb 10ms DropTail
$ns duplex-link $fuente1 $extr3 2Mb 10ms DropTail
$ns duplex-link $fuente2 $extr3 2Mb 10ms DropTail
$ns simplex-link $extr3 $core4 6Mb 10ms dsRED/
edge
$ns simplex-link $core4 $extr3 6Mb 10ms dsRED/
core
$ns simplex-link $core4 $extr5 2Mb 10ms dsRED/
core
$ns simplex-link $extr5 $core4 2Mb 10ms dsRED/
edge
$ns duplex-link $dest6 $extr5 2Mb 10ms DropTail
$ns duplex-link $dest7 $extr5 2Mb 10ms DropTail
$ns duplex-link $dest8 $extr5 2Mb 10ms DropTail
```

RED (RandomEarlyDetection) es un algoritmo activo de manejo de colas. Monitorea el tamaño promedio de la cola y desecha paquetes basado en probabilidades estadísticas. Si el búfer está casi vacío, todos los paquetes entrantes son aceptados. A medida que la cola crece, la probabilidad de rechazar un paquete entrante aumenta también. Cuando el búfer está lleno, la probabilidad ha alcanzado el valor de 1 y todos los paquetes entrantes son rechazados [8], [10].

Se crean las colas entre los enrutadores de borde y el de núcleo.

```
set cola_Ex1_core [[$ns link $extr3 $core4] queue]
set cola_Ex2_core [[$ns link $extr5 $core4] queue]
set cola_core_Ex1 [[$ns link $core4 $extr3] queue]
set cola_core_Ex2 [[$ns link $core4 $extr5] queue]
```

Las políticas de prioridades se configuran en los enrutadores de borde, que son los que marcan los paquetes con el Code Point. Una vez que los paquetes son marcados y salen de los enrutadores de borde el enrutador de núcleo simplemente lee el CP del paquete y lo enruta dependiendo de su prioridad.

A continuación se muestra el script que

realiza la configuración en uno de los enrutadores de borde. Además se configuran las prioridades que se le darán a los paquetes de las distintas fuentes de la red.

```
#se configuran parametros RED de Edge1 a Core:
1.$cola_Ex1_core meanPktSize 500 #Numero de paquetes
2.$cola_Ex1_core set numQueues_ 1 #Numero de colas fisicas
3.$cola_Ex1_core setNumPrec 3 #No. colas virt por cola fis
4.$cola_Ex1_core addPolicyEntry [$fuente0 id] [$dest6 id] TokenBucket 10 2000000500
5.$cola_Ex1_core addPolicyEntry [$fuente1 id] [$dest7 id] TokenBucket 20 2000000500
6.$cola_Ex1_core addPolicyEntry [$fuente2 id] [$dest8 id] TokenBucket 30 2000000500
7.$cola_Ex1_core addPolicerEntryTokenBucket 10 10
8.$cola_Ex1_core addPolicerEntryTokenBucket 20 20
9.$cola_Ex1_core addPolicerEntryTokenBucket 30 30
10.$cola_Ex1_core addPHBEntry 10 0 0
11.$cola_Ex1_core addPHBEntry 20 0 1
12.$cola_Ex1_core addPHBEntry 30 0 2
13.$cola_Ex1_core configQ 0 0 1 50 0.001
14.$cola_Ex1_core configQ 0 1 5 20 0.6
15.$cola_Ex1_coreconfigQ 0 2 10 20 0.9
```

En las líneas 1, 2, 3 del script anterior, se configura la cantidad de colas físicas y virtuales que se le darán a cada uno de los tipos de tráfico, es decir cada tráfico estará en la misma cola física pero tendrá su propia cola virtual. En las líneas 4, 5, 6 se configuran las políticas de entrada en donde a cada ruta entre fuente y destino se le asigna un codepoint (10, 20 o 30) y además se le distribuye parte de la cola física a su cola virtual, en esta primera parte entre en enrutador de borde (extr3) y el enrutador de núcleo (core4) como se manejan 6 Mb de capacidad de canal, se le asignan 2 Mb (2000000) a cada cola virtual, con un tamaño promedio de paquete de 500 bytes. Las líneas 7, 8, 9, asignan el codepoint a un número de cola virtual (en este caso el codepoint 10 a una cola virtual llamada de igual forma y asi con las otras dos). Ya en las líneas 10, 11, 12, alcodepoint 10 se le asigna a la cola física0 y cola virtual 0 (es decir primera cola virtual), el 20 a la misma cola física0 (ya que solo se utilizara una cola física) y cola virtual 1 y el 30 a la cola física0 y cola virtual 2.

Finalmente se configuran los parámetros para dar a conocer la prioridad.$cola_Ex1_core configQ 0 0 1 50 0.001 el primer dato posterior a configQ (0), indica la cola física, el siguiente indica la cola virtual (0), los dos siguientes indican paquetemínimo y máximo a priorizar en un instante de tiempo, y el último ítem indica la probabilidad de desechar un paquete que llegue a dicha cola virtual. Estos parámetros y la asignación de ancho de banda a cada cola virtual dependiendo de la capacidad en la cola física permiten definir la prioridad a los paquetes.

A continuación se observa el script donde se configuran los parámetros entre el core4 y el extr5 (Edge2).

```
#se configuran parametros RED de Edge2 a Core:
1.$cola_Ex1_core meanPktSize 500 #Numero de paquetes
2.$cola_Ex1_core set numQueues_ 1 #Numero de colas fisicas
3.$cola_Ex1_core setNumPrec 3 #No. colas virt por cola fis
4.$cola_Ex1_core addPolicyEntry [$fuente0 id] [$dest6 id] TokenBucket 10 1500000 500
5.$cola_Ex1_core addPolicyEntry [$fuente1 id] [$dest7 id] TokenBucket 20 300000 500
6.$cola_Ex1_core addPolicyEntry [$fuente2 id] [$dest8 id] TokenBucket 30 200000 500
7.$cola_Ex1_core addPolicerEntryTokenBucket 10 10
8.$cola_Ex1_core addPolicerEntryTokenBucket 20 20
9.$cola_Ex1_core addPolicerEntryTokenBucket 30 30
10.$cola_Ex1_core addPHBEntry 10 0 0
11.$cola_Ex1_core addPHBEntry 20 0 1
12.$cola_Ex1_core addPHBEntry 30 0 2
13.$cola_Ex1_core configQ 0 0 1 50 0.001
```

```
14.$cola_Ex1_core configQ 0 1 5 20 0.6
15.$cola_Ex1_core configQ 0 2 10 20 0.9
```

Este código script es similar al anterior, debido a que los parámetros deben ser los mismos para que cada enrutador identifique correctamente a que cola virtual pertenece cada paquete y que prioridad tiene. Esta parte difiere, en las líneas 4, 5, 6, debido a que el canal entre el core4 y el extr5 posee tan solo 2 Mb de capacidad y por ende se le asigna a la cola virtual correspondiente a los paquetes de mayor prioridad (en este caso marcada con codepoint 10 cola virtual 0) 1.5 Mb para utilizar en el envio de dichos paquetes y a las otras dos colas se les asigna 300 Kb (cola virtual 1 codepoint 20) y 200 Kb (cola virtual 2 codepoint 30), para así completar un total de 2 Mb que existen en la cosa física.

2.3 Topología con servicios integrados

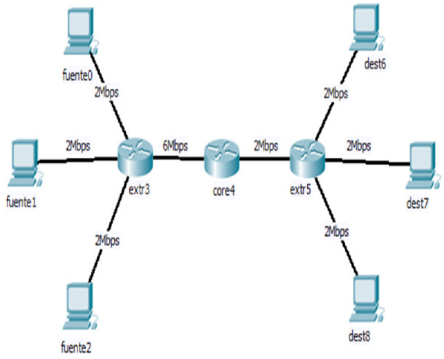

Fig. 4. Topología de red que se implementará con modelo IntServ para dar calidad de servicio mediante reservación de recursos.

Se presenta a continuación el script que implementa el modelo de servicios integrados sobre NS-2. Se hará énfasis en los comandos que se diferencian de manera relevante de los scripts anteriores.

La conexión entre los nodos tiene el siguiente esquema:

```
ns duplex-rsvp-link <node_1><node_2><bw><dela
y><reservable><rsvp><queue><adc><est>
```

Donde:

- ns: es una instancia del simulador
- bw: indica el ancho de banda del enlace
- delay: representa el retardo del enlace
- reservable: determina la cantidad de ancho de banda disponible para ser reservada por RSVP.
- rsvp: representa el ancho de banda (bps) reservado para los mensajes de control RSVP. Si este valor está en cero todos los mensajes de control se envían como paquetes "mejor esfuerzo".
- queue: es el tamaño de la cola en bytes asignada para servir a los paquetes "mejor esfuerzo".
- adc: Determina el algoritmo de control de admisión utilizado.

Actualmente existen 5 algoritmos de control de admisión implementados [9], se nombran a seguir:

- Algoritmo de "Suma Simple" basado en parámetros (Param)
- Suma medida (MS)
- Límites de Hoeffding (HB)
- Región tangente de aceptación en el origen (ACTO)
- Región tangente de aceptación en el pico (ACTP)

Los enlaces entre los host y los enrutadores se crean utilizando un proceso cuyos parámetros variables son el nodo fuente, el nodo destino y el ancho de banda del enlace.

```
proccrea_link {nodo_fuentenodo_dest tasa} {
global ns
setretardo 10ms
setreservable1.0
settasa_rsvp 0.0
settam_cola_be 50000
$nsduplex-rsvp-link $nodo_fuente $nodo_dest $tasa
$retardo $reservable $tasa_rsvp $tam_cola_bePa-
ramNull;}
```

Se invoca el proceso "crea_link" que crea los enlaces.

```
crea_link $fuente0 $extr3 2Mb
```

Con el fin de establecer las sesiones de servicios integrados en todos los nodos se deben crear agentes del tipo RSVP.
RSVP (Resource Reservation Protocol) es un protocolo de la capa de transporte diseñado para reservar recursos a través de una red para servicios integrados.

```
set rsvp0 [$fuente0 add-rsvp-agent]
```

Es necesario crear un flujo y una sesión para los datos a los que se requiere reservar ancho de banda.
Se crea la sesión para los datos que se enviarán desde la fuente0 hasta el destino 6. El agente rsvp6 hace la solicitud de reserva de 1.4 Mbps. La fuente0 está asociada a una fuente de tráfico CBR que tiene un tamaño de paquete de 500 bytes y un intervalo de envío de 3 ms. Siendo así la tasa de transmisión es de 1.33 Mbps por lo que resulta apropiada una reservación de 1.4 Mbps.

```
set flowid0 1
setsesionrsvp [$rsvp0 session $dest6 $flowid0]
$ns at 0.1 "$rsvp0 sender $sesionrsvp +1400000
5000 32"
$ns at 0.15 "$rsvp6 reserve $sesionrsvpff +1400000
5000 $fuente0"
```

3. SIMULACIONES Y RESULTADOS DE LOS MODELOS EN NS-2

Seguidamente se presentan los resultados obtenidos para cada modelo. Se obtuvieron gráficas y tablas que se analizarán para cada caso.

3.1 Modelo "Best effort"

El modelo inicial se basa en la idea de ofrecer toda la capacidad de canal para transmitir todo el tráfico generado. Es por esto que no se presentan pérdidas de paquetes ni encolamientos, sin embargo, variables como el jitter y el delay sí tienen ciertas variaciones.

A continuación se presentan las figuras relacionadas con las variables de interés y se hace una breve descripción de lo observado.

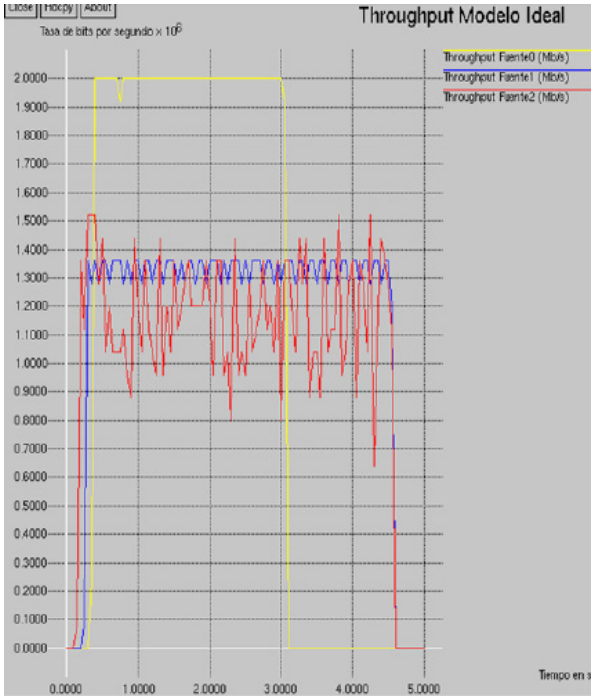

Fig. 5. Throughput para topología de modelo Best Effort.

Se puede apreciar que todos los tráficos utilizan la disponibilidad que requieren del canal. El tráfico CBR utiliza 2 Mbps, el Pareto 1.3 Mbps en promedio y el exponencial varía entre 0.7 y 1.5 Mbps. Las variaciones observadas se presentan por el tipo de tráfico.

Como es de esperarse, por el hecho de tener garantizado todo el canal requerido por los tráficos, no se presentan pérdidas de paquetes para ninguno de ellos en ningún instante de tiempo, esto se observa en la Fig. 6.

En la Fig. 7, 8 y 9 se observa el jitter para el modelo de mejor esfuerzo (ideal ya que se garantiza ancho de banda suficiente para cada tráfico). A pesar de tener una topología de red ideal, se presenta jitter debido al tipo de tráfico que se está simulando. Para el tráfico CBR el jitter es cero dado que su tasa de bits es constante. Sin embargo, para

los flujos 2 y 3 asociados a las fuentes de tráfico Pareto y Exponencial respectivamente, se presenta jitter causado por la variación en las tasas de bits transmitidas.

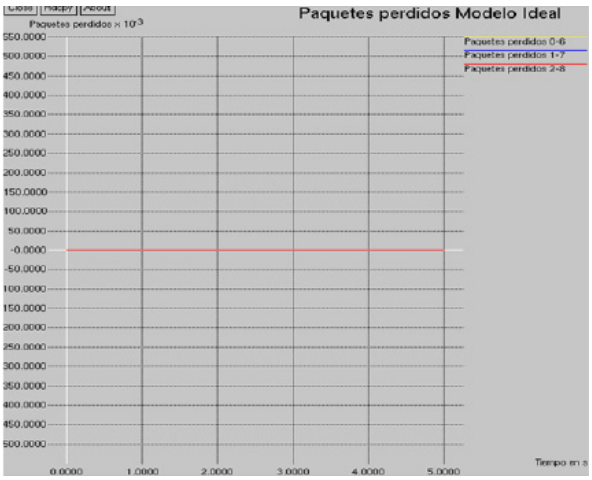

Fig. 6. Paquetes perdidos para topología con Modelo Best Effort

Fig. 7. Gráfica de jitter para cada uno de los flujos con Modelo Best Effort.

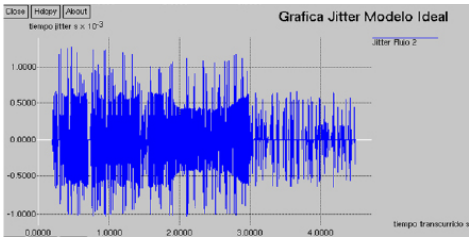

Fig. 8. Gráfica de jitter para cada uno de los flujos con Modelo Best Effort.

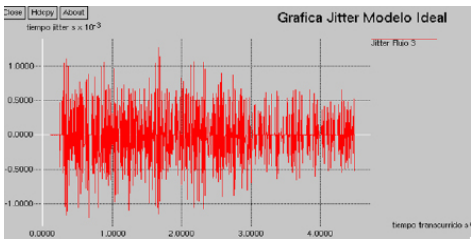

Fig. 9. Gráfica de jitter para cada uno de los flujos con Modelo Best Effort.

En la Fig. 8 se aprecia el delay para cada uno de los flujos analizados. Al igual que en las gráficas de jitter, la principal causa de las variaciones del delay es el tipo de tráfico implementado. En el caso del flujo 1, que es originado por la fuente de tráfico CBR el delay alcanza un valor constante de 46ms, de los cuales 40ms se atribuyen al retardo de la conexión entre nodo y nodo (10ms X 4).

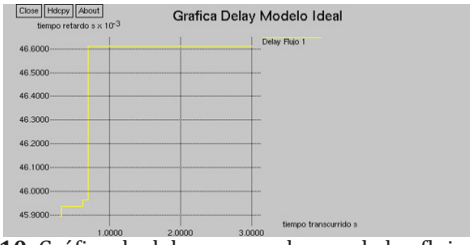

Fig. 10. Gráfica de delay para cada uno de los flujos con topología Best Effort.

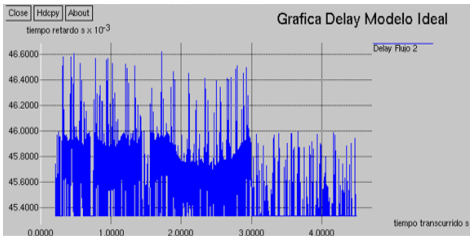

Fig. 11. Gráfica de delay para cada uno de los flujos con topología Best Effort.

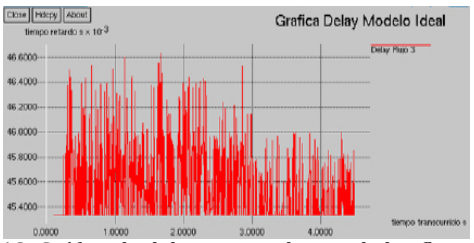

Fig. 12. Gráfica de delay para cada uno de los flujos con topología Best Effort.

3.2 Topología con Servicios Diferenciados

En la Fig. 13 se observa el throughput de cada uno de los tipos de tráfico para el modelo de servicios diferenciados. En el tiempo 0.1s inicia a transmitir la fuente2 asociada al tráfico exponencial.

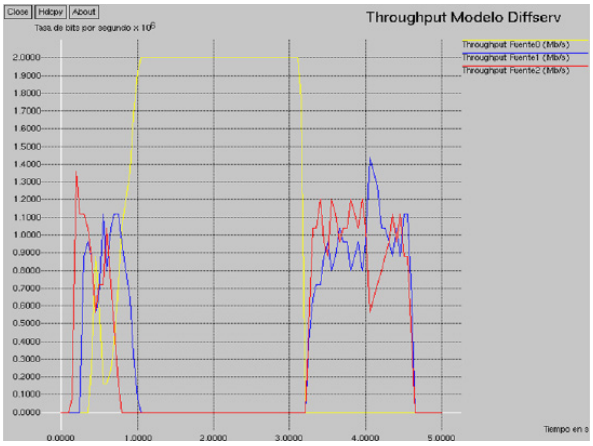

Fig. 13. Throughput para topología con modelo Diffserv

Este tráfico empieza a ocupar disponibilidad del canal alcanzando un máximo de 1.35 Mbps en el instante en que inicia a trasmitir la fuente1, asociada al trafico Pareto, esto ocurre a 0.2s. Estos dos tráficos empiezan a compartir la disponibilidad del canal y la cantidad ocupada por la fuente2 disminuye (promedio ancho de banda de 0.9 Mbps). En el instante 0.3s cuando la fuente0 asociada al trafico CBR empieza a transmitir, la disponibilidad del canal tanto para la fuente1 como para la fuente2 comienza a disminuir. Mientras transcurre más tiempo la fuente0 asignada con paquetes de mayor prioridad empieza a tomar la mayor parte de la disponibilidad del canal (alcanzando 2 Mbps), dejando a las fuente1 y fuente2 sin ancho de banda disponible para su transmisión de paquetes. Dicho suceso ocurre a 1s y concluye a los 3s cuando la fuente0 deja de transmitir paquetes. Luego de los 3s la fuente1 y fuente2 empiezan de nuevo una competencia por disponibilidad del canal (con un promedio de 1 Mbps).

En la Fig. 14 se observan los paquetes que pierden las fuentes de tráfico a través del tiempo en el que transcurre la simulación.

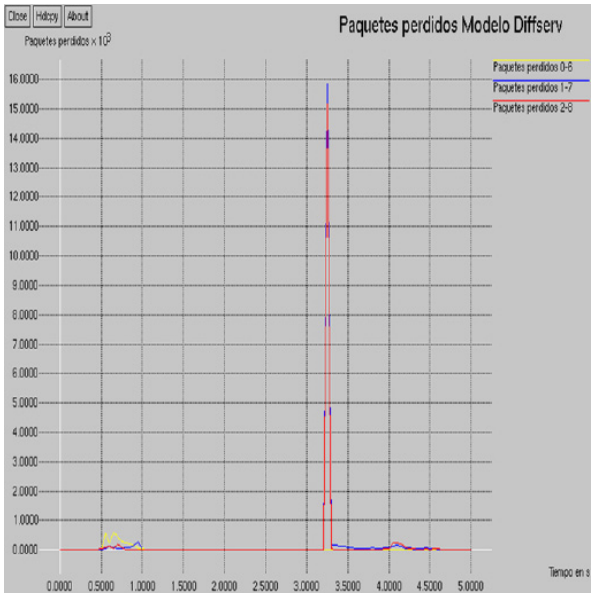

Fig. 14. Paquetes perdidos en la topología con modelo Diffserv.

En los instantes menores a 0.5s ninguna de las fuentes ha perdido paquetes, pero se conoce que debido a las características de disponibilidad del canal, los paquetes se encolaran, hasta un punto tal que comiencen a descartarse paquetes y se pierdan durante la transmisión.

Debido a que la fuente2 comienza un pequeño instante de tiempo antes que la fuente1, en el instante 0.5s se observa que la fuente1 pierde un poco más de paquetes que la fuente2, y esto es debido a que la cola está siendo ocupada mayormente por la fuente2 y cada paquete que llegue de esta misma fuente o de la fuente1 serán desechados. En ese mismo instante (0.5s) se ve que la fuente que pierde más paquetes es la fuente0, debido a que inicia a transmitir en último lugar, la cola se encuentra completamente llena y debe esperar a que poco a poco se liberen espacios y pueda tener lugar para sus paquetes. Una vez que la fuente0 empieza a hacer uso de toda la disponibilidad del ca-

nal las fuentes1 y 2 pierden todos los paquetes enviados sin embargo en la gráfica se nota esto hasta que haya disponibilidad del canal nuevamente. Es decir, el agente que permite calcular los paquetes perdidos compara el último paquete que llega con el inmediatamente anterior, como el canal estuvo saturado durante casi 2 segundos la cantidad de paquetes perdidos se calcula cuando llega el primer paquete después de que la fuente0 deja de transmitir. Esto se ve reflejado en el pico que se observa en la Fig. 14 en el tiempo 3.3s.

En la Fig. 15 se observa el jitter de la fuente0 asociada al tráfico CBR.

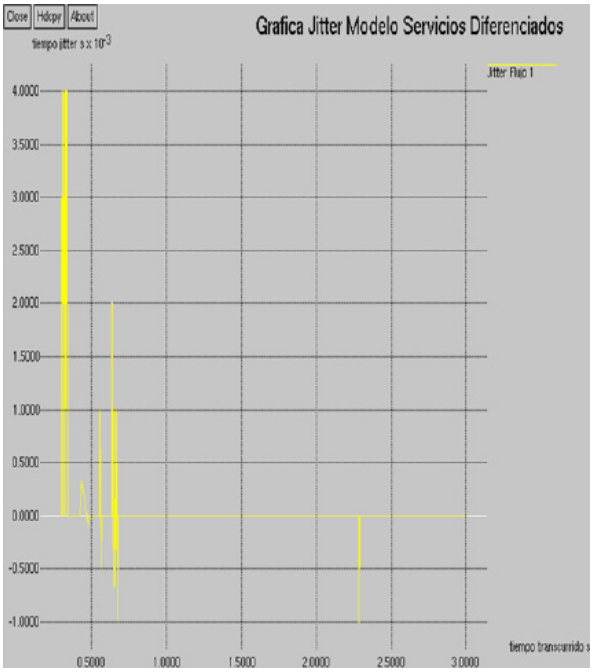

Fig. 15. Jitter de la fuente0 en la topología con modelo Diffserv.

En el instante 0.3s se observa que el jitter es de 4ms debido a que como en ese instante de tiempo la cola se encuentra ocupada por las fuentes1 y 2 se presenta un diferencial de retardo de los paquetes por encolamiento. Una vez se empieza a dar prioridad a los paquetes de la fuente0, su jitter empieza a disminuir, sufriendo pequeñas variaciones mientras se establece la prioridad por completo.

En la Fig. 16 se observa el jitter de la fuente1 asociada al tráfico Pareto. Los paquetes de la fuente1 poseen desde un poco después de su inicio de transmisión un jitter elevado, debido a que la fuente2 (0.1s) que inició su transmisión antes que la fuente1 (0.2s), está ocupando gran parte de la cola, haciendo algunos de los paquetes de la fuente1 sean descartados y que los que llegan a su destino tengan un diferencial de retardo.

Desde el tiempo 0.7s hasta 3.2s pareciese que los paquetes de la fuente1 no poseen jitter, es decir llegan en orden y no se descartan, pero esto no es así, en este intervalo de tiempo lo que realmente sucede es que debido a la prioridad de paquetes asignada a la fuente0, los paquetes de la fuente1, son descartados y no llegan a su destino, por lo que no se puede medir el jitter si no hay paquetes con los cuales comparar. Cuando la fuente0 deja de transmitir datos y le es permitido a la fuente1 llevar algunos de sus paquetes al destino, se observa que el jitter aumenta y se mantiene variando pues el canal se comparte con la fuente2.

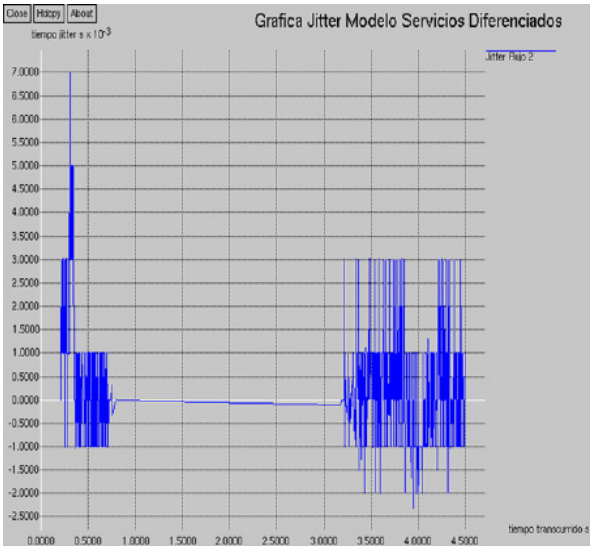

Fig. 16. Jitter de la fuente1 en la topología con modelo Diffserv

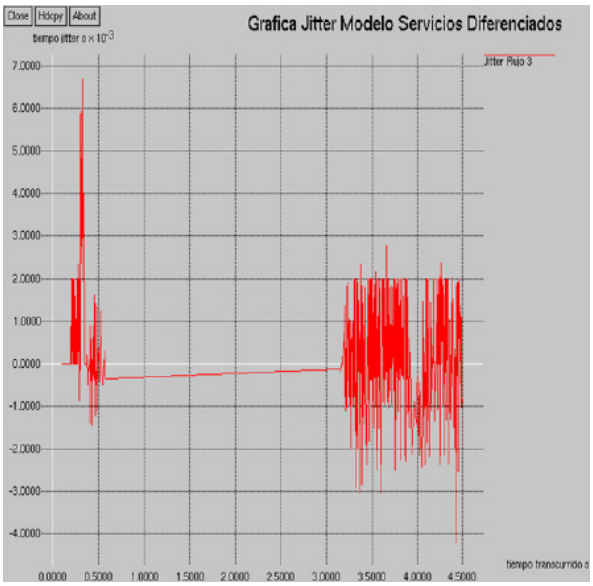

Fig. 17. Gráfica Jitter Flujo 3 con Modelo de Servicios Diferenciados Implementado.

El jitter para el flujo3 (Fig. 17) tiene un comportamiento similar al del flujo2. Antes de que inicie la transmisión de la fuente0 se presentan variaciones en el jitter debido al encolamiento. Una vez que empieza a transmitir la fuente0, ocupa toda la disponibilidad del canal y el siguiente paquete de la fuente2 llega cuando el canal tiene disponibilidad nuevamente. Este intervalo va entre un poco más de 0.5s y 3.2s.

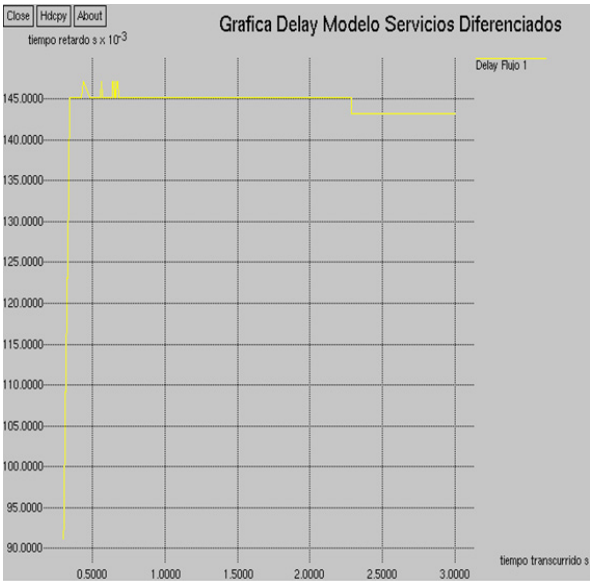

Fig. 18. Gráfica Delay Flujo 1 para Modelo de Servicios Diferenciados.

Debido a que la fuente0 ocupa toda la disponibilidad del canal presenta un retardo a causa del encolamiento. Sin embargo ese retardo es constante en casi todo el transcurso de la simulación (Fig. 18).

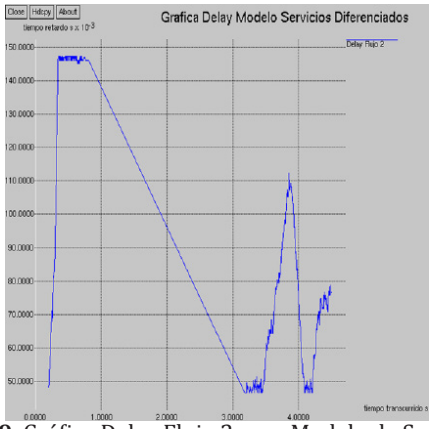

Fig. 19. Gráfica Delay Flujo 2 para Modelo de Servicios Diferenciados.

Antes de 1s, el flujo2 (Fig. 19) presenta un delay cercano a 150ms debido al encolamiento de paquetes del flujo3 y del flujo2. Una vez que el canal se satura con el flujo1, dejan de llegar paquetes del flujo2 al destino, razón por la cual no se tiene información de retardo sino hasta que llega el siguiente paquete después de que el flujo 1 se detiene después de 3s. En ese tiempo es cuando el retardo se reduce a menos de 50ms, pero empieza a aumentar conforme empiezan a encolarse paquetes de los flujos 2 y 3.

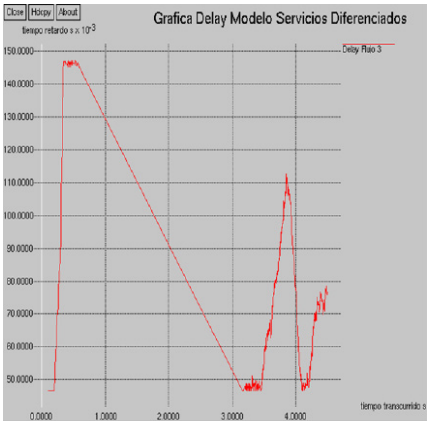

Fig. 20. Gráfica Delay del Flujo3 para Modelo de Servicios Diferenciados.

Para el flujo 3 (Fig. 20) el comportamiento del delay es similar al del flujo 2. Alcanzando el valor más alto de 145ms en 0.5s y el mínimo de 50ms en 3.2s.

En la tabla 1, 2, 3 se muestran los puntos de código asociados a cada tráfico además del total de paquetes transmitidos.

El flujo 1 que es el que tiene la más alta prioridad está asociado al codepoint 10. En la misma tabla también se presentan los paquetes "ldrops" que hacen referencia a los paquetes perdidos por encolamiento y los flujos "edrops" que representan los "early-drops" o paquetes eliminados tempranamente debido a mecanismos RED implementados.

En coherencia con lo que se ha presentado hasta el momento se puede apreciar que el flujo 1 no tiene pérdida de paquetes asociada con los mecanismos implementados en servicios diferenciados. Después de 1s el flujo 1 ni siquiera presenta pérdida por encolamiento, pues todo el canal está disponible para él.

Tabla 1. Estadísticas de paquetes en 1 segundo.

CP	TotPkts	TxPkts	ldrops	edrops
All	879	468	323	88
10	339	194	145	0
20	259	149	53	57
30	281	125	125	31

Tabla 2. Estadísticas de paquetes en 2 segundos.

CP	TotPkts	TxPkts	ldrops	edrops
All	2003	968	947	88
10	839	694	145	0
20	592	149	386	57
30	572	125	416	31

Tabla 3. Estadísticas de paquetes en 3 segundos.

CP	TotPkts	TxPkts	ldrops	edrops
All	3120	1467	1564	89
10	1339	1193	145	1
20	925	149	719	57
30	856	125	700	31

3.3 Topología con Servicios Diferenciados

En la Fig. 21 se aprecia el comportamiento del tráfico cuando se implementan servicios integrados. En este caso la conexión tiene un reservable para RSVP del 100% y la sesión asociada al flujo 1 solicita reservar la totalidad del reservable para su tráfico. Es decir, que todo el tráfico de la sesión 1 es transmitido a través del canal que se ha reservado previamente.

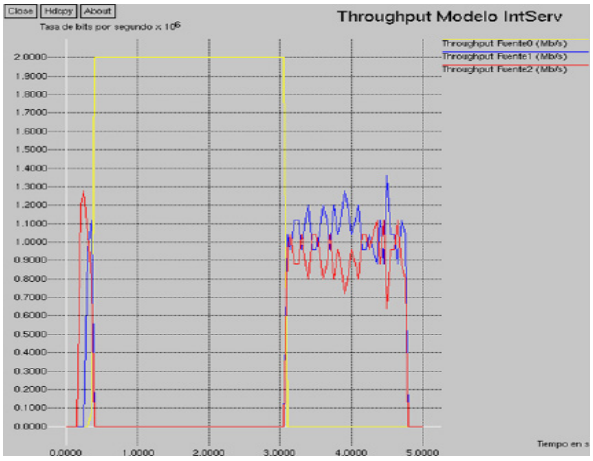

Fig. 21. Throughput para modelo de servicios integrados

En al Fig. 22 se observa el resultado a nivel de paquetes perdidos en Servicios integrados.

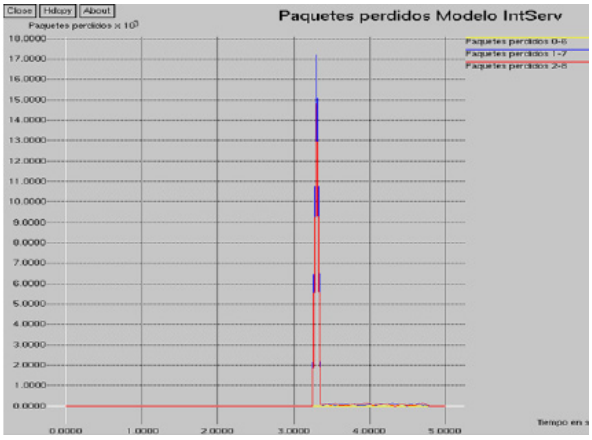

Fig. 22. Gráfica de paquetes perdidos modelo de servicios integrados.

Inicialmente y hasta que se termina la

transmisión del flujo 1 pareciera que no hay pérdida de paquetes. Sin embargo, lo sucedido muestra que justo al final de la transmisión del flujo 1, cuando se reciben paquetes del flujo 2 y 3, se registran los paquetes perdidos. En ese punto el máximo registrado es de 17.000 paquetes.

En la Fig. 23, 24, 25 el jitter máximo se presenta en 0.3s debido a que la cola ya se encuentra ocupada. Sin embargo en adelante el valor de jitter es 0, para cualquiera de los flujos mencionados (ideal, diffserv, intserv).

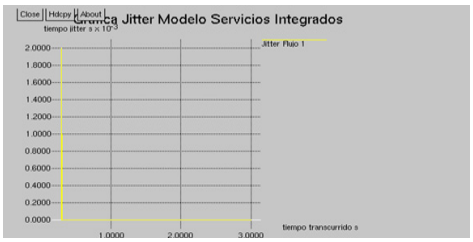

Fig. 23. Gráfica Jitter de Modelo Servicios Integrados

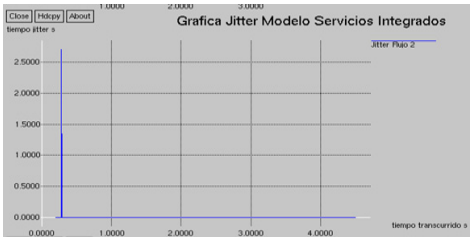

Fig. 24. Gráfica Jitter de Modelo Servicios Integrados

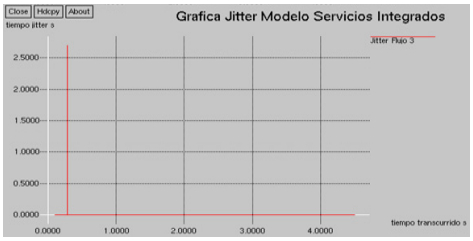

Fig. 25. Gráfica Jitter de Modelo Servicios Integrados

En la Fig. 26, 27, 28 se muestra el retardo de 49ms para el flujo 1 y de 2.7s para el flujo 2 y 3. Este fenómeno se presenta porque las fuentes 1 y 2 no pueden transmitir mientras el canal esté ocupado con tráfico CBR de la fuente0, luego, una vez que se detenga el flujo 1 se empiezan a recibir paquetes de los flujos 2 y 3 y es allí cuando se sigue guardando registro de retardo.

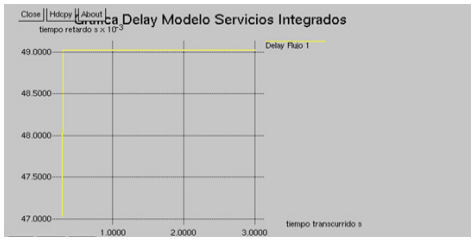

Fig. 26. Gráfica Delay Modelo Servicios Integrados.

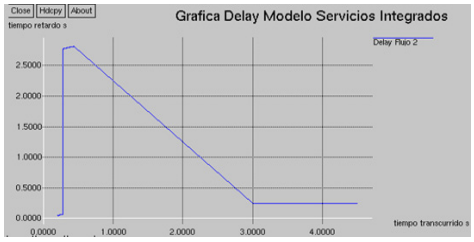

Fig. 27. Gráfica Delay Modelo Servicios Integrados.

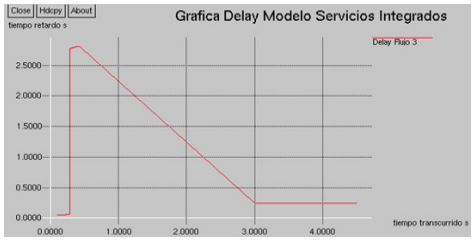

Fig. 28. Gráfica Delay Modelo Servicios Integrados.

4. ANÁLISIS DE RESULTADOS

A partir de las gráficas se han obtenido tablas comparativas de cada una de las variables de interés para cada flujo y cada modelo analizado.
Se debe recordar que los flujos están asociados de la siguiente manera:

Flujo 1: Fuente0 >> dest6 [cbr]
Flujo 2: Fuente1 >> dest7 [pareto]
Flujo 3: Fuente 2 >> dest8 [exponencial]

Tabla 4. Paquetes perdidos.

CASO	FLUJO 1	FLUJO 2	FLUJO 3
BEST EFFORT	0	0	0
DIFFSERV	146	957	849
INTSERV	0	954	830

Debido al modelo propuesto por servicios integrados el flujo 1 que es el asociado a la sesión que solicita RESERVA no presenta ni una pérdida de paquetes. Esto se da porque toda la capacidad del canal está reservada al flujo 1 que está asociado a la sesión que solicita la reserva.

Tabla 5. Throughtput promedio.

CASO	FLUJO 1	FLUJO 2	FLUJO 3
BEST EFFORT	2Mbps	1.35Mbps	1.15Mbps
DIFFSERV	2Mbps	0.272Mbps	0.272Mbps
INTSERV	2Mbps	0.506Mbps	0.396Mbps

En cualquiera de los 3 modelos el flujo con mayor prioridad es el flujo 1. Sin embargo, se puede considerar el throughput para otros flujos, con lo que IntServ tiene la ventaja sobre Diffserv pues no sólo garantiza el throughput para el flujo 1 sino que los flujos 2 y 3 tienen mayor throughput.

Tabla 6. Jitter máximo.

CASO	FLUJO 1	FLUJO 2	FLUJO 3
BEST EFFORT	600ms	1.2ms	1.2ms
DIFFSERV	4ms	7ms	6.8ms
INTSERV	2ms	2.7s	2.7s

IntServ presenta una desventaja sobre Diffserv y radica en que si el flujo ocupa todo el canal disponible, como el caso que se analizó, el jitter de los flujos 2 y 3 se vuelve muy grande. En este sentido Diffserv tiene ventaja sobre Intserv, pues el jitter tiene un valor promedio de 6ms para los tres flujos.

Tabla 7. Delay máximo.

	FLUJO 1	FLUJO 2	FLUJO 3
IDEAL	46.6ms	46.6ms	46.6ms
DIFFSERV	145ms	145ms	145ms
INTSERV	49ms	2.8s	2.8s

Similar a lo que sucede con el jitter, el delay también toma valores bastante grandes con IntServ cuando un flujo ocupa todo el canal disponible. Ventaja que toma DiffServ sobre IntServ.

Conocido el funcionamiento tanto de DiffServ como de IntServ es difícil decir cuál de los dos modelos es mejor.

Se debe analizar desde el punto de vista de la aplicación o servicio que hará uso del modelo. Por ejemplo, si se habla de la red de un banco para conectar sus cajeros automáticos es posible que se prefiera utilizar IntServ que a pesar de tener un retardo mayor la pérdida de paquetes es muy baja, lo que evitaría errores en las transacciones.
Pero si lo que queremos es utilizar varias aplicaciones con diferentes tipos de fuentes de tráfico es posible que se prefiera usar DiffServ.

5. CONCLUSIONES

Una topología de red sin QoS, presenta muchos problemas en cuanto al envío de paquetes, debido a que no garantiza que la información llegue completa y la tasa de pérdidas en un canal que tenga baja disponibilidad, seria grandísima y no permitiría un envío de datos confiable en tiempo real.

La topología de red con QoS utilizando el modelo Diffserv, presenta problemas en cuanto a que cada nodo perteneciente a la red debe conocer los codepoints para dar la prioridad a los paquetes y que para cada administrador de red habrá cierta información que es de mayor prioridad que otra, esto quiere decir que establecer un estándar para la prioridad no es un resultado inmediato.

A diferencia de Intserv, Diffserv no necesita de enviar una “reservación” de disponibilidad de canal a los nodos que pertenecen a la red o mejor a los nodos que pertenecen a la trayectoria necesaria para enviar el paquete al destino, sino que al establecer las prioridades, hace que la disponibilidad del canal sea dinámica y varíe para cada paquete.

La principal desventaja de IntServ es que muchos estados deben almacenarse en cada enrutador dependiendo de las sesiones que se requieran iniciar, por lo tanto puede ser eficiente a pequeña escala pero a una escala como la de internet es muy difícil mantener el registro de todas las reservaciones necesarias.

Referencias Bibliográficas

[1] U. De Jaén. (2011, Abril) Calidad De Servicio En Redes Ip. Trasparencias,[en línea]. Consultado en Marzo 20 de 2010, disponible en: Http://Www4.Ujaen.Es/ Jccuevas/ Data/Aattaa2/ Transparencias/

[2] Tiphon 22td047 Problems With The Behavior Of Jitter Buffers And Their Influence On The End-To-End Speech Quality, Source Kpn Research, March 2001.

[3] M. Baldi, Y. Ofek, End-To-End Delay Analysis Of Videoconferencing Over Packet-Switched Networks. Ieee/Acm Trans. On Networking, 8(4):479.492, Aug. 2000.

[4] Hierarchy of classes,[en línea]. Consultado en Septiembre 10 de 2010, disponible en: http://www.planete.inria.fr/software/ns-doc/ns-current/LossMonitor.html

[5] The Network Simulator, [en línea]. Consultado en Agosto 11 de 2010, disponible en: http://Www.Isi.Edu/Nsnam/Ns/

[6] M. Greis. Rsvp/Ns: An Implementation Of Rsvp For The Network Simulator Ns-2. Rsvp/Ns Documentation, 2000.

[7] Vint Group. Tutorial For Network Simulator Ns2,[en línea]. Consultado en Abril de 2010, disponible en: http://www.Isi.Edu/Nsnam/Ns/Tutorial/Examples/Template.Tcl

[8] Floyd, Sally; Jacobson, Van (August 1993). "Random Early Detection (Red) Gateways For Congestion Avoidance". Ieee/Acm Transactions On Networking 1 (4): 397–413. Doi:10.1109/90.251892,2008.

[9] S. Jamin, S. J. Shenker, P. B. Danzig, “Comparison Of Measurement-Based Admission Control Algorithms For Controlled-Load Service", Proc. Ieee-Infocom 97, 19 97.

[10] X. Zhou, J. Wei, And C. Xu, "Quality-Of-Service Differentiation On The Internet: A Taxonomy", Presented At J. Network And Computer Applications, Pp.354-383, 2007.