IMPACTO DEL MECANISMO DE PREEMPTION EN REDES MPLS

ANEXOS

MARÍA DEL MAR IBARRA VIVAS

MARIO IVAN LÓPEZ MORA

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELECOMUNICACIONES
GRUPO DE NUEVAS TECNOLOGÍAS EN TELECOMUNICACIONES
GESTIÓN INTEGRADA DE REDES, SERVICIOS Y ARQUITECTURAS DE TELECOMUNICACIONES
POPAYÁN
2009

IMPACTO DEL MECANISMO DE PREEMPTION EN REDES MPLS
ANEXOS

MARÍA DEL MAR IBARRA VIVAS
MARIO IVAN LÓPEZ MORA

Trabajo de grado para optar por el título de
Ingeniero en Electrónica y Telecomunicaciones

Director
Ing. OSCAR J. CALDERON C.

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELECOMUNICACIONES
GRUPO DE NUEVAS TECNOLOGÍAS EN TELECOMUNICACIONES
GESTIÓN INTEGRADA DE REDES, SERVICIOS Y ARQUITECTURAS DE TELECOMUNICACIONES
POPAYÁN
2009

CONTENIDO

ANEXO A – HERRAMIENTA DE SIMULACIÓN NETWORK SIMULATOR (NS-2)

PARTE 1. DESCRIPCION DE NS2

Ns es un simulador de redes orientado a eventos desarrollado por la Universidad de Berkeley que simula una gran variedad de redes. Tiene implementado protocolos de red como TCP y UDP, el tráfico de fuente se puede comportar como FTP, Telnet, Web, CBR y VBR, el mecanismo de encolamiento usado en los enrutadores es DropTail, RED Y CBQ, usa algoritmos de enrutamiento como Dijkstra, y más. NS también implementa algunos de los protocolos de la capa MAC para simulaciones de LAN´s. Actualmente el proyecto NS forma parte del proyecto VINT. NS versión 2 esta escrito en C++ y OTcl (Lenguaje con Scripts TCL orientado a objetos).

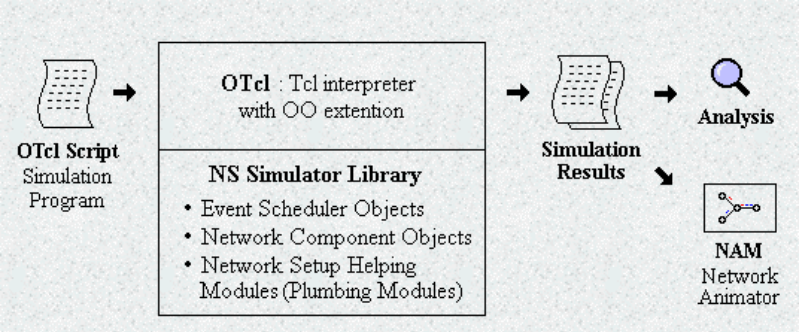


Figura 1. Vista de NS para el usuario.

La figura 1 muestra como ve el usuario al simulador NS, donde NS es un intérprete de script Tcl orientado a objetos (OTcl) que tiene un planificador de eventos de simulación y librerías de objetos que representan los componentes de la red. Para usar NS, se programa en el lenguaje de script OTcl, para establecer y correr una simulación de red, se escribe un script OTcl que inicie el planificador de eventos, se crea la topología de red usando los objetos de red, y se define cuando se inicia y se termina la transmisión de paquetes por parte de las fuentes.

Por razones de eficiencia NS no solo esta escrito en OTcl sino también en C++. Porque se necesita separar la trayectoria de las implementaciones de los datos de la trayectoria de las implementaciones de control. Para reducir el tiempo de procesamiento de los eventos y de los paquetes (no se refiere al tiempo de simulación), los objetos que representan los componentes básicos de la red y el planificador de eventos son escritos y compilados

usando C++. Estos objetos compilados están diseñados de tal manera que el interprete OTcl cree un objeto OTcl por cada uno de los objetos en C++, y se hace que las funciones de control y las variables de configuración especificadas por el objeto C++ se asocien con las variables del objeto OTcl correspondiente. De esta forma se le da a OTcl el control de los objetos C++. También es posible agregar funciones y variables a C++ enlazados con objetos OTcl. Los objetos en C++ que no son controlados en una simulación no necesitan ser enlazados con OTcl. La figura 2 muestra un ejemplo de jerarquía de objetos en C++ y OTcl. Se puede observar que los objetos C++ tienen su similitud OTcl donde se forma una jerarquía muy similar a la de C++.

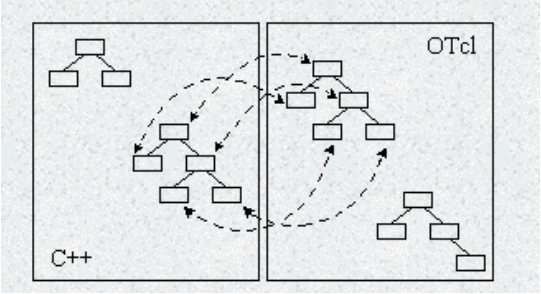


Figura 2. La dualidad C++ y OTcl

La figura 3 muestra la arquitectura general de NS. En ella un usuario general (no un desarrollador) se ubica en la esquina inferior izquierda, diseñando y corriendo simulaciones en Tcl usando los objetos simuladores en la librería OTcl. El planificador de eventos y la mayoría de los componentes de red son implementados en C++ y disponible para OTcl a través de un enlace OTcl que se implementa usando tclcl. Todo esta interacción crea en conjunto a NS, el cual es un interprete Tcl extendido orientado a objetos equipado con librerías de simulador de red.

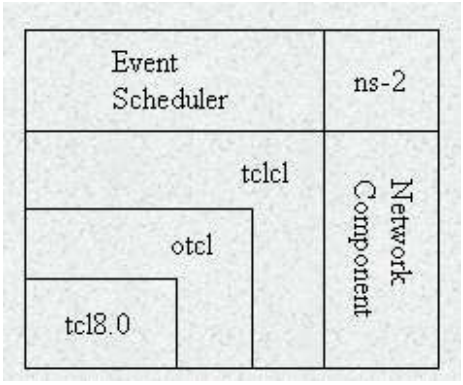


Figura 3. Arquitectura de NS

Para obtener los resultados de la simulación se debe especificar en el script la generación de archivos de traza de la simulación, para que una vez terminada la simulación NS produzca uno o más archivos de salida de texto que contienen datos detallados de la simulación. Los datos pueden ser usados para análisis de la simulación o servir de entrada a la herramienta de simulación grafica de NS llamada NAM que ha sido desarrollada como parte del proyecto VINT.

La figura 4 muestra al simulador gráfico NAM que presenta una interfaz de usuario muy amigable, con botones para iniciar, adelantar, retroceder o parar la animación de la simulación ya realizada también se cuenta con la posibilidad de controlar la velocidad a la que se despliega la animación.

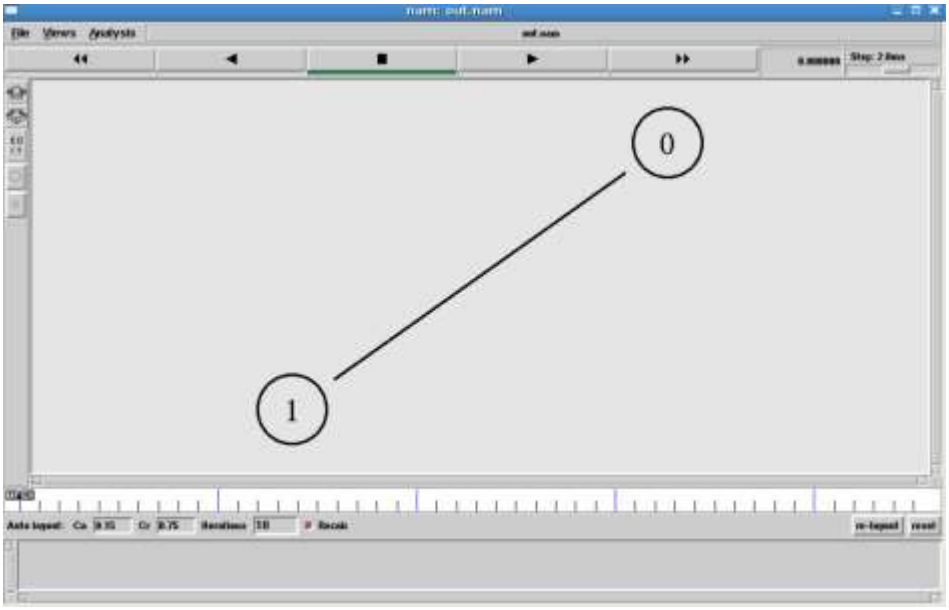


Figura 4. Animador gráfico NAM.

PARTE 2. MNS (MPLS NETWORK SIMULATOR)

MNS (MPLS network Simulator – simulador de redes MPLS) fue implementado realizando extensiones a NS (Network Simulator). El principal propósito de su creación fue desarrollar un simulador que tenga la capacidad de simular varias aplicaciones MPLS sin la necesidad de construir una red MPLS real.

MNS soporta el establecimiento de CR-LSP (Constraint-based routing LSP – LSP que cumple con restricciones) para tráfico con QoS, también funciones básicas de MPLS como LDP y conmutación de etiquetas. Para poder implementar estas características, MNS tiene varios componentes; CR-LDP, clasificador de MPLS, clasificador de servicios, control de admisión, gestor de recursos, y planificador de paquetes.

Consideraciones para el diseño de MNS

Para el diseño de MNS se tuvo en cuenta las siguientes consideraciones:

Extensibilidad: Sigue el paradigma orientado a objetos para poder soportar varias de las aplicaciones de MPLS.
Usabilidad: Esta diseñado para que el usuario lo pueda usar fácilmente.
Portabilidad: Reducir las modificaciones al código de NS para que el modulo no se encuentre atado a una versión específica de NS.
Reusabilidad: Se desarrollo para que opere como un switch real.

MNS en su versión 2.0 soporta las siguientes funciones de MPLS:

- Conmutación de etiquetas: se habilitan operaciones con etiquetas de intercambio (swapping) y apilamiento (apilamiento) de etiquetas, decremento del TTL,
- Protocolo LDP: Se manejan los mensajes LDP (Request, Mapping, Withdraw, Relegase, y Notification)
- Protocolo CR-LDP: Para este protocolo se manejan los mensajes CR-LDP (Request y Mapping)
- Agregación de Flujo: La agregación de flujos muy pequeños sobre un flujo mas robusto.

Las siguientes son las capacidades que brinda MNS respecto al establecimiento de LSPs:

- En el esquema de distribución y asignación: soporta los esquemas de downstream on demand y upstream on demand.
- ER-LSP basada en CR-LDP: se establece el LSP sobre información de la ruta predefinida por el usuario.
- CR-LSP basado en CR-LDP: el LSP se establece en base a parámetros como tasa del tráfico, tamaño del buffer.
- Preemption de recursos: se posibilita la acción de apropiarse de los recursos de los CR-LSP existentes teniendo en cuenta la prioridad de establecimiento y la prioridad de retención.
- Enrutamiento basado en restricciones: Consiste en obtener la ruta para un CR-LSP utilizando información acerca de los recursos que hay en la red para luego establecerlo por esta ruta.
- Se pueden crear rutas explicitas basadas en restricciones para el CR-LSP

ARQUITECTURA DEL NODO MPLS

MNS se constituye en una extensión de NS que es un simulador basado en IP. En NS un nodo consiste de clasificadores y agentes. Un agente es un objeto de protocolo emisor – receptor, y un clasificador es el objeto responsable de la clasificación de paquetes necesaria para el envío a el siguiente nodo. Con el propósito de realizar un nuevo nodo MPLS de un nodo IP, el 'MPLS Classifier' y el 'LDP agent' son insertados dentro del nodo IP.

La figura 5 muestra la arquitectura de un nodo MPLS dentro de MNS. 'Nodo de entrada' retorna un punto de entrada para el nodo. Este es el primer elemento que maneja la llegada de paquetes al nodo. La variable de instancia de nodo 'entry_', guarda la referencia para este elemento. Cuando el multicasting no es soportado la variable debe estar referenciada a el 'MPLS Classifier', el cual primero clasifica los paquetes entrantes en etiquetados y no etiquetados. Los paquetes no etiquetados se tratan de una manera convencional. Cuando los paquetes están etiquetados el 'MPLS Classifier' es responsable de realizar el intercambio de etiquetas. El 'Addr Classifier' es responsable del envío de paquetes basado en las direcciones IP de destino y finalmente el 'Port Classifier' se encarga de la selección del agente.

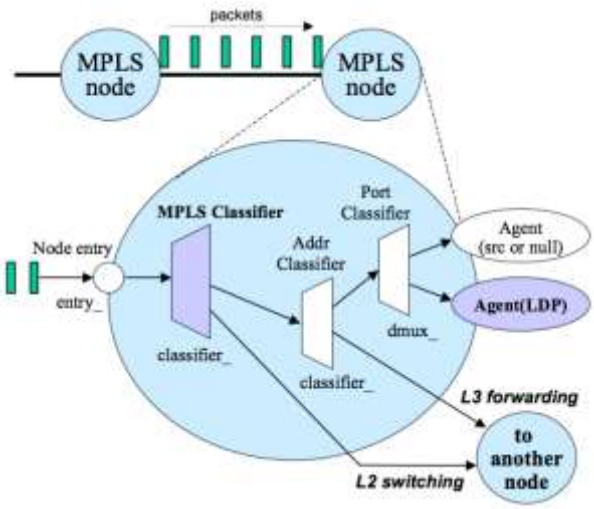


Figura 5. Arquitectura del nodo MPLS.

APIs PARA CREAR UNA RED MPLS CON MNS

A continuación se muestran algunas de las APIs definidas para crear una red MPLS:

MPLSnode: crea un nuevo nodo MPLS
configure-ldp-on-all-mpls-nodes: asigna agentes LDP para todos los nodos MPLS.
enable-on-demand: permite operar a los LSRs en el modo de asignación de etiquetas bajo demanda.
make-explicict-route: establecer un ER-LSP
flow-erlsp-binding: enlaza un flujo sobre un ER-LSP establecido.
flow-aggregation: se agregan flujos pequeños dentro de un gran flujo.
MPLSnode setup-crlsp $fec $lspid $TRate $BSize $SPrio $HPrio: Se crea un CR-LSP especificando una fec, una ruta explicita y un identificador del LSP definido. Con este comando, un mensaje de petición de Request CR-LDP se envía hacia el nodo con fec definida por la ruta explicita (er) determinada para crear un LSP explicito basado en restricciones.
Para este comando los parámetros tienen el siguiente significado

TRate representa la tasa de bit.
BSize se refiere al tamaño del Buffer.
SPrio es el valor de la prioridad de establecimiento.
HPrio es el valor de la prioridad de retención.

ANEXO B – INSTALACIÓN DE NS-2 Y EL MÓDULO MNS EN LINUX

Pasos:

1. Instalar debian etch.

Modificación del archivo sources.list.

2. Desde un terminal del root ubicarse en el directorio /etc/apt/ y editar el archivo sources.list:

```
#cd /etc/apt
/etc/apt# gedit sources.list
```

El archivo debe quedar así:

```
# deb cdrom:[Debian GNU/Linux 4.0 r0 _Etch_ - Official i386 CD Binary-1
20070407-11:55]/ etch contrib main

#deb cdrom:[Debian GNU/Linux 4.0 r0 _Etch_ - Official i386 CD Binary-1
20070407-11:55]/ etch contrib main

# Line commented out by installer because it failed to verify:
deb http://security.debian.org/ etch/updates main contrib
# Line commented out by installer because it failed to verify:
deb-src http://security.debian.org/ etch/updates main contrib

deb http://mirrors.kernel.org/debian/ stable main
deb-src http://mirrors.kernel.org/debian/ stable main
deb http://mirrors.kernel.org/debian/ stable non-free
```

Descarga de NS y MNS

3. Descargar los siguientes archivos:

   - ns-allinone-2.26.tar.gz
   - mns-for-2.26.tar.gz
   - ns-2.26-rsvp.patch.gz

4. Ubicarse en el directorio en donde se instalará NS (en este caso /usr/local/) y desempaquetarlo :

```
usr/local# tar -xzvf ns-allinone-2.26.tar.gz
```

Esto creará el directorio /ns-allinone-2.26/

Instalación de las librerías necesarias

5. Exportar el proxy:

```
# export http_proxy="http://proxy.unicauca.edu.co:3128"
```

6. Instalar las librerías:

```
apt-get install libstdc++2.10-glibc2.2
apt-get install libx11-dev
apt-get install libxt-dev
apt-get install xspecs
apt-get install libxmu-dev
apt-get install xlibs-static-dev
```

Instalación y configuración del compilador de C

7. La versión del compilador de C que viene con debian etch es la 4.1. Dado que el parche de MNS fue compilado con una versión más antigua, es necesario instalar esa versión:

```
apt-get install gcc-2.95
apt-get install g++-2.95
```

8. Ubicarse en /usr/local/ns-allinone-2.26/. La siguiente línea aparece cuatro veces en tres archivos de configuración:

```
system=MP-RAS-`awk '{print }' /etc/.relid'`
```

Borrar el penúltimo caracter (') de tal manera que la línea quede así:

```
system=MP-RAS-`awk '{print }' /etc/.relid`
```

Este paso debe hacerse para los siguientes 3 archivos:

./tcl8.3.2/unix/configure
./tk8.3.2/unix/configure
./otcl-1.0a8/configure

En el primer archivo la línea aparece dos veces.

9. Desde el terminal en el que vaya a hacerse la instalación de NS, modificar las variables de entorno para que apunten al compilador gcc-2.95:

```
export CC=/usr/bin/gcc-2.95;
export CPP=/usr/bin/cpp-2.95;
export CXX=/usr/bin/g++-2.95;
```

10. Editar el archivo /usr/local/dload/ns-allinone-2.26/tclcl-1.0b13/Makefile.in. En este archivo aparece la siguiente línea:

```
CCOPT = @V_CCOPT@
```

Cambiarla por ésta:

```
CCOPT = -fpermissive @V_CCOPT@
```

Aplicar el parche para MNS y RSVP-TE

11. Copiar mns-for-2.26.tar.gz dentro del directorio /ns-allinone-2.26 y descomprimirlo:

```
#tar –xzvf mns-for-2.26.tar.gz
```

12. Cambiar el nombre del directorio /ns-allinone-2.26/ns-2.26 por ns-allinone-2.26/ns-2.26-orig
13. Crear un directorio ns-allinone-2.26/ns-2.26-mod
14. Copiar el contenido de ns-allinone-2.26/ns-2.26-orig a ns-allinone-2.26/ns-2.26-mod
15. dentro de /ns-allinone-2.26/ copiar el archivo ns-2.26-rsvp.patch.gz y desempaquetarlo:

```
#gunzip ns-2.26-rsvp.patch.gz
```

16. Aplicar el parche:

```
#patch -p0< ns-2.26-rsvp.patch
```

17. Cambiar el nombre de ns-allinone-2.26/ns-2.26-mod a ns-allinone-2.26/ns-2.26
18. Dentro de /ns-allinone-2.26/

```
#./install
```

ANEXO C - COMPORTAMIENTO Y RESULTADOS PARA EL CASO DE ESTUDIO 3

Comportamiento escenario 1: sin preemption

Con el establecimiento de los 9 LSPs el ancho de banda libre en el enlace es de 1064Kb, en consecuencia para la prueba 1 y 2 cuando el requerimiento de ancho de banda del tráfico de prueba es de 512 Kb y 1024 Kb respectivamente, el LSP de prueba se puede crear utilizando el ancho de banda libre del enlace.

Desde la prueba número 3 en adelante el ancho de banda libre no es suficiente para satisfacer la petición del nuevo LSP, entonces, el control de admisión envía un mensaje NOTIFICATION donde se indica que no existen recursos disponibles y el LSP no se establece.

Comportamiento escenario 2: con preemption

De nuevo en este caso de estudio para las primeras dos pruebas el nuevo LSP (de datos) se establece satisfactoriamente. En la prueba 3 el ancho de banda del LSP de prueba es de 1536 Kb con lo que se supera el ancho de banda disponible en el enlace y como la prioridad del tráfico de prueba es baja, este no podrá *apropiar* a ninguno de los LSPs en curso en la red así que el control de admisión rechaza la petición generando pérdidas en el tráfico de datos.

Esta situación la podemos observar en la figura 6 que muestra el throughput para los diferentes LSPs con respecto al tiempo para la prueba tres, donde se identifica que el LSP 10 de color verde, correspondiente al LSP de prueba (con tráfico de datos) no se puede establecer y su throughput es de 0 Kbps.

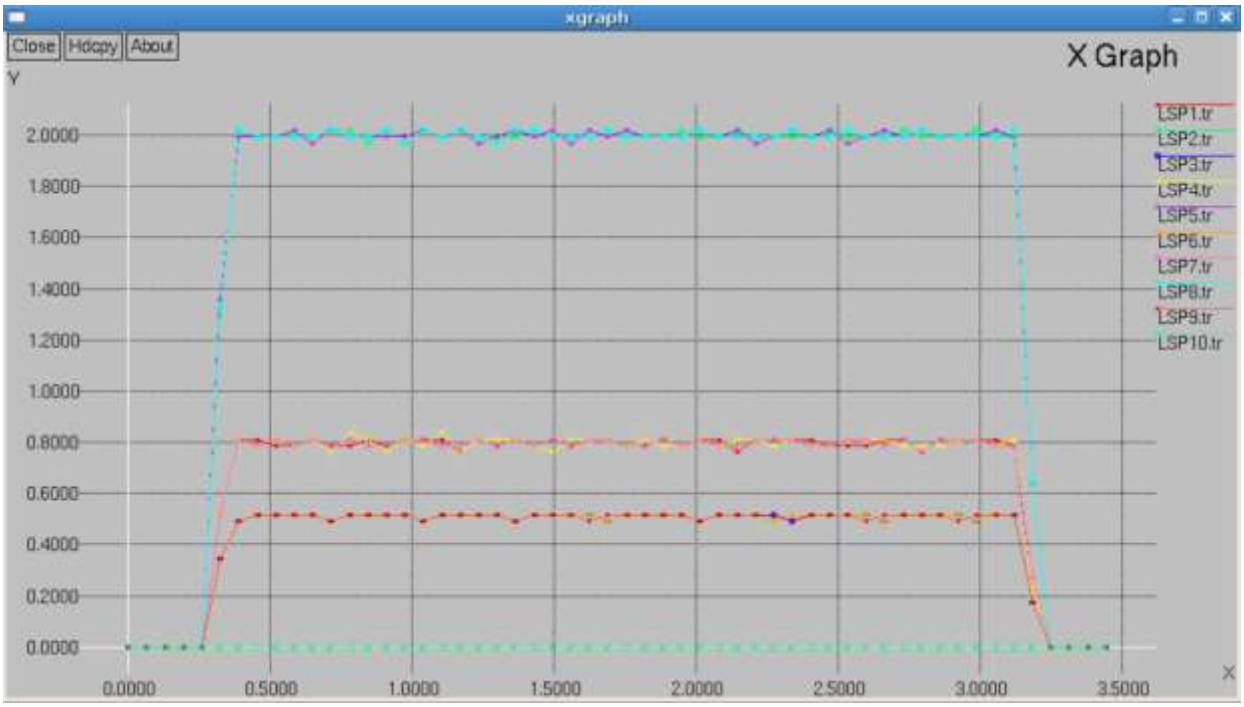


Figura 6 Throughput de los LSPs del enlace en la prueba 3**.**

Para las pruebas posteriores: de la 4 hasta la 14 se presentará la misma situación de la prueba tres, ya que el tráfico de datos no puede realizar *apropiación* de los recursos utilizados por los diferentes LSPs de datos video y de voz ya establecidos en la red.

Los LSPs *apropiados* en cada una de las pruebas para este caso de estudio se muestran en la tabla 1

Caso de estudio 3				
Escenario 2				
LSPs *apropiados*				
Prueba	BW LSP de prueba (Kb)	BW que es necesario *apropiar* * (Kb)	Cantidad de LSPs *apropiados*	Prioridad de los LSPs *apropiados*
1	512	0	0	--
2	1024	0	0	--
3	1536	472	0	--
4	2048	984	0	--
5	2560	1496	0	--
6	3072	2008	0	--
7	3584	2520	0	--
8	4096	3032	0	--
9	4608	3544	0	--
10	5120	4056	0	--
11	5632	4568	0	--
12	6144	5080	0	--
13	6656	5592	0	--
14	7168	6104	0	--

** BW LSP de prueba – BW libre*

Tabla 1 LSPs *apropiados* en el caso de estudio 3

Evaluación del desempeño

Se realizaron las medidas de throughput, perdida de paquetes, jitter, retardo para ambos escenarios para evaluar el desempeño de la red MPLS con y sin preemption.

Evaluación del desempeño para los tres tipos de tráfico

En las tablas 2 y 3 se muestran las medidas de los parámetros de desempeño para el tráfico de datos en las 14 pruebas realizadas.

Caso de estudio 3					
Escenario 1: sin preemption					
Prueba	BW del LSP de prueba	**Medidas del desempeño para el tráfico de datos**			
		Throughput promedio (Kbps)	% Pérdida de paquetes	Retardo (ms)	Jitter (ms)
1	512	2834,75	0	41,03	0,49
2	1024	3304,27	0	41,15	0,51
3	1536	2365,24	37,32	40,96	0,44
4	2048	2365,24	44,25	40,96	0,44
5	2560	2365,24	49,80	40,96	0,44
6	3072	2365,24	54,35	40,96	0,44
7	3584	2365,24	58,13	40,96	0,44
8	4096	2365,24	61,29	40,96	0,44
9	4608	2365,24	63,97	40,96	0,44
10	5120	2365,24	66,38	40,96	0,44
11	5632	2365,24	68,41	40,96	0,44
12	6144	2365,24	70,28	40,96	0,44
13	6656	2365,24	72,06	40,96	0,44
14	7168	2365,24	73,41	40,96	0,44

Tabla 2 Desempeño para el tráfico de datos en el escenario sin preemption

Caso de estudio 3					
Escenario 2: con preemption					
Prueba	BW del LSP de prueba	**Medidas del desempeño para el tráfico de datos**			
		Throughput promedio (Kbps)	% Pérdida de paquetes	Retardo (ms)	Jitter (ms)
1	512	2833,65	0,01	41,86	0,7
2	1024	3289,37	0,43	42,71	0,84
3	1536	2365,24	37,32	41,58	0,813
4	2048	2365,24	44,25	41,58	0,813
5	2560	2365,24	49,8	41,58	0,813
6	3072	2365,24	54,35	41,58	0,813
7	3584	2365,24	58,13	41,58	0,813
8	4096	2365,24	61,29	41,58	0,813
9	4608	2365,24	63,97	41,58	0,813
10	5120	2365,24	66,38	41,58	0,813
11	5632	2365,24	68,41	41,58	0,813
12	6144	2365,24	70,28	41,58	0,813
13	6656	2365,24	72,06	41,58	0,813
14	7168	2365,24	73,41	41,58	0,813

Tabla 3 Desempeño para el tráfico de datos en el escenario con preemption

Las tablas 4 y 5 muestran la medida de los parámetros de desempeño para el tráfico de video en el escenario sin preemption y en el escenario con preemption en cada una de las 14 pruebas:

Caso de estudio 3					
Escenario 1: sin preemption					
Prueba	BW del LSP de prueba	**Medidas del desempeño para el tráfico de video**			
		Throughput promedio (Kbps)	% Pérdida de paquetes	Retardo (ms)	Jitter (ms)
1	512	5912,82	0,04	41,53	0,27
2	1024	5912,82	0,04	41,62	0,26
3	1536	5912,82	0,04	41,28	0,28
4	2048	5912,82	0,04	41,28	0,28
5	2560	5912,82	0,04	41,28	0,28
6	3072	5912,82	0,04	41,28	0,28
7	3584	5912,82	0,04	41,28	0,28
8	4096	5912,82	0,04	41,28	0,28
9	4608	5912,82	0,04	41,28	0,28
10	5120	5912,82	0,04	41,28	0,28
11	5632	5912,82	0,04	41,28	0,28
12	6144	5912,82	0,04	41,28	0,28
13	6656	5912,82	0,04	41,28	0,28
14	7168	5912,82	0,04	41,28	0,28

Tabla 4 Desempeño para el tráfico de video en el escenario sin preemption

Caso de estudio 3					
Escenario 2: con preemption					
Prueba	BW del LSP de prueba	**Medidas del desempeño para el tráfico de video**			
		Throughput promedio (Kbps)	% Pérdida de paquetes	Retardo (ms)	Jitter (ms)
1	512	5915,58	0	40,85	0,26
2	1024	5915,58	0	40,87	0,27
3	1536	5915,58	0	40,87	0,26
4	2048	5915,58	0	40,87	0,26
5	2560	5915,58	0	40,87	0,26
6	3072	5915,58	0	40,87	0,26
7	3584	5915,58	0	40,87	0,26
8	4096	5915,58	0	40,87	0,26
9	4608	5915,58	0	40,87	0,26
10	5120	5915,58	0	40,87	0,26
11	5632	5915,58	0	40,87	0,26
12	6144	5915,58	0	40,87	0,26
13	6656	5915,58	0	40,87	0,26
14	7168	5915,58	0	40,87	0,26

Tabla 5 Desempeño para el tráfico de video en el escenario con preemption

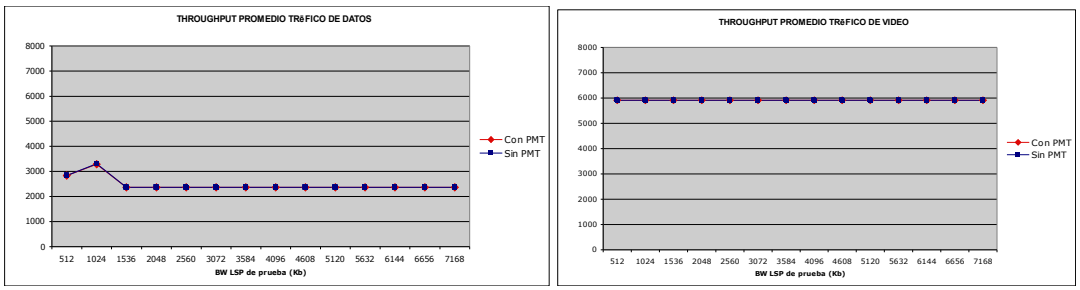
Las medidas de los parámetros de desempeño para el tráfico de voz en los escenarios sin y con preemption durante las 14 pruebas realizadas se muestran en las tablas 6 y 7

Caso de estudio 3					
Escenario 1: sin preemption					
Prueba	BW del LSP de prueba	**Medidas del desempeño para el tráfico de voz**			
		Throughput promedio (Kbps)	% Pérdida de paquetes	Retardo (ms)	Jitter (ms)
1	512	1514,48	0	41,14	0,56
2	1024	1514,48	0	41,19	0,55
3	1536	1514,48	0	40,99	0,47
4	2048	1514,48	0	40,99	0,47
5	2560	1514,48	0	40,99	0,47
6	3072	1514,48	0	40,99	0,47
7	3584	1514,48	0	40,99	0,47
8	4096	1514,48	0	40,99	0,47
9	4608	1514,48	0	40,99	0,47
10	5120	1514,48	0	40,99	0,47
11	5632	1514,48	0	40,99	0,47
12	6144	1514,48	0	40,99	0,47
13	6656	1514,48	0	40,99	0,47
14	7168	1514,48	0	40,99	0,47

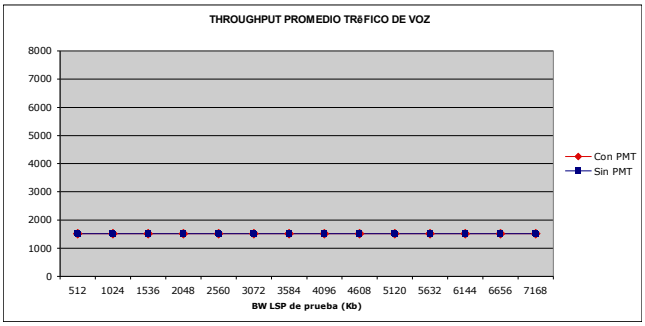
Tabla 6 Desempeño para el tráfico de voz en el escenario sin preemption

Caso de estudio 3					
Escenario 2: con preemption					
Prueba	BW del LSP de prueba	**Medidas del desempeño para el tráfico de voz**			
		Throughput promedio (Kbps)	% Pérdida de paquetes	Retardo (ms)	Jitter (ms)
1	512	1514,48	**0**	40,67	0,16
2	1024	1514,48	**0**	40,69	0,18
3	1536	1514,48	**0**	40,65	0,15
4	2048	1514,48	**0**	40,65	0,15
5	2560	1514,48	**0**	40,65	0,15
6	3072	1514,48	**0**	40,65	0,15
7	3584	1514,48	**0**	40,65	0,15
8	4096	1514,48	**0**	40,65	0,15
9	4608	1514,48	**0**	40,65	0,15
10	5120	1514,48	**0**	40,65	0,15
11	5632	1514,48	**0**	40,65	0,15
12	6144	1514,48	**0**	40,65	0,15
13	6656	1514,48	**0**	40,65	0,15
14	7168	1514,48	**0**	40,65	0,15

Tabla 7 Desempeño para el tráfico de voz en el escenario con preemption



a) b)



c)

Figura 7 Throughput promedio de cada tipo de tráfico. Caso de estudio 3

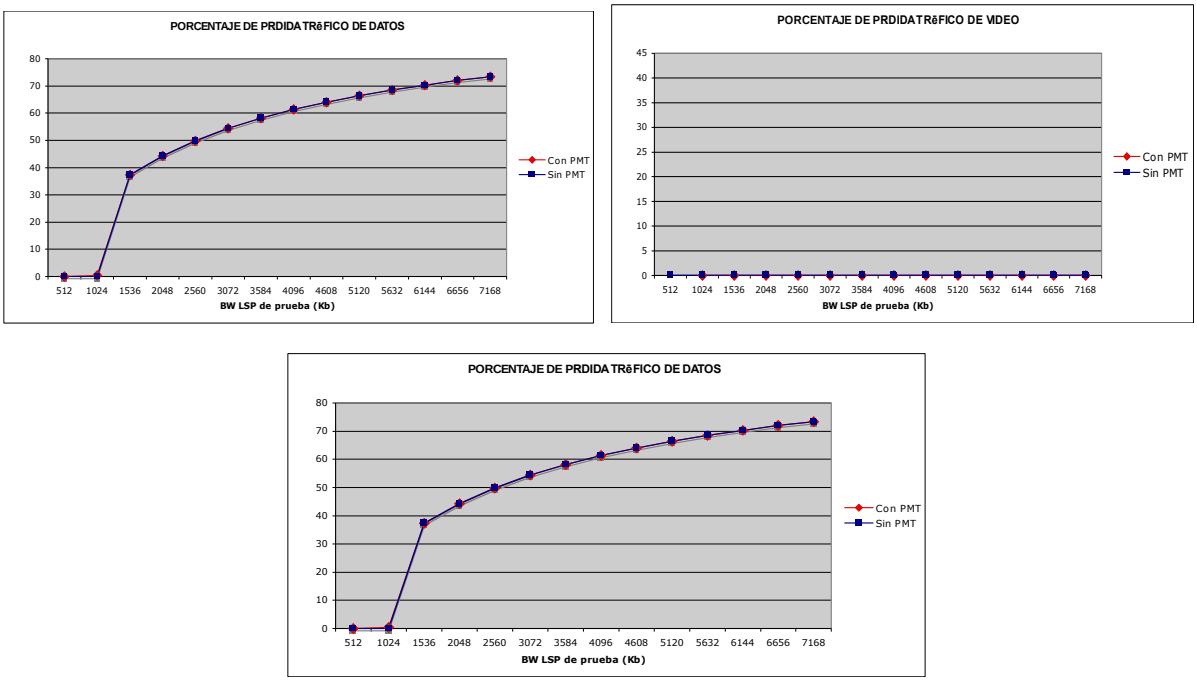


Figura 8 Porcentaje de perdida de paquetes de cada tipo de tráfico. Caso de estudio 3

Las tablas 2, 3, 4, 5, 6 y 7 indican las medidas de los parámetros de desempeño para los tres tipos de tráfico y las figuras 7 y 8 muestran el throughput y el porcentaje de perdida de paquetes para cada tipo de tráfico (datos, video y voz) en los escenarios con y sin preemption en el caso de estudio 3. Se observa que los parámetros como: throughput, porcentaje de perdida de paquetes, retardo y jitter para cada tipo de tráfico son iguales en el escenario sin y con preemption. Esto ocurre porque el LSP de prueba de baja prioridad (de datos) no puede *apropiarse* de los recursos que son usados por los LSPs establecidos en el enlace y el LSP es rechazado. Debido a que no se originan procesos de preemption, los parámetros de desempeño para el tráfico de video y de voz presentan los mismos valores en ambos escenarios.

Conclusiones para el caso de estudio 3

Ya que el comportamiento de la red en los dos escenarios es el mismo, el desempeño tanto de la red en general como de cada tipo de tráfico es también el mismo. El tráfico de prioridad baja se ve afectado en este escenario presentando pérdidas, retardo y jitter elevados pero esto sucede, no porque ocurran *apropiaciones*, sino porque el tráfico que llega a la red que es de prioridad baja simplemente se rechaza incrementando las pérdidas.