ANÁLISIS DE LA INCIDENCIA DE FALLAS MÚLTIPLES EN REDES MPLS. ANEXO.

WILLIAM GIRALDO SANDOVAL
RUBÉN DARÍO GUERRERO ENRÍQUEZ

Director: Ing. Oscar Josué Calderón Cortés

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
Grupo de Nuevas Tecnologías en Telecomunicaciones
Línea de Investigación: Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones
Popayán, Junio de 2009

ANEXO

INSTALACIÓN DEL SIMULADOR NS, DEL MÓDULO MNS (MPLS NETWORK SIMULATOR) Y DE LA HERRAMIENTA DE ANÁLISIS DE TRAZAS TRACEGRAPH.

PROCESO DE INSTALACION DE NS-2 EN LINUX.

Inicialmente se procede a descargar los archivos binarios ejecutables del simulador NS correspondientes a la versión 2.26, para lo cual se accede a la página http://www.isi.edu/nsnam/dist/ y se selecciona la versión allinone ns-allinone-2.26.tar.gz. Esta versión contiene en su interior las versiones pre-compiladas de los paquetes necesarios para la instalación del simulador, lo cual agiliza dicho proceso al no requerir la instalación de un número considerable de paquetes adicionales. Posteriormente se ingresa a una consola de comandos como usuario *root* y se digita el siguiente comando para descomprimir el paquete:

#tar -xzvf ns-allinone-2.26.tar.gz

Seguidamente, se requiere aplicar un parche de compatibilidad para el soporte de versiones recientes del compilador gcc sobre la carpeta donde se encuentran los archivos de instalación del simulador, puesto que la versión del simulador NS utilizada está diseñada para correr sin problemas usando el compilador gcc v2.95, pero la mayoría de distribuciones de Linux traen incorporadas versiones más recientes de este. Para aplicar el parche se deben seguir los siguientes pasos:

- Descargar el parche ns-2.26-gcc410.patch del enlace http://www.tekno.chalmers.se/~yusheng/projects.htm.
- Copiarlo en la misma ruta donde se encuentra la carpeta donde se descomprimió el instalador del simulador.
- Ingresar el comando *patch –p0 < ns-2.26-gcc410.patch*

Seguidamente se procede a instalar el simulador, para lo cual se ingresa a la carpeta de instalación y se teclea el comando *./install*. Para mayor información, leer el archivo README que se encuentra dentro de la carpeta base del instalador.

#cd ns-allinone-2.26
#./install

Al final del proceso de instalación, se indica un reporte sobre la instalación exitosa o no de cada uno de los paquetes que integran el simulador, además de unas indicaciones adicionales sobre la configuración de las variables de entorno necesarias para ejecutarlo de manera correcta. Para llevar a cabo dicha acción se ingresa al archivo *~/.bashrc* con el comando *gedit ~/.bashrc* y se ingresan las siguientes líneas.

 # LD_LIBRARY_PATH
OTCL_LIB=/ruta de instalación/ns-allinone-2.26/otcl-1.13
NS2_LIB=/ruta de instalación/ns-allinone-2.26/ ns-allinone-2.26/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$USR_LOCAL_LIB

 # TCL_LIBRARY
TCL_LIB=/ruta de instalación/ns-allinone-2.26/tcl8.4.14/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB

 # PATH
XGRAPH=/ruta de instalación/ns-allinone-2.26/bin:/ruta de instalación/ns-allinone-2.26/tcl8.4.14/unix:/ruta de instalación/ns-allinone-2.26/tk8.4.14/unix
NS=/ruta de instalación/ns-allinone-2.26/ns-2.31/
NAM=/ruta de instalación/ns-allinone-2.26/nam-1.13/
PATH=$PATH:$XGRAPH:$NS:$NAM

Finalmente, para hacer efectivas estas modificaciones, se ingresa el siguiente comando:

#source ~/.bashrc

Para comprobar si el simulador está bien instalado y las variables de entorno se configuraron correctamente, tecleamos el comando ns, después de lo cual un % deberá aparecer en la pantalla. Para validar la instalación después de los pasos realizados, se puede correr la suite de validación mediante los siguientes líneas:

 # cd /ns-allinone-2.26/ns-2.26
 # ./valídate

INSTALACIÓN DE MNS (MPLS NETWORK SIMULATOR).

Una vez se haya instalado el simulador NS-2.26 y comprobado su funcionamiento, se procede a instalar el módulo MNS. Para ello, se descarga el archivo desde la página http://heim.ifi.uio.no/~johanmp/ns-2/mns-for-2.26.tar.gz. Luego se copia el paquete dentro de la carpeta base de instalación de NS-2.26 y se descomprime mediante el siguiente comando:

#tar –zxvf mns-for-2.26.tar.gz

Seguidamente, se debe copiar el archivo ns-mpls-te-reroute.tcl (adjunto en la documentación entregada) dentro de la carpeta /ruta de instalación/ns-2.26/tcl/mns_v2.0/, reemplazando el que ya existía por este último. Finalmente se debe compilar y validar el módulo de simulación MPLS para su correcto funcionamiento. Para ello se siguen los siguientes pasos:

 # cd /ns-allinone-2.26/ns-2.26
 # make clean
 # ./configure
 # make

Finalmente se recomienda el uso de las distribuciones Fedora Core 4 y Debian Sarge 3.1, ya que ellas el simulador se desempeña de manera óptima. Sin embargo, también puede ejecutarse en distribuciones más modernas de estos sistemas operativos.

DESCRIPCION E INSTALACIÓN DEL SOFTWARE TRACEGRAPH PARA LINUX Y WINDOWS.

Tracegraph es una herramienta de software libre para el análisis de archivos de trazas desarrollada para el simulador NS-2, la cual complementa el proceso de estudio de diferentes parámetros y métricas de interés en las redes de comunicaciones simuladas, facilitando el análisis de los resultados entregados por el simulador que no cuenta con ninguna opción para llevar a cabo dicho proceso. Tracegraph fue implementado en Matlab 6.0, sin embargo, puede ejecutarse sin necesidad de tenerlo instalado. Esta herramienta provee a sus usuarios muchas opciones de análisis de resultados, incluyendo más de 250 gráficas y reportes estadísticos entre los que se encuentran el retardo, RTT (Round trip time), tiempo de procesamiento, jitter, gráficas de throughput, etc., y además permite almacenar los resultados obtenidos en archivos de texto y procesar sus scripts para llevar a cabo los cálculos de manera automática.

Esta herramienta puede soportar cualquier formato de archivo de traza si este se convierte al formato de trazas de NS-2 o al formato propio de Tracegraph. Soporta formatos de trazas de redes cableadas, redes inalámbricas, redes hibridas (cableadas e inalámbricas) y redes satelitales, incluyendo tanto trazas nuevas como de versiones anteriores. Además, es compatible con plataformas Windows, Linux, UNIX y MAC OS. Para conocer detalles acerca de cómo usar el simulador e interpretar sus resultados, el programa incluye un archivo de documentación. En la versión para Windows, este se encuentra en la carpeta C:\mgl\bin\win32\doc, donde C:\mgl\ es el directorio base donde se copian los archivos del programa.

Instrucciones de instalación.

Instalación en Linux.

1) Descargar el tracegraph y mglinstaler de la página web http://www.tracegraph.com/download.html. En dicha página se deben ingresar algunos datos de usuario, después de lo cual se selecciona la versión del simulador; aquí se escoge la opción *2.02 for Linux.* Finalmente aparece una página donde se pueden descargar los dos archivos mencionados.

2) Dentro del directorio donde se almacenaron los dos archivos descargados, ingresar el siguiente comando:

tar -xvzf tracegraph202.linux.tar.gz

Posteriormente se ingresa el comando gunzip mglinstaller.gz para descomprimir el archivo mglinstaller.

3) Ejecutar el comando *sh mglinstaller*. Cuando el script solicite una carpeta de ubicación, ingresar /home/user1/mgl u otra ubicación según el criterio del usuario.

4) Añadir en el archivo de variables de entorno del sistema, al cual se accede utilizando el comando *gedit ~/.bashrc*, la siguiente variable:

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/user1/mgl/bin/glnx86.

5) Ejecutar el comando *source ~/.bashrc* para hacer efectivos los cambios llevados a cabo en el archivo de variables de entorno del sistema.

6) Ingresar al directorio */home/user1/Tracegraph202* y ejecutar la herramienta tracegraph mediante el comando ./trgraph.

Cuando se ejecuta la herramienta por primera vez, suelen presentarse dos errores comunes, estos son:

- ./trgraph: error while loading shared libraries: libmwcl.so: cannot open shared object file: No such file or directory.

Solución: Chequear que la variable LD_LIBRARY_PATH esté bien definida en el archivo ~/.bashr o también actualizar el contenido del anterior archivo usando el comando source ~/.bashrc, tal como se describe en el paso 5.

- ./trgraph: error while loading shared libraries: libXp.so.6: cannot open shared object file: No such file or directory.

Solución: Instalar el paquete libXp.so.6 usando el comando yum install libXp.so.6.

Instalación en Windows.

1) Descargar el tracegraph y mglinstaler de la página web http://www.tracegraph.com/download.html. En dicha página se deben ingresar algunos datos de usuario, después de lo cual se selecciona la versión del simulador; aquí se escoge la opción *2.02 for Windows.* Finalmente aparece una página donde se pueden descargar los dos archivos mencionados.

2) Descomprimir el archivo tracegraph202.zip en el directorio C:\ o en cualquier otro según el criterio del usuario.

3) Ejecutar el archivo mglinstaller.exe. Cuando este programa pida un nombre de directorio, ingresar el directorio C:\tracegraph202.

4) Añadir el directorio C:\tracegraph202\bin\win32 dentro de las variables de entorno del sistema. Para acceder a ellas, se da clic derecho en Mi PC y luego en propiedades. Allí se da clic en la pestaña Opciones avanzadas y posteriormente en Variables de entorno, donde se selecciona PATH en las variables del sistema.
5) Finalmente, en la variable PATH se ingresa la ubicación D:\tracegraph202\bin\win32, después de ello se da clic en aceptar para aplicar los cambios realizados.

Para ejecutar la herramienta se debe correr el archivo trgraph, ubicado en la carpeta C:\tracegraph202\bin\win32.

MODIFICACIÓN DE CÓDIGO FUENTE PARA PERMITIR LA SIMULACIÓN DE ARCHIVOS DE PAQUETES UDP SUPERIORES A 1000 BYTES.

En su versión estándar, el simulador NS en su versión 2.26 no permite la simulación de escenarios de red con tamaños de paquetes superiores a 1000 bytes. Cuando se trabaja con paquetes de tamaño mayor, el simulador automáticamente fragmenta los paquetes en partes de 1000 bytes y construye otro paquete con el residuo. Puesto que en el presente proyecto se trabaja con diferentes tráficos (voz, video y datos) donde algunos de ellos poseen un tamaño de paquete que supera este tope, se hizo necesario modificar el código fuente para aumentar dicho límite.

Para realizar el ajuste requerido, se debe acceder al archivo ns-default.tcl ubicado en /directorio base/ns-2.26/tcl/lib/ y se abre este archivo usando el siguiente comando:

gedit ns-default.tcl.

Posteriormente se busca la linea Agent/UDP set packetSize_ 1000 y se cambia este valor por 2000, el cual será el nuevo límite del tamaño de paquete con que puede operar NS-2. Por último, para hacer efectiva esta modificación se accede al directorio /directorio base/ns-2.26/ y se recompila el código de NS-2 utilizando los siguientes comandos:

#make clean

#./configure

#make

MÓDULO DE GENERACIÓN ALEATORIA DE FALLAS.

Se ha desarrollado un módulo de generación aleatoria de fallas con el propósito de programar varias fallas en diversos enlaces de la red y así simular el efecto que podría tener un evento de gran impacto en su infraestructura. El módulo inicialmente se programó teniendo en cuenta la topología de red establecida y bajo la estructura de nodos y enlaces desarrollada en este proyecto, sin embargo este se puede configurar mediante unos sencillos cambios para adaptarse a cualquier escenario de red y con el número de fallas que el usuario determine. El procedimiento en lenguaje Tcl que define el módulo se adjunta en un archivo txt.

MANUAL DE MNS VERSIÓN 2.0.

A continuación se presenta la lista de comandos del módulo MNS y la descripción de su funcionalidad dentro del simulador NS. Esta información se basa en el manual del módulo MNS versión 2 [1].

ns simulator - ns [new Simulator]
Este comando se usa para crear una nueva instancia del objeto de Simulación.

APIs para LSP/ER-LSP y LDP

$ns MPLSnode - set LSR0 [$ns MPLSnode]
Crea un LSR y le agrega la funcionalidad MPLS al mismo.

$ns configure-ldp-on-all-mpls-nodes
Configura agentes LDP en todos los nodos MPLS creados.

$MPLSnode enable-control-driven
Permite que el nodo MPLS $MPLSnode opere con base a una estrategia de disparo dirigida por control.

$MPLSnode enable-data-driven
Permite que el nodo MPLS $MPLSnode opere con base a una estrategia de disparo dirigida por datos.

$MPLSnode enable-on-demand
Permite que el nodo MPLS $MPLSnode opere en modo bajo demanda. El esquema de distribución y asignación de etiquetas se configura en modo descendente, donde se utilizan los mensajes LDP Request y LDP Mapping para negociar el establecimiento del LSP. Al utilizar la estrategia de disparo dirigida por control este comando queda sin efecto.

$MPLSnode enable-ordered-control
Permite que el nodo $MPLSnode opere en modo de control ordenado. El mecanismo de control de la distribución de etiquetas se configura en modo ordenado, donde el LER de entrada lleva a cabo la petición de un LSP mediante un mensaje LDP Request dirigido al LER de salida, que a su vez contesta con un mensaje LDP Mapping hasta el establecimiento de un LSP. Al utilizar la estrategia de disparo dirigida por control este comando queda sin efecto.

$ns enable-control-driven
Permite que todos los nodos MPLS creados operen con una estratega de disparo dirigida por control.

$ns enable-data-driven
Permite que todos los nodos MPLS creados operen con una estrategia de disparo dirigida por datos.

$ns enable-on-demand
Permite que todos los nodos MPLS creados operen en modo bajo demanda.

$ns enable-ordered-control
Permite que todos los nodos MPLS creados operen en modo de control ordenado.

$MPLSnode send-ldp-release-msg $fec
Envía un mensaje LDP Release hacia el nodo con FEC $fec en sentido ascendente para liberar el LSP establecido con dicha FEC.

$MPLSnode send-ldp-withdraw-msg $fec
Envía un mensaje LDP Withdraw hacia el nodo de subida con FEC $fec para liberar el LSP establecido con dicha FEC.
$MPLSnode aggregate-flows $fec1 $fec2
Este comando se usa para agregar un flujo del nodo $MPLSnode al nodo con FEC $fec1 dentro del flujo del nodo $MPLSnode al nodo con FEC $fec2, con la condición de que en la configuración de los parámetros el flujo con FEC $fec2 debe tener mayor capacidad para soportar el flujo con FEC $fec1.

APIs para ER-LSP

$MPLSnode setup-erlsp $fec $er $lspid
Crea una ruta explicita (ER-LSP: Explicit Route Label Switching Path) cuya FEC es $fec usando la ruta especificada por $er y cuyo identificador es $lspid.

Parámetros de configuración.

- $fec: El valor de la FEC
- $er: Ruta explicita.
- $lspid: El valor del LSPID.

Ejemplos:

$LSR1 setup-erlsp 7 5_4_8_6_7 3000

Crea a través de la ruta explícita que atraviesa los LSRs 5, 4, 8, 6, 7 un ER-LSP cuya FEC es 7 y el LSPID es 3000.

$LSR2 setup-erlsp 9 2_L3000_8 4000

Crea a través de la ruta explícita que atraviesa los LSRs 2, L3000, 8 un ER-LSP cuya FEC es 9 y el LSPID es 4000. L3000 representa el LSPID que se usa para identificar el punto de ingreso de un túnel LSP como el siguiente salto dentro de la ER en cuestión.

$MPLSnode bind-flow-erlsp $fec $flowid $lspid
Une un flujo con FEC $fec e identificador de flujo $flowid a un ER-LSP ya establecido con LSPID $lspid que se dirige al mismo destino.

Ejemplo:

$LSR2 bind-flow-erlsp 9 100 3000"

Une un flujo cuya FEC es 9 e identificador de flujo 100 al ER-LSP establecido cuyo LSPID

es 3000.

APIs para CR-LSP y CR-LDP.

$ns cfg-cbq-for-SBTS $qlim $cbq_qtype $okborrow $bw $maxidle $extradelay
Configura CBQ (CBQ: Class Based Queueing) en todos los nodos MPLS para soportar servicios con tráfico best effort simple.

$ns cfg-cbq-for-HBTS $qlim $cbq_qtype $okborrow $bw $maxidle $extradelay
Configura CBQ en todos los nodos MPLS para soportar servicios con tráfico best-effort de prioridad alta.

$ns cfg-cbq-for-STS $qlim $cbq_qtype $okborrow $bw $maxidle $extradelay
Configura CBQ en todos los nodos MPLS para soportar un tráfico de señalización.

$ns cfg-cbq-for-RTS $qlim $cbq_qtype $okborrow $bw $maxidle $extradelay
Configura CBQ en todos los nodos MPLS para soportar tráfico de tiempo real.

Los parámetros de configuración para los cuatro comandos mencionados anteriormente son:

- $qlim: Tamaño de la cola dado en paquetes.
- $cbq_qtype: Tipo del objeto cola CBQ.
- $okborrow: Booleano que indica si la clase está habilitada para solicitar ancho de banda asignado a un vecino.
- $bw $maxidle: Representa la máxima cantidad de tiempo que una clase dispone para tener sus paquetes encolados antes de que estos se encuentren habilitados para su envío.
- $extradelay: Aumenta el retardo experimentado por una clase en el tiempo especificado.

Para mayor información correspondiente a los parámetros de configuración, consultar la sección de CBQ en el manual de NS [2].

$ns bind-flowid-to-SBTS $id1 [$id2]
Asocia los paquetes que poseen el identificador de flujo $id1, incluyendo los que están comprendidos en el rango de $id1 a $id2, al servicio SBTS.

$ns bind-flowid-to-HBTS $id1 [$id2]
Asocia los paquetes que poseen el identificador de flujo $id1, incluyendo los que están comprendidos en el rango de $id1 a $id2, al servicio HBTS.

$ns bind-flowid-to-STS $id1 [$id2]

Asocia los paquetes que poseen el identificador de flujo $id1, incluyendo los que están comprendidos en el rango de $id1 a $id2, al servicio STS.

$ns bind-ldp-to-SBTS.
Asocia los paquetes LDP con el servicio SBTS.

$ns bind-ldp-to-HBTS
Asocia los paquetes LDP con el servicio HBTS.

$ns bind-ldp-to-STS
Asocia los paquetes LDP con el servicio STS.

$MPLSnode setup-crlsp $fec $er $lspid $TRate $BSize $PSize $SPrio $HPrio
Este comando crea un CR-LSP cuya FEC es $fec, la ruta explícita está identificada por $er y el identificador del LSP es $lspid. Adicionalmente envía un mensaje CR-LDP Request hacia el nodo con la FEC $fec por la ruta especificada por $er.

Parámetros de configuración.

$fec: Valor de la FEC.
$er: Ruta explícita;
$lspid: Valor del LSPID.
$TRate: Valor de la tasa de tráfico en Mbps.
$BSize: Valor del tamaño del buffer.
$PSize: Valor del tamaño de los paquetes.
$SPrio: Valor que especifica la prioridad del flujo.
$HPrio: Valor que especifica la prioridad propietaria (holding).

Ejemplo:

$LSR2 setup-erlsp 4 2_3 5000 450K 400B 200B 7 3

Crea a través de la ruta explícita que atraviesa los LSRs 2 y 3 un ER-LSP cuya FEC es 4 y LSPID 5000. Mediante este comando se pueden establecer CR-LSPs que incluyan parámetros de tráfico.

$MPLSnode send-crldp-release-msg $lspid
Envía un mensaje CR-LDP Release hacia un LSR ascendente con el propósito de liberar un ER-LSP/CR-LSP cuyo identificador de flujo es $lspid.

Ejemplo:

$LSR1 send-crldp-release-msg 3000

Libera el ER-LSP cuyo LSPID es 3000 usando un mensaje CR-LDP Release.

$MPLSnode send-crldp-withdraw-msg $lspid
Envía un mensaje CR-LDP Withdraw hacia un LSR descendente para retirar la etiqueta distribuida previamente por el nodo $MPLSnode, liberando así el ER-LSP/CR-LSP establecido cuyo identificador de flujo es $lspid.

$MPLSnode send-crldp-notifícation-msg $status $lspid $tr
Envía un mensaje CR-LDP Notification hacia los nodos ascendentes para informarlos acerca de una notificación. Los parámetros de configuración utilizados son:

$status: Información de status definida en el protocolo CR-LDP.
$lspid: Identificador del LSP.
$tr: Información de tráfico.

$MPLSnode set-flow-prio $fec $flowid $priority

Permite que un tráfico con FEC $fec e identificador de flujo $flowid tenga una prioridad dada por $priority. Este último parámetro puede tener dos valores, 1 significa alta prioridad y 0 baja prioridad respectivamente.

Funciones call-back en MNS.

- proc notify-erlsp-setup {node lspid}
Esta función se utiliza para notificar al usuario que el mensaje CR-LDP Mapping con LSPID $lspid ha llegado al nodo $node.

- proc notify-erlsp-fail {node status lspid tr}
Función utilizada para notificar al usuario que el mensaje CR-LDP Notification con información de status $status, identificador LSPID $lspid e información de tráfico $tr ha llegado al nodo $node.

- proc notify-erlsp-release {node lspid}
Se usa para notificar al usuario que el mensaje CR-LDP Release/Withdraw con identificador LSPID $lspid ha llegado al nodo $node.

APIs para enrutamiento basado en restricciones.

$ns collect-resource-info $itime
Comando usado para recopilar la información de recursos de todos los nodos MPLS de manera periódica en cada intervalo $itime.

$MPLSnode constraint-based-routing $dstid $bw
Este comando permite calcular rutas explicitas, retornando como resultado la ruta si efectivamente la encuentra o -1 en caso de que dicho proceso no sea exitoso.

Parámetros de configuración:

$dstid: ID del LSR destino.
$bw: Ancho de banda solicitado.

APIs para simulación de re-enrutamiento en redes MPLS.

$ns enable-reroute $option
Permite que todos los nodos MPLS ejecuten la función de restauración de rutas. Este comando cuenta con 5 opciones a saber:

-drop: No se crea ninguna ruta alternativa.
-L3: Hace uso de una tabla de encaminamiento L3.
-notify-prenegotiated: El nodo que detecta un falla de enlace envía un mensaje LDP Notification a sus LSRs ascendentes.
-simple-dynamic: Crea una ruta alternativa entre el nodo que detecta la falla y el nodo PML en caso de que no exista alguna disponible.
-shortest-dynamic: Crea una ruta alternativa entre el nodo que detecta la falla y el próximo nodo en caso de que no exista alguna disponible.

$ns set-protection-lsp $stime $itime $lspid
Este comando se usa para verificar cuando ocurren fallos en los enlaces que comunican los nodos MPLS. Sus parámetros de configuración son:

$stime: Tiempo para empezar la detección de fallas en los enlaces de la red.
$itime: Intervalos de tiempo para la detección de fallas.
$lspid: Identificador del LSP de trabajo.

$MPLSnode reroute-lsp-binding Sw_lspid a_lspid
Este comando se usa para unir un camino de trabajo con uno de respaldo. Se configura utilizando los siguientes parámetros:

- $w_lspid: Identificador del camino de trabajo.
- $a_lspid: Identificador del camino de respaldo.

$MPLSnode enable-reroute-egress-lsr
Permite que el nodo $MPLSnode opere con funciones PML (Protection Merge LSR).
APIs para trazas de paquetes MPLS/LDP y volcado de tablas de encaminamiento.

$MPLSnode trace-mpls
(VER) Trazas de los paquetes en el nodo MPLS $MPLSnode.

$MPLSnode trace-ldp
Trazas LDP de los paquetes en el nodo MPLS $MPLSnode.

$MPLSnode pft-dump
Muestra una tabla PFT (PFT: Partial Forwarding Table) gestionada en el nodo MPLS $MPLSnode.

$MPLSnode erb-dump
Muestra una tabla ERB (ERB: Explicit Route information Table) gestionada en el nodo MPLS $MPLSnode.

$MPLSnode lib-dump
Muestra una tabla LIB (LIB: Label Information Base) gestionada en el nodo MPLS $MPLSnode.

APIs para utilitarios.

$ns ldp-request-color $color
Configura un color especificado por $color para el mensaje de petición LDP Request.

Ejemplo:

$ns ldp-request-color $green
Configura el color verde para el mensaje LDP Request.

$ns ldp-mapping-color $color
Configura un color especificado por $color para el mensaje LDP Mapping.

$ns ldp-withdraw-color $color
Configura un color especificado por $color para el mensaje LDP Withdraw.

$ns ldp-release-color $color
Configura un color especificado por $color para el mensaje LDP Release.

$ns ldp-notifícation-color $color
Configura un color especificado por $color para el mensaje LDP Notification.

BIBLIOGRAFÍA

[1] Ahn, G. et al., " Architecture of MPLS Network simulator (MNS) for the setup of CR-LSP". Chungnam National University. Korea, 2001.

[2] Fall, K., "The NS Manual". VINT Project. URL: http://www.isi.edu/nsnam/ns/ns-documentation. Diciembre 2003.