MANUAL DE PRÁCTICAS CON NS-2

AUTORES:

Jhon Jairo Padilla Aguilar
Ingeniero Electrónico, Universidad del Cauca
MsC. en Informática, Universidad Industrial de Santander
PhD. en Ingeniería Telemática, Universidad Politécnica de Cataluña

Line Yazmin Becerra
Ingeniera Electrónica, Universidad Pontificia Bolivariana
MsC. en Telecomunicaciones, Universidad Pontificia Bolivariana

Grupo de Investigación en Telecomunicaciones
Facultad de Ingeniería Electrónica
UNIVERSIDAD PONTIFICIA BOLIVARIANA
Bucaramanga

url: http://git.upbbga.edu.co

Tabla de Contenido

1. Generalidades del Network Simulator

Network Simulator (NS) es un software con el cual se puede simular eventos discretos y fue ideado para la ayuda a la investigación de redes telemáticas. NS proporciona soporte para la simulación de multitud de protocolos de las capas de aplicación (http, ftp, cbr, etc), transporte (TCP, UDP, RTP, SRM), protocolos de enrutamiento monodifusión y multidifusión (multicast), etc, tanto para redes cableadas como no cableadas locales o vía satélite, y con topologías complejas con un gran número de generadores de tráfico.

NS comenzó en 1989 como una variante al ya existente REAL Network Simulator y ha evolucionado sustancialmente en los últimos años, habiendo sido objeto de desarrollo por DARPA con ayuda de varias instituciones de investigación en redes, como LBL, Xerox PARC, UCB, USC/ISI,etc.

También contempla mecanismos referentes a la capa de Enlace de Datos en Redes de Area Local (LAN), tales como protocolos MAC (Control de Acceso al Medio) del tipo CSMA/CD (Acceso Múltiple con Detección de Portadora y Detección de Colisiones). Así mismo, incluye diversos algoritmos para la planificación de Colas: FQ (Encolamiento Justo), SFQ (Encolamiento Estocástico Justo), DRR (Deficit Round Robin), FIFO (Primero en Entrar es Primero en Salir), etc. Posee herramientas para graficar como NAM y XGRAPH. La utilidad de NAM es que puede representar gráficamente la red, que previamente hayamos construido mediante comandos escritos en un lenguaje de programación llamado Tcl y posteriormente compilados por NS. Así mismo, NAM puede visualizar dinámicamente el desplazamiento de los paquetes de la simulación y que NS ha almacenado en un fichero junto con la propia topología de la red. Estos resultados que NS produce a partir de los ficheros que contienen los comandos escritos en lenguaje Tcl o Scripts, lógicamente, dependerán de la topología, protocolos, parámetros, y demás actividad que en ellos se definan. Por otra parte, con las gráficas bi-dimensionales de XGRAPH se pueden hacer análisis de paquetes recibidos en la comunicación, paquetes perdidos, ancho de banda y de retardos.

NS (Network Simulator) es básicamente un simulador orientado a objetos, escrito en **C++**, cuya interfaz de usuario se presenta como un intérprete de lenguage Tcl orientado a objetos o, en otras palabras, de lenguaje OTcl. El simulador soporta una jerarquía de clases escrita en C++, también llamada jerarquía compilada, y otra jerarquía de clases similar a la anterior pero dentro del intérprete que se presenta al programador en OTcl, y que también se conoce como jerarquía interpretada. Estas dos jerarquías están estrechamente relacionadas entre sí, de modo que, desde la perspectiva del usuario, hay una correspondencia uno a uno entre una clase de la jerarquía interpretada y otra de la compilada. La raíz de esta jerarquía es una clase que se llama TclObject. Cuando el usuario crea un objeto simulador desde el intérprete, cosa con la que generalmente se comienza un

script, este objeto es creado dentro del intérprete y se crea una estrecha relación con otro objeto idéntico, pero dentro de la jerarquía compilada. La jerarquía de clases interpretada se establece automáticamente a través de métodos definidos dentro de la clase llamada TclClass. Además, podemos crear otras jerarquías a nuestro gusto en los scripts, escritas en OTcl, y que no estén desdobladas en la jerarquía compilada.

Es muy importante hacer notar que, para trabajar a fondo con NS, usted debe construirse sus propias clases, así como sus propios protocolos y modificar según sus necesidades los ya existentes. Todo ello conlleva hacerlo en el programa fuente del propio simulador NS, escrito en C++, y luego volver a compilar y vincular todos los módulos para crear un nuevo ejecutable de NS.

También se puede ampliar la funcionalidad de NS a través de una segunda posibilidad: mediante OTcl, dentro de los scripts. Dependiendo de qué necesitemos hacer, será más apropiado ampliar NS haciéndolo en C++, o bien en OTcl. Por una parte, las simulaciones detalladas de protocolos requieren el uso de C++, al ser un lenguaje de programación que puede manipular eficientemente bytes, paquetes, cabeceras, e implementar algoritmos que manejen grandes cantidades de datos. Para estas tareas la velocidad de ejecución es más importante que el tiempo que pueda invertirse en su desarrollo. Por otro lado, una gran parte del trabajo de investigación de redes conlleva una modificación continua de parámetros de configuración, la exploración de muchos escenarios de simulación, etc. En estos casos, el tiempo de iteración (cambiar el modelo y volver a ejecutar) es muy importante, y en estos casos sólo cobra sentido hacerlo en OTcl.

2. Documentación de NS

En la dirección http://www.isi.edu/nsnam/ns/ usted podrá abrir el sitio donde se encuentran todas las versiones de NS para que lo instale en su computador, el manual de NS y algunos tutoriales de utilidad. Un tutorial muy bueno para iniciar con el NS fue desarrollado por Marc Greiss. El presente Manual de prácticas está basado en dicho tutorial. De otra parte, NS es un software que trabaja bajo el sistema operativo Linux, por lo que se debe tener previamente instalado Linux en el computador para luego si instalar NS.

3. PRACTICAS CON NS-2

3.1. Práctica 1: Familiarización con el Linux e instalación del NS-2

Objetivos

Con la presente práctica el estudiante conocerá:

- El ambiente del sistema operativo Linux
- Las herramientas de lectura y creación de archivos de texto
- El manejo básico de archivos desde el Terminal de Linux

Requisitos

Computador con el sistema operativo Linux instalado (cualquiera de sus versiones).

A. Familiarización con el sistema Operativo Linux

Procedimiento

El Linux es un sistema operativo multi-usuario y multi-tarea. Esto significa que puede operar con varios usuarios simultáneamente y con cada usuario tener múltiples tareas ejecutándose.

- *Cómo ingresar al Linux?*

Si su computador posee dos sistemas operativos (*Windows* y *Linux*), deberá seleccionar el sistema operativo al encender el computador. El computador posee un menú de opciones que aparece al iniciar. En este menú, que suele activarse oprimiendo las teclas de flechas, se deberá seleccionar el inicio con el sistema operativo *Linux*.

Una vez el sistema haya iniciado todos los programas, aparecerá una ventana solicitando el nombre de usuario y la contraseña. Ingréselos terminando con la tecla *Enter*.

- *El ambiente del sistema operativo Linux*

Una vez haya ingresado su nombre de usuario y clave, el computador pasará una presentación gráfica similar a la del *Windows*. Podría aparecer en la parte de

arriba o en la parte de abajo, un menú de opciones donde se muestran los programas clasificados ya sea por herramientas de oficina, herramientas del sistema, herramientas de Internet y en alguna de estas opciones aparecerá la posibilidad de apagar el computador. Esta última deberá seleccionarla al terminar su sesión.

Para nuestros propósitos, y ya que no es nuestro objetivo hacer un tutorial completo de *Linux*, sólo explicaremos algunas operaciones indispensables para realizar las simulaciones en el simulador de redes NS.

Fundamentalmente, se requieren dos tipos de operaciones:

- Utilizar el navegador de archivos para pasar de un directorio a otro y para ver archivos de texto.
- Utilizar el editor de texto para ver los archivos de texto plano y para hacer modificaciones a estos archivos.
- Utilizar el Terminal de Linux para dar órdenes al computador, tales como navegar por los directorios o ejecutar el NS.

➢ *El navegador o administrador de archivos*

Al seleccionar esta opción del menú principal, aparecerá una ventana con una apariencia similar a la del explorador de archivos de Windows. En esta ventana se mostrarán las carpetas (con su símbolo de carpeta) y los archivos, a los que se les da una apariencia que depende del tipo de archivo. La navegación por las carpetas es bastante intuitiva, sencillamente para entrar a una carpeta, deberá hacer doble clic sobre la misma. Para pasar a la carpeta anterior, puede dar clic sobre el botón “Atrás”, que se muestra como una flecha hacia arriba o hacia la izquierda.

➢ *Leer o modificar archivos de texto plano con el editor de texto de Linux*

Para mirar el contenido de un archivo de texto plano, sencillamente ubíquese sobre él en la ventana del navegador de archivos y de doble clic. El Linux abrirá el editor de texto y le mostrará el contenido del archivo. Si usted desea hacer cambios, haga clic sobre la posición donde quiere ubicar el cursor y luego borre o inserte el texto que quiera. Luego de la orden de “Guardar” con el puntero del ratón y luego cierre la aplicación. El Linux retornará inmediatamente a la ventana del navegador de archivos.

➢ *Utilizar el Terminal de Linux*

El Terminal de Linux es una interfaz de usuario que permite dar órdenes al Linux en forma de comandos en lugar de utilizar la interfaz tipo Windows. Para ejecutar esta aplicación, algunos computadores muestran esta opción como un ícono en el escritorio, otros tendrán esta opción dentro del menú del Linux.

Una vez seleccionada la opción "Abrir Terminal", aparecerá una ventana con un prompt como por ejemplo: *"/home>"* . A continuación, usted podrá escribir el comando que requiera y luego debe dar *Enter* para que el computador lea el comando y lo ejecute.

Los dos comandos básicos para navegar por los directorios desde el Terminal son:

ls: Este comando permite ver los directorios y archivos que se encuentran ubicados en el directorio actual

cd nombredirectorio: Este comando permite cambiar al directorio inferior cuyo nombre es "*nombredirectorio*". Para pasar al directorio inmediatamente superior, deberá teclear el comando "*cd ..*". Finalmente, si quiere pasar al directorio raíz de todos los directorios, deberá dar el comando "*cd /*".

Examen

-Con el navegador de archivos, crear una carpeta llamada "sunombre".
-Con el editor de texto crear un archivo de texto "sunombre.tcl", en donde el nombre del archivo es "sunombre" y la extensión es ".tcl".
-Dentro del archivo escriba algún texto como: "estoy aprendiendo a crear archivos de texto"
-Luego, guardar el archivo en la carpeta "sunombre"
-Abrir el Terminal de Linux y mediante las instrucciones de cambio de directorio llegar hasta la carpeta donde se encuentra el archivo "sunombre.tcl". Verificar el contenido de la carpeta para determinar si allí se encuentra el archivo.

B. Instalación del NS-2

Para instalar el Network Simulator bajo el sistema operativo de Linux se deben seguir los siguientes pasos:

- ✓ Primero que todo se descarga el paquete "todo en uno", el cual trae todos los paquetes del simulador en uno. Para descargarlo se puede acceder a esta página: http://www.isi.edu/nsnam/dist/ y escoger la versión más conveniente para el usuario. Se debe colocar el archivo descargado en */home/nombre-usuario/*, donde nombre-usuario es la cuenta personal en el computador.

- ✓ Para descomprimir el archivo se habilita un terminal y se aplica el comando *gunzip ns-allinone-2.31.tar.gz*; el nombre puede cambiar dependiendo de la versión del NS-2 que se haya descargado.

- ✓ Se reconstruye el sistema de archivos, aplicando el comando *tar –xvf ns-allinone-2.31.tar*.

- ✓ Otra forma de hacer la operación comprendida por los dos pasos anteriores es dar click derecho sobre el archivo comprimido *ns-allinone-2.31.tar.gz* y hacer click en la opción "descomprimir", estando en el navegador de archivos en el ambiente gráfico del Linux.

- ✓ Pasar a modo superusuario (esto permite hacer cualquier tipo de operación sobre el sistema operativo), aplicando el comando *su* y el *password* correspondiente (Esto también puede hacerse mediante el ambiente gráfico ingresando al navegador de archivos en el modo superusuario).

- ✓ Modificar el PATH, lo que permite encontrar los archivos necesarios. Para ello se edita el archivo */home/nombre-usuario/.bashrc/* insertando las siguientes líneas (respetar mayúsculas-minúsculas):

  PATH=$PATH:/usr/X11R6/bin:/usr/X11R6/lib:/usr/X11R6/lib/X11:/usr/X11R6/include/X11:/home/nombre-usuario/ns-allinone-2.30/bin: /home/nombreusuario/ns-allinone-2.30/tcl8.4.13/unix:/home/nombre-usuario/ns-allinone-2.30/tk8.4.13/unix:/home/nombre-usuario/ns-allinone-2.30/xgraph-12.1:/home/nombre-usuario/ns-allinone- 2.30/nam-1.12
  export LD_LIBRARY_PATH=/home/nombre-usuario/ns-allinone-2.30/otcl-1.12: /home/nombre-usuario/ ns-allinone-2.30/lib
  export TCL_LIBRARY=/home/nombre-usuario/ns-allinone-2.30/tcl8.4.13/library

- ✓ Activar el PATH, aplicando el comando *source .bashrc*.

- ✓ Ir al directorio de instalación del *NS-2* con el siguiente comando, *cd ns-allinone-2.31.*

- ✓ Instalar NS-2, aplicando el comando *./install*

- ✓ Una vez finalizada la instalación, ir al directorio */home/nombre-usuario/ns-allinone-2.31/ns-2.31/*, y correr el programa de validación, aplicando el comando "*./valídate".*

- ✓ Para verificar si quedó instalado correctamente el NS-2 teclee el comando "ns" estando en la ventana del terminal. Si se ha instalado correctamente el programa, aparecerá una línea de texto así "%ns". En ese momento ya se está ejecutando el NS-2. Para salir, interrumpa el programa dando el comando "CTRL-C" en la ventana del terminal.

3.2. Práctica 2: Creación y Simulación de un Script simple con NS-2

Objetivos

Con la presente práctica el estudiante aprenderá:

- A crear un script en lenguaje Tcl
- A crear una topología simple en NS-2 (creación de nodos, enlaces y fuentes de tráfico)
- A ejecutar una simulación en el NS-2
- A mostrar los resultados de la simulación en la herramienta NAM y a utilizar la interfaz gráfica de dicha herramienta

Requisitos

Para realizar simulaciones en NS es necesario en primer lugar instalar el programa en un computador que tenga el sistema operativo Linux (ver sección *pasos de instalación de NS-2*). En segundo lugar se requiere manejar algunas herramientas básicas del Linux, tales como un editor de texto y saber trabajar desde un Terminal con los comandos básicos de Linux (cambiar de directorio, ejecutar comandos, etc). Una vez tenga estas destrezas, se le facilitará mucho el manejo del NS-2.

Procedimiento

- *Ejecución de una simulación con NS*

El NS-2 se ejecuta desde un Terminal en *Linux*. Para ejecutar una simulación se debe escribir el siguiente comando en la pantalla del Terminal:

*ns prueba1.tc*l

Al ejecutar este comando, se ejecuta el simulador NS como si fuese un comando del Linux y toma como parámetro de entrada el nombre del archivo de texto plano con extensión *.tcl*, que contiene las instrucciones de la simulación (en nuestro ejemplo el archivo tiene el nombre *prueba1.tcl*. A este archivo también se le conoce como un script Tcl. Este comando debe ejecutarse en la pantalla del Terminal, que debe estar ubicado en el directorio donde se encuentra el archivo

que contiene el script Tcl a simular. El script Tcl puede ser creado mediante una herramienta de edición de texto sin formato de Linux.

- *Creación de un script básico*

Para escribir su primer *script Tcl*, deberá ejecutarse un editor de texto plano de *Linux*. En este editor se escriben las instrucciones que se describen a continuación y luego se almacena en el disco duro bajo un nombre con extensión *.tcl*. Un ejemplo de un nombre de archivo podría ser *prueba1.tcl*. Ahora iniciaremos la explicación de las diferentes instrucciones que se requieren para hacer una simulación muy sencilla.

Primero se necesita crear un objeto simulador, esto se puede hacer con el siguiente comando:

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

Si se desea obtener una grafica de la topología de la red usando el graficador NAM entonces se debe abrir un archivo de traza, donde NS guarde todas las trazas necesarias para que NAM pueda realizar la respectiva grafica. Esto se hace de la siguiente manera:

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

En la primera línea se abre el archivo “out.nam” y además se vincula a la variable “nf”. En la segunda línea se le dice al objeto simulador que puede escribir en este archivo todos los datos de la simulación necesarios para NAM.

El siguiente paso es adicionar un procedimiento “finish” en el cual se cierre el archivo de traza y se ejecute NAM.

```
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
}
```

La siguiente línea le dice al objeto simulador que ejecute el procedimiento “finish” después de 5.0 segundos de tiempo de simulación.

```
$ns at 5.0 "finish"
```

La última línea finalmente comienza la simulación.

```
$ns run
```

Ahora se puede salvar el archivo con el nombre que el usuario desee y con la extensión .tcl (nombre.tcl) y tratar de correrlo así: “ns nombre.tcl”, pero al ejecutarlo se obtienen mensajes de error puesto que el archivo out.nam esta vacío ya que no se han definido objetos como nodos, enlaces o eventos.

- *Creación de una topología simple.*

Para realizar la siguiente simulación, debe tomarse el *script* creado en la sección anterior y agregarle las instrucciones que se describen a continuación.

Ahora se va a definir una topología simple con dos nodos que están conectados por un enlace. Las siguientes dos líneas definen los dos nodos. (Nota: Este código se debe insertar antes de la línea “$ns run” o aun mejor, antes de la línea “$ns at 5.0 “finish”).

```
set n0 [$ns node]
set n1 [$ns node]
```

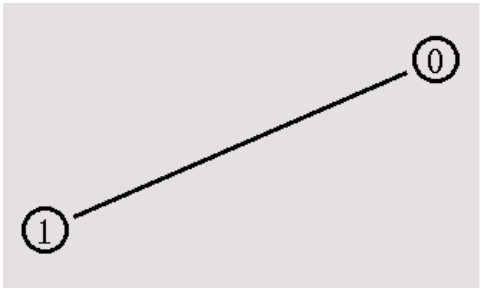
Un nuevo objeto *node* es creado con el comando “$ns node”. Por tanto, se han creado dos nodos que son asignados a las variables “n0” y “n1” respectivamente.

Ahora se conectan los dos nodos mediante la siguiente instrucción:

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

Esta línea le dice al objeto simulador que conecte los nodos *n0* y *n1* con un enlace *full-duplex* con ancho de banda de 1Mbps, un retardo de 10ms y un tipo de cola *Drop Tail* (Cola *FIFO* que desecha los paquetes que llegan si está llena).

Luego, se puede salvar el archivo y ejecutar el script con el comando “ns nombre.tcl” , NAM se ejecutara automáticamente y se podrá ver una ventana de salida parecida a la siguiente grafica.



- *Enviando datos*

Con los pasos anteriores sólo se puede apreciar la topología pero aún nada está sucediendo. El siguiente paso es enviar datos desde el nodo *n0* hasta el nodo *n1*. En NS, los datos siempre son enviados desde un Agente de la capa de Transporte a otro. Por lo tanto, el próximo paso es crear un objeto Agente que envía datos desde el nodo *n0*, y otro objeto agente que reciba los datos en *n1*.

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

Estas líneas crean un agente UDP (llamado *udp0*) y lo adjuntan al nodo *n0*, luego se adjunta un generador de trafico CBR al agente UDP. Por CBR se entiende "tasa constante de bits". El tamaño de paquete se fija a 500 bytes y un paquete será enviado cada 0.005 segundos (200 paquetes por segundo).

Las próximas líneas crean un Agente *Null*, el cual actúa como sumidero de tráfico y se adjunta al nodo *n1*.

```
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
```

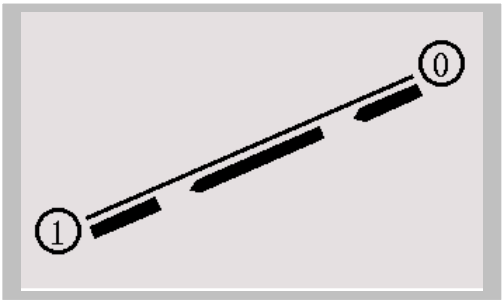
Entonces se deben conectar los dos agentes a nivel de capa de transporte así,

```
$ns connect $udp0 $null0
```

Luego se le debe decir al agente CBR cuándo enviar datos y cuándo parar el envío. Para ello, es mejor colocar las siguientes líneas justo antes de la línea "$ns at 5.0 "finish".

```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

Ahora se puede salvar el archivo y comenzar la simulación de nuevo. Cuando se de un clic sobre la tecla "*play*" de la ventana de NAM, se podrá apreciar que después de 0.5 segundos de simulación, el nodo *n0* comienza a enviar paquetes de datos al nodo *n1*.



Examen

a. Cómo se da la orden de ejecutar una simulación en NS-2? Explique las partes de la orden.
b. Qué hace la instrucción: *set ns [new Simulator]?*
c. Qué hace la instrucción *$ns run*?
d. Qué pasos del script son necesarios para mostrar los resultados en la herramienta NAM?
e. Por qué es necesario el procedimiento *finish* en un *script* de Tcl?

3.3. Práctica 3: Creación de Topologías más complejas

Objetivos

La presente práctica permitirá al estudiante:

- Crear topologías con varios nodos y enlaces
- Crear simulaciones con varias comunicaciones que ocurren en diferentes intervalos de tiempo
- Dar colores a los paquetes de diferentes comunicaciones para hacer un mejor seguimiento a los mismos
- Mostrar el comportamiento de la cola en un nodo
- Utilizar diferentes tipos de colas en los nodos

Requisitos

Haber realizado la práctica 2

Procedimiento

- *Definición de la topología de red*

Cuando se va a efectuar una simulación, el primer paso es definir una topología. Por lo tanto se puede crear cualquier topología y agregarla al código ya elaborado anteriormente; como ya se mencionó, siempre se debe crear un objeto simulador, es decir que siempre se debe comenzar la simulación con el mismo comando, y si se desea correr NAM automáticamente, se debe siempre abrir un archivo de traza y definir un procedimiento el cual cierre los archivos de traza y ejecute NAM.

Entonces para este caso se va a definir una topología con cuatro nodos.

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
```

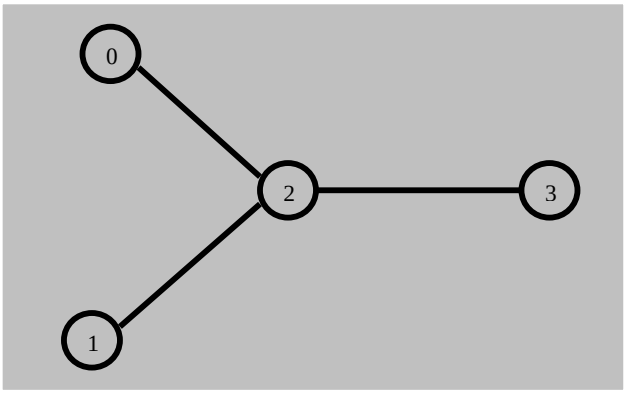
Luego se crean tres enlaces para conectar los nodos.

```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail
```

Ahora se puede salvar y ejecutar el script. Si al visualizar en NAM la topología se ve un poco desordenada, se puede dar un clic en el botón "re-layout" de la ventana de NAM para reorganizar la topología, pero si se quiere obtener un poco más de control sobre la ubicación de los nodos en la pantalla, se pueden adicionar las próximas líneas en el script y ejecutarlo de nuevo.

```
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
```

Entonces se podrá apreciar la topología de la siguiente manera:



Las opciones de orientación de los enlaces son *right*, *left*, *up*,*down* y combinaciones de estas orientaciones.

- *Creación de los Agentes y los eventos*

Ahora se crean dos agentes UDP con fuentes de tráfico CBR y se adjuntan a los nodos *n0* y *n1*. Luego se crea un agente Null y se adjunta al nodo *n3*.

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

# Create a CBR traffic source and attach it to udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
```

Los dos agentes CBR tienen que ser conectados al agente *Null*.

```
$ns connect $udp0 $null0
$ns connect $udp1 $null0
```

Para este ejemplo el primer agente CBR comienza a 0.5 segundos y se detiene a los 4.5 segundos mientras el segundo agente CBR comienza a 1.0 segundos y se detiene a los 4.0 segundos.

```
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
```

El usuario de este simulador puede salvar y ejecutar este *script*, con lo cual podrá apreciar que hay más tráfico sobre el enlace de *n0* a *n2* y *n1* a *n2* que en el enlace de *n2* a *n3*. Realizando cálculos simples se puede confirmar que: Se están enviando 200 paquetes por segundo sobre cada uno de los dos primeros enlaces y el tamaño del paquete es de 500 bytes; esto resulta un ancho de banda de 0.8 Mbps por los enlaces de *n0* a *n2* y de *n1* a *n2*; para un total de ancho de banda

1.6Mbps, pero el enlace entre *n2* y *n3* solo tiene una capacidad de 1Mbps, por lo tanto algunos paquetes se pierden. Como poder saber cuales son los que se pierden, si se observa en NAM que todos los flujos son de color negro? Las siguientes instrucciones ayudan a diferenciar entre diferentes flujos y así se podrá tener un mejor control de estos.

- *Marcando flujos*

Se pueden agregar las siguientes líneas en las definiciones de agentes CBR.

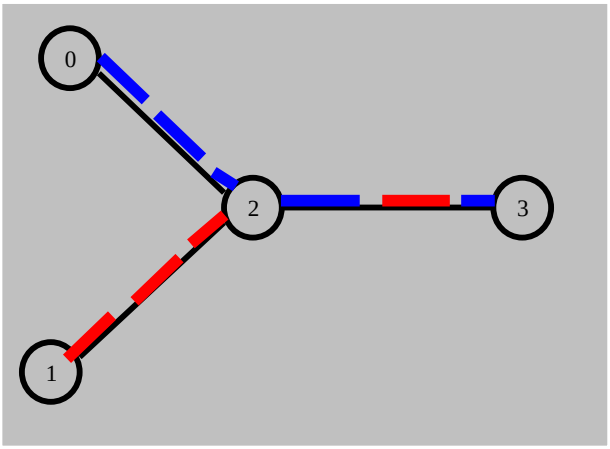
```
$udp0 set class_ 1
$udp1 set class_ 2
```

Donde el parámetro "*fid_*" se entiende por "*flow id*".

Las siguientes líneas se pueden agregar preferiblemente en el comienzo del *script* después de que el objeto simulador haya sido creado.

```
$ns color 1 Blue
$ns color 2 Red
```

Este código permite fijar diferentes colores para cada flujo, tal como se observa en la figura siguiente.

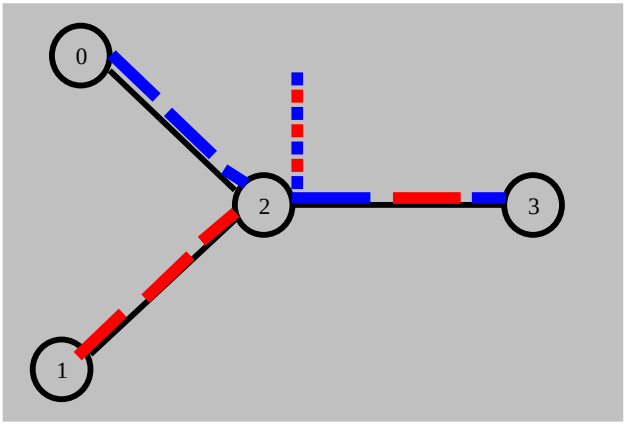


- *Monitoreando una cola*

Para monitorear la cola en el enlace n2 a n3 se agrega la siguiente línea.

```
$ns duplex-link-op $n2 $n3 queuePos 0.5
```

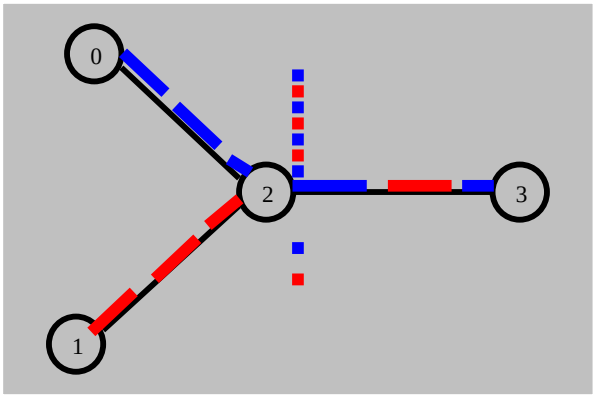
Al correr el script, se podrá apreciar una gráfica así:



En la gráfica se pueden ver los paquetes en la cola y cómo se pierden solo los paquetes azules. Si se quiere una pérdida más justa para todos los usuarios, entonces se puede probar usando SFQ (stochastic fair queueing) para el enlace n2 a n3 como en aparece en la siguiente linea:

```
$ns duplex-link $n3 $n2 1Mb 10ms SFQ
```

Entonces se pierde la misma cantidad de paquetes azules y rojos.



Examen

1. Qué pasos deben seguirse para darle colores a los paquetes de los diferentes flujos en una simulación?
2. Cómo se puede hacer que diferentes aplicaciones funcionen en diferentes momentos?
2. Qué instrucción permite monitorear una cola?
3. Investigar: Qué otros tipos de colas pueden colocarse en los enlaces de salida?

3.4. Práctica 4: Uso de Arreglos para crear topologías complejas

Objetivos

La presente práctica permitirá al estudiante:

- Aprender a utilizar Arreglos (o vectores) de nodos
- Aprender a utilizar ciclos *for* para optimizar el código *Tcl* en la creación de topologías con un gran número de nodos
- Simular la caída de un enlace y analizar lo que ocurre
- Ver el efecto de utilizar enrutamiento estático y dinámico

Requisitos

Haber realizado la Práctica 3.

Procedimiento

- *Creación de la topología usando arreglos de elementos*

El siguiente código crea siete nodos y los almacena en un Arreglo (o vector) llamado n().

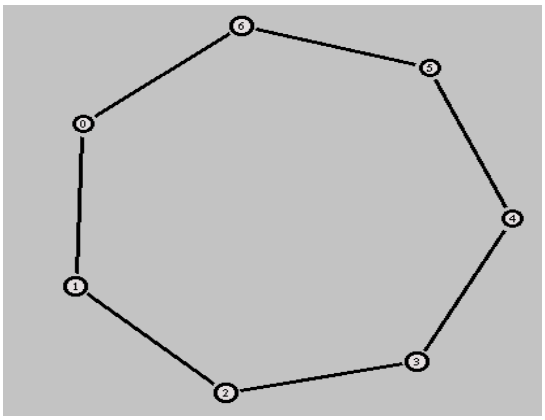
```
for {set i 0} {$i < 7} {incr i} {
  set n($i) [$ns node]
}
```

Los Arreglos, igual que otras variables en Tcl, no tienen que ser declarados primero. Ahora se conectan los nodos para crear una topología circular.

```
for {set i 0} {$i < 7} {incr i} {
  $ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms DropTail
}
```

Este ciclo "*for*" conecta cada nodo con el próximo en el arreglo, con la excepción del último nodo que se conecta con el primero. Esto se logra con la expresión %7, que indica que se hace el conteo en módulo 7.

Si al correr el programa la topología aparece un poco desordenada, se puede oprimir "*re-layout*" en la ventana de NAM para poderla visualizar mejor, esta topología luce de la siguiente manera



- *Creación de los Agentes y los eventos*

El próximo paso es enviar algunos datos desde el nodo *n(0)* al nodo *n(3)*.

```
#Create a UDP agent and attach it to node n(0)
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0

$ns connect $udp0 $null0

$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

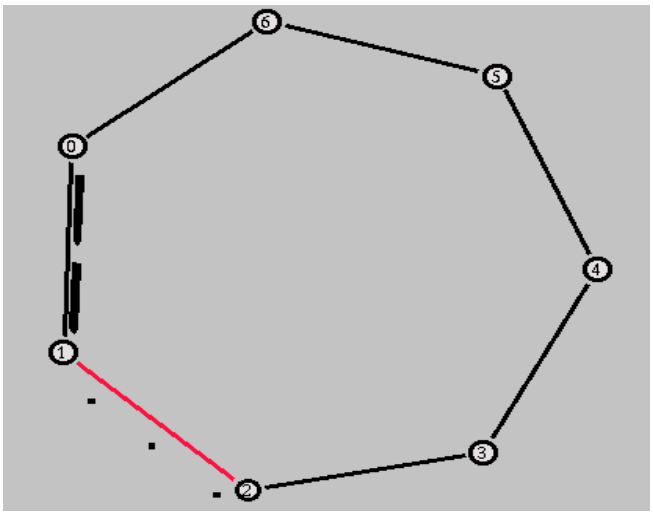
El código anterior es similar al ya utilizado; la única diferencia es que en este último se usa un arreglo de elementos.

Al ejecutar el *script*, se observa que el tráfico toma el camino más corto desde el nodo 0 al nodo 3 a través de los nodos 1 y 2.

Si se requiere se puede bajar cualquier enlace por un tiempo determinado. Para este ejemplo se bajará el enlace entre el nodo 1 y el nodo 2 por un segundo con las siguientes instrucciones:

```
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
```

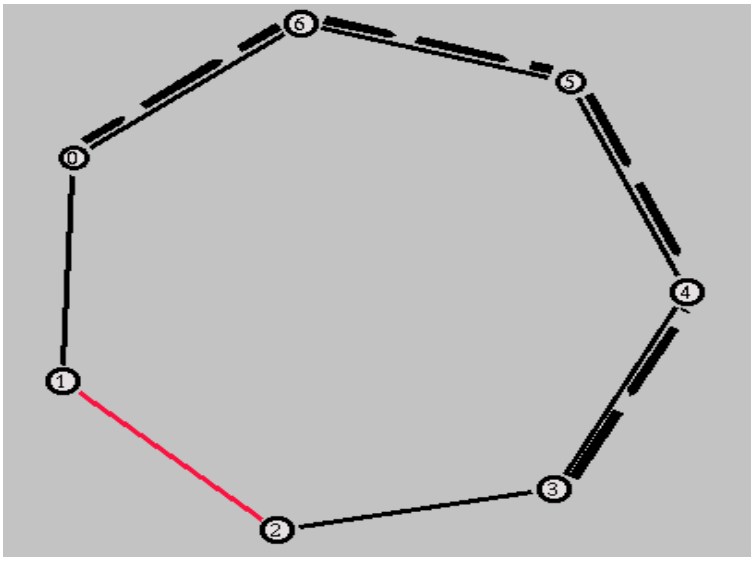
Al correr el *script* de nuevo, se podrá apreciar que entre el instante 1.0s y el instante 2.0s el enlace estará fuera de servicio y todos los datos enviados desde el nodo 0 se pierden.



Ahora se muestra como usar un enrutamiento dinámico para resolver el problema. Se puede agregar la siguiente línea al comienzo del *script*, después de haber creado el objeto simulador.

```
$ns rtproto DV
```

Al ejecutar el programa se puede notar que cuando el enlace se ha caído en el segundo 1.0, el enrutamiento se actualiza y el tráfico se vuelve a encaminar a través de los nodos 6, 5 y 4.



Examen

1. Describa las partes que componen la instrucción *for* en Tcl.
2. Explique cómo se logra que se conecte el nodo 0 con el nodo 6 en esta simulación.
3. Cómo se puede hacer que un enlace se caiga y luego se vuelva a poner en operación?
4. Qué otros tipos de fuentes de tráfico pueden usarse en NS?
5. Qué otros parámetros se le pueden modificar a una fuente de tráfico CBR?

3.5. Práctica 5: Creación de gráficos de salida con Xgraph

Objetivos

La presente práctica permitirá al estudiante:

- Utilizar sumideros de tráfico que permiten graficar variables
- Calcular el ancho de banda de un flujo de datos
- Crear y almacenar la variación de ciertas variables en archivos de texto
- Graficar el comportamiento de las variables almacenadas mediante la herramienta *XGraph*

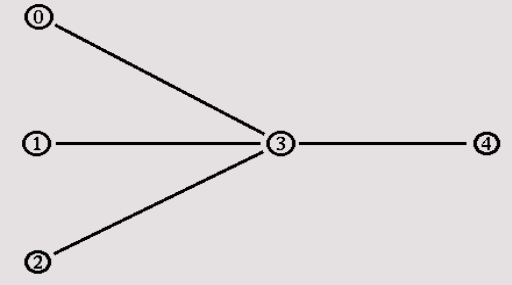
Requisitos

Haber realizado la práctica 3.

Procedimiento

- *Creación de la Topología de red*

Primero se crea la topología. Para este ejemplo utilizaremos la siguiente:



Entonces se crean los nodos de la misma forma como ya se había mencionado.

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

$ns duplex-link $n0 $n3 1Mb 100ms DropTail
$ns duplex-link $n1 $n3 1Mb 100ms DropTail
$ns duplex-link $n2 $n3 1Mb 100ms DropTail
$ns duplex-link $n3 $n4 1Mb 100ms DropTail
```

Luego, se adjuntan las fuentes de tráfico a los nodos *n0*, *n1* y *n2*.

```
proc attach-expoo-traffic { node sink size burst idle rate } {
        #Get an instance of the simulator
        set ns [Simulator instance]

        #Create a UDP agent and attach it to the node
        set source [new Agent/UDP]
        $ns attach-agent $node $source

        #Create an Expoo traffic agent and set its configuration parameters
        set traffic [new Application/Traffic/Exponential]
        $traffic set packet-size $size
        $traffic set burst-time $burst
        $traffic set idle-time $idle
        $traffic set rate $rate

      # Attach traffic source to the traffic generator
      $traffic attach-agent $source
        #Connect the source and the sink
        $ns connect $source $sink
        return $traffic
```

Primero, el procedimiento crea una fuente de tráfico y lo adjunta al nodo. Luego, este crea un objeto *Traffic/Expoo*, establece sus parámetros de configuración y lo adjunta a la fuente de trafico. Antes, la fuente y el sumidero han sido conectados. Ahora, se adjuntan las fuentes de tráfico (con diferentes tasas pico) a *n0*, *n1*, y *n2* y entonces, se los conecta a los tres sumideros de tráfico en el nodo *n4*, que ha sido creado previamente.

```
set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
$ns attach-agent $n4 $sink0
$ns attach-agent $n4 $sink1
$ns attach-agent $n4 $sink2

set source0 [attach-expoo-traffic $n0 $sink0 200 2s 1s 100k]
set source1 [attach-expoo-traffic $n1 $sink1 200 2s 1s 200k]
set source2 [attach-expoo-traffic $n2 $sink2 200 2s 1s 300k]
```

En este ejemplo se usan objetos *Agent/Loss Monitor* como sumideros de tráfico, ya que ellos almacenan la cantidad de datos recibidos, lo cual puede ser usado para calcular el ancho de banda.

- *Grabando datos en archivos de salida*

Para este ejemplo, se abren tres archivos de salida. Para ello, estas líneas deben aparecer en el principio del *script Tcl*:

```
set f0 [open out0.tr w]
set f1 [open out1.tr w]
set f2 [open out2.tr w]
```

Como se había mencionado anteriormente, estos archivos deben ser cerrados en un procedimiento llamado "*finish*" y además llamar al programa *Xgraph* para desplegar los resultados. También se puede adaptar el tamaño de la ventana (800x400) al tamaño de la pantalla.

```
proc finish {} {
        global f0 f1 f2
        #Close the output files
        close $f0
        close $f1
        close $f2
        #Call xgraph to display the results
        exec xgraph out0.tr out1.tr out2.tr -geometry 800x400 &
        exit 0
}
```

Ahora, se puede escribir el procedimiento que escribe los datos actuales a los archivos de salida.

```
proc record {} {
        global sink0 sink1 sink2 f0 f1 f2
        #Get an instance of the simulator
        set ns [Simulator instance]
        #Set the time after which the procedure should be called
again
        set time 0.5
        #How many bytes have been received by the traffic sinks?
        set bw0 [$sink0 set bytes_]
        set bw1 [$sink1 set bytes_]
        set bw2 [$sink2 set bytes_]
        #Get the current time
        set now [$ns now]
        #Calculate the bandwidth (in MBit/s) and write it to the files
        puts $f0 "$now [expr $bw0/$time*8/1000000]"
        puts $f1 "$now [expr $bw1/$time*8/1000000]"
        puts $f2 "$now [expr $bw2/$time*8/1000000]"
        #Reset the bytes_ values on the traffic sinks
        $sink0 set bytes_ 0
        $sink1 set bytes_ 0
        $sink2 set bytes_ 0
          #Re-schedule the procedure
        $ns at [expr $now+$time] "record"
}
```

Este procedimiento lee el número de *bytes* recibidos desde los tres sumideros de tráfico. Luego calcula el ancho de banda (en MBits/s ó Mbps) y lo escribe en los tres archivos de salida, junto con tiempo actual. Luego, se ponen en cero los

valores del parámetro *bytes_* sobre los diferentes sumideros de tráfico y finalmente se vuelve a llamar a sí mismo el procedimiento "*record*".

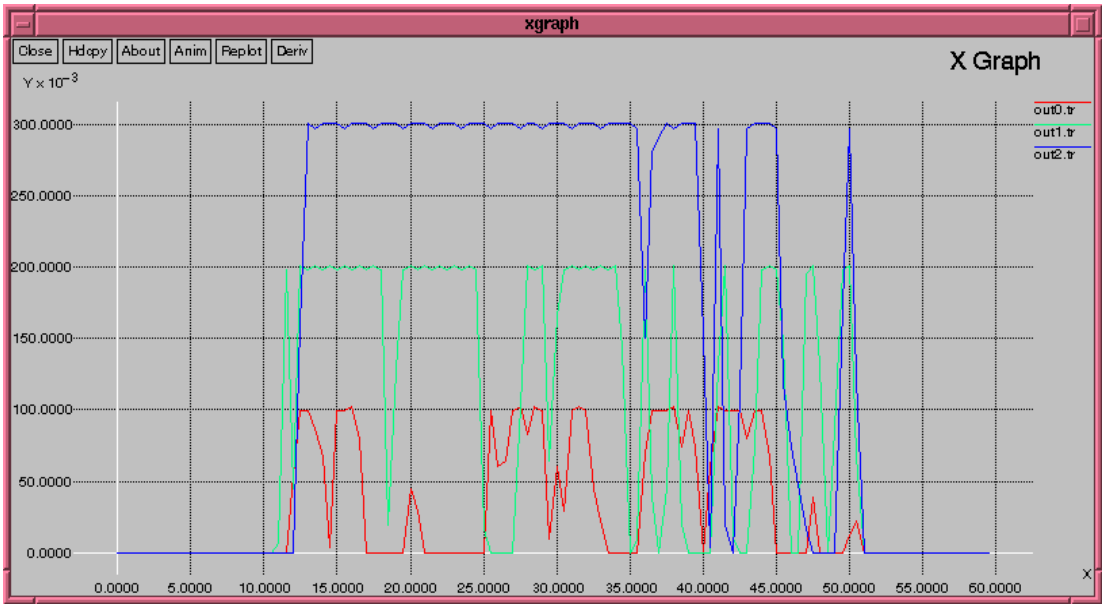
- *Corriendo la simulación*

Se pueden programar los siguientes eventos:

```
$ns at 0.0 "record"
$ns at 10.0 "$source0 start"
$ns at 10.0 "$source1 start"
$ns at 10.0 "$source2 start"
$ns at 50.0 "$source0 stop"
$ns at 50.0 "$source1 stop"
$ns at 50.0 "$source2 stop"
$ns at 60.0 "finish"
$ns run
```

Primero, se llama al procedimiento "*record*" y después, él mismo se reprograma periódicamente cada 0.5 segundos. Luego, las tres fuentes de tráfico comienzan a generar tráfico a los 10 segundos y detienen a los 50 segundos el envío de datos. A los 60 segundos se llama el procedimiento "*finish*".

Cuando se corre la simulación, una ventana de *Xgraph* muestra algo similar a esto:



Examen

1. Qué pasos deben seguirse para utilizar la herramienta Xgraph para graficar una variable?
2. Cómo se puede calcular el ancho de banda de un flujo de datos en NS?
3. Por qué es necesario utilizar un agente Loss/Monitor en la simulación?
4. Investigar: Qué otros parámetros permite monitorear el agente Loss/Monitor, además del parámetro bytes_