EVALUACIÓN Y DOCUMENTACIÓN DE LA HERRAMIENTA NETWORK SIMULATOR

Ing. LEONARDO ANDRES MOZO JACOME

Director:
Ph.D OSCAR GUALDRON GONZALEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERIAS ELECTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
ESPECIALIZACIÓN EN TELECOMUNICACIONES
BUCARAMANGA
2004

EVALUACIÓN Y DOCUMENTACIÓN DE LA HERRAMIENTA NETWORK SIMULATOR

Ing. LEONARDO ANDRES MOZO JACOME

Monografía presentada como requisito
parcial para obtener el título de
especialista en telecomunicaciones

Ph.D OSCAR GUALDRON GONZALEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERIAS ELECTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
ESPECIALIZACIÓN EN TELECOMUNICACIONES
BUCARAMANGA
2004

DEDICATORIA

Gracias a mi familia que me ha apoyado, en especial a mis padres Enrique y Amparo ya que representan lo más valioso de mi ser. Al igual que mis hermanas Maria Ángela y Erika Tatiana, a mi hermano Alfredo Enrique y a mi hijo Andrés Felipe por convertirse en la base que sostiene este proyecto de ser lo que soy .

Abuela, gracias por estar conmigo siempre en el momento en que te he necesitado.

Lizbeth Liliana, tu cariño y comprensión hacen que el trasegar diario se llene de fuerza para apurar los pasos y así llegar a mi meta.

Adelita, gracias por tu abnegada colaboración, aunque ya no estés yo se que me guiaste por el camino correcto brindándome tu cariño, ternura y comprensión. Y aunque nunca te dije lo que representabas para mi, tu deseo de verme especialista ya está cumplido, además yo se que mi dicha es y será la misma tuya hasta el día que nos encontremos.

Inés y Victoria, haber llegado a ésta casa me trajo grandes alegrías, mil gracias por hacer parte de este logro, es de ustedes también.

Amigos: Víctor Deioxes, Fabián Alberto, Rixon Leonardo, Yamit Danilo, Arthurito, Mónica, Maira, Luz Daris, Walter, José Luis, Ramiro, Carlos, Juan Manuel, Leonardo, Rafa, Memo, Yhozep, Alvin... Es bonito saber que cuento con cada uno de ustedes y espero que tengan claro que en el momento en que me necesiten espero no defraudarlos, con ustedes viví gran parte de las experiencias como estudiante y puedo decir que los valoro por lo que son y espero de ustedes su reciproco sentimiento.

A todos mis familiares, ellos saben que lo digo de corazón esperando ver reflejado en este logro un pedacito de su valiosa estimación.

Leonardo Andrés Mozo Jácome

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

Pág.

TITULO:
EVALUACIÓN Y DOCUMENTACIÓN DE LA HERRAMIENTA NETWORK SIMULATOR.[*]

AUTORES:
MOZO JÁCOME, Leonardo Andrés [**]

PALABRAS CLAVES:
Network Simulator – TCL – Simulador de red – Linux – Redes – E3T.

DESCRIPCIÓN O CONTENIDO:

Network Simulator (NS) es un simulador de red discreto orientado a objetos, desarrollado en la Universidad de Berkely escrito en C++ y OTCL. NS es ante todo usado para la simulación de redes LAN y WAN. Esta monografía nace ante la necesidad de estudiar y evaluar dicha herramienta, para su posterior implantación en la Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones (E3T).

Los objetivos que guiaron el desarrollo de ésta, es dar a los nuevos usuarios una idea básica de cómo trabaja el simulador, cómo configurar redes para simularlas y cómo crear nuevos componentes de red, dando principalmente explicaciones breves y ejemplos simples basados en esta experiencia; además, se propone realizar el modelo de un segmento de la red de la Universidad Industrial de Santander (UIS), con el fin de generar diversos escenarios que conlleven al análisis de la misma. Sin embargo, todos los usos del simulador o posibles configuraciones para la simulación de redes no pueden ser cubiertas aquí.

El aporte fundamental de esta monografía está enmarcado dentro del apoyo y ayuda para los nuevos usuarios que se inician en esta herramienta y se considera como un primer aporte y punto de partida para el análisis del tráfico de la red de la UIS del grupo de investigación CPS (Conectividad y Procesado de Señales) de la E3T de la UIS.

---

[*] Monografía

[**] Facultad de Ingenierías Físico – Mecánicas. Especialización en Telecomunicaciones. OSCAR GUALDRÓN GONZÁLEZ

TITLE:
EVALUATION AND DOCUMENTATION OF THE NETWORK SIMULATOR TOOL.[*]

AUTHOR:
MOZO JÁCOME, Leonardo Andrés [**]

KEY WORDS: Network Simulator – TCL – Linux – Networks – E3T

DESCRIPTION OR CONTENT:

Network Simulator (NS) is a discreet net simulator oriented to objects, developed in the University of Berkely written in C++ and OTCL. NS is used above all for the simulation of nets LAN and WAN. This monograph is born in the face of the necessity of to study and to evaluate this tool, for its later implantation in the School of Electric, Electronic Engineering and Telecommunications (E3T).

The objectives that guided the development of this are, to give the new users a basic idea of how the simulator works, how to configure nets to simulate them and how to create new net components, giving brief explanations and simple examples based on this experience mainly; also, it is intended to realize the model of a segment of the net of the Industrial University of Santander (UIS), with the purpose of generating diverse scenarios that bear to the analysis of the same one. However, all the uses of the simulator or possible configurations for the simulation of nets cannot be covered here.

The fundamental contribution of this monograph is framed inside the support and help for the new users that begin in this tool and it is considered as a first contribution and starting point for the analysis of the traffic of the net of the UIS of the investigation group CPS (Connectivity and Processed of Signs) of the E3T of the UIS.

---

[*] Monograph

[**] Phisycs and Mechanicals enginnering Faculty. Specialization in Telecommunications. OSCAR GUALDRÓN GONZÁLEZ

INTRODUCCÓN

Network Simulator (NS) es un simulador de red discreto orientado a objetos, desarrollado en la Universidad de Berkely escrito en C++ y OTCL. NS es ante todo usado para la simulación de redes LAN y WAN. Aunque es fácil de usar cuando se tiene experiencia en el simulador, es muy difícil para usuarios principiantes, debido a los pocos manuales que existen. Hay mucha documentación escrita por los desarrolladores que hacen una explicación profunda del simulador, pero el estilo se orienta a usuarios expertos.

El propósito de esta monografía es dar a los nuevos usuarios una idea básica de cómo trabaja el simulador, cómo configurar redes para simularlas, dónde buscar información adicional acerca de los componentes de red en el código del simulador y cómo crear nuevos componentes de red, dando principalmente explicaciones breves y ejemplos simples basados en esta experiencia; sin embargo, todos los usos del simulador o posibles configuraciones para la simulación de redes no pueden ser cubiertas aquí.

Esta monografía se constituye como una ayuda para los nuevos usuarios que se inician en esta herramienta y se considera como un primer aporte y punto de partida para el análisis del tráfico de la red de la UIS del grupo de investigación CPS (Conectividad y Procesado de Señales) de la E3T de la Universidad Industrial de Santander.

1. INTRODUCCIÓN A NETWORK SIMULATOR

1.1 NETWORK SIMULATOR.

NS, una herramienta de OpenSource (fuente libre) que se ejecuta bajo ambiente Linux, es un simulador de eventos discretos enfocado a la investigación de redes y provee un soporte sustancial para la simulación de rutas, protocolos de multidifusión y protocolos IP, tales como UDP, TCP, RTP y SRM bajo redes (local y satélite) cableadas e inalámbricas. NS tiene muchas ventajas que la hacen una herramienta útil, tanto para el soporte de múltiples protocolos como la capacidad de detallar gráficamente el tráfico de la red. Adicionalmente, soporta varios algoritmos para enrutamiento en LAN´S y de colas incluyendo déficit round-robin y fifo.

NS comenzó como una variante del simulador de red REAL en 1989. REAL fue creado inicialmente para el estudio del comportamiento dinámico de flujo y el diseño de control de congestión en redes de intercambio de paquetes de datos. Actualmente NS es desarrollado por el grupo VINT, soportado por la agencia de investigación de proyectos avanzados de defensa (DARPA) con SAMAN y por NSF con CONCER, ambos en colaboración con otros investigadores incluyendo a ACIRI. NS está disponible para varias plataformas tales como: Linux, SunOS, FreeBSD y Solaris; además se ha desarrollado para ejecutarse bajo ambiente Windows. Claro está que en escenarios simples debería ejecutarse en cualquier computador razonable; sin embargo escenarios muy grandes requieren cantidades superiores de memoria. Adicionalmente NS requiere los siguientes soportes para ejecutarse: Tcl, Tk, Otcl, TclCL.

El propósito de este documento es hacer que el lector aprenda a utilizar de manera más fácil NS y NAM y cree sus propios escenarios de simulación para estas herramientas, además de adicionar nuevas funcionalidades a NS. El objetivo principal es que después de un corto tiempo el usuario sea capaz de manejar eficientemente NS. La Web es probablemente el mejor medio para encontrar la información que ayude en el proceso de aprendizaje.

1.1.1 Ejecución de NS.

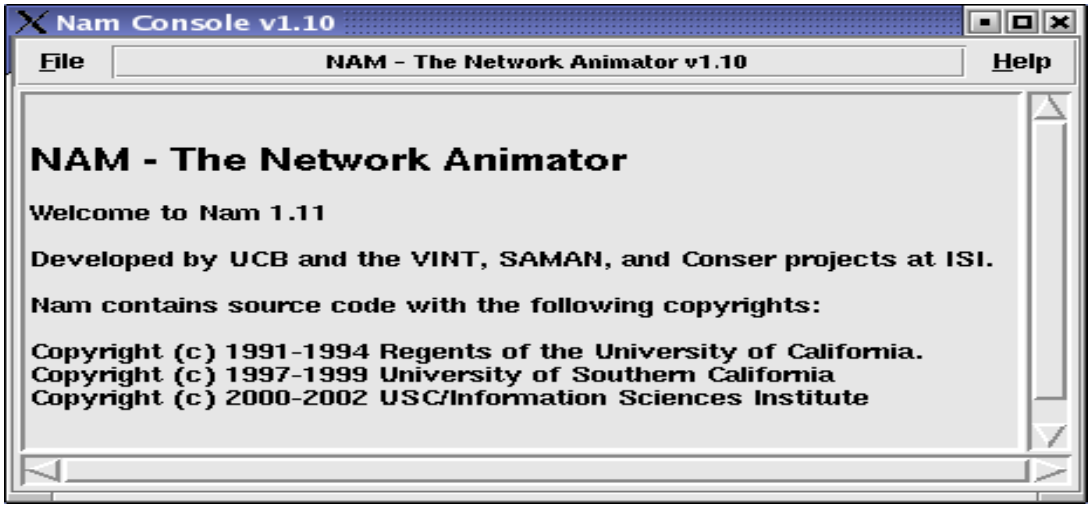
NS se ejecuta con el comando ./ns seguido del nombre del script asumiendo que el usuario se encuentra dentro del directorio donde está el ejecutable ns. El script tcl es un archivo que define un escenario de simulación. NS también se puede ejecutar sin argumentos entrando los comandos tcl directamente en el shell tcl. Todo lo demás depende del script tcl, el cual podrá crear una salida generando un archivo trace o ejecutando NAM para visualizar la simulación.

1.2 NETWORK ANIMATOR.

NAM es una herramienta de animación basada en TCL/TK para la visualización de trazas en la simulación de redes y el seguimiento de paquetes de datos. Normalmente el archivo de traza o de seguimiento lo genera NS y cuya extensión es .tr, el cual contiene una serie de parámetros que describen las características del tráfico (paquetes enviados, paquetes recibidos, tamaño del paquete, envío entre nodos, tiempo simulado, etc). La idea inicial de NAM, fue crear un animador que fuera capaz de leer grandes conjuntos de datos y ser suficientemente extensible para así usarse en cualquier situación de visualización de red. Bajo esta restricción NAM fue diseñado para leer comandos de eventos de simples animaciones desde un gran archivo de traza.

Con el fin de manejar grandes conjuntos de datos, una mínima cantidad de información se guarda en memoria. Los comandos de eventos se guardan en el archivo de traza y se vuelve a leer desde el archivo en cualquier momento si es necesario.

Figura 1. Ventana principal de NAM.



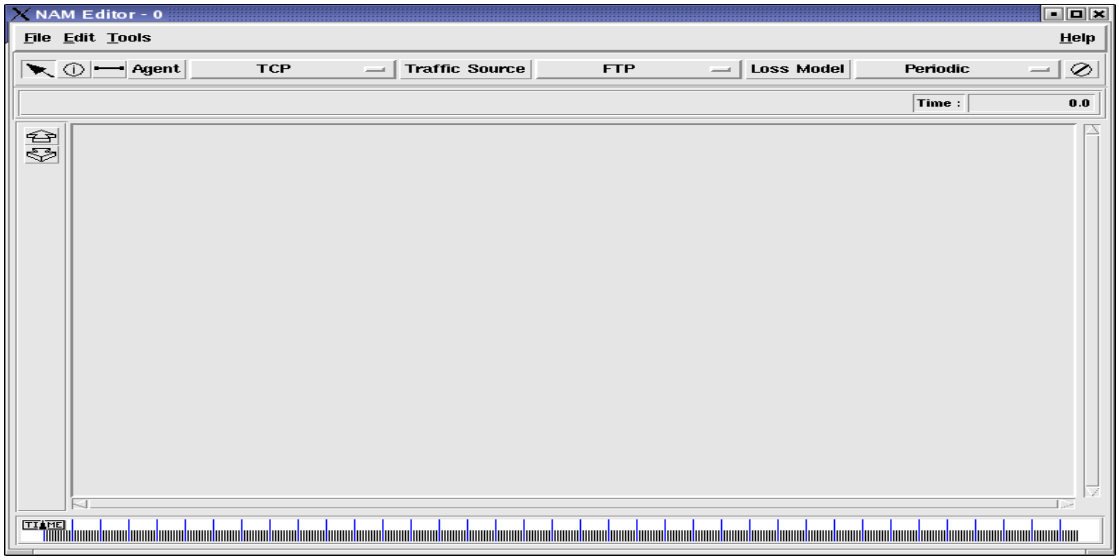
El primer paso para usar NAM es producir el archivo de traza. Este archivo contiene información de la topología, por ejemplo nodos, enlaces y trazas de paquetes.

Sin embargo, cualquier aplicación puede generar un archivo de traza NAM. Cuando el archivo de traza se genera, esta listo para ser animado por el NAM. Al inicio, en el NAM se lee el archivo de traza, se genera la topología, se despliega una ventana, se hace el diseño y se coloca en el tiempo 0.

A través de la interfaz de usuario, NAM provee el control de muchos aspectos de la animación.

También se encuentra la herramienta NAM Editor, ésta ayuda a desarrollar ejercicios visualizando y haciendo correctivos directamente cuando se creen simulaciones de redes con sus características como topologías, nodos, switches y enlaces.

Figura 2. Ventana principal de NAM Editor.



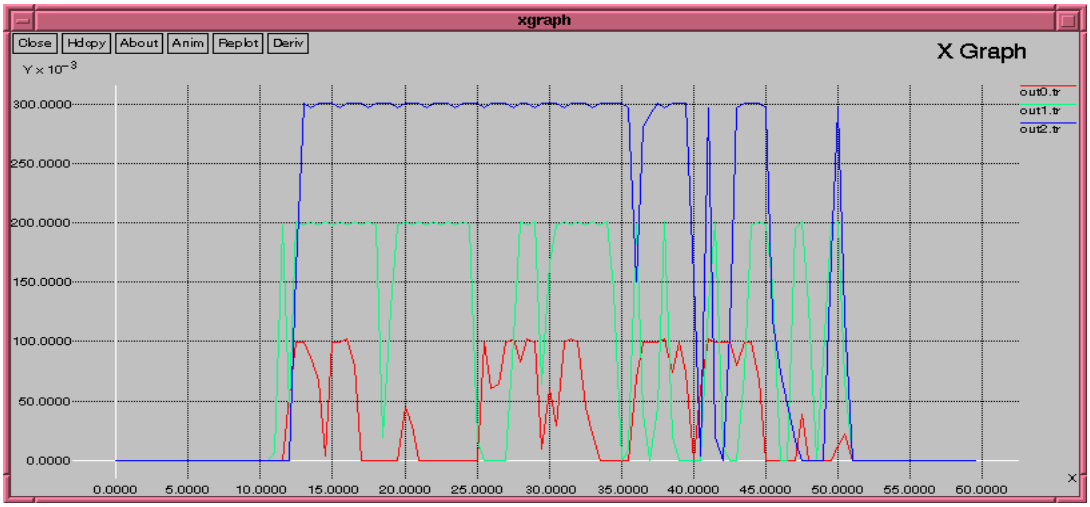
1.2.1 Ejecutando NAM.

Network Animator se ejecuta con el comando ./nam seguido del nombre del archivo que NS genera o puede generarse directamente por fuera del script de simulación tcl para la simulación que quiere visualizar.

1.3 XGRAPH.

Una parte de la herramienta NS es el Xgraph, un programa de visualización que puede ser usado para crear representaciones gráficas de los resultados de la simulación.

Figura 3. Ventana del XGRAPH.



En la figura anterior se observan 3 archivos de traza graficados (out1.tr, out2.tr y out3.tr), con el fin de mostrar el formato de este tipo de archivo se hará la interpretación de un segmento tipo trace, es sencillo y presenta el siguiente aspecto:

Figura 4. Segmento de seguimiento.

+	1	0	2	cbr	210	-------	0	0.0	3.0	0	0
-	1	0	2	cbr	210	-------	0	0.0	3.0	0	0
r	1.002336	0	2	cbr	210	-------	0	0.0	3.0	0	0
+	1.002336	2	3	cbr	210	-------	0	0.0	3.0	0	0
-	1.002336	2	3	cbr	210	-------	0	0.0	3.0	0	0
+	1.00375	0	2	cbr	210	-------	0	0.0	3.0	1	1
-	1.00375	0	2	cbr	210	-------	0	0.0	3.0	1	1
r	1.006086	0	2	cbr	210	-------	0	0.0	3.0	1	1

La primera columna contiene el tipo de evento: (**+**) significa que un paquete ingresa a una cola, (**-**) significa que un paquete sale de una cola, y (**r**) significa que un paquete se recibe en un nodo. La siguiente columna muestra el instante de tiempo simulado en el que el evento ocurre. Las dos siguientes columnas muestran los nodos que participan en el evento. La siguiente columna contiene el nombre (cbr en este caso) del tipo de paquete de seguimiento. Lo siguiente es el tamaño del paquete de seguimiento.

A continuación viene un campo dedicado a banderas, que pueden indicar notificaciones relacionadas con la congestión, o con la prioridad de los paquetes, etc. En el segmento mostrado no hay banderas activadas. A continuación viene el identificador de flujo, los dos campos siguientes muestran los identificadores de los nodos fuente y destino (0.0 y 0.3 en el caso de la traza mostrada). El penúltimo campo es un número de secuencia, solamente usado para paquetes generados por fuentes que requieran que ese número vaya incrementándose de paquete a paquete, y la última columna es el identificador del paquete.

2. INSTALACIÓN DE NETWORK SIMULATOR

El procedimiento de instalación de la herramienta Network Simulator (NS), se hará bajo ambiente LINUX debido a las disposiciones legales en cuanto a que es un software de libre distribución. Algunas recomendaciones para la lectura del documento son:

Toda instrucción se colocará con letra en negrita.
Los mensajes de información visualizados serán escritos en letra cursiva.

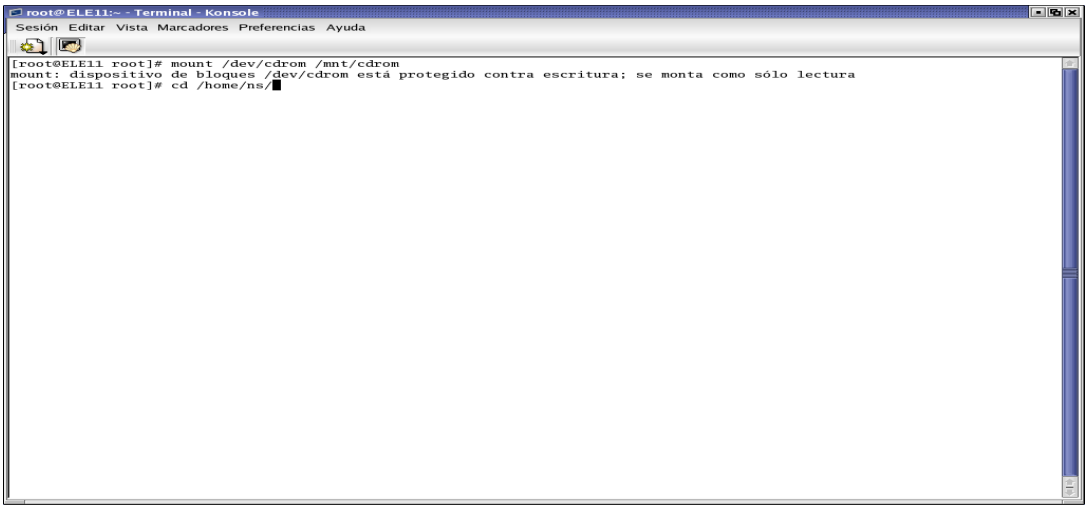
Inicialmente se tiene el archivo donde residen los fuentes de la herramienta, en este caso se utilizó el archivo ns-allinone-2.27.tar.gz (todo en uno). Aunque también se pueden instalar todos sus componentes (NS, NAM, XGRAPH, TCL/TK, OTCL) individualmente.

El proceso de instalación de Network Simulator se hará a través de los siguientes pasos:

1. Teniendo en cuenta que el archivo se encuentra en un cdrom, se debe abrir una terminal y se monta el cdrom como lo muestra la siguiente instrucción:

[root@ELE11 root]# mount /dev/cdrom /mnt/cdrom
Mount: dispositivo de bloques /dev/cdrom está protegido contra escritura; se monta como solo lectura

Figura 5. Instrucción para montar el cdrom.



Luego se debe copiar el archivo fuente ns-allinone-2.27.tar.gz en un nuevo directorio (se sugiere como nombre para el directorio : “ns”, el cual puede ser ubicado dentro de /home) como lo muestran las figuras 6, 7 y 8.

Figura 6. Creación de un nuevo directorio.

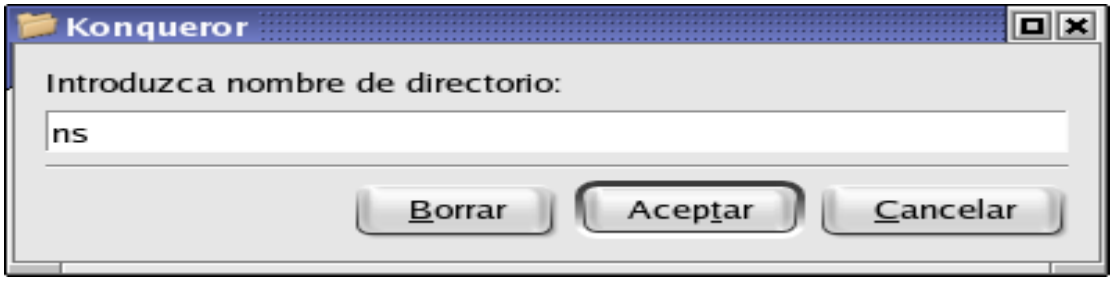


Figura 7. Visualización del proceso de copia del archivo.

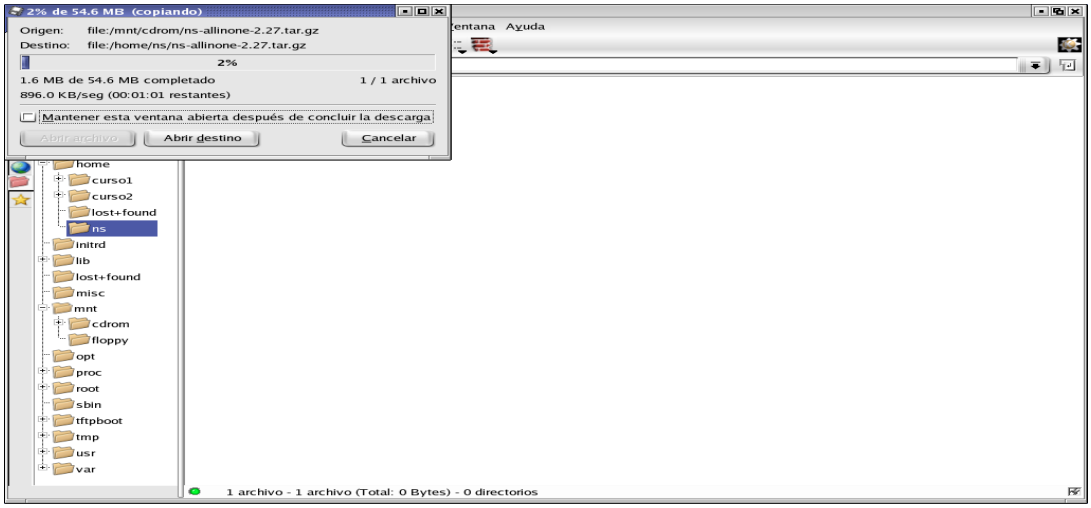
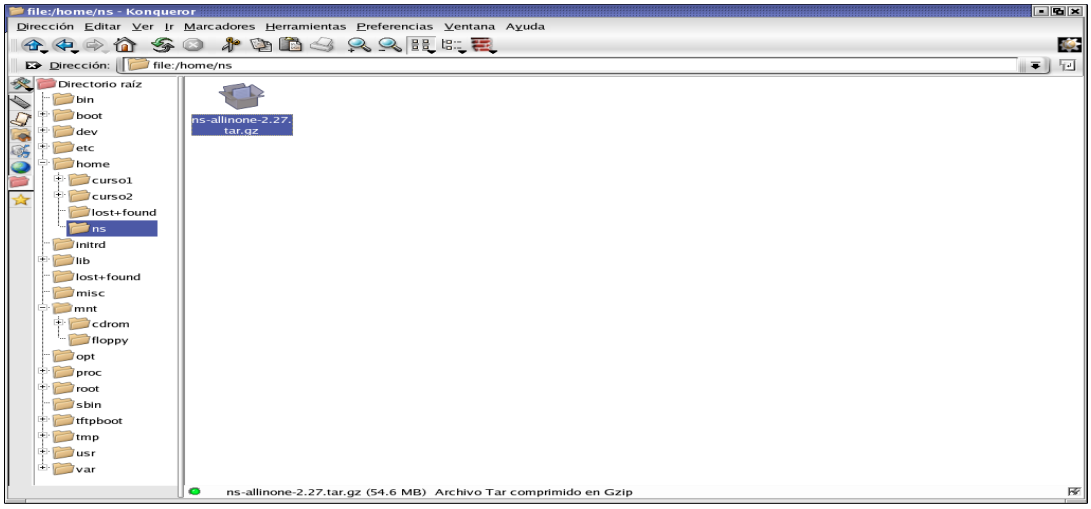


Figura 8. Visualización del archivo copiado en el nuevo directorio.



Una vez terminado el proceso de copia del archivo ns-allinone-2.27.tar.gz se procede a descomprimirlo, inicialmente con la aplicación gunzip y luego con tar.

2. En la terminal, dentro del directorio ns creado se descomprime el archivo con gunzip mediante la siguiente instrucción:

[root@ELE11 ns]# gunzip -df ns-allinone-2.27.tar.gz

Luego se debe descomprimir el archivo .tar que resulta de ejecutar la anterior instrucción, a través del siguiente comando:

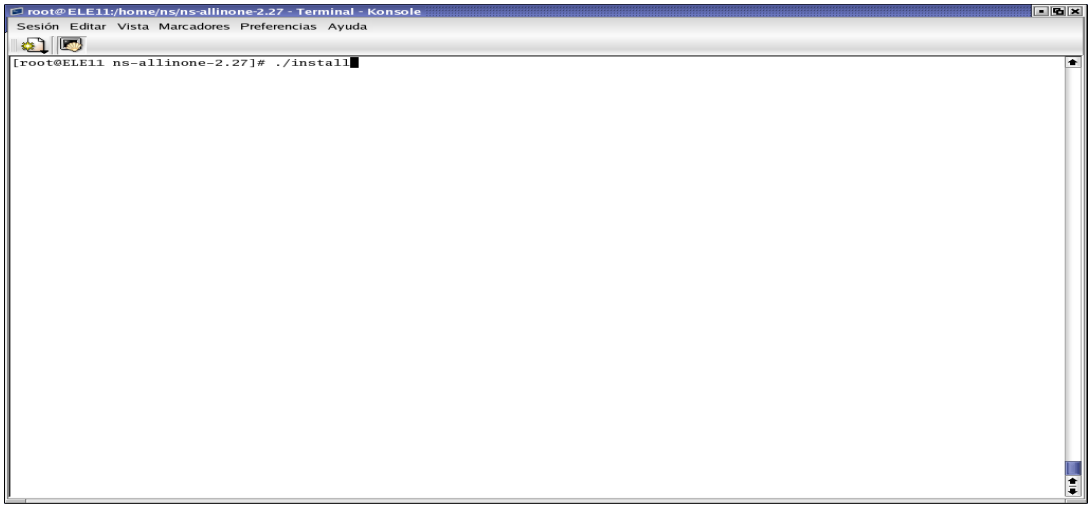
[root@ELE11 ns]# tar -xf ns-allinone-2.27.tar

3. De los dos pasos anteriores, se obtiene un directorio de salida llamado ns-allinone-2.27, donde residen todos los archivos fuente y ejecutables de la herramienta. Ahora se debe ubicar dentro del directorio ns-allinone-2.27, con la siguiente instrucción:

[root@ELE11 ns]# cd ns-allinone-2.27

4. Finalmente, para proceder a instalar Network Simulator se ejecuta el siguiente comando en la terminal: **./install**, como lo ilustra la siguiente figura.

Figura 9. Visualización del comando ./install.



En las figuras 10 y 11 se muestran algunos de los resultados del proceso de instalación del NS. Como esta versión es “todo en uno”, el detalla uno a uno cada comprimido.

Figura 10. Visualización del inicio de la instalación del NS.

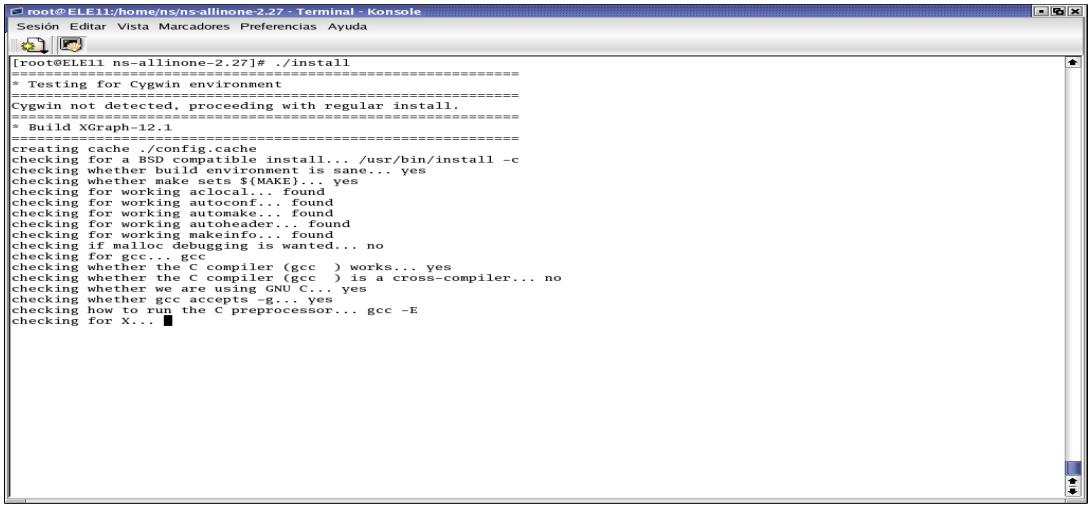
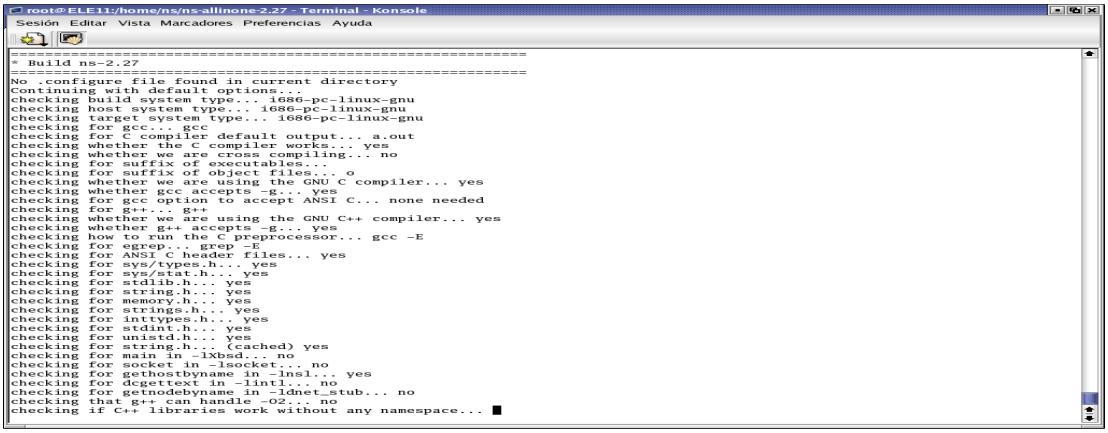
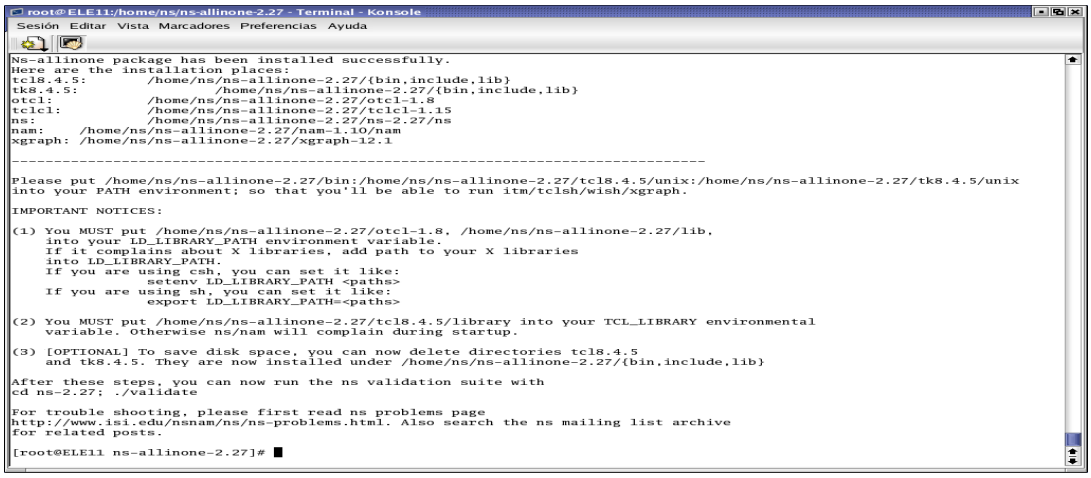


Figura 11. Visualización en detalle del componente ns-2.27.



La siguiente figura muestra el momento en que el NS termina la instalación de los paquetes quedando el “prompt” en la línea de comandos, en esta instancia ya se puede ejecutar la herramienta digitando la instrucción **./ns**.

Figura 12. Visualización del final de la instalación.

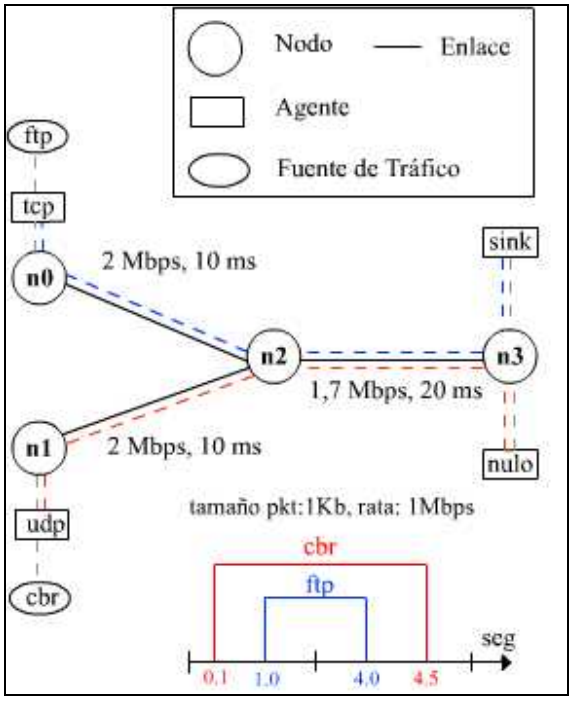


En la figura anterior se muestran notas importantes en la finalización de la instalación. Estas notas son configuraciones de variables de entorno de Linux.

3. SIMULACIÓN DE RED SIMPLE

Esta sección muestra un script simple de simulación y explica que hace cada línea. El ejemplo 1 es un script OTcl que crea la configuración simple de red y ejecuta el escenario de simulación de la figura 12.

Figura 13. Topología simple de red y escenario de simulación.



Esta red consiste en 4 nodos (n0, n1, n2 y n3) como se muestran en la figura anterior. Los enlaces duplex entre n0 - n2, y n1 - n2, tienen un ancho de banda de 2 Mbps y 10 ms de retardo. El enlace duplex entre n2 y n3 tiene un ancho de banda de 1,7 Mbps y 20 ms de retardo. Cada nodo usa una cola DropTail, del cual el máximo tamaño es 10. Un agente TCP es adjuntado a n0 y una conexión es

establecida a un agente sink adjuntado a n3. Por defecto, el máximo tamaño de un paquete que un agente TCP puede generar es 1 KByte. Un agente sink genera y envía los paquetes ACK (acuses) al emisor (agente TCP) y libera los paquetes recibidos. Un agente UDP que es adjuntado a n1 se conecta al agente nulo adjunto a n3. Un agente nulo libera los paquetes recibidos. Un generador de tráfico FTP y CBR son adjuntados a los agentes TCP y UDP respectivamente, y el CBR se configura para generar paquetes de 1 KByte a una rata de 1 Mbps. El CBR se programa para iniciar en 0,1 segundos y parar en 4,5 segundos, y FTP se programa para iniciar en 1,0 segundos y terminar en 4,0 segundos.

El código fuente del script ejemplo1.tcl se observa a continuación:

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

#Define diferentes colores para flujos de datos (en el NAM)
$ns color 1 Blue
$ns color 2 Red

#Abre el archivo de traza NAM
set nf [open out.nam w]
$ns namtrace-all $nf

#Define el procedimiento 'finish'
proc finish {} {
        global ns nf
        $ns flush-trace
        #Cierra el archivo de traza NAM
        close $nf
        #Ejecuta NAM en el archivo de traza
        exec ./nam out.nam &
```

```
        exit 0
}

#Crea cuatro nodos
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Crea enlaces entre los nodos
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Configura el tamaño de cola del enlace (n2-n3) a 10
$ns queue-limit $n2 $n3 10

#Da la posición del nodo (en el NAM)
$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

#Monitorea la cola para el enlace (n2-n3). (en el NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5

#Configura una conexión TCP
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
```

#Configura una conexión FTP sobre TCP

```
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
```

#Configura una conexión UDP

```
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2
```

#Configura una conexión CBR sobre UDP

```
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
```

#Programa eventos para los agentes CBR y FTP

```
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
```

 # Desconecta los agentes tcp y sink

```
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"
```

#Llama el procedimiento finish después de 5 segundos de simulación
$ns at 5.0 "finish"

#Imprime el tamaño e intervalo del paquete CBR
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

#Ejecuta la simulación
$ns run

La siguiente es la explicación del script anterior. En general, un script NS inicia haciendo una instancia del objeto simulador.

set ns [new Simulator]: genera una instancia del objeto simulador NS, y lo asigna a la variable ns. La línea hace lo siguiente:

- ◊ Inicializa el formato del paquete
- ◊ Crea un programador
- ◊ Selecciona el formato de dirección por defecto

El objeto "Simulador" tiene funciones miembro que hacen lo siguiente:

- ◊ Crea objetos compuestos tales como nodos y enlaces
- ◊ Conecta los objetos de componentes de red creados (ejemplo. adjunta-agente)
- ◊ Coloca los parámetros de componente de red (principalmente para objetos compuestos)
- ◊ Crea conexiones entre agentes (ejemplo. hace la conexión entre un "tcp" y un "sink")
- ◊ Especifica las opciones de visualización de NAM

$ns color fid color: es para configurara el color de los paquetes para un flujo especificado por el identificador de flujo (fid). Esta función miembro del objeto simulador es para visualizar los colores con NAM, y no tiene efectos sobre la actual simulación.

$ns namtrace-all file-descriptor: Esta función miembro llama al simulador para grabar las trazas de la simulación en el formato de entrada NAM. También da el nombre del archivo que la traza será escrita después por el comando $ns flush-trace. Similarmente, la función miembro trace-all es para grabar la traza de la simulación en un formato general.

proc finish {}: Se llama luego que la simulación ha terminado por el comando $ns at 5.0 "finish". En esta función, se especifican procesos posteriores a la simulación.

set n0 [$ns node]: La función miembro node crea un nodo. Un nodo en NS es un objeto compuesto hecho de dirección y clasificadores de puerto. Los usuarios pueden crear un nodo, estableciendo los objetos dirección y clasificador de puerto separadamente y conectándolos a la vez. Sin embargo, esta función miembro del objeto simulador hace el trabajo mas fácil.

$ns duplex-link node1 node2 bandwidth delay queue-type: Crea dos enlaces simples de un ancho de banda y retardo especificado, y conecta los dos nodos descritos. En NS, la cola de salida de un nodo se implementa como una parte de un enlace, por consiguiente los usuarios deberían especificar el tipo de cola cuando se crean los enlaces. En el anterior script de simulación, se usa una cola DropTail. Si el lector quiere usar una cola RED, simplemente remplace la palabra DropTail con RED. Igual que un nodo, un enlace es un objeto compuesto, y los usuarios pueden crear sus subobjetos y conectarlos a los nodos. Algo para notar es que se puede

insertar módulos de error en un componente para simular un enlace perdido (actualmente los usuarios pueden hacer e insertar cualquier componente de red).

$ns queue-limit node1 node2 number: Esta línea configura el límite de la cola de los dos enlaces simples que conecta el nodo 1 y el nodo 2 al número especificado.

$ns duplex-link-op node1 node2 ...: Esta línea configura el canal de comunicación para los dos enlaces, nodo1 y nodo2.

Luego que la configuración básica de la red está hecha, lo siguiente es hacer la configuración de los agentes de tráfico tales como TCP Y UDP, las fuentes de tráfico tales como FTP Y CBR y adjuntarlos a los nodos y a los agentes respectivamente.

set tcp [new Agent/TCP]: Esta línea muestra como crear un agente TCP. Pero en general, los usuarios pueden crear cualquier agente o fuentes de tráfico de esta forma. Los agentes y las fuentes de tráfico son de hecho objetos básicos (no son objetos compuestos), que en su mayoría son implementados en C++ y enlazados a OTcl. Por consiguiente, no hay funciones miembro del objeto simulador específico que creen estas instancias de objeto. Para crear agentes o fuentes de tráfico, un usuario debería conocer los nombres de las clases de estos objetos (Agent/TCP, TCPSink, Application/FTP, etc).

$ns attach-agent node agent: La función miembro attach-agent adjunta un objeto agente al objeto nodo. Lo que ésta función hace es llamar la función miembro del nodo específico, la cual adjunta el agente dado a él mismo. Por lo tanto, un usuario puede hacer lo mismo con $n0 attach $tcp. Similarmente, cada objeto agente tiene

una función miembro attach-agent que adjunta un objeto de fuente de tráfico a él mismo.

$ns connect agent1 agent2: Después de que los agentes sean creados, se comunicarán entre sí, luego se establece la conexión lógica de red entre ellos. Esta línea establece una conexión de red configurando la dirección de destino y el par de direcciones del puerto para la otra red.

Asumiendo que toda la configuración de red está hecha, lo siguiente es programar el escenario de simulación. El objeto simulador tiene muchas funciones miembro programadas. Sin embargo, la más usada es la siguiente:

$ns at time "string": Esta función miembro de un objeto simulador hace el programador (scheduler_ es la variable que apunta al objeto programador creado con el comando [new Scheduler] al principio del script) para programar la ejecución de la cadena especificada en un tiempo de simulación dado. Por ejemplo, $ns at 0.1 "$cbr start" hará que el programador ejecute la función miembro del objeto fuente de tráfico CBR, la cual inicia el CBR para transmitir datos. En NS, normalmente una fuente de tráfico no transmite los datos actuales, pero notifica al agente fundamental que tiene una cantidad de datos para transmitir, y el agente, que sabe cuantos datos transferir, crea paquetes y los envía.

Después que se hace toda la configuración de la red, se programa un procedimiento posterior a la simulación, finalmente la simulación se puede ejecutar mediante la instrucción **$ns run**.

3.1 RESUMEN DE OBJETOS Y ORDENES QUE SE UTILIZAN EN LA HERRAMIENTA NETWORK SIMULATOR

Tabla 1. Lista de objetos y órdenes.

Set ns [new simulator]	Inicializa la simulación
Open archivo tipo_de_acceso	Permite abrir los archivos que se van a usar para generar las trazas del modelo el tipo de acceso en la practica será w (escritura), Los archivos abiertos se deben cerrar con close.
Set variable_valor	Asigna un valor a una variable, estas se pueden consultar prefijando con $ el nombre de la variable. Así $var devolvería el valor que contiene la variable var.
[...]	El efecto de encerrar con corchetes una sentencia equivale a que interesa el valor que devuelve.
Puts	Escribe datos (cadena de caracteres, variables...) en la salida estándar.
Expr	Evalúa la expresión que le sigue, ejemplo: expr 10*5.
Exec programa_argumentos	Ejecuta el programa que se indique
$objeto **set** var_valor	Asigna un valor al miembro var del objeto sobre el que se hace la operación, si no se especifica un valor, la sentencia devuelve el valor actual almacenado en la variable var.
Agente/UDP **Agente/ Nul**	Clases de objetos agentes, **Agent/Null** es un sumidero de tráfico, sin ninguna capacidad adicional. **Agent/TCP** y **TCPSink,** que se utilizan para simular transferencias TCP unidireccionales en los datos (los datos se

Agente/TCP **Agente/TCPSink**	originan en el agente TCP y terminan en TCPSink. Obviamente hay tráfico de ACK en sentido contrario. Si se desea hacer transferencias TCP de datos vi direccionales, se debe usar un objeto de clase agent/FullTcp en ambos extremos y ocupa más recursos que los otros.
Aplicación/Trafico/CBR	Clase de objeto generador de tráfico de tasa constante. Los paquetes también tienen tamaños constantes. Miembros de un objeto de la clase: **Rate_** Tasa de envío. **Interval_** (opcional) intervalo entre paquetes **Packetsize_** Tamaño de los paquetes. **Random_** Si es distinto de 0, se introduce una cierta variación en los instantes de transmisión de los paquetes. **Maxpkts_** Máximo número de paquetes a enviar. Por defecto es 2^{28}.
Aplicación/FTP	Clase de objeto generador de tráfico de transferencia de archivos. Produce una transferencia masiva de datos que un objeto TC debe hacer legar al destino. Si FTP es un objeto de clase Aplicación/FTP, se tiene: $**ftp start.** Se produce un máximo de paquetes. **$ftp produce N.** Se produce N paquetes instantáneamente. $**ftp stop.** Dejan de producirse paquetes. $**ftp attach agent.** Asocia el objeto FTP a un agente. $**ftp producemore c.** Se producen c paquetes

	adicionales. Como variable miembro del objeto FTP se tiene **maxpkts_** máximo número de paquetes que se van a producir.
$ **ns now**	Guarda huella del tiempo en una simulación.
$ **ns halt**	Detiene o hace una pausa en el tiempo de simulación.
$ **ns at**	Fija un evento (qué normalmente es un pedazo de código) para ser ejecutado en un tiempo específico.
# **;#**	Se utilizan para introducir comentarios en el texto del programa, si el primer caracter es # se interpreta como comentario. Donde la línea presenta una orden, se debe colocar el (punto y coma).
$ **ns simples-link**	**nodo0-nodo1,** Genera enlaces simples entre los nodos 0-1 **bandwidth,** Ancho de banda. **Delay** Retardo de propagación. **queue-type,** tipo de cola.
$**src_(0) publish**	Orden para empezar una fuente del ping del (remitente).
Set **src [new Application/DiffApp/PingSender]**	Esta orden se usa para crear una aplicación del ping-remitente.
Set **snk [new Application/DiffApp/PingReceiver**]	Esta orden se usa para crear una aplicación del ping-receptor.
LossMonitor	Es un sumidero del paquete que verifica pérdidas.

3.2 QUÉ SE PUEDE HACER CON NETWORK SIMULATOR?.

Network Simulator es una herramienta que por su escasa documentación y difusión aparenta ser menos compleja de lo que en realidad es. Debido a la dispersión de las necesidades de cada quien y a que los desarrolladores de esta herramienta no la unifican en un solo sitio sino que lo hacen libremente en la red, se encuentran muchas aplicaciones hechas para las necesidades de cada desarrollador mas no para Ns en general.

Algunas de estas aplicaciones y el campo de implementación que corresponde para las mismas lo definimos a continuación :

3.2.1 Enrutamiento.

Unicast: para especificar la estrategia o protocolo de enrutamiento unicast de una simulación, el script de usuario requiere de un comando dado. La estrategia de enrutamiento es el mecanismo por el cual ns calculará rutas en una simulación. En ns hay cuatro estrategias de enrutamiento: Static, Session, Dynamic y Manual. Por el contrario, un protocolo de enrutamiento es la realización de un algoritmo especifico. Actualmente, el enrutamiento static y session usan el algoritmo Dijkstra's all-pairs SPF; el algoritmo distribuido Bellman-Ford es un tipo de estrategia de enrutamiento dinámico que actualmente se encuentra implementado. En ns, no se distinguen entre estrategia y protocolo para enrutamiento static y session, considerándolos simplemente protocolos.

En enrutamiento dinámico, cada nodo puede ser ejecutado por más de un protocolo de enrutamiento; de esta manera, más de un protocolo puede tener una ruta al mismo destino. Además, cada protocolo adjunta un valor de preferencia a cada una

de sus rutas. Estos valores son enteros no negativos en el rango de 0..255. El mas bajo valor, es el preferido por la ruta. Cuando múltiples agentes de protocolos de enrutamiento tienen una ruta al mismo destino, la ruta más escogida se selecciona y se instala en la tabla de reenvío del nodo. Si más de un agente presenta la ruta más corta, el agente recurre a la métrica para determinar cual camino elegir. A la ruta mínima del protocolo escogido se le llama ruta “candidata”. Si existen múltiples rutas candidatas para el mismo o diferentes protocolos, entonces, una de las rutas de los agentes se selecciona de forma aleatoria.

Multicast: el transporte multicast requiere un incremento a los nodos y enlaces en la topología. Además, el usuario debe especificar los requerimientos multicast a la clase Simulator antes de crear la topología. Se puede hacer de dos formas:

```
set ns [new Simulator -multicast on]
```

o

```
set ns [new Simulator]
$ns multicast
```

Cuando las extensiones multicast son habilitadas, los nodos serán creados con clasificadores y replicadores adicionales para el transporte, y los enlaces contendrán elementos para asignar etiquetas a todos los paquetes que llegan a un nodo. Una estrategia de enrutamiento multicast es el mecanismo por el cual el árbol de distribución es calculado en la simulación. Ns soporta tres estrategias de enrutamiento multicast: centralizada, modo denso (DM) y modo de árbol compartido (ST). Además, existe una estrategia llamada modo bidireccional de árbol compartido que se encuentra en experimentación.

Multicast centralizado: es un modo de implementación escaso similar a PIM-SM. Un agente de cómputo centralizado se usa para calcular el transporte y configurar el estado multicast en los nodos relevantes como nuevos receptores asociados al

grupo. Los paquetes de datos desde los emisores a un grupo son unicast al punto de reunión (RP), aún si no existen receptores para el grupo.

Modo denso: dependiendo del valor de la clase DM CacheMissMode, se puede ejecutar en dos modos. Si CacheMissMode se establece a pimdm (por defecto), las reglas de transporte como PIM-DM son usadas. Alternativamente, CacheMissMode se puede establecer a dvmrp . La principal diferencia entre estos dos modos es que DVMRP mantiene una relación padre-hijo en medio de los nodos para reducir el número de enlaces bajo el cual los paquetes son emitidos. La implementación trabaja sobre enlaces punto a punto, como también a LANs y se adapta a redes dinámicas.

Modo de árbol compartido: la variable vector RP_ indexado por un grupo determina cual nodo es el RP para un grupo en particular. Por ejemplo:

```
ST set RP_($group) $node0
$ns mrtproto ST
```

Cuando la simulación inicia, el protocolo crea e instala objetos encapsuladores en los nodos que tienen emisores multicast, objetos decapsuladores en los RPs y los conecta. Para asociar un grupo, un nodo envía un mensaje hacia los RP del grupo. Para salir del grupo, se envía un mensaje caído. El protocolo no soporta cambios dinámicos ni LANs.

3.2.2 Transporte.

Agente UDP: un agente UDP acepta datos en una variable de tamaño pequeño de una aplicación y segmenta los datos si es necesario. Los paquetes UDP contienen un número secuencial y un tiempo de marca RTP. Aunque los paquetes reales UDP

no contienen números de secuencia o tiempos de marca, este número de secuencia no incurren en algún aviso simulado y puede ser útil para el análisis de archivos de traza o para la simulación de aplicaciones basadas en UDP.

Agente TCP: hay dos tipos principales de agentes TCP: agentes de un camino y de dos caminos. Los agentes de un camino se subdividen en un conjunto de emisores TCP (los cuales obedecen a congestión y error en las técnicas de control) y receptores (sumideros). El agente de dos caminos es simétrico en el sentido que representa al emisor y al receptor.

Agente SCTP: los agentes SCTP (protocolo de transmisión de control de flujo) son todos de dos caminos, los cuales significan que ellos son simétricos en el sentido en que cada instancia representa a un emisor o a un receptor.

Agente SRM: adiciona los encabezados SRM, establece la dirección de destino al grupo multicast y envía el paquete a su objetivo. El encabezado SRM contiene el tipo de mensaje, la identidad del emisor, el número de secuencia del mensaje y (para el control de mensajes), la ruta por el cual el mensaje está siendo enviado. Cada parte de los datos en SRM se identifica como (identificador del emisor, número de secuencia del mensaje).

3.2.3 Programación y manejo de colas.

Las colas representan localizaciones donde los paquetes pueden ser mantenidos o perdidos. La programación de paquetes se refiere a la decisión de procesos usados para seleccionar cuales paquetes deberían ser servidos o perdidos. El manejo del buffer se refiere a cualquier disciplina usada para regular la ocupación de una cola en particular. Ns soporta colas droptail (FIFO), manejo del buffer RED (Random

Early Detection), CBQ (Class-Based Queueing, incluyendo una prioridad y programación round-robin), y variantes de colas normales incluyendo, colas normales (FQ), colas normales estocásticas (SFQ) y déficit Round-Robin (DRR). Para el caso donde el elemento delay (retardo) es flujo abajo de una cola, ésta debe ser bloqueada hasta que sea habilitada por su vecino flujo arriba. Este es el mecanismo por el cual el retardo de transmisión es simulado.

Además, las colas pueden ser forzadas a bloquearse o desbloquearse en tiempos arbitrarios por sus vecinos. Los paquetes perdidos son implementados de tal manera que las colas contienen un "destino perdido"; es decir, un objeto que recibe todos los paquetes perdidos por una cola. Esto puede ser útil para guardar estadísticas sobre los paquetes perdidos.

3.2.4 Manejo de redes móviles inalámbricas y satelitales:

Redes inalámbricas: el modelo inalámbrico consiste del MobileNode en el núcleo, con características adicionales de soporte que permiten simulaciones de redes multi-hop ad-hoc, LANs inalámbricas, etc. El objeto MobileNode es un objeto dividido. La clase MobilNode de C++ es derivada de la clase padre Node. Un MobileNode es el objeto Node básico con funciones adicionales de nodos móviles e inalámbricos, con habilidades para desplazarse en una topología dada, habilidades para recibir y transmitir señales desde y hacia un canal inalámbrico etc. La principal diferencia entre ellos, es que el nodo móvil no está conectado por medio de enlaces a otros nodos o a nodos móviles.

Redes satelitales: simulaciones exactas de redes satelitales requieren un detallado modelamiento de características de radio frecuencia (interferencia, desvanecimiento), interacciones de protocolos (interacciones de errores violentos sobre el enlace con código de verificación de error) y efectos orbitales de segundo

orden (presesión, anomalías gravitacionales, etc). Sin embargo, con el fin de estudiar las características de redes satelitales desde una perspectiva de red, ciertas características pueden ser abstraídas afuera. Por ejemplo, el rendimiento de enlaces satelitales bajo TCP es impactado usando un modelo aproximado de canal que puede ser caracterizado inicialmente por la probabilidad de todos los paquetes perdidos. Se puede crear una estructura en donde se estudien los protocolos de transporte, enrutamiento y MAC en un ambiente satelital, que consiste en satélites geoestacionarios o constelaciones de satélites de órbita polar o órbita baja (LEO).

Como se pudo observar anteriormente, Network Simulator posee una gran cantidad de herramientas complementarias que implementan todo tipo de funciones para la simulación de redes. Específicamente algunas actividades que se pueden desarrollar con NS y NAM son:

CREAR :

- ◊ Redes terrestres, satelitales e inalámbricas con varios algoritmos de enrutamiento (dv, ls, pim-dm, pim-sm, aodv, dsr).

- ◊ Fuentes de tráfico ó generadores de tráfico como : web, ftp, telnet, cbr, tráfico estocástico.

- ◊ Fallas, incluyendo determinísticas, pérdidas probabilísticas, fallas de enlace, entre otras.

- ◊ Varias disciplinas de colas (drop-tail, red, fq, sfq, drr, fred, csfq, etc.) y calidad de servicio “QoS” (ejemplo, IntServ y Diffserv).

VISUALIZAR :

- ◊ Flujo de paquetes, incremento de colas y pérdida de paquetes.
- ◊ Comportamientos de protocolos : inicio lento TCP, registro de si mismo, control de congestión, retransmisión y recuperación rápida.
- ◊ Movimiento de nodos en redes inalámbricas.
- ◊ Anotaciones para resaltar eventos importantes.
- ◊ Estado del protocolo (ejemplo : TCP cwnd).

4. TALLER PROPUESTO

OBJETIVO: se busca que el estudiante comprenda el proceso de creación de escenarios (nodos, enlaces, fuentes de tráfico en los nodos y programar eventos) implementados en el lenguaje de script Tcl y ejecutados con la herramienta Network Simulator. Además, que el estudiante complemente el ejercicio agregando nuevos objetos al escenario de simulación.

Se proponen otros incisos para que el estudiante desarrolle, a medida que vaya profundizando en sus conocimientos tanto de la sintaxis de programación Tcl como de las diferentes topologías de redes y posea un concepto más claro de la herramienta.

1. En este punto se configura un escenario de simulación donde un generador envía tráfico bajo una conexión TCP sobre los enlaces de los dos nodos de la red. Para este caso, se desarrolla el código Tcl y se deja para el estudiante las actividades a desarrollar. Ejecute un editor de texto, por ejemplo kate para comenzar a digitar el código Tcl.

◊ Se inicia creando una instancia objeto del simulador.

```
set ns [new Simulator]  ;# Crea una instancia de la clase Simulador
                        ;# "ns" se refiere a la instancia
```

◊ Se configura la animación.

```
set namfile "tcp.nam"    ;# Establece "namfile" como nombre de la animación
set nf [open $namfile w]          ;# Abre el archivo ("nf" establece el puntero)
$ns namtrace-all $nf     ;# Habilita la animación en la instancia del simulador
```

Recuerde que las variables en Tcl se configuran mediante el comando set [nombre_variable]. Se necesita utilizar el prefijo “$” para referirse a la variable después en el programa.

◊ Se configuran los colores para el flujo de tráfico.

```
$ns color 0 blue        ;# Establece el color de clase 0 al tráfico
$ns color 1 red         ;# Establece el color de clase 1 al tráfico
```

◊ Se crean los nodos.

```
set nodo(0) [$ns node] ;# Crea el nodo ("nodo(x)" se refiere a un nuevo nodo)
set nodo(1) [$ns node] ;# Crea el nodo ("nodo(x)" se refiere a un nuevo nodo)
```

◊ Se crea un enlace full-duplex entre los nodos. En este caso, el enlace es de 10Mb/s con un retardo de 10ms y una cola drop-tail (el último paquete se borra si la cola está llena).

```
$ns duplex-link $node(0) $node(1) 10Mb 10ms DropTail
```

◊ Se inicia la cola en el nodo 0 para el flujo de tráfico sobre el enlace hacia el nodo 1.

```
$ns duplex-link-op $node(0) $node(1) queuePos 0.5
```

◊ Se crea un generador de tráfico CBR (Rata de Bit Constante) el cual envía paquetes de 1460 bytes con un intervalo de 0.003 segundos.

```
set cbr(0) [new Application/Traffic/CBR]   ;#Crea un generador de tráfico CBR
$cbr(0) set packetSize_ 1460  ;#Establece el tamaño del paquete
$cbr(0) set interval_ 0.003;#Establece el intervalo de transmisión del paquete
```

◊ Se Crea un emisor TCP (src) con el tamaño máximo de ventana de 100 paquetes y con una clase de tráfico 0 (color azul).

```
set src(0) [new Agent/TCP]  ;# Crea una instancia de una fuente TCP
$src(0) set window_ 100     ;# Establece la máxima ventana de la fuente TCP
$src(0) set class_ 0        ;# Establece la clase de tráfico (color) del TCP
```

◊ Se crea un receptor TCP (sink).

```
set sink(0) [new Agent/TCPSink]      ;# Crea la instancia del sumidero TCP
```

◊ Se conectan el generador de tráfico, el emisor TCP, el receptor TCP y los nodos.

```
$ns attach-agent $node(0) $src(0) ;# La fuente TCP se adjunta a un nodo específico.
$ns attach-agent $node(1) $sink(0);# El sumidero TCP se adjunta a un nodo específico
$cbr(0) attach-agent $src(0)       ;# El generador de tráfico CBR usa una fuente específica TCP
```

◊ Se establece la conexión entre el emisor TCP y el receptor TCP.

```
$ns connect $src(0) $sink(0)   ;# Conecta la fuente TCP y el sumidero TCP
```

◊ Se establecen los eventos necesarios para el escenario. Primero se debe iniciar el generador de tráfico CBR.

```
$ns at 0.0 "$cbr(0) start"     ;# Inicia el generador de tráfico CBR al tiempo 0.0
```

◊ Se termina el escenario.

```
# Se inicia el animador y se finaliza el simulador a los 2.0 segundos.
$ns at 2.0 "$ns flush-trace; exec nam -a $namfile; exit 0"
```

◊ Finalmente, se adiciona la línea con la cual se ejecuta la simulación del escenario.

```
$ns run
```

Actividades a desarrollar: inicialmente se guarda el archivo con un nombre, con extensión .tcl y se ejecuta con la instrucción "./ns nombre.tcl". La topología se mostrará en una ventana del NAM. Para arreglar los nodos y enlaces se hace clic en el botón re-layout o en el botón edit (de la izquierda) y manualmente se mueven los nodos con el mouse.

Se Inicia la animación. El estudiante debe observar cómo el flujo de tráfico varía?. Describa lo que se observa. Se detiene la animación en la mitad del escenario y se hace una imagen, se describe brevemente que muestra la imagen.

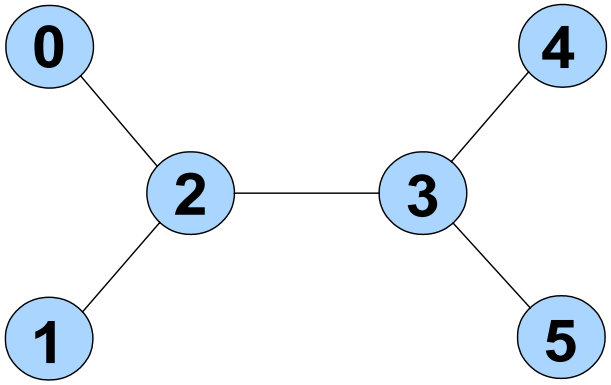
2. El escenario anterior será extendido a una red de 6 nodos donde dos conexiones TCP comparten la capacidad de un enlace. Además, se debe elaborar el código correspondiente para generar tres archivos de trazas con el fin de determinar gráficamente el tamaño de la cola, el ancho de banda y los paquetes perdidos del escenario. El script anterior se usa como base para el desarrollo de este punto.

◊ Se hace la definición de los tres archivos de trazas.

```
set qsize [open queuesize.tr w];# Abre un archivo queuesize.tr y crea la
                                variable qsize
set qbw   [open queuebw.tr   w] ;# Abre un archivo queuebw.tr y crea la
                                variable qbw
set qlost [open queuelost.tr w] ;# Abre un archivo queuelost.tr y crea la
                                variable qlost
```

◊ Se realiza el código para crear seis nodos y se hacen los enlaces respectivos con el fin de obtener la topología de la figura 14.

Figura 14. Topología de red de 6 nodos.



◊ Cada enlace será full-duplex con un ancho de banda de 10Mb/s, un retardo de 10ms y una cola drop-tail. Verifique el escenario ejecutando el script.

```
set nodo(0) [$ns node] ;# Crea el nodo ("nodo(x)" que se refiere a un nuevo
set nodo(1) [$ns node]     nodo).
set nodo(2) [$ns node]
set nodo(3) [$ns node]
set nodo(4) [$ns node]
set nodo(5) [$ns node]
$ns duplex-link $nodo(2) $nodo(3) 10Mb 10ms DropTail
$ns duplex-link $nodo(0) $nodo(2) 10Mb 10ms DropTail
$ns duplex-link $nodo(1) $nodo(2) 10Mb 10ms DropTail
$ns duplex-link $nodo(4) $nodo(3) 10Mb 10ms DropTail
$ns duplex-link $nodo(5) $nodo(3) 10Mb 10ms DropTail
```

◊ Se crea y se configura un nuevo generador de tráfico cbr(1) con la misma especificación del cbr(0).

```
set cbr(1) [new Application/Traffic/CBR] ;# Crea un generador de tráfico CBR
$cbr(1) set packetSize_ 1460     ;# Establece el tamaño del paquete
$cbr(1) set interval_ 0.003     ;# Establece el intervalo de transmisión del paquete
```

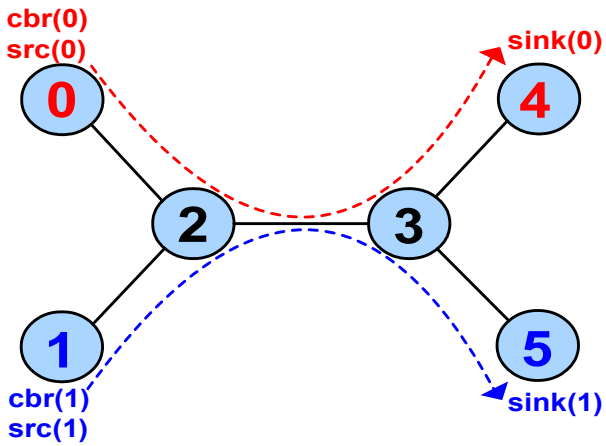
◊ Se crea y se configura otra fuente TCP y otro destino TCP con la misma especificación de src(0) y sink(0).

```
set src(1) [new Agent/TCP]      ;# Crea la instancia de la fuente TCP
$src(1) set window_ 100          ;# Establece la máxima ventana de la fuente TCP
$src(1) set class_ 0          ;# Establece la clase de tráfico (color) de la fuente TCP
set sink(1) [new Agent/TCPSink]          ;# Crea la instancia del sumidero TCP
```

◊ Se conectan el generador de tráfico, las fuentes TCP, los destinos TCP y los nodos como se muestra en la configuración de la figura 15.

```
$ns attach-agent $node(1) $src(1)        ;# La fuente TCP se adjunta al nodo 1
$ns attach-agent $node(5) $sink(1)       ;# El sumidero TCP se adjunta al nodo 5
$cbr(1) attach-agent $src(1)             ;# El generador CBR usa la fuente TCP
```

Figura 15. Generadores de tráfico, fuentes, flujos y sumideros en la topología.



◊ Se configuran las conexiones entre las fuentes TCP y los destinos TCP como lo muestra la figura anterior (línea azul y roja).

```
$ns connect $src(0) $sink(0)        ;# Se conectan la fuente y el sumidero
$ns connect $src(1) $sink(1)        ;# Se conectan la fuente y el sumidero
```

◊ Se ejecuta xgraph para visualizar el tamaño de la cola, el ancho de banda y la tasa de pérdida de paquetes.

```
exec ./xgraph queuesize.tr -geometry 800x400 -t "Tamano de la Cola" -x "seg" -y
"# paquetes" &
exec ./xgraph queuebw.tr -geometry 800x400 -t "Ancho de Banda" -x "seg" -y
"Kbps" -fg white &
exec ./xgraph queuelost.tr -geometry 800x400 -t "# Paquetes Perdidos" -x "seg" -
y "paquetes" -fg red &
```

◊ Se instala un evento en 0.3 segundos que inicie el nuevo generador de tráfico. (el anterior cbr(0) aún comienza en 0.0 segundos).

```
$ns at 0.3 "$cbr(1) start"  ;# Se inicia el generador de tráfico CBR a los 0.3 seg.
```

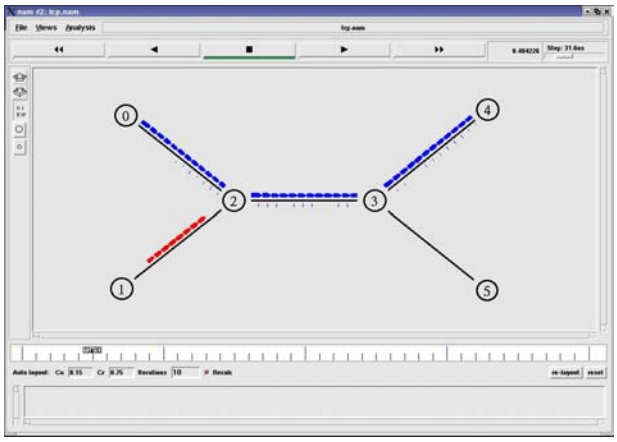
◊ Se ejecuta el escenario, y se arregla la topología para iniciar la animación.

```
$ns run
```

Actividades a desarrollar: Se observa y se describe el movimiento de la animación en el instante en que suceden los eventos. Realice la descripción utilizando las figuras y las gráficas de trazas de la simulación.

El flujo de tráfico comienza a los 0.0 segundos desde el nodo 0 hacia el nodo 4, luego a los 0.3 segundos se genera el segundo flujo desde el nodo 1 hacia el nodo 5.

Figura 16. Inicio del flujo en el escenario.



A partir de los 0.5 segundos empieza un represamiento en la cola del nodo 2, el cual indica que el flujo de entrada es mayor que el flujo de salida, lo que hace que en determinado instante los datos se pierdan.

Figura 17. Visualización del represamiento de la cola en el nodo 2.

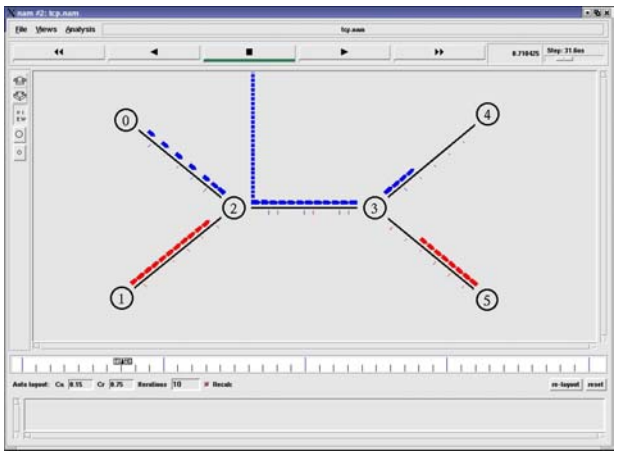
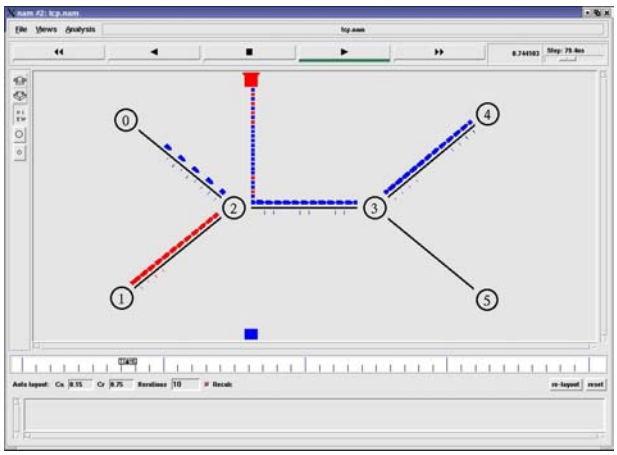
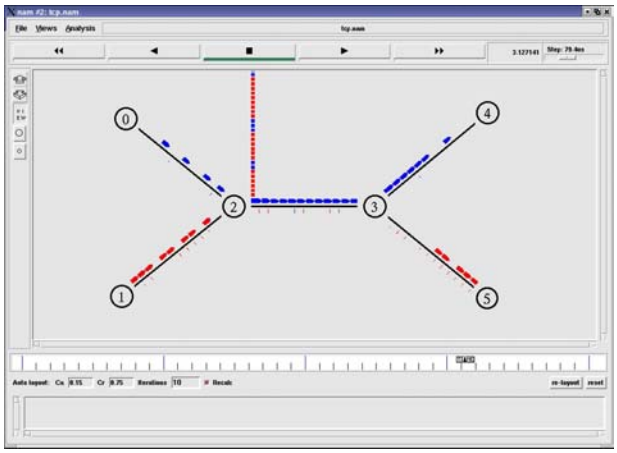


Figura 18. Visualización de la pérdida de paquetes.



Luego de un corto tiempo, la cola de tráfico comienza a disminuir hasta que el flujo se estabiliza al finalizar la simulación.

Figura 19. Flujo de tráfico normal.



En las siguientes figuras se muestran el ancho de banda utilizado, el tamaño de la cola y los paquetes perdidos durante la simulación.

Figura 20. Ancho de banda utilizado.

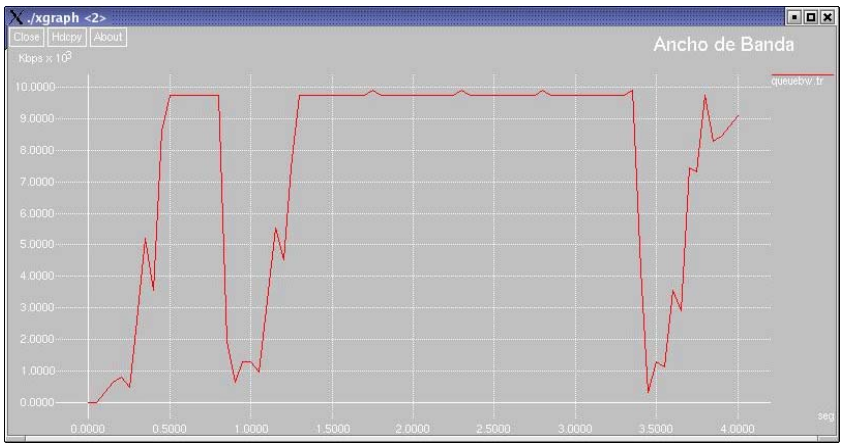


Figura 21. Tamaño de la cola durante el tiempo simulado.

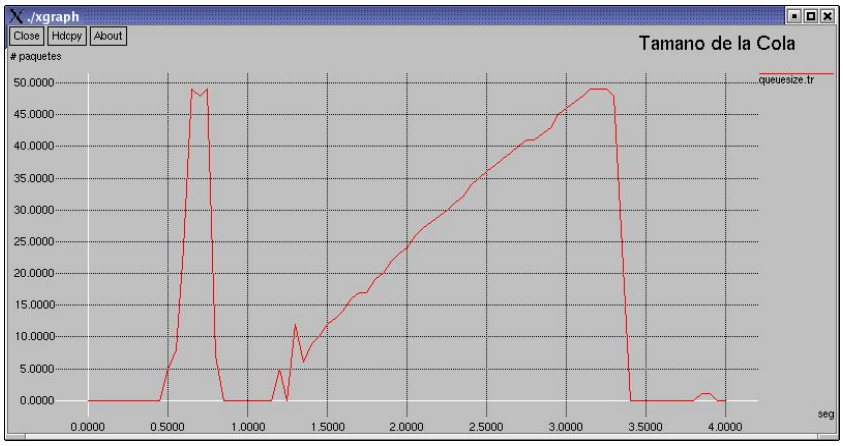
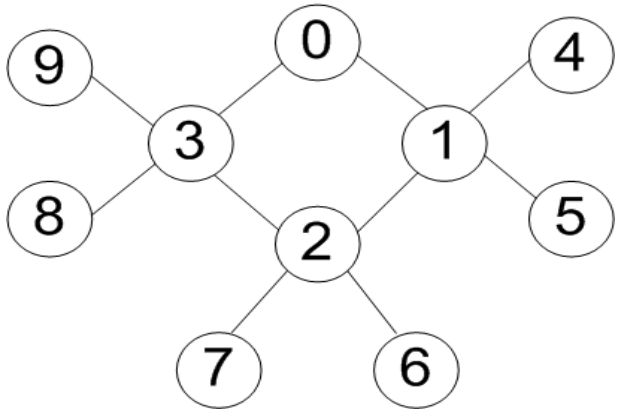


Figura 22. Paquetes perdidos en el nodo 2 durante la simulación.

3. El punto a continuación se deja como trabajo para que el estudiante lo desarrolle. Éste debe crear una red con funcionalidad multicast.

◊ Se crean 10 nodos y se hacen los enlaces respectivos con el fin de obtener la topología de la figura siguiente. Todos los enlaces tendrán la configuración de los puntos anteriores.

Figura 23. Topología de red con capacidad multicast.



◊ Se define que el enlace entre el nodo 0 y el nodo 1 sea dinámico. Esto con el fin de que el enlace pueda ser desconectado durante la simulación.

◊ El modo de cambio (PIM-SIM) de árbol compartido es el tipo de multicast a ser simulado.

◊ Se elimina un generador de tráfico de tal manera que el generador cbr(0) quede sólo.

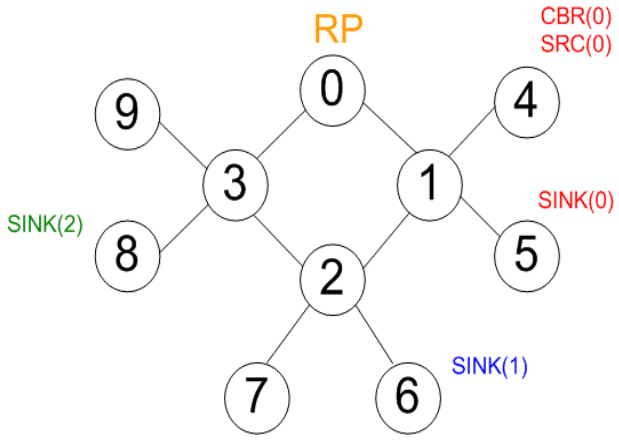
◊ Se eliminan las dos fuentes TCP y se crea una fuente UDP.

◊ Se eliminan los receptores TCP y se crean 3 nuevos receptores de tipo LossMonitor.

◊ Se eliminan las configuraciones de las conexiones TCP.

◊ Se conectan el generador de tráfico, la fuente UDP, los receptores LossMonitor y los nodos de tal manera que la configuración sea la mostrada en la siguiente figura.

Figura 24. Configuración de la topología a simular.



◊ Se crea un evento tal que el generador de tráfico inicie en el tiempo 0.1 y el receptor sink(0) registre en el grupo multicast a los 0.2 segundos.

◊ Defina la simulación en 1.0 segundo.

◊ Arregle la topología antes de iniciar la simulación y describa lo que sucede.

◊ El tráfico multicast es visible. Cuáles son las diferencias si se crea una simulación unicast?.

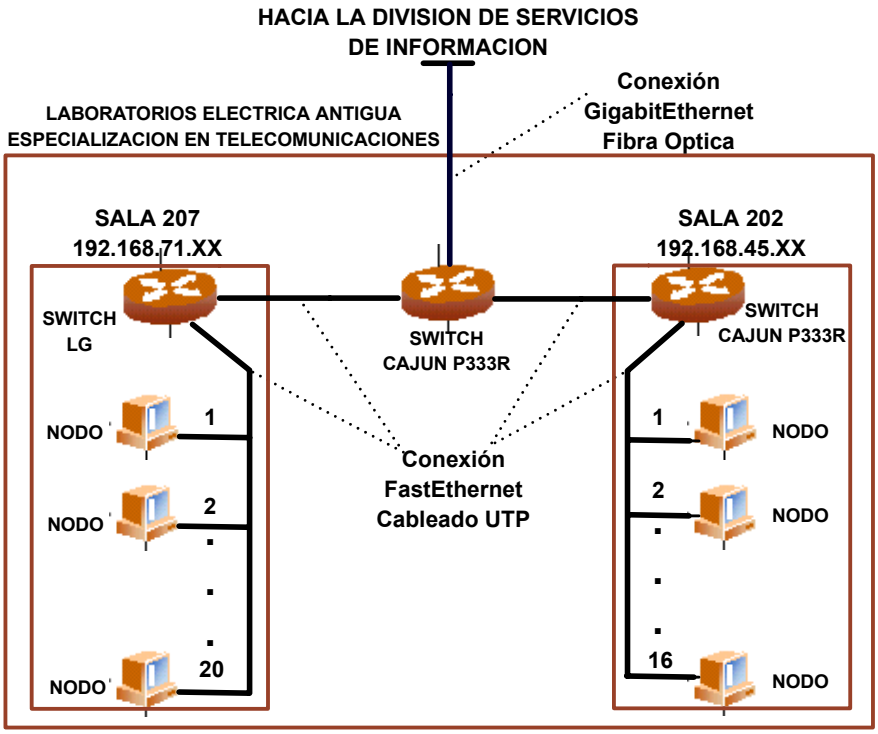
4. Al escenario anterior, se le debe crear un evento en el que el enlace entre los nodos 0 y 1 se rompa a los 0.4 segundos. Describa brevemente lo que sucede con el flujo de datos.

5. PROPUESTA DE UN MODELO DE SIMULACIÓN PARA UN SEGMENTO DE LA RED LAN DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER

Llevando a la práctica los fundamentos y ejercicios de los capítulos anteriores hacia la Universidad Industrial de Santander, se propone un modelo de red correspondiente al segmento de la LAN que se encuentra en el edificio de Eléctrica antigua. El objetivo principal de esta propuesta es realizar el planteamiento de la topología de la red con la herramienta Network Simulator y sentar la base para que con experiencias futuras se puedan simular una gran diversidad de escenarios que permitan la investigación, el análisis de los resultados obtenidos y en caso de encontrar fallas, la apropiada toma de decisiones que conlleven al mejoramiento de la red de la Universidad Industrial de Santander.

La descripción del segmento de red diseñado en la siguiente figura corresponde a 3 switches y 36 estaciones de trabajo con conexión de cable UTP (FastEthernet 10/100Mbps) repartidas en dos laboratorios (202 con 16 estaciones y con las direcciones IP de la serie 192.168.45.XX y 207 con 20 estaciones y con las direcciones IP de la serie 192.168.71.XX), soportando a cada uno de los laboratorios un switch (Cajun P333R y LG), además estos switches se conectan a otro Cajun P330 ubicado en el primer piso, el cual presenta una conexión de fibra óptica (GigabitEthernet) hacia el switch central P880 de la División de Servicios de Información.

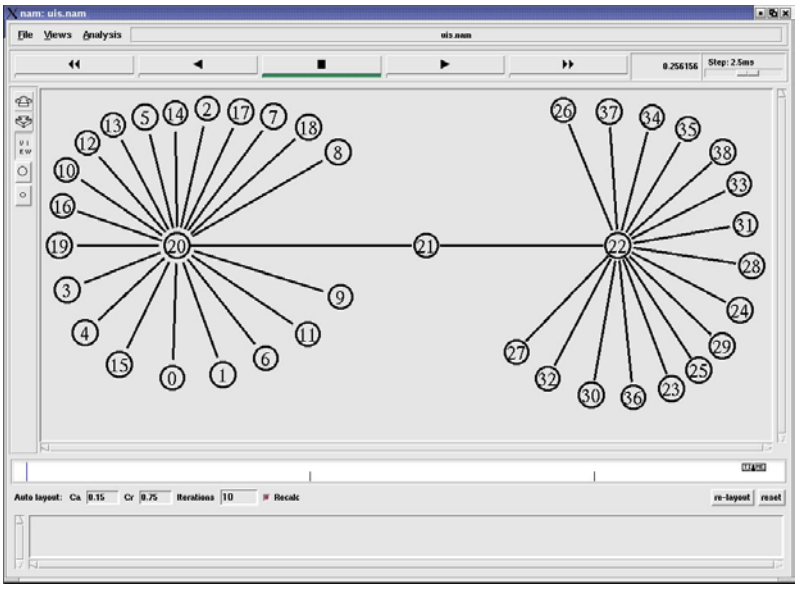
Figura 25. Diseño del segmento de red.



A continuación se define y se presenta la topología simulada en Network Simulator la cual muestra la estructura básica de la figura anterior.

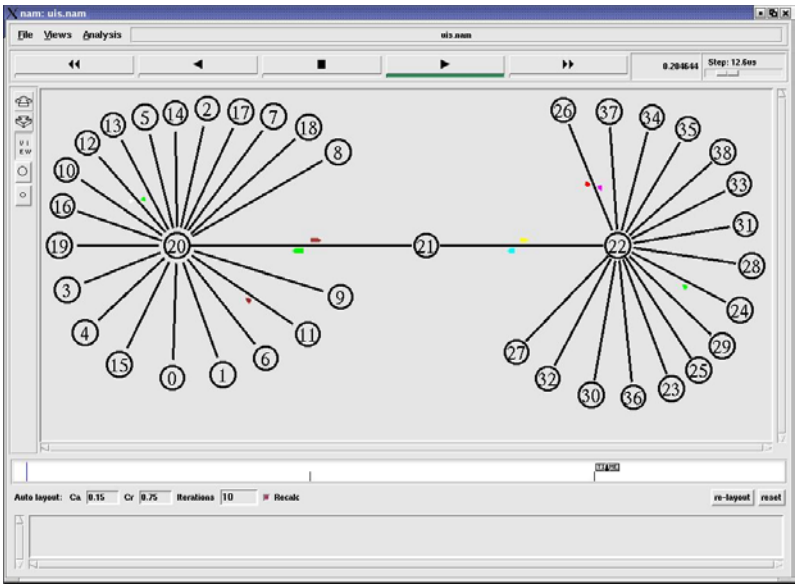
La correspondencia de los nodos es la siguiente: los nodos 20, 21 y 22 representan los 3 switches, los nodos del 0 al 19 representan a las estaciones de trabajo del laboratorio 207, los nodos del 23 al 39 representan a las estaciones de trabajo del laboratorio 202 y los enlaces representan únicamente a las conexiones FastEthernet.

Figura 26. Topología de red visualizada en NAM.



A manera de presentar una funcionalidad del modelo, se realiza una simulación Ping (que permite hacer pruebas de conexión en la red). Se crean 10 agentes ping y se adjuntan a los nodos nodo(0), nodo(5), nodo(6), nodo(11) y nodo(12) que están en el grupo del nodo 20 y a los nodos nodo(25), nodo(27), nodo(32), nodo(36) y nodo(38) que están en el grupo del nodo 22. Además, a cada agente Ping se le asigna un color para diferenciarlo. Por último, se deben conectar los agentes por pares de nodos. La simulación tiene una duración de 0.3 segundos que son suficientes para apreciar el evento. A continuación se muestra gráficamente el modelo en ejecución cuando los Ping se encuentran sobre los enlaces.

Figura 27. Modelo en ejecución Ping.



Finalmente, el programa genera un archivo de seguimiento donde se aprecian todos los comportamientos de la simulación. Un segmento del archivo de seguimiento se presenta a continuación junto con la respectiva descripción.

Figura 28. Segmento para el modelo de la UIS.

+	0.202	6	20	ping	64 ------- 2	6.0	31.0	-1	4
-	0.202	6	20	ping	64 ------- 2	6.0	31.0	-1	4
r	0.203051	6	20	ping	64 ------- 2	6.0	31.0	-1	4
+	0.203051	20	21	ping	64 ------- 2	6.0	31.0	-1	4
-	0.203051	20	21	ping	64 ------- 2	6.0	31.0	-1	4
r	0.204102	20	21	ping	64 ------- 2	6.0	31.0	-1	4
+	0.204102	21	22	ping	64 ------- 2	6.0	31.0	-1	4
-	0.204102	21	22	ping	64 ------- 2	6.0	31.0	-1	4
r	0.205154	21	22	ping	64 ------- 2	6.0	31.0	-1	4
+	0.205154	22	31	ping	64 ------- 2	6.0	31.0	-1	4
-	0.205154	22	31	ping	64 ------- 2	6.0	31.0	-1	4
r	0.206205	22	31	ping	64 ------- 2	6.0	31.0	-1	4

La descripción se va a realizar utilizando el movimiento de un ping que inicia en el nodo 6 y finaliza en el nodo 31. Se puede observar que cada 3 filas el ping se encuentra dentro de los mismos nodos, debido a que el ping debe ingresar a la cola (+), salir de la cola (-) y ser recibido en el nodo destino (r). La trayectoria recorrida por el ping se da desde el nodo 6 al nodo 20, del nodo 20 al 21, del nodo 21 al 22 y por último del nodo 22 al nodo 31 que es el que recibe el ping. A través de la columna 6 se puede determinar que el tamaño del paquete es de 64 bytes. El resto de las columnas (7 a la 11), son banderas, identificadores del flujo, número de secuencia (únicamente para fuentes que lo requieran) y el identificador del paquete (en este caso del ping) que es igual a 4.

5.1 LIMITACIONES DE NETWORK SIMULATOR

Algunas de las limitaciones encontradas a través de la práctica y de fuentes bibliográficas son:

- ◊ La sintaxis del código TCL para la programación de escenarios en Network Simulator, requiere de considerable tiempo para su aprendizaje.

- ◊ El hecho de que la herramienta se encuentre dividida en varias aplicaciones independientes (NS, NAM, XGRAPH), dificulta la ejecución de los scripts TCL. Además, para un principiante se le hace difícil encontrar los archivos generados por las aplicaciones.

- ◊ Cuando se está en Network Animator en el modo edición, algunas veces al hacer clic sobre el área de la ventana, se cierra el programa de forma repentina.

- ◊ Cuando se realiza el modelo para redes con una gran cantidad de nodos, es difícil de visualizar claramente su estructura. Además, su simulación se torna lenta y no es apreciable en detalle, teniendo que recurrir al archivo de seguimiento generado. En algunos casos el programa se cierra inesperadamente.

6. CONCLUSIONES

La presente monografía se constituye en un punto de partida para los estudiantes que se inicien en el estudio de Network Simulator. Además, se reconoce la labor del grupo de investigación CPS (Conectividad y Procesado de Señales) de la E3T, con la profundización en la línea de redes bajo el ambiente de trabajo Linux.

Network Simulator brinda la posibilidad de realizar modelos de simulación donde se pueden definir topologías, generadores de tráfico, programación y manejo de colas, transporte, manejo de redes móviles inalámbricas y satelitales, enrutamiento, entre otros.

Debido a que la herramienta es de libre distribución “OpenSource”, hace más viable su utilización dentro del campus universitario sin preocupaciones de costos de licencias o violaciones de derechos de autor.

A partir de la realización del modelo de simulación propuesto en la parte práctica de la monografía, se considera una base general para profundizar en escenarios con características particulares. De tal manera que en su avance, se llegue a simular completamente la red de la Universidad Industrial de Santander.

El aprendizaje de Network Simulator requiere de considerable tiempo, no es viable la implementación como módulo dentro de la especialización. Es interesante implantarlo para una asignatura de pregrado ya que se cuenta con el tiempo suficiente para lograr los objetivos propuestos.

BIBLIOGRAFÍA

CHUNG, Jae and CLAYPOOL, Mark. NS by Example. Worcester Polytechnic Institute, Computer Science Department. e- http://nile.wpi.edu/NS/menu.html.

FALL, Kevin. The ns Manual. The VINT Project. e- http://www.isi.edu/nsnam/ns/ns-documentation.html.

FASCIANA, Michele Luca. SIP Module for Network Simulator 2. UNIVERSITA' DI PALERMO, CNIT,
Progetto Tango.

GREIS, Marc. Tutorial for the Network Simulator "ns". UCB/LBNL/VINT Group. e- http://www.isi.edu/nsnam/ns/tutorial/nsintro.html.

HALDAR, Padmaparna and CHEN, Xuan. Ns Tutorial 2002. 2002. e- http://www.isi.edu/nsnam/ns/.

HUANG, Polly. Beginning ns-2. USC Information Sciences Institute. 1999. e- http://www-scf.usc.edu/~bhuang.

LABORATORIO DE INGENIERIA de Redes y Servicios Telemáticos. Introducción a la simulación con NS. 2003. e- http://www.lab.dit.upm.es/~labrst/03-04/p.int/prac-int.htm.

LABORATORIO DE INGENIERIA de Redes y Servicios Telemáticos. Simulación de protocolos de encaminamiento multicast. 2004. e- http://www.lab.dit.upm.es/~labrst/03-04/p.sim/prac-sim.htm.

RUIZ, Francisco Javier et al. Implantación de un Laboratorio Docente para Redes de Comunicaciones. Universidad Politécnica de Madrid. e- http://www.lab.dit.upm.es.

YU, Haobo and SALEHI, Nader. Ns2 tutorial. USC Information Sciences Institute.