UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIA Y TECNOLOGIA

ESCUELA DE INGENIERIA ELECTRÓNICA

“Implementación del “laboratorio Virtual” para simulación de redes, basado en el software libre “NS-2”, para la Universidad del Azuay”

Trabajo de graduación previo a la obtención del título de Ingeniero Electrónico

Autor:

Humberto Damián Jaramillo Paredes

Director:

Hugo Marcelo Torres Salamea

Cuenca, Ecuador

2011

Resumen

El propósito de esta monografía consiste en una guía práctica basada en el software libre Network Simulator destinado a desarrollar y simular diferentes topologías de redes de telecomunicaciones, la propuesta está compuesta en dos partes principales: la primera es una revisión de conceptos relacionados con la programación de scripts y las topologías más comunes. La segunda agrupa un compendio de programas; cada programa será usado por el profesor y el estudiante en las clases prácticas. Como resultado final tenemos una muy completa guía teórica y práctica para ser usada dentro de las actividades académicas de la escuela de Ingeniería Electrónica

Abstract

The purpose of this monograph consists in a practice guide based on Network Simulator software oriented to develop and simulate different telecommunications networks topologies. The proposal is composed of two main parts: first one reviews a set of concepts related to scripts programming and most used network topologies. The second one groups a set of programs; each program will be use by teacher and students in a practical session. As a final result a very complete theorical and practical guide was implement to be applied within academic activities at Electronics Engineer School.

ÍNDICE GENERAL.

CAPITULO 4.- PRÁCTICAS

ÍNDICE DE FIGURAS

INDICE DE TABLAS

Jaramillo Paredes Humberto Damián

Trabajo de Grado

Ing. Hugo Torres

Enero 2012

Implementación del “laboratorio Virtual” para simulación de redes, basado en el software libre “NS-2”, para la Universidad del Azuay

INTRODUCCIÓN

Dentro del estudio de las redes de telecomunicaciones, los conceptos teóricos quedan muy aislados sin la preparación practica de los mismo, para lo cual se han creado diferentes herramientas virtuales, siendo una de ellas el Software llamado Network Simulator, el cual es uno de los mas utilizador y más desarrollado dentro de las investigaciones y proyectos en muchos campos afines a las telecomunicaciones.

Es por eso, que este tema monográfico realiza una guía de inicialización para la simulación de redes utilizando como herramienta principal el NS-2, en el cual se trata temas desde la configuración de una red y sus parámetros principales hasta la concepción de una red completa, mediante la utilización de scripts que se guía desde los primeros pasos.

CAPITULO 1
INTRODUCCIÓN Y MANEJO DE LINUX

Linux es un sistema operativo con su primera aparición oficial en el año de 1991, el cual propone ser multiusuario y multitarea, lo que nos brinda la facilidad que varios usuarios utilicen el computador a la vez con varios programas al mismo tiempo.

Desde la primera aparición, Linux, ha buscado la facilidad en el manejo de sus interfaces, por lo que constantemente ha estado innovando, dando así sistemas más estables, robustos y seguros. Siendo Fedora 15 una de las tantas distribuciones que tiene Linux y una de las más comunes bajo una licencia Red Hat.

1.1.- Instalación

1.1.1.- Preinstalación

El primer paso para instalar Fedora 15 es preparar tu equipo y descargar el disco de instalación desde el sitio oficial de Fedora, el mismo que es gratuito por ser un software libre, en esta página encontraras los requisitos de tu computadora para dar paso a la instalación.

En este trabajo monográfico, daremos seguimiento a la instalación mediante el DVD como principal medio de instalación, en el mismo que viene incluido las aplicaciones principales.

1.1.2.- Preparación del equipo para el arranque desde el disco

Con el disco previamente grabado el primer paso a seguir es configurar la máquina para el arranque desde el cd.room, para esto vamos a reiniciar el ordenador y en la

pantalla de comprobación del BIOS (primera pantalla de arranque) pulsamos la tecla de configuración que nos indica en la parte inferior con el nombre de setup.

Con la pantalla de configuración abierta nos desplazamos hasta la pestaña boot donde procedemos a ordenar las unidades de arranque de tal forma que quede como principal el cd-room, concluido esto guardamos los cambios y cerramos.

El ordenador se iniciará y observaremos que arranca con una pantalla azul con una letra f, esto nos indica que estamos listos para empezar la instalación.

1.1.3.- Pasos para la Instalación

Fedora nos brinda una instalación por medio de ventanas, las cuales son muy fáciles y razonables de seguir. La primera pantalla nos da la opción de instalar o actualizar un sistema operativo ya instalado, para nuestro caso escogeremos la primera opción.

La segunda pantalla nos muestra la opción de verificar los datos del cd o dvd, el cual si sabemos que es seguro podemos obviar este paso, caso contrario es conveniente hacerlo.

Luego daremos paso a la elección del idioma tanto para la instalación como para el teclado, seguido por el tipo de almacenamiento y pregunta si vamos a realizar una actualización o una nueva instalación, la segunda opción es la que vamos a tomar.

El siguiente paso es dar un nombre a nuestro equipo y luego la ubicación geográfica y la configuración de fecha y hora para nuestro equipo.
Con esto realizado seguimos por dar una contraseña de usuario y dar paso al paso de particionado del disco duro, en donde tenemos que indicar que tipo de instalación queremos realizar. En este punto debemos tener el disco duro asignado una partición dedicada para Fedora.

Una vez que llegamos a la pantalla de instalación de archivos, nos encontramos que hay dos maneras generales de instalar: Eligiendo una instalación por defecto, o

diciéndole al instalador mediante un “diseño personalizado“, de qué manera y sobre qué particiones uno quiere instalar los paquetes.

Una vez elegida la opción de “crear un diseño personalizado” procedemos a crear los puntos de montaje, que son directorios básico para Fedora 15, que para nuestro caso son necesarios 4: /boot, /sawp, /root y /home.

Le indicamos el tamaño y la ubicación de cada punto de montaje dentro del espacio del disco duro que hemos elegido para la instalación de Fedora 15.

Colocamos cada punto de montaje en una partición independiente. Si disponemos sólo de una partición libre, es suficiente: el instalador la convierte automáticamente en una partición extendida, y la irá subdividiendo de acuerdo a como nosotros se lo indiquemos. Estrictamente, lo único que necesitamos es disponer de suficiente espacio libre o “no asignado” en el disco, y para ello recomendamos al menos 20 GB.

Lo que tenemos que hacer ahora es crear una partición primaria para el punto de montaje /boot. Para hacerlo, seleccionamos la parte “libre” del disco, y hacemos click en “crear”. Vamos a observar una ventana que nos va a preguntar qué tipo de partición queremos crear, y elegimos “generar partición estándard“.

Una vez que se abre la siguiente ventana, elegimos el “punto de montaje” /boot, indicamos el tipo de formato “ext3”, o “ext4” y le decimos que ocupe un espacio de 200 MB. Aceptamos y volvemos a la pantalla anterior, en la que ahora vemos una partición nueva, más el espacio sin asignar, o libre.

Ahora tenemos que crear otra partición estándard que contenga el punto de montaje: / (root). Repetimos el proceso anterior de selección del espacio libre disponible, indicamos que queremos generar otra partición estándard, y le damos sus características. El punto de montaje / pide formato “ext3” o “ext4“, y en él van a ir todos los archivos que necesita fedora 15 para poder funcionar, más todo el soft que le vayamos instalando. Para este punto de montaje, como mínimo aconsejamos 9 GiB, y como máximo 15 GiB.

Repetimos el proceso para crear el punto de montaje /home, en donde van a ir todos los archivos personales que utilizamos cotidianamente en nuestros equipos: textos, fotos, música, pelis, etc., y todo el espacio libre que queramos tener disponible para seguir guardando este tipo de cosas. Así que cada uno conoce cuanta capacidad necesita para este punto de montaje, que también necesita formato “ext3”, o “ext4“.

Por último, volvemos a repetir todo el proceso para generar el punto de montaje swap, que no necesita tipo de formato, y que tiene que tener el doble de capacidad de nuestra memoria RAM, aunque no es necesario que supere los 4 GB, no importa que tengamos 10 GiB de RAM. Swap es algo así como un espacio virtual que fedora 15 va a ir utilizando en función de las necesidades de la memoria RAM de que disponga.

Una vez que tenemos todo esto configurado, guardamos las modificaciones en el disco, y seguimos adelante.

En la siguiente pantalla indicamos dónde queremos instalar el gestor de arranque (boot loader), también conocido como grub (grand boot unified bootloader). Esto es lo que nos va a permitir elegir con qué sistema operativo arrancamos nuestro equipo cada vez que lo prendemos, si lo queremos protegido con contraseña, y cuál será el orden de prioridad de arranque en la lista de los sistemas instalados, se recomienda dejar todo como está.

En seguida llegamos a la pantalla del tipo de sofware que queremos instalar, y de los repositorios que queremos configurar. En esta pantalla, si estamos online, tenemos la posibilidad de instalar paquetes y repositorios que no vienen en el DVD. No se lo recomendamos a quienes sea la primera instalación que hacen de fedora, ya que este es un paso muy sencillo de hacer luego, una vez que todo esté instalado y funcionando.

Una vez que todo esto está configurado, empiezan a instalarse los paquetes y cuando terminan de cargarse tenemos que reiniciar la máquina. La primer pantalla que nos encontramos es la que nos permite crear un usuario, (con password diferente por favor a la que creamos para el usuario “root”).

En las siguientes pantallas controlamos que el día y la hora estén correctas. Decidimos si queremos o no enviar nuestro perfil de hardware a smolt, y listo.

1.1.4.- Postinstalacion

- **Conexión a Internet.-** Si utilizamos una conexión a Internet por cable, o wireless, vamos a tener que 'encender' la red a la que nos queramos conectar desde el ícono del Network Manager: cableada o inalámbrica, de acuerdo a cual sea nuestra red.

  Si utilizamos una conexión ADSL, hay que configurarla desde el Administrador de redes haciendo click en Aplicaciones > Otras > Conexiones de red. Vamos a la pestaña DSL, hacemos click en añadir, ingresamos nuestros datos de conexión y tildamos la casilla 'Conectar automáticamente' si queremos que se active cada vez que prendamos el equipo.

- **Primera actualización del sistema.-** Una vez online, lo primero que tenemos que hacer es actualizar nuestro equipo. En un entorno de escritorio Gnome, lo hacemos desde Actividades> Aplicaciones> Actualizacion de software.

- **Configuración de Sudo y de Yum.-** Dentro de Fedora el medio para instalar un software, es la utilización de repositorios llamados RPM (*Red Hat Package Manager*), que en Fedora 15 se puede instalar mediante un doble click, pero la forma optima es la utilización de "yum", el mismo que puede servir para descargar e instalar o simplemente instalar, esto se puede hacer mediante el uso de "sudo", para el cual es importante localizarnos en el sistema como súper usuarios; lo que hacemos es abrir el terminal e ingresar el comando:

  ```
  su -
  ```

  Damos un enter e ingresamos la clave establecida durante la instalación.

- **Instalación de software**.- Para poder proceder a instalar un software es imprescindible haber instalado primero todos los repositorios necesarios, una vez hecho esto yum es la herramienta que nos ayuda con la instalación, para lo

cual se debe indicar correctamente el nombre, caso contrario yum nos dará un error, la forma de utilizar yum es:

yum install nombredelsoftquequeremosinstalar

Una vez instalado un software este se ira colocando de forma automática en el menú de aplicaciones o sistema.

Si queremos saber cuáles son los paquetes disponibles para descargar:

yum list available

CAPITULO 2
TEORÍA SOBRE REDES.

Redes LAN

LAN en su traducción al español es la definición de una red de área local, que es un sistema de transmisión de datos que nos permite compartir recursos e información informática, voz, multimedia, etc, dentro de una red de equipos y ordenadores. El Comité IEEE (Institute of Electrical and Electronics Engineers) ofrece una definición oficial de red local: “una red local es un sistema de comunicaciones que permite que un número de dispositivos independientes se comuniquen entre sí.”

Redes WLAN

WLAN, traducidas sus siglas al español, es la forma en que se conoce a una red de area local inalámbrica, este tipo de redes son una alternativa a las redes LAN, mediante la comunicación de datos de forma inalámbrica, cabe añadir que las dos pueden coexistir en una misma topología.

Mediante el uso de la tecnología de radio difusión permite a los usuarios la ventaja de movilidad y minimiza costos de redes cableadas. Cada día se reconocen más este tipo de redes es un amplio número de negocios y se augura una gran extensión de las mismas y altas ganancias.

Redes Satelitales

Para este tipo de redes se utiliza a los satélites como medio de transmisión, el cual cumple la función de un repetidor radioeléctrico de señales generadas en la tierra y receptadas en el espacio, lo que hace es amplificar y retransmitir al punto final de recepción de la señal.

En una red satelital encontramos un dispositivo receptor-transmisor, una estación terrestre, destinada al control y proporcionar las pautas para la transmisión y recepción del tráfico, y una red de usuario.

Topología

El término “topología” se emplea para referirse a la disposición geométrica de las estaciones de una red y los cables que las conectan, y al trayecto seguido por las señales a través de la conexión física. La topología de la red es pues, la disposición de los diferentes componentes de una red y la forma que adopta el flujo de información.

Las topologías fueron ideadas para establecer un orden que evitase el caos que se produciría si las estaciones de una red fuesen colocadas de forma aleatoria. La topología tiene por objetivo hallar cómo todos los usuarios pueden conectarse a todos los recursos de red de la manera más económica y eficaz; al mismo tiempo, capacita a la red para satisfacer las demandas de los usuarios con un tiempo de espera lo más reducido posible. Para determinar qué topología resulta más adecuada para una red concreta se tienen en cuenta numerosos parámetros y variables, como el número de máquinas que se van a interconectar, el tipo de acceso al medio físico deseado, etc.

2.1.- Topologías

2.1.1 Topologías de las redes LAN

Topología en estrella

La topología en estrella es uno de los tipos más antiguos de topologías. Se caracteriza porque en ella existe un nodo central al cual se conectan todos los equipos, de modo similar al radio de una rueda.

En esta topología, cada estación tiene una conexión directa a un acoplador (conmutador) central. Una manera de construir esta topología es con conmutadores telefónicos que usan la técnica de conmutación de circuitos.

Otra forma de esta topología es una estación que tiene dos conexiones directas al acoplador de la estrella (nodo central), una de entrada y otra de salida (la cual lógicamente opera como un bus). Cuando una transmisión llega al nodo central, este la retransmite por todas las líneas de salida.

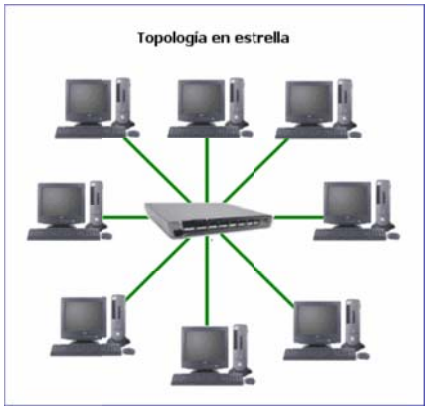

Figura 2.1: Topología Estrella
Fuente: www.slidershare.net

Topología en bus

Al contrario que en la topología en estrella no existe un nodo central, sino que todos los nodos que componen la red quedan unidos entre sí linealmente, uno a continuación del otro. Es necesario incluir en ambos extremos del bus unos dispositivos denominados terminadores, que evitan posibles rebotes de la señal.

Esta topología permite que todas las estaciones reciban la información que se transmite, una estación transmite y todas las restantes escuchan. Consiste en un cable con un terminador en cada extremo del que se cuelgan todos los elementos de una red. Todos los nodos de la red están unidos a este cable: el cual recibe el nombre de "Backbone Cable". Tanto Ethernet como Local Talk pueden utilizar esta topología.

El bus es pasivo, no se produce regeneración de las señales en cada nodo. Los nodos en una red de "bus" transmiten la información y esperan que ésta no vaya a chocar con otra información transmitida por otro de los nodos. Si esto ocurre, cada nodo espera una pequeña cantidad de tiempo al azar, después intenta retransmitir la información.

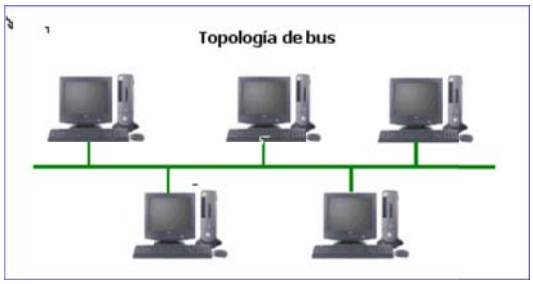

Figura 2.2: Representa la topología de Bus

Fuente: www.slidershare.net

Topología en anillo

En esta topología, las estaciones están unidas unas con otras formando un círculo por medio de un cable común. El último nodo de la cadena se conecta al primero cerrando el anillo. Las señales circulan en un solo sentido alrededor del círculo, regenerándose en cada nodo. Con esta metodología, cada nodo examina la información que es enviada a través del anillo. Si la información no está dirigida al nodo que la examina, la pasa al siguiente en el anillo. La desventaja del anillo es que si se rompe una conexión, se cae la red completa.

El cableado es el más complejo de todos, debido, en parte, al mayor coste del cable, así como a la necesidad de emplear dispositivos MAU (Unidades de Acceso Multiestación) para implementar físicamente el anillo.

Cuando existen fallos o averías, es posible derivar partes de la red mediante los MAUs, aislando las partes defectuosas del resto de la red mientras se determina el problema.

Así, un fallo en una parte del cableado no detiene la red en su totalidad. Cuando se quieren añadir nuevas estaciones de trabajo se emplean también los MAUs, de modo que el proceso no posee una complicación excesiva.

Figura 2.3: Representa la topología en anillo
Fuente: www.slidershare.net

Topologías híbridas

Son las más frecuentes y se derivan de las tres anteriores, conocidas como topologías puras. Las más frecuentes son la topología en árbol y la topología estrella-anillo.

La topología en árbol es una variante de la topología en bus. Esta topología comienza en un punto denominado cabezal o raíz (headend). Uno o más cables pueden salir de este punto y cada uno de ellos puede tener ramificaciones en cualquier otro punto. Una ramificación puede volver a ramificarse. En una topología en árbol no se deben formar ciclos.

Una red como ésta representa una red completamente distribuida en la que computadoras alimentan de información a otras computadoras, que a su vez alimentan a otras. Las computadoras que se utilizan como dispositivos remotos pueden tener recursos de procesamientos independientes y recurren a los recursos en niveles superiores o inferiores conforme se requiera.

La topología en estrella-anillo combina las tecnologías de las topologías en estrella y anillo. El cable que une cada estación con la siguiente pasa a través de un nodo central que se encarga de desconectarla de la red si sufre una avería.

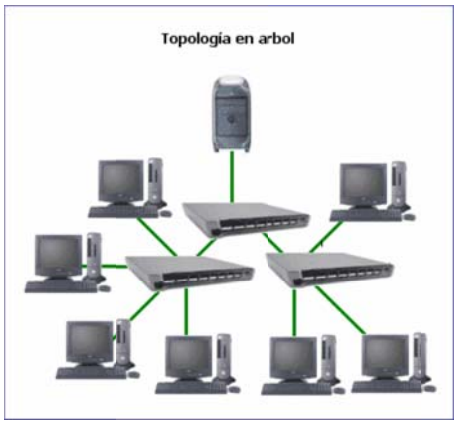

Figura 2.4: Representa la topología en árbol
Fuente: www.slidershare.net

2.1.2.- Topologías redes WLAN

Topología Ad-Hoc

Cada dispositivo se puede comunicar con todos los demás. Cada nodo forma parte de una red Peer to Peer o de igual a igual, para lo cual sólo vamos a necesitar el disponer de un SSID igual para todos los nodos y no sobrepasar un número razonable de dispositivos que hagan bajar el rendimiento. A más dispersión geográfica de cada nodo más dispositivos pueden formar parte de la red, aunque algunos no lleguen a verse entre sí.

Topología Infraestructura

En el cual existe un nodo central (Punto de Acceso WiFi) que sirve de enlace para todos los demás (Tarjetas de Red Wifi). Este nodo sirve para encaminar las tramas hacia una red convencional o hacia otras redes distintas. Para poder establecerse la comunicación, todos los nodos deben estar dentro de la zona de cobertura del AP.

CAPITULO 3

INSTALACIÓN DEL NS-2

NS (Network simulator) es un simulador de eventos centrado en la investigación sobre redes. Ns dispone simulación para TCP (*Transmission Control Protocol*), *routing* y multicast sobre redes cableadas o inalámbricas.

Este capítulo hablara requerimientos y pasos a seguir de forma ordenada para una instalación óptima del simulador y posterior funcionamiento óptimo. En el cual encontraremos ayuda e información como: donde descargar el programa, que parámetros necesitamos, etc.

Tratando de dar mayor ayuda al estudiante diseñaremos un instructivo paso a paso para la instalación y puesta a punto del simulador NS-2.

Teniendo en cuenta que el NS-2 es un simulador que puede ser instalado en cualquier sistema operativo que trabajen bajo UNIX, este instructivo nos guiara a través de plataformas LINUX.

3.1.- Descarga e instalación

3.1.1.- Descarga

Para la descarga el mismo desarrollador del NS-2 nos brinda el programa sin ningún costo desde su página oficial, siendo el link:

http://www.isi.edu/nsman/ns/

Una vez en esta página tenemos la posibilidad de descargar de tres diferentes vías, la que nosotros elegiremos es la ns-allinone-2.26.tar.gz, que es el que tiene el código

fuente del programa, mas todas las librería y programas necesarios para un buen desempeño del NS-2, y lo bajamos completo al NS-2 versión 7.

3.1.2.- Instalación

Con el programa ya descargado con el fichero completo la primera acción es proceder a descomprimir el programa eligiendo un directorio para el mismo, que en nuestro caso será el mismo /home.

```
tar zxvf ns-allinone-2.lb7a.tar.gz
```

Luego nos ubicamos en el directorio e instalamos.

```
#cd ns-allinone-2.lb7a
#./install  install.out
```

3.1.3.- Errores comunes

Entre los dos errores más comunes encontramos los siguientes:

- *"tools/ranvar.cc: in member function `virtual double gammarandomvariable::value()': tools/ranvar.cc:219:70:error:cannot call constructor`GammaRandomVariable: :GammaRandomVariable'tools/ranvar.cc:219:70: error: for a function-style cast, remove the redundant `::GaammaRandomVariable' make: *** [tools/ranvar.o] Error 1 ns make failed"*

Para solucionar este error lo que procedemos a hacer es modificar un archivo llamado nakagami.cc ubicado en la dirección ./ns-allinone-2.34/ns-2.34/mobile/nakagami.cc, lo que podemos hacerlo desde el terminal o directamente con un gestor de texto; modificamos la línea numero 183 y la remplazamos con "resultPower = ErlangRandomVariable(Pr/m, int_m).value();" y lo mismo la línea 185 debemos remplazarla por "resultPower = GammaRandomVariable(m, Pr/m).value();"

- tools/ranvar.cc tools/ranvar.cc:In member function ‘virtual double GammaRandomVariable::value()’:tools/ranvar.cc:219:70: error: cannot call constructor ‘GammaRandomVariable::GammaRandomVariable’directly tools/ranvar.cc:219:70: error: for a function-style cast, remove the redundant ‘::GammaRandomVariable’make: ***[tools/ranvar.o]Error1 Ns make failed!

Igual que en el error anterior lo solucionaremos modificando un archivo, en este caso será el llamado ranvar.cc ubicado en ./ns-allinone-2.34/ns-2.34/tools/ranvar.cc con la modificación de la línea numero 219, reemplazándola por “return GammaRandomVariable(1.0 + alpha_, beta_).value() * pow (u, 1.0 / alpha_);”

Una vez instalado el programa procedemos a configurar el entorno Shell mediante los siguientes comandos:

```
setenv NS2HOME $HOME/ns-allinone-2.lb7a
setenv TCL_LIBRARY $NS2HOME/tcl-8.04/library
setenv TK_LIBRARY $NS2HOME/tk-8.04/library
set path = ($NS2HOME/bin $path)
```

Una vez acabado este procedimiento lo aconsejable es reiniciar el sistema operativo para que el programa trabaje sin contratiempos quedando el programa constituido como muestra la figura.

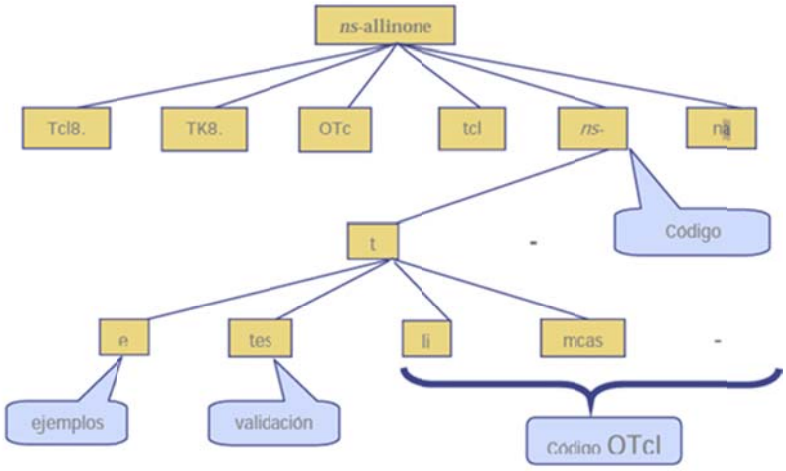

Figura 3.1: Distribución del simulador NS-2

Fuente: www.isi.edu/nsnam/ns/doc

3.2.- Lenguaje TLC

3.2.1.- Introducción

El software NS-2 se encuentra desarrollado en un lenguaje C++, pero se lo invoca mediante un lenguaje TCL (Tool Command Lenguage), para lo cual debe existir una armonía entre los dos lenguajes. Para nuestro caso de estudio desarrollaremos únicamente conceptos fundamentales del lenguaje TCL que nos ayudaran en el manejo de topologías a simular, en el caso que se necesite modificar ciertas herramientas del NS-2 será necesario manejar el lenguaje C++

Podemos decir que TLC es un lenguaje interpretado, donde, sus programas van a ser formados por ficheros o scripts de ordenes TLC que van a ser interpretados y procesados en tiempo de ejecución por el interprete.

3.2.2.- Conceptos Básicos

TCL es un lenguaje de programación interpretado y multiplataforma de distribución gratuita, basado en la interpretación de comandos por medio de un tclsh que su función más grande es la de enlazar funciones de C++ dentro del lenguaje TCL.

El desarrollador de TCL (Sun Microsystems Laboratories) creó junto con TCL una herramienta llamada Tk (Tool Kit), que es distribuida de forma conjunta con TCL y se denominada TCL/Tk.

Ventajas

- Sencillez de programación
- Rapidez en el desarrollo de aplicaciones
- Gran velocidad comparada con otros lenguajes interpretados.

- Fácil modificación
- Multiplataforma
- Grand numero de extensiones gratuitas
- Posibilidad de incorporar nuevos comandos en lenguaje C++

3.2.3.- Elementos de programación

- **Variables**

Para crear una variable no es necesario declararla en el inicio del programa, una vez que se inicializa la variable se crea automáticamente utilizando el comando set, que consta de dos argumentos, el nombre y el valor de la variable.

```
Set var1 5
```

Para borrar la variable se utiliza el comando unset

```
Unset var1
```

Y para saber si la variable existe utilizamos info, donde devolverá 1 si existe, caso contrario devuelve 0.

Comentarios

Para poder comentar una línea de programación usaremos el signo #.

- **Comillas y llaves**

Las comillas y llaves tienen la misma función, la de agrupar una cadena de caracteres, teniendo la única diferencia que con las comillas podemos ingresar una variable dentro de la cadena.

- **Evaluación de comandos**

Es posible evaluar cualquier tipo de expresiones, matemáticas o no, que se encuentre entre corchetes; ejemplo:

```
Set longitud [string length avión]
5
```

El ejemplo lo que hace es evaluar la cadena de caracteres “avión” y setearla en una variable llamada longitud.

- **Expresiones Matemáticas**

Debido a que todas las variables son de tipo cadena de caracteres para realizar operaciones matemáticas debemos indicar al intérprete que considere dichas variables como números, esto mediante la utilización del comando expr.

```
Expr 7/2
3
```

Si deseamos el resultado con decimales solo es necesario utilizar un dato con coma.

```
Eprt 7.0/2
3.5
```

- **Principales operaciones**

+ ; -	Suma ; Resta
* ; / ; %	Multiplicación ; División ; Resto
== ; =!	Igual ; no igual
> ; < ; >= ; <=	Mayor que; Menor que ; Mayor igual que ; Menor igual que
&& ; ||	OR lógico ; AND lógico

Tabla 3.1: Simbología de operaciones principales

- **Estructuras de control de flujo**

La estructura más básica es la estructura If, la cual constara de una condición expresada de forma boleana, una acción primaria que es la instrucción que se ejecutara en caso de que la condición sea verdadera, y una acción secundaria que se ejecutara en el caso de que la condición se falsa mediante la utilización del comando else. En esta estructura se permite ir anidando estructuras if mediante la utilización elseif.

```
If { $x==0} {puts “si”} else {puts “no”}
```

El comando for sirve para realizar una acción repetidas veces, mediante la utilización de 4 argumentos que servirán para: dar una pauta de inicio, comprobación del condicionante, incrementar la variable y por último la acción a realizar dentro del bucle.

```
for {inicialización} {test} {incremento} {instrucción}
```

El bucle while es utilizado para realizar una misma actividad varias veces mientras se cumpla una condición.

```
while {condición} {instrucción}
```

- **Procedimientos**

Los procedimientos son una reunión de comandos bajo el mismo nombre, que nos servirán para usarlos como un comando más del lenguaje, permitiendo agilitar el proceso de programación, debido a su automatización de la secuencia de varios comandos.

Estos procesos podrán ser guardados en el mismo fichero del programa o en otro, incluso en otro directorio.

Se lo define de la siguiente manera:

proc nombre_del_proceso {parámetros} {cuerpo}

Donde, distinguimos tres argumentos: el primero que da el nombre de dicho procedimiento, el segundo son todos los parámetros que regirán en el procedimiento y el ultimo es el procedimiento en sí.

Una vez ejecutado el procedimiento, este, nos retornara el valor de la última acción realizada, en caso de querer visualizar algún otro valor deberemos usar el comando return dentro del procedimiento.

- **Ámbito de las variables**

Partiremos desde el concepto que una variable puede considerarse global o local; una variable global es aquella que va a ser utilizada durante todo el programa y debe ser declarada mediante el comando global al inicio del programa, mientras que una variable local es aquella que va a ser utilizada durante determinado tiempo de ejecución de algún comando o procedimiento caso contrario la variable es indefinida.
Muchos autores no recomiendan el uso de muchas variables globales, debido a que en algún momento del programa esta puede ser modificada por alguna desconcentración del programador.

- **Listas**

Una lista está conformada por varios strings separados por un espacio en blanco y agrupados mediante el uso de las comillas o de las llaves. Estos string podrán ser manipulados mediante diferentes comandos que veremos a continuación:

- El comando list crea una lista que a su vez puede ser asociada a una variable.
- El comando lappend agrega elementos al final de una lista ya creada o crea argumentos dentro de la misma.
- El comando concat nos ayuda a unir dos o más listas dentro de una sola.

- **Cargar ficheros**

Para la carga de ficheros importantes durante la ejecución del programa se utilizara el comando source, dichos ficheros pueden contener variables o procesos que se desean ser utilizados para el funcionamiento de nuestro programa.

```
source nombre_fichero
```

- **Ficheros entrada/salida**

Para abrir un fichero de entrada/salida de datos ocupamos el comando open.

```
open fichero acceso
```

Donde el primer argumento es en nombre del fichero a abrir y el segundo es el acceso que tendremos dentro del mismo; así mismo se puede utilizar el comando close para cerrar algún fichero.

- **Accesos de ficheros**

R	Abre para la lectura
r+	Abre para lectura y escritura
W	Abre para la escritura, sobre fichero existente o crea uno
w+	Abre para la lectura y escritura sobre fichero existente o crea uno
A	Abre para escritura sobre fichero existente
a+	Abre para escritura y lectura sobre fichero existente

Tabla 3.2: código de ficheros

3.3 Inicialización del NS-2

3.3.1.- Componentes

NS-2 es un simulador de redes dirigidos por eventos, el cual está diseñado para ser ejecutado en modo batch, donde el cerebro es el simulados NS. Mediante un script definimos dos aspectos fundamentales:

- La pila del protocolo y otros aspectos básicos del tipo de red.
- Datos del escenario a simular y el tipo y características del tráfico a utilizar.

En momento de ejecutar la simulación el script será llamado desde el terminal mediante el comando:

ns script

una vez este corriendo la simulación, se irán generando datos de salida, los cuales serán almacénanos en un fichero de traza, los cuales mediante algunos lenguajes, que trataremos más adelante, serán filtrados para así obtener resultados que servirán para ser evaluados o graficados mediante la herramienta nam del mismo NS-2.

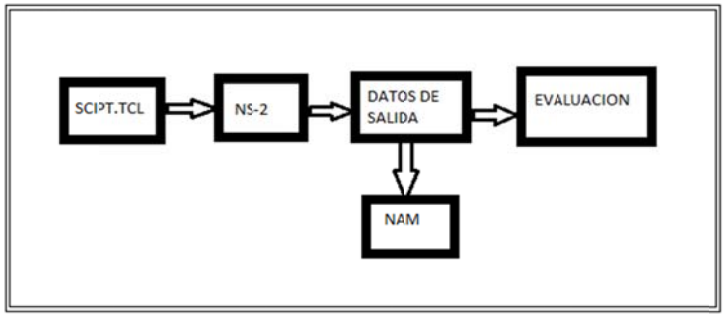

Figura 3.2: Esquema del proceso de simulación

Aspectos para la simulación

- Topología de red.- define los elementos estáticos (nodos y enlaces) y dinámicos de la red así como el encaminamiento (*routing*) de la misma.

- Eventos.- consiste en la planificación de la actividad de eventos de todos los elementos de la red y dejarla configurada.
- Datos estadísticos.- es la parte donde recopilaremos todos los datos generados en la simulación dentro de una traza.

- **Inicio del simulador**

Para proceder a definir una simulación lo primero que debemos hacer es crear una instancia en el simulador por medio del uso del comando:

set ns [new Simulator]

Así asignamos a la variable ns una instancia de clase *simulator* y *new* es la condición de creación.

Dentro de esta instancia configuraremos tres aspectos básicos:

- Nodos.- son los *routers* y *hosts* que existen dentro de la red, incluyendo su actividad de nivel inferior al nivel de red.
- Enlaces.- es aquí donde se define la topología de la red; es decir enlaces entre nodos como referimos en el capitulo anterior.
- Agentes.- aquí hablamos de nivel de transporte, aplicaciones y módulos de encaminamiento.

- **Obtención de la salida**

Para completar la simulación es necesario almacenar los datos obtenidos en un fichero de salida para posteriormente analizarlos y graficarlos. Esto incluyendo en el script la instrucción:

set <variable a asignar fichero> [open<nombre fichero> <modo fichero>]

Recordemos que existen diversos tipos de ficheros, en nuestro caso para poder ir almacenando las salidas siempre abriremos el fichero en tipo escritura, estos ficheros serán los que posteriormente iremos visualizando.

Para concluir la simulación es importante crear un procedimiento que valla cerrando todas las trazas que fuimos creando o en su defecto simplemente cerrar y visualizar mediante la herramienta grafica.

3.3.2.- Configuración de eventos

El NS-2 está regido por la simulación de eventos en un determinado tiempo que se da en las llamadas a los procedimientos, estos son configurados dentro del script mediante el comando:

ns at <tiempo> “<evento>”

Donde tiempo es el parámetro medido en segundos y evento son las instrucciones que serán representadas en forma de string. Así también, at devuelve un indicador de evento que puede ser utilizado como argumento para cancelar algún procedimiento.

De esta manera, para dar inicio a la simulación ocuparemos la herramienta run del simulador.

ns run

Así mismo la simulación termina una vez realizados todos los procedimientos, o por la llamada de un stop o por el procedimiento exit.

3.3.3 Nodos

Para la creación de los nodos se utilizara del simulador el método Node con el cual iremos asignando una dirección de red única. Cada nodo se irá guardando en un array Node_ que se indexa con identificadores de nodos.

set <nombre nodo> [$ns node]

Para la creación automatizada de varios nodos se los puede ir realizando mediante sentencias for, pero, teniendo en cuenta que el acceso a cada uno será como acceder a un vector.

Los nodos poseen dos 2 métodos principales: id, que permite dar un identificador al nodo; y el neightbors, que permite obtener los identificadores de los nodos vecinos.
El numero de nodos en son limitados a 256 pero pueden ser expandidos a 2^{22} mediante:

Node expandaddr

3.3.4 Enlaces

Una vez creados los nodos, para completar la red procedemos a comunicar los mismos mediante la creación de los enlaces, los mismos que pueden ser de dos tipos; simple en el caso que la comunicación sea en un solo sentido y doble si requerimos una comunicación full dúplex. Estos enlaces se caracterizan por:

- Los nodos que unen
- El ancho de banda
- El retardo requerido para las comunicaciones
- Tipo de cola.

Utilizando el mismo criterio que para la creación de los nodos, podemos crear los enlaces uno por uno o con la ayuda de un bucle, mediante la siguiente nomenclatura:

```
ns simplex_link $nodo1 $nodo2 $ancho_banda $retardo $tipo_cola
ns duplex_link $nodo1 $nodo2 $ancho_banda $retardo $tipo_cola
```

Para esto tenemos que definir previamente las variables correspondientes a los nodos (ordenados), retardo, ancho de banda y tipo de cola, en el caso de no definir el ancho de banda y retardo NS-2 tomara como valores por defecto de 1,5mbps y 100ms respectivamente, para los otros dos parámetros si es importante definirlos previamente. Par obtener una referencia de un enlace creado utilizamos:

```
Set mi_enlace [$ns link $nodo1 $nodo2]
```

En el caso de querer cambiar el valor del retardo en el transcurso del programa utilizaremos:

```
$ns delay $nodo1 $nodo2 $nuevo_retardo
```

3.3.5 Distribución de los enlaces

En lo que respecta al estado de los enlaces (activo o inactivo), es posible simular cuando un enlace esta caído mediante el cambio de estado del mismo, para lo que nos ayudaremos en la herramienta up y donw que sirven para activar y desactivar el enlace respectivamente, ejemplo:

```
if(![$mi_enlace up?]) {
        $mi_enlace up
}else{
        $mi_enlace donw
}
```

Donde “up?” nos ayuda a saber si un enlace esta o no activo.

Ahora bien, para poder realizar una distribución aleatoria de los estados de los enlaces podemos utilizar el método rtmodel el cual nos da un handle que nos servirá para

modelar un enlace, en este método es necesario la configuración de los siguientes parámetros:

- Distribución de los enlaces.
- Parámetros de distribución.
- Par de nodos que definen el enlace.

Por otra parte, el método rtmodel-delete lo que hace es tomar un handle mencionado y eliminarlo, poniendo así al enlace en un nuevo estado constante.

- **Tipos de distribución**

Determinístico

En este modelo es importante tratar 4 parámetros que son los que regirán en la creación de los enlaces. El primero y el ultimo corresponden al comienzo y al final de la dinámica, se recomienda siempre al primero dar un número mayor a 0,5 debido que los procesos de routing necesitan cierto tiempo de arranque, y el ultimo en caso de no expresar se dará hasta el final de la simulación; en cuanto al segundo y tercero nos configura el tiempo en el cual los nodos van a estar activo o no.
$ns rtmodel Deterministic 0,5 800ms 200ms $nodo1 $nodo2

Exponencial

Para este modelo necesitamos configurar dos parámetros que son la media en la cual se irá incrementando el tiempo de actividad o inactividad, los valores por defecto son de 10 y 1s respectivamente.

$ns rtmodel Exponential 0,2s 0,1s $mi_nodo

Manual

Este modelo nos permite asignar tiempos en los cuales queremos que un enlace cambie de estado, utilizando los métodos up y donw, pudiendo así considerar eventos especiales en la simulación de nuestra red.

Toma el primer argumento como el tiempo en el cual sucederá el evento, seguido del evento a simular (up o donw) y especificando el enlace o nodo a modificar.

$ns rtmodel-at $tiempo evento $nodo

Traza

En este método lo que utilizaremos es una traza que contiene el handle del dinamismo del enlace.

Para lo cual setearemos una traza, la misma que será utilizada.

```
Set mi_traza [open mi_fichero_de_trazas.tr r]
$ns rtmodel Trace $mi_traza $nodo1 $nodo2
```

3.3.6 Colas

Las colas son las encargadas de ir gestionando los mensajes que van llegando al nodo, debido a que están llegan y no son leídas aleatoriamente por lo que debemos tener un algoritmo que nos ayude a ir organizando dichos mensajes. Por lo que, para cada nodo será importante asignar una cola que encargue de dar prioridad a algunos mensajes y así la información no se pierda en el momento de arribar a un nodo.

Tipos de colas

Drop Tail

Este tipo de cola basa su estrategia en ir atendiendo paquetes según vallan llegando al nodo, en caso de descartar algún paquete, este será el último.

Fair Queuing (FQ)

Este proceso lo que hace es asignar un tiempo determinado de acceso del paquete a la cola, lo que garantiza una forma equitativa de acceso en el flujo.

Stochastic Fair Queing (SFQ)

Basado en el FQ con la única variación que los paquetes a los flujos de entrada al nodo serán asignados estocásticamente, por lo que habrá menos flujos de entrada que paquetes que tipos de paquetes.

Parámetros:

- Maxqueue_ es el tamaño máximo que puede tener una cola asociada a un flujo de entrada
- Buckets_ es el numero de flujos que pueden existir

Deficit round Robin Schedule (DRRS)

Este proceso lo que va hacienda es, a partir, de un conjunto de flujos, ir asignando un tiempo determinado a cada uno de ellos, conocida como política de Roun Robin (RR). Lo optimo seria usar SFQ para asignar los flujos a las colas de entrada y RR para ir escogiendo dichas colas.

Parámetros:

- Buckets_ es el numero de flujos que existen

- Blimit_ es el tamaño de búfer compartido expresado en bytes
- Quantum_ es la cantidad que puede transmitir cada flujo durante su turno
- Mask_ valdrá 1 cuando los paquetes de un mismo flujo tengan un mismo identificador de nodo.

Ramdon Early-detection (RED)

Este método es el más utilizado dentro del NS-2, y trata en calcular matemáticamente el tamaño de cola a utilizar para de esta forma poder marcar un paquete o desecharlo.

Parámetros:

- Bytes_ activa el modo byte de RED y hará que conforme varíe el tamaño del paquete de entrada varíe la posibilidad que este se descarte
- Queue-in-bytes_ mediante su activación permite medir el tamaño de la cola en bytes o en paquetes.
- Tres_ Es el valor mínimo del tamaño de la cola y se mide en paquetes.
- Mean-pktsize_ tamaño en bytes estimado de un paquete.
- Q_weight_ peso de la cola
- wait_ su activación permite mantener un intervalo de los paquetes descartados.
- linterm_ controla la probabilidad de descarte de un paquete.
- Setbit_ marca los paquetes indicando la congestión de los mismos.
- drop-tail_ activamos en el caso de utilizar una política FIFO par el descarte de paquetes.

Class Based Queuing (CBQ)

Este método trata en dar distintos tipos de clases a los paquetes dependiendo del nodo del cual provengan, pudiendo así dar diferentes tratamientos a los distintos paquetes. Para esto utilizaremos las siguientes sentencias:

```
$cbq insert <clase>
```

Le da una clase de tráfico a un enlace asociado con cbq

$cbq bind <mi clase> <id1> [id2]

Le da una clase “mi clase” a un paquete de identificador id1 o a un grupo con rango id1,id2

$cbq algorithm <algoritmo>

Da la clase por medio de un algoritmo interno.

CBQ/WRR

Asocia colas por medio de una política CBQ de clases, pero con prioridad según el peso de cada una de ellas (WRR).

Parámetro:

- Maxpkt_ es el ancho de banda máximo que tendrá un paquete expresado en bytes

3.3.7 Configuración de colas

Cada uno de los tipos de colas mencionados tiene parámetros y métodos específicos, para nuestro estudio veremos los más comunes entre todos para un alcance más amplio.

Lo primero a tratar va a ser la capacidad de la cola, la cual puede ser modificada mediante el método queue-limit o modificando la variable limit de la cola.

```
$ns queue-limit $nodo1 $nodo2 $nuevo_tamaño
$mi_cola cola set limit_ $nuevo_tamaño
```

Para referenciar una cola usaremos el método queue

```
set mi_enlace $nodo1 $nodo2
set cola [$mi_enlace queue]
```

existe dos variables tipo bandera que sirven para saber cuándo una cola está bloqueada (bloqued) y para cuando una cola debe ser desbloqueada debido a que el último paquete fue enviado y no recibido (unblock_on_resume), las mismas que se expresaran de manera boleana, es decir, es verdadera (T,t,1) o falsa (F,f,0) respectivamente.

3.3.8 Agentes

Un agente es el encargado de definir cuando un paquete es transmitido o receptado, es decir, dentro del simulador NS-2 es de vital importancia el manejo de agentes debido a que estos son los encargados del tratamiento de los paquetes.

- Agente Emisor.- es el que recoge los paquetes enviados de un nodo y encaminarlos por la red hacia otro u otros nodos.
- Agente Receptor.- este recoge los paquetes y les dará el tratamiento indicado por los mismos sin transmitirlos a ningún lado.

```
set <nuevo_agente> [new Agent/<tipo>]
```

Dentro de un agente podemos variar algunos parámetros que citaremos a continuación:

- Addr_ asigna una dirección del nodo de origen
- Dst_ asigna una dirección del nodo de destino
- Size_ expresa el tamaño del paquete
- Type_ nos da el tipo del paquete
- Fid_ identificador del flujo IP
- Prio_ da la prioridad IP
- Flags_ son las banderas de un paquete
- Defttl_ es el tiempo de vida de un paquete

Estos pueden ser modificados mediante:

$<mi_agente> set <parámetros>

Este agente será asignado a un nodo mediante:

$ns attach-agent $nodo1 $mi_agente

Para deslindar un agente a un nodo haremos igual solo que con la sentencia detach-agent.

Con los nodos asignados sus agentes el siguiente paso es el conectarlos entre ellos, para lo cual vamos a usar connect:

$ns connect $<Agente_fuente> <Agente_destino>

Con los nodos conectados, se puede ir haciendo interacciones entre ellos como:

- reset.- que inicializa un agente
- dts.- devuelve la dirección del nodo
- port.- Devuelve el puerto del nodo
- dts-addr.- nos da la dirección del puerto de destino
- dts-port.- nos da el puerto de destino.

- **Tipos de Agentes.**

Agentes de transporte

Para motivo de estudio del simulador NS-2, indicaremos dos tipos protocolos de transporte que son los más utilizados, UDP (*User Datagram Protocol*) y TCP.

- El protocolo UDP lo utilizaremos para aplicaciones generadoras de tráfico, que quiere decir que podrá ofrecer servicios de transporte a agentes de aplicación que no requieran de una conexión.

set <transporte> [new Agent/UDP]

Un agente UDP debe ser asignado a un nodo que soporte un agente generador de tráfico y conectado con un node del mismo tipo o uno de modo null

- El agente TCP se utilizara mas para la conexión de servicios más enfocado a las aplicaciones con comunicaciones unidireccionales, tales como, Telnet y Fttp.

De la misma manera un agente TCP debe ser asignado en el nodo emisor y ser conectado a un nodo null (vacio).

Agentes de aplicación

Las aplicaciones se apoyan en los agentes de conexión y servirán únicamente para realizar simulaciones de aplicaciones mediante el envío de bytes, estas aplicaciones simuladas son Telnet y Ftp, como también pueden existir aplicaciones que simulen una generación de tráfico.

El ciclo de vida de las aplicaciones es: se crean, se asocian a un tipo de agente de transporte, se lanzan y se finalizan.

3.3.9 Encaminamiento o Routing

Encaminamiento Unicast

Los métodos de enrutamiento unicast pueden ser de tres tipos: Static, Session, Dymanic; los mismo que para su implementación dentro del simulador NS-2 se hará con la utilización del método llamado rtproto, definiendo dos parámetros principales: el primero será el tipo de encaminamiento a utilizar que en caso de no ser especificado tomara por defecto el encaminamiento unicast estático, esto es, que no va a cambiar la topología durante la simulación; y el segundo parámetro es el asocia al grupo de nodos que van a utilizar el encaminamiento deseado, esto se hace por medio de una lista, en caso de no especificar se tomara a todos los nodos.

La forma más eficaz de implementar un encaminamiento es usando el encaminamiento unicast dynamic, debido a que en la concepción de la red siempre a los paquetes enviamos por la ruta más corta, con lo que si un enlace se cae los paquetes se estarán perdiendo continuamente, este método lo que permite es ir asignando por medio de una algoritmo la ruta más corta disponible, este algoritmo tiene el nombre de Distance Vector, y hace que se busque un enlace alternativo hasta que se recupere el enlace perdido.

Entonces, lo habilitaremos al comienzo del programa mediante la sentencia:

```
$ns rtproto DV
```

Encaminamiento Multicast

Este protocolo de encaminamiento lo que permite es la comunicación punto-multipunto en la red, y lo habilitaremos mediante la utilización de –multicast on en la misma sentencia donde creamos una nueva simulación:

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

Es posible también agrupar nodos para darles el mismo tipo de encaminamiento solo a ciertos nodos, esto se realiza mediante la sentencia join-group, que lo que hace es ir añadiendo nodos a un grupo determinada, para ello es importante a su vez crear una dirección de multicast usando allocaddr. Este método es parecido al conect que habíamos ya estudiado, la diferencia está en la utilización del parámetro dist_ que nos otorga la facultad de saber que nodo esta dentro de su grupo. Así mismo, un nodo puede abandonar el grupo mediante la sentencia leave-group, siendo utilizado de la misma forma.

La forma de crear un encaminamiento multicast es con el método mrtproto:

```
$ns mrtproto $protocolo_multicast
```

3.3.10 Redes de Área Local

En el simulador NS-2 se utiliza un tipo de redes el cual basa en la conexión de estaciones mediante un medio compartido el cual funciona con el envío de un paquete a través del medio compartido, y por medio de las direcciones de origen y destino se determinara cual será la estación que recibirá dicho paquete, además, cada estación se puede comunicar con otra a través de enlaces punto a punto tal como lo ya mencionado.

Para lo cual, la forma de inicializar una red será mediante:

```
$ns make-lan
```

{lista_nodos,ancho_banda,retardo,subcapa_802,2,tipo_cola,acceso, canal}

Un ejemplo real seria:

```
set lan [$ns make-lan $lista_nodo 10Mb 1ms LL Queue/droptail Mac/802_3 chanel]
```

Donde el ancho de banda es de 10 Mb con retardo de 1ms, LL es la representación de la subcapa 802,2, con un tipo de cola dropa tail y acceso mac, el canal será un canal físico denominado chanel.

3.3.11 Perdidas

Con el fin que la simulación sea lo más acercado a la realidad, dentro de la misma introduciremos un modelo simulado de errores que introducirá perdidas de paquetes o en su defecto enviara paquetes a otro destino, este método dentro del NS-2 se lo llama ErrorModel.

Lo visualizaremos en un ejemplo práctico.

“ #crear modelo de error y asignar taza de error de 1 por ciento
set loss_module [new ErrorModel]
$loss_module set rate_ 0.01

#configurar la unidad de error y la variable aleatoria
$loss_module unit pkt
$loss_module ranvar [new RandomVariable/Uniform]

#Asignar el destino de los paquetes descartados
$loss_module dropa-target [new Agent/null]

#insertar modelo de error en el enlace n0-n1
$ns lossmodel $loss_module $n0 $n1 ”
(www.isi.edu.ns.ns-documentacion.htlm)

3.3.12 Trazas

Las trazas son ficheros donde se almacenaran todos los resultados de la simulación para ello hemos de abrir un fichero previamente:

set mi_fichero [open nombre_fichero w]

Por medio de la sentencia trace-all el simulador realizar una traza de todos los enlaces de la red simulada:

set [open out.tr.w]
$ns trace-all $f

El cual nos daría un resultado como el de la figura

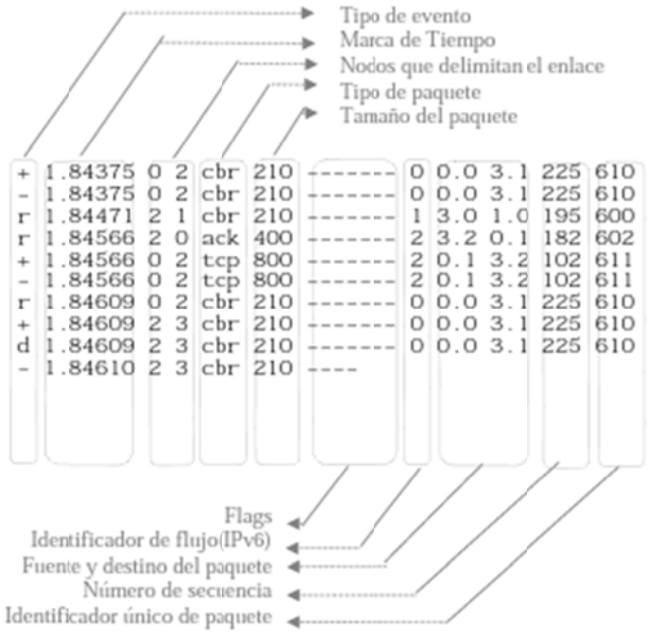

Figura 3.3: Grafico de traza

Fuente: www.isi.edu.ns.ns-documentation.htlm

Para realizar una traza para monitorear únicamente la actividad entre dos nodos se utilizara el método create-trace, donde únicamente indicaremos el tipo de traza, el fichero donde descargar y los nodos relacionados.

Para ver la actividad de una cola lo que utilizaremos es trace-queue, y de igual forma, indicamos el fichero y los nodos relacionados con la cola a monitorear

CAPITULO 4

PRACTICAS

En este capítulo desarrollaremos una guía de prácticas destinadas al alumno y al profesor, diferenciando correctamente cada una de ellas mediante la creación de una guía para alumnos y otra para el profesor, lo que nos ayudara a incentivar al estudiante y al correcto seguimiento del profesor.

Las prácticas tienen como objetivo crear en el alumno una destreza para la simulación de redes de telecomunicaciones básicas y mediante la herramienta grafica observar su funcionamiento así como sus falencias, y tener la capacidad de solucionar las mismas.

4.1.- Practica 1 (Alumno y profesor)

4.1.1.- Problema

Esta es un ejemplo inicial el cual iremos desarrollando paso a paso para crear un ejercicio básico que se basa en conectar dos nodos por un enlace dúplex de 1Mb de ancho de banda, 10ms de retardo y con un tipo de cola “DropTail”. Usaremos un protocolo UDP con trafico “Contant Bit Rate” (CBR), con un tamaño de paquetes de 500 bytes a 200 paquetes por segundo.

4.1.2.- Pasos

Paso 1.- Creación de un objeto simulador. Este es el paso principal en toda simulación, aquí incluiremos todo nuestro script.

set ns [new Simulator]

Paso 2.- Crear los ficheros necesarios para la obtención de los resultados. Con estos datos lograremos la grafica de salida.

```
Set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
```

Paso 3.- Crear un procedimiento para la finalización de la simulación. Dentro de este procedimiento se almacena todos los datos en el fichero de salida y será llamado oportunamente.

```
proc finish {} {
        global ns nf f0
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
        }
```

Paso 4.- Crear los nodos con su respectivo enlace.

```
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

Paso 5.- Configurar el tipo de protocolo y trafico para los nodos.

```
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
```

Paso 6.- Configuración de paquetes y enviarlos.

```
$cbr0 set packetSize 500
$cbr0 set interval 0.005
```

```
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
$ns connect $udp0 $null0
```

Paso 7.- Planificación de tiempos, empiezo, parada y finalización.

```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
```

Paso 8.- Llamar al procedimiento final y correr el programa.

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

4.1.3.- Grafico final

4.2.- Practica 2 (Alumno)

4.2.1.- Problema

Crear una simulación de 4 nodos, los cuales dos se conectan independientemente (Nodos Fuente) a uno que se encarga de enviar de forma sinconizada (Nodo Router) los paquetes a un nodo de sumidero. Teniendo los enlaces de los nodos fuente al nodo router un ancho de banda de 5Mbps, un retardo de 2ms y un tipo de cola “DropTrail” y el enlace de los nodos router y sumidero es de 1,5Mbps, un retardo de 10ms y una

cola tipo “DropTrail”. El protocolo del primer nodo al tercero es UDP con un tráfico CBR y el protoco del segundo nodo al tercero será TCP y para el nodo tendremos un sumidero capas de recibir UDP y TCP.

4.2.2.- Pasos

Paso 1.- Creación de un objeto simulador.

……

Paso 2.- Crear los ficheros necesarios para la obtención de los resultados

……..

……

……

……

Paso 3.- Crear un procedimiento para la finalización de la simulación.

……

……

……..

……

……

……

……..

Paso 4.- Crear los nodos con su respectivo enlace.

……..

……..

……..

……..

……..

……..

……..

……..

……..
……..

Paso 5.- Configurar el tipo de protocolo y trafico para el primer nodo.

……
……
……
……
……
……

Paso 6.- Configurar nodo de sumidero y conectar.

……
……
……
……
……

Paso 7.- Planificación de tiempos, empiezo, parada y finalización del primer enlace.

……
……

Paso 8.- Realizar el paso 6,7 y 8 para el segundo enlace. Tener en cuenta que debe hacerse de manera síncrona.

……
……..
……..
……..
……..

……..
……..
……..
……
……..
……..
……..
……..
……..
……
……
……..
……

Paso 9.- Correr el programa.

……..

4.2.3.- Grafico final

4.2.4.- Solución. (Profesor)

Paso 1.- Creación de un objeto simulador.

set ns [new Simulator]

Paso 2.- Crear los ficheros necesarios para la obtención de los resultados

```
Set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
```

Paso 3.- Crear un procedimiento para la finalización de la simulación.

```
proc finish {} {
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        exit 0
        }
```

Paso 4.- Crear los nodos con su respectivo enlace.

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 5Mb 2ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
```

Paso 5.- Configurar el tipo de protocolo y trafico para el primer nodo.

```
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
```

Paso 6.- Configurar nodo de sumidero y conectar.

```
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0
```

Paso 7.- Planificación de tiempos, empiezo, parada y finalización del primer enlace.

```
$ns at 0.0 "$cbr0 start"
$ns at 3.0 "$cbr0 stop"
```

Paso 8.- Realizar el paso 6,7 y 8 para el segundo enlace. Tener en cuenta que debe hacerse de manera síncrona.

```
set tcp [new Agent/TCP]
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns connect $tcp $sink
$ns at 1.0 "$ftp start"
$ns at 1.3 "$ftp stop"
$ns at 3.0 "terminar"
```

Paso 9.- Llamar al procedimiento final y correr el programa.

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

4.3.- Practica 3 (Alumno)

4.3.1.- Problema

Este ejemplo es muy similar al anterior, pero en este vamos a observar que sucede si los paquetes de los nodos fuente son enviados al mismo tiempo, para esto creamos dos nodos fuente con enlaces al nodo router de 1Mb, 10ms y tipo de cola droptail; y el enlace del nodo router al nodo sumidero será igual solo que con tipo de cola SFQ, el trafico será CBR con topología UDP.

4.3.2.- Pasos

Paso 1.- Creación de un objeto simulador.

…….

Paso 2.- Definimos colores para distinguir la procedencia de los paquetes.

…….
…….

Paso 3.- Crear los ficheros necesarios para la obtención de los resultados

…….
…….
…….
…….

Paso 4.- Crear un procedimiento para la finalización de la simulación.

…….
…….
……..

……
……
……
……..

Paso 5.- Crear los nodos con su respectivo enlace y ubicar dentro del simulador.

……..
……..
……..
……..
……..
……..
……..
……..
……..
……..

Paso 6.- Monitorear colas.

……

Paso 7.- Configurar el tipo de protocolo y trafico para los nodos fuentes y asignar colores a sus paquetes.

……
……
……
……
……
……
……
……

Paso 8.- Configurar paquetes.

……
……
……..
……
……..
……

Paso 9.- Configurar nodo de sumidero y conectar.

……..
……
……
……
……

Paso 10.- Planificación de tiempos de empiezo, parada y finalización.

……
……
……..
……..
……...

Paso 11.- Correr el programa.

……

4.3.4.- Grafico final

4.3.5.- Solución. (Profesor)

Paso 1.- Creación de un objeto simulador.

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

Paso 2.- Definimos colores para distinguir la procedencia de los paquetes.

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

Paso 3.- Crear los ficheros necesarios para la obtención de los resultados

```
Set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
```

Paso 4.- Crear un procedimiento para la finalización de la simulación.

```
proc finish {} {
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        exit 0
        }
```

Paso 5.- Crear los nodos con su respectivo enlace y ubicar dentro del simulador.

```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
```

```
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link $n2 $n3 1Mb 10ms SFQ
$ns duplex-link-op $n2 $n3 orient right
```

Paso 6.- Monitorear colas.

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

Paso 7.- Configurar el tipo de protocolo y trafico para los nodos fuentes y asignar colores a sus paquetes.

```
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
$udp0 set fid 1 # Color Azul
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
$udp1 set fid 2 # Color Rojo
set cbr0 [new Application/Traffic/CBR]
set cbr1 [new Application/Traffic/CBR]
```

Paso 8.- Configurar paquetes.

```
$cbr0 set packetSize 500
$cbr1 set packetSize 500
$cbr0 set interval 0.005
$cbr0 attach-agent $udp0
$cbr1 set interval 0.005
$cbr1 attach-agent $udp1
```

Paso 9.- Configurar nodo de sumidero y conectar.

```
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0
```

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

Paso 10.- Planificación de tiempos de empiezo, parada y finalización.

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

Paso 11.- llamar al procedimiento final y correr el programa.

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

4.4.- Practica 4 (Alumno y profesor)

4.4.1.- Problema

En esta práctica aprenderemos a simular eventos multicast y monitorear errores, que para efecto de esta práctica hemos propuesto que se den entre los nodos 1 y 2 y entre los nodos 1 y 3; está conformada por 7 nodos con enlaces dúplex que conectan a todos, con un ancho de banda 1Mb, un retardo de 10ms y enrutamiento DropTrail.

4.4.2.- Pasos

Paso 1.- Creación de un objeto simulador y activar el modo multicast.

```
set ns [new Simulator –multicats on]
```

Paso 2.- Crear los ficheros necesarios para la obtención de los resultados

```
Set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
```

```
$ns namtrace-all $nf
set f_recvr2_loss [open recvr2.tr w]
set f_recvr4_loss [open recvr2.tr w]
set f_recvr6_loss [open recvr2.tr w]
set f_recvr2_ recv [open recvr2.tr w]
set f_recvr4_ recv [open recvr2.tr w]
set f_recvr6_recv [open recvr2.tr w]
```

Paso 3.- usando un bucle for crear las estaciones y los nodos que formaran la red.

```
Set nmax 6
for {set i 0} {$i <= $nmax} {incr i} {
set n($i) [$ns node]
```

Paso 4.- definir grupos de direcciones multicast

```
set grupo1 [Node allocaddr]
set grupo2 [Node allocaddr]
```

Paso 5.- Crear los enlaces y ubicamos en el simulador.

```
$ns duplex-link $n(0) $n(1) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(3) 1Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 1Mb 10ms DropTail
$ns duplex-link $n(3) $n(5) 1Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 1Mb 10ms DropTail
$ns duplex-link-op $n(0) $n(1) orient right
$ns duplex-link-op $n(1) $n(2) orient right
$ns duplex-link-op $n(1) $n(3) orient right-down
$ns duplex-link-op $n(3) $n(4) orient right
$ns duplex-link-op $n(3) $n(5) orient right-down
$ns duplex-link-op $n(5) $n(6) orient right
$ns duplex-link-op $n(0) $n(1) queuePos 0.5
```

```
$ns duplex-link-op $n(1) $n(0) queuePos 0.5
$ns duplex-link-op $n(3) $n(1) queuePos 0.5
```

Paso 6.- Crear los Errores de los enlaces N1-N2 y N1-N3.

```
set loss_module1 [new ErrorModel]
$loss_module 1 set rate_0.005
$loss_module 1 set unit pkt
$loss_module 1 ranvar [new RandomVariable/Uniform]
$loss_module 1 drop-target [new Agent/Null]
$ns lossmodel $loss_module 1 $n(1) $n(2)
set loss_module2 [new ErrorModel]
$loss_module 2 set rate_0.1
$loss_module 2 set unit pkt
$loss_module 2 ranvar [new RandomVariable/Uniform]
$loss_module 2 drop-target [new Agent/Null]
$ns lossmodel $loss_module 2 $n(1) $n(3)
```

Paso 7.-Aplicar parámetros de configuracion de la coneccion multicast aplicando DenseMode.

```
set mproto DM
set mrthandle [$ns mrtproto $mproto {}]
```

Paso 8.- Configurar el tipo de protocolo y trafico para los nodos fuentes y asignar colores a sus paquetes.

```
set emisor1 [new Agent/UDP]
$ns attach-agent $n(0) $emisor1
set cbr1 [new Application/traffic/CBR]
$cbr1 attach-agent $emisor1
$emisor1 set dst_$grup1
$emisor1 set dst_port_0
```

```
set emisor2 [new Agent/UDP]
$ns attach-agent $n(0) $emisor2
set cbr2 [new Application/traffic/CBR]
$cbr2 attach-agent $emisor2
$emisor2 set dst_$grup2
$emisor2 set dst_port_0

set rcvr2 [new Agent/LossMonitor]
set rcvr4 [new Agent/LossMonitor]
set rcvr6 [new Agent/LossMonitor]
$ns attach-agent $n(2) $rcvr2
$ns attach-agent $n(4) $rcvr4
$ns attach-agent $n(6) $rcvr6
```

Paso 9.- Planificación de tiempos de empiezo, parada y finalización.

```
$ns at 1.0 "$cbr1 start"
$ns at 1.0 "$cbr2 start"
$ns at 0.0 "$n(2) join-group $rcvr2 $group1”
$ns at 0.0 "$n(4) join-group $rcvr4 $group2”
$ns at 0.0 "$n(6) join-group $rcvr6 $group2”
$ns at 0.0 "comenzar_medidas"
$ns at 0.1 "medir"
$ns at 5.0 "terminar"
```

Paso 10.- Procedimiento para inicializar variables para monitorar.

```
proc comenzar_medidas {} {

        global ns rcvr2 rcvr4 rcvr6
        $rcvr2 set bytes_0
        $rcvr2 set nlost_0
        $rcvr4 set bytes_0
        $rcvr4 set nlost_0
```

```
        $rcvr6 set bytes_0
        $rcvr6 set nlost_0
}
```

Paso 11.- procedimiento para medir bytes recibidos y perdidos por cada agente.

```
proc medir {} {
global ns f_recvr2__loss f_recvr4_loss f_recvr6_loss f_recvr2_recv f_recvr4_recv
f_recvr6_recv rcvr2 rcvr4 rcvr6

set ns [Simulator instance]
set frecuencia 0.1
set now [$ns now]
set recibidos2 [$rcvr2 set bytes_]
set perdidos2 [$rcvr2 set nlost_]
set recibidos4 [$rcvr4 set bytes_]
set perdidos4 [$rcvr2 set nlost_]
set recibidos6 [$rcvr6 set bytes_]
set perdidos6 [$rcvr6 set nlost_]
puts $f_recvr2_loss “$now $perdidos2”
puts $f_recvr4_loss “$now $perdidos4”
puts $f_recvr6_loss “$now $perdidos6”
puts $f_recvr2_recv “$now $recibidos2”
puts $f_recvr4_recv “$now $recibidos4”
puts $f_recvr6_recv “$now $recibidos6”
$rcvr2 set bytes_0
$rcvr2 set nlost_0
$rcvr4 set bytes_0
$rcvr4 set nlost_0
$rcvr6 set bytes_0
$rcvr6 set nlost_0
$ns at [expr $now+$frecuencia] “medir”
}
```

Paso 12.- crear el procedimiento de finalización.

```
proc terminar {} {

global ns f_recvr2_loss f_recvr4_loss f_recvr6_loss f_recvr2_recv f_recvr4_recv
f_recvr6_recv
close $f_recvr2_loss
close $f_recvr4_loss
close $f_recvr6_loss
close $f_recvr2_recv
close $f_recvr4_recv
close $f_recvr6_recv
$ns flush-trace
exit 0
}
```

Paso 13.- llamar al procedimiento final y correr el programa.

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

4.4.3.- Grafico final

4.5.- Practica 5 (Alumno)

4.5.1.- Problema

En esta práctica simularemos una red dinámica, es decir usaremos un encaminamiento dinamico, lo que nos permite prever una eventual caída en algún enlace. Esta es una topología circular con 7 nodos, donde el N0 es el nodo fuente y el N3 es el nodo destino, usaremos sentencias for para crear nodos y conectarlos.

4.5.2.- Pasos

Paso 1.- Creación de un objeto simulador y habilitar el encaminamiento dinámico.

……
……

Paso 2.- Crear los ficheros necesarios para la obtención de los resultados

……..
……..
……..
……..

Paso 3.- Crear un procedimiento para la finalización de la simulación.

……
……
……
……
……
……
……
……

Paso 4.- Crear los nodos y conectarlos.

……
……
……

Paso 5.- Configurar el tipo de protocolo y trafico para los nodos fuentes y asignar colores a sus paquetes.

……
……
……
……

Paso 6.- Configurar paquetes.

……
……
……
……

Paso 7.- Configurar nodo de sumidero y conectar.

……
……
……
……

Paso 8.- Planificación de tiempos de empiezo, parada y finalización.

……
……
……

……

Paso 9.- llamar al procedimiento final y correr el programa.

……
……

4.5.3.- Grafico final

4.5.4.- Solución. (Profesor)

Paso 1.- Creación de un objeto simulador y habilitar el encaminamiento dinamico.

```
set ns [new Simulator]
$ns rtproto DV
```

Paso 3.- Crear los ficheros necesarios para la obtención de los resultados

```
Set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
```

Paso 4.- Crear un procedimiento para la finalización de la simulación.

```
proc finish {} {
```

```
        global ns nf f0
        $ns flush-trace
        close $tf
        close $nf
        exec nam out.nam &
        exit 0
        }
```

Paso 5.- Crear los nodos y conectarlos.

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

Paso 7.- Configurar el tipo de protocolo y trafico para los nodos fuentes y asignar colores a sus paquetes.

```
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0

set cbr0 [new Application/Traffic/CBR]
```

Paso 8.- Configurar paquetes.

```
$cbr0 set packetSize_500
$cbr0 set interval_0.005
$cbr0 attach-agent $udp0
```

Paso 9.- Configurar nodo de sumidero y conectar.

```
set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0
$ns connect $udp0 $null0
```

Paso 10.- Planificación de tiempos de empiezo, parada y finalización.

```
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
$ns at 4.5 "$cbr0 stop"
```

Paso 11.- llamar al procedimiento final y correr el programa.

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

4.6.- Practica 6 (Alumno)

4.6.1.- Problema

Esta práctica nos ayudara a analizar el ancho de banda de un enlace, que para esta práctica será un enlace que hace la función de “cuello de botella”, es decir une diferentes comunicaciones. Es una topología con 8 nodos que se comunican de dos en dos, todos los enlaces son de 100Mbps con un retardo de 10ms y enrutamiento droptail, y el enlace “cuello de botella” es de 20 Mbps, 11ms y enrutamiento RED.

4.6.2.- Pasos

Paso 1.- Creación de un objeto simulador e inicializar variables.

```
……
……..
……..
……..
……..
……..
……..
……..
```

……..
……..
……..

Paso 2.- Crear los ficheros necesarios para la obtención de los resultados

……..
……..
……..
……..

Paso 3.- Crear un procedimiento para la finalización de la simulación.

……..
……..
……..
……..
……..
……..
……..
……..
……..
……..

Paso 4.- Crear los nodos con su respectivo enlace y ubicarlos dentro del simulador.

……..
……..
……..
……..
……..
……..
……..
……..

……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..

Paso 5.- Configurar el tipo de protocolos y tráficos para la primera comunicación.

……..
……..
……..
……..
……..
……..

Paso 6.- Creamos una aplicación de tipo exponencial.

……..
……..
……..
……..
……..
……..
……..
……..

Paso 6.- creamos el segundo enlace, por medio de un bucle for configuramos el agente TCP.

……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..

Paso 7.- Planificación de tiempos para el enlace FTP.

……..
……..
……..
……..
……..
……..
……..
……..
……..
……..

Paso 9.- Llamar al procedimiento final y correr el programa.

……..
……..

4.6.3.- Grafico final

4.6.4.- Solución. (Profesor)

Paso 1.- Creación de un objeto simulador e inicializar variables.

```
set ns [new Simulator]
set rtt 11ms
set bw 20Mb
set bw_link 20
```

```
set queue_size [expr ($bw_link*11)/2]
set r_max_thresh [expr $queue_size/2]
set num_tcp 3
set fin_sim 30.0
```

Paso 2.- Crear los ficheros necesarios para la obtención de los resultados

```
Set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
```

Paso 3.- Crear un procedimiento para la finalización de la simulación.

```
proc terminar {} {
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        puts ëjecutando nam…”
        exec nam out.nam
        exit 0
        }
```

Paso 4.- Crear los nodos con su respectivo enlace y ubicarlos dentro del simulador.

```
Set n0 [$ns node]
Set n1 [$ns node]
Set n2 [$ns node]
Set n3 [$ns node]
Set n4 [$ns node]
Set n5 [$ns node]
Set n6 [$ns node]
Set n7 [$ns node]
```

```
$ns duplex-link $n0 $n4 100Mb 10ms DropTail
$ns duplex-link $n6 $n4 100Mb 10ms DropTail
$ns duplex-link $n4 $n5 $bw $rtt RED
$ns duplex-link $n5 $n1 100Mb 10ms DropTail
$ns duplex-link $n2 $n4 100Mb 10ms DropTail
$ns duplex-link $n5 $n3 100Mb 10ms DropTail
$ns duplex-link $n5 $n7 100Mb 10ms DropTail

Queue/RED set thresh_5
Queue/RED set maxthresh_50
$ns queue-limit $n4 $n5 $queue_size

$ns duplex-link-op $n0 $n4 orient right-down
$ns duplex-link-op $n2 $n4 orient right-up
$ns duplex-link-op $n6 $n4 orient right
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n5 $n1 orient right-up
$ns duplex-link-op $n5 $n7 orient right
$ns duplex-link-op $n5 $n3 orient right-down
$ns duplex-link-op $n4 $n5 queuePos 0.5
```

Paso 5.- Configurar el tipo de protocolos y traficos para la primera comunicación.

```
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set null [new Agent/Null]
$ns attach-agent $n1 $null
$ns connect $udp0 $null
```

Paso 6.- Creamos una aplicacion de tipo exponencial.

```
set [new Application/Traffic/Exponential]
$exp set packetSize_512
```

```
$exp set burst_time 500ms
$exp set idle_time 500ms
$exp set rate_500k
$exp attach-agent $udp
$ns at 0.0 “$exp start”
```

Paso 6.- creamos el segundo enlace, por medio de un bucle for configuramos el agente TCP.

```
for {set i 0} {$i < $num_tcp} {incr i} {
        set tcps($i) [new Agent/TCP/Reno]
        set tcps($i) [new Agent/TCPSink]
        set ftp ($i) [new Source/FTP]

        $ftp ($i) set agent_$tcps($i)
        $tcps ($i) set fid_ [expre 20 + $i]
        $tcps($i) set window_1024

        $tcps($i) set packetSize_512

        $ns attach-agent $n6 $tcps($i)
        $ns attach-agent $n7 $tcpr($i)
        $ns connect $tcps($i) $tcpr($i)
}
```

Paso 7.- Planificación de tiempos para el enlace FTP.

```
for {set i 0} {$i < $num_tcp} {incr i} {
        set start_time_tcp($i) [expr 5.0+2*$i]
        $ns at $start_time_tcp($i) “$ftp($i) start”
}

for {set i 0} {$i < $num_tcp} {incr i} {
        set end_time_tcp($i) [expr 20.0+2*$i]
```

```
        $ns at $end_time_tcp($i) “$ftp($i) stop”
}
```

Paso 9.- Llamar al procedimiento final y correr el programa.

```
$ns at $fin_sim "terminar"
$ns run
```

CONCLUSIONES

Linux representa una plataforma de gran ayuda ya que al ser un software libre donde se puede acceder a su programación y modificar estructuras permite el fácil manejo de redes siendo muy utilizado dentro de esta rama de las telecomunicaciones; además de su fácil instalación y posterior uso.

Las redes de telecomunicaciones son extensas en las que hemos profundizado únicamente en las más utilizadas y las de mejor rendimiento, las cuales fueron simuladas dentro del formato del software llamado Network Simulator.

En el NS-2 hallamos una herramienta muy útil y de gran expansión en el desarrollo de simulaciones, que dentro de nuestro tema se sentaron pautas importantes y bases para el manejo de este software.

Por el hecho de que NS-2 maneja una programación específica, hemos logrado aprender a establecer la correcta configuración de los parámetros a utilizar para la concepción de una red, lo que nos ayuda al éxito en una simulación asemejando lo máximo posible a la vida real.

Dentro del manual de prácticas se consigue sentar bases no solo de programación sino también de conceptos reales sobre redes y tratamientos de datos y paquetes importantes dentro de una comunicación, con una metodología que llega hasta una parte grafica, lo que se considera como un incentivo para que el estudiante aborde mas el tema y pueda utilizar estos conocimientos para un posterior estudio ya sea de modo personal o para material didáctico.

Como conclusión final podemos considerar a este documento como una guía importante para crear fundamentos importantes para el desarrollo de múltiples proyectos dentro del área de telecomunicaciones, así como también es una guía didáctica para virtualizar clases y conceptos que se dictan en las cátedras afines al tema.

ACRONIMOS

NS-2 (Network Simulator 2)

DVD (Digital Video Disk)

BIOS (*basic input/output system*)

GB (Giga Bytes)

RAM (random-access memory)

ADSL (Asymmetric Digital Subscriber Line)

DSL (Digital Subscriber Line)

IEEE (Institute of Electrical and Electronics Engineers)

LAN (local area network)

WLAN (wireless local area network)

MAU (Unidades de acceso multiestacion)

RPM (Red Hat Package Manager),

NS (Network simulator)

TCP (Transmission Control Protocol)

TCL (Tool Command Lenguage),

Tk (Tool Kit)

FQ (Fair Queuing)

SFQ (Stochastic Fair Queing)

DRRS (Deficit round Robin Schedule)

RED (Ramdon Early-detection)

CBQ (Class Based Queuing)

UDP (User Datagram Protocol)

BIBLIOGRAFÍA

REFERENCIAS BIBLIOGRÁFICAS

1. D. Cavin, Y. Sasson, A. Schiper. "*On the Accuracy of MANET simulators*". ACM *Workshop on Principles of Mobile Computing .* ACM Press. USA. 2002.
2. D. Vergara, R. Yañez. Simulacion de Protocolos y Algoritmos de Redes usando NS2. Memoria Ing. Civ. Electrónica. España. 2004
3. FALL Kevin. The ns Manual. USA. 2010.
4. FEDORA Documentation Project. Fedora 15 installation guide. Copyright © 2011 Red Hat, Inc. and others.
5. MATTEW S. GAST, 802.11 Wireless Networks: The Definitive Guide. USA. O´Reilly Networking, 2006
6. TRIVIÑO-CABRERA, E. CASILARI-PÉREZ, A. ARIZA-QUINTANA. Implementacion de protocolos en el network simulator (NS-2). España.
7. TURREGANO Javier. Simulador NS-2. España. SETAS. 2006. 6

REFERENCIAS ELECTRÓNICAS

1. ISI. (2009). "The Network Simulator". [En línea]. Disponible en: http://www.isi.edu/nsnam/ns/. [Accesado el día 15 de septiembre del 2011].
2. ISI. (2009). "The VINT Project". [En línea]. Disponible en: http://www.isi.edu/nsnam/ns/. [Accesado el día 23 de septiembre del 2011].
3. Red Hat, Inc. (2011). "Fedora Project". [En línea]. Disponible en: http://fedoraproject.org/. [Accesado el día 10 de Agosto del 2011].