ANEXOS

ANEXO A: Definiciones estadísticas.

ANEXO B: Simulador NS-2.

ANEXO C: Calidad de servicio en redes IP.

ANEXO D: Métodos de estimación del parámetro de Autosimilitud.

ANEXO A.

DEFINICIONES ESTADISTICAS

Este capitulo describe brevemente algunas definiciones estadísticas necesarias en el estudio (análisis y modelamiento) del tráfico en las redes de comunicaciones.

Variable aleatoria

Una variable aleatoria es una función que acota un conjunto de eventos, o resultados de un experimento, sobre un conjunto de valores. Por ejemplo, si nosotros consideramos el experimento de lanzar una moneda, una variable aleatoria podría ser el número de veces que la moneda muestra la cara después de diez lanzamientos. La variable aleatoria en este experimento puede solamente asumir un número finito de valores $0, 1, \ldots, 10$, y es también llamada una *variable aleatoria discreta*. Así mismo si nosotros observamos la tasa de fallas de los componentes de una maquina, una variable aleatoria podría ser el tiempo de vida de un componente en particular. La variable aleatoria en este experimento puede asumir infinitos valores reales, y es también llamada una *variable aleatoria continua* [1].

Una variable aleatoria, no es más que un número real asociado al resultado de un experimento aleatorio, Se trata, por tanto, de una función real con dominio en la σ-algebra del espacio probabilístico, [2].

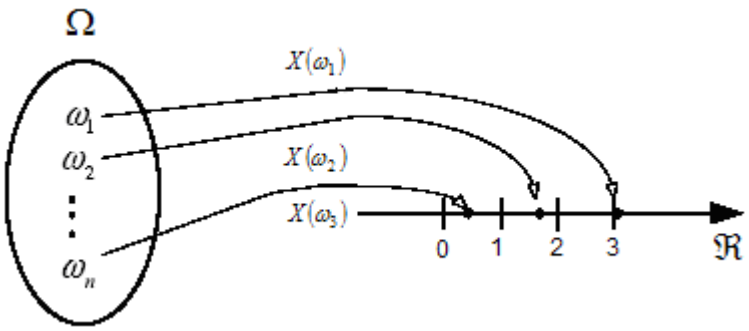

Figura A.1 Variable aleatoria

Proceso estocástico

Un *proceso estocástico*, X_t, es una secuencia de variables aleatorias en el tiempo.

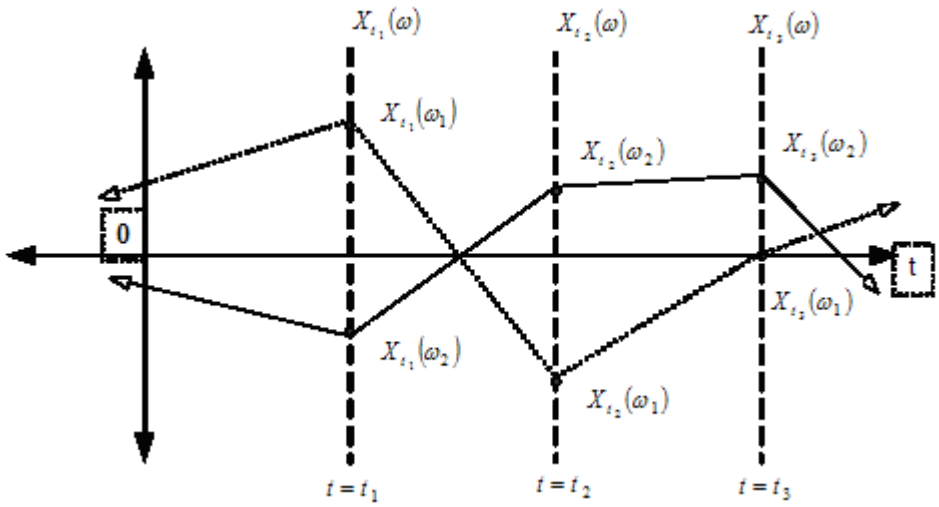

Figura A.2 Proceso estocástico

Note que para un tiempo fijo t, $X_t(\varpi)$ es una variable aleatoria y para toda muestra ϖ, $X_t(\varpi)$ es una funcion de t (o una señal) llamada una realización de ϖ. Para todo ϖ las funciones $X_t(\varpi)$ de t son llamadas funciones de conjunto [2].

Serie temporal Una serie temporal es una simple realización de un proceso estocástico o determinístico con un ϖ fijo. En general, una serie temporal es denotada por

$$\{X(t) : t \in T\}, \tag{A.1}$$

donde $X(t)$ es la observación en el instante t, y T es el conjunto de tiempos en los cuales fueron hechas las observaciones. Si el tiempo entre observaciones es igual entonces la notación se simplifica a

$$\{X_t : t \in Z\} \tag{A.2}$$

si un modelo matemático puede describir exactamente una serie temporal este es llamado un *modelo determinístico*, en muchos casos la serie temporal es *estocástica* por que los valores futuros solo pueden ser parcialmente obtenidos por sus valores históricos [2].

Densidad de probabilidad y Función de distribución acumulativa

La *función de densidad de probabilidad* (pdf) para una variable aleatoria provee una descripción completa de las características probabilísticas de la variable. Si X es una variable aleatoria discreta, entonces su función de densidad $f_x(x)$ esta definida como

$$f_x(x) = P(X = x) \tag{A.3}$$

En palabras, la densidad da la probabilidad de que X asuma un valor finito particular x. Por que de su definición, $f_x(x)$ es algunas veces referido a la *función de frecuencia* para una variable aleatoria discreta. Para $f_x(x)$ ser valida, esta debe ser no negativa y la suma de todas las posibles probabilidades debe ser igual a uno.

$$\sum_{i=1}^{n} f_x(x_i) = 1 \tag{A.4}$$

donde X puede asumir los valores $x_1, x_2, \ldots, x_n$.

Para una variable aleatoria continua Y, la densidad $f_Y(y)$ es usada para encontrar la probabilidad que Y asuma un rango de valores $a < Y < b$:

$$P(a < Y < b) = \int_a^b f_Y(y) \tag{A.5}$$

Dado que una variable aleatoria puede asumir infinitos valores reales, la probabilidad que Y sea igual a cualquier valor único es cero.

$$P(Y = a) = \int_a^a f_Y(y) = 0 \tag{A.6}$$

Así como las variables discretas, la probabilidad para todos los posibles valores de una variable continua deben ser no negativa y la suma igual a uno.

$$\int_{-\infty}^{\infty} f_Y(y)\, dy = 1 \tag{A.7}$$

Es algunas veces conveniente considerar la *funcion de distribución acumulativa* (cdf), la cual también describe las características probabilísticas de una variable aleatoria. Para una variable aleatoria discreta X, la distribución $F_x(x)$ es la probabilidad que X sea menor a algún valor x. La distribución acumulativa es hallada sumando las probabilidades para todo valor real menor o igual a x:

$$F_X(x) = P(X \leq x) = \sum_{t \leq x} f_X(t) \tag{A.8}$$

Si Y es una variable aleatoria continua, la funcion de distribución acumulativa $F_Y(y)$ toma la siguiente forma:

$$F_Y(y) = P(Y \leq y) = \int_{-\infty}^{y} f_Y(y) \tag{A.9}$$

Esta ecuación ilustra una relación entre la densidad y la función de distribución para una variable aleatoria. Si una función es conocida, la otra puede ser fácilmente calculada. Por que de su relación, los términos *distribución* y *densidad* son a menudo intercambiables cuando describen las características globales de probabilidad de una variable aleatoria [1].

Media: La media o valor esperado de una variable aleatoria describe el centro de la funcion de densidad de la variable. Si X es una variable aleatoria discreta y asume los valores $x_1, x_2, \ldots, x_n$ con probabilidades $f_x(x_1), f_x(x_2), \ldots, f_x(x_n)$, entonces la media μ_x esta dada por la media de la suma

$$\mu_x = \sum_{i=0}^{n} x_i f_x(x_i) \tag{A.10}$$

Si Y es una variable aleatoria continua con función de densidad de probabilidad $f_Y(y)$, la media μ_Y esta dada por [1]

$$\mu_Y = \int_{-\infty}^{\infty} y f_Y(y) dy \tag{A.11}$$

Función de Auto-covarianza El segundo momento de un proceso estocástico es conocido como la auto-covarianza del proceso y es calculada por

$$\gamma(t_1, t_2) = Cov(X_{t_1}, X_{t_2}) = E\{[X_{t_1} - \mu(t_1)][X_{t_2} - \mu(t_2)]\} \tag{A.12}$$

La auto-covarianza mide la similaridad entre dos puntos, en la misma serie observada, en diferentes tiempos [2].

Varianza y desviación: La varianza y la desviación estándar de una variable aleatoria son medidas de la dispersión. La varianza es el valor promedio del cuadrado de la desviación de la media de la variable, y la desviación estándar es la raíz cuadrada de la varianza. Si X es una variable aleatoria discreta con funcion de densidad $f_x(x)$ y media μ_x, la varianza $\sigma^2{}_x$ esta dada por la expresión

$$\sigma_x^2 = \sum (x_i - \mu_x)^2 \, f_x(x_i). \tag{A.13}$$

La desviación estándar de X, σ_x, provee un indicador de cómo están dispersos los valores $x_1, x_2, \ldots, x_n$ alrededor de μ_x. En la práctica, algunas veces es deseable calcular la *desviación media absoluta* de una variable aleatoria en lugar de su varianza. Para una variable discreta X, la desviación media es $\sum_i | x_i - \mu_x | \, f_x(x_i)$.

Así mismo, si Y es una variable aleatoria continua con función de densidad $f_Y(y)$ y media μ_Y, la varianza $\sigma^2{}_Y$ esta definida por

$$\sigma^2{}_Y = \int_{-\infty}^{\infty} (y - \mu_Y)^2 \, f_Y(y) dy \,. \tag{A.14}$$

La desviación estándar de Y es σ_Y, y la desviación media absoluta es [1]

$$\int_{-\infty}^{\infty} | \, y - \mu_Y \, | \, f_Y(y) dy \,. \tag{A.15}$$

Función de auto-correlación La función de auto-correlación normalizada (ACF), ρ, de una serie de datos, $X_t = \{X_1, X_2, \ldots\}$ es calculada como sigue

$$\rho(t_1, t_2) = Cor(X_{t_1}, X_{t_2}) = \frac{Cov(X_{t_1}, X_{t_2})}{\sqrt{Var(X_{t_1}) Var(X_{t_2})}} = \frac{\gamma(t_1, t_2)}{\sigma(t_1)\sigma(t_2)} \,. \tag{A.16}$$

La auto-correlación es una medida de la predecibilidad lineal de la serie en el tiempo t_1 usando solamente el valor X_{t_2} [2].

Momentos: El *rth* momento de una variable aleatoria X esta definido como el valor esperado de la cantidad X^r. En la práctica, los *momentos centrales* son a menudo usados en lugar de los momentos ordinarios. Si una variable aleatoria X tiene media μ_x, el *rth* momento central esta definido como el valor esperado de la cantidad $(X - \mu_x)^r$. El primer momento es similar a la desviación media absoluta, y el segundo, momento central es la varianza de una distribución. El tercer momento central es llamado el *skewness*, y es una medida de la asimetría en una función densidad de probabilidad. El cuarto momento central es llamado el *kurtosis*, y es una medida de *peakedness* en una funcion de densidad.

REFERENCIAS

[1] S-Plus 6 for Windows Guide to Statistics, Volume 1, Insighful Corporation. Seattle, Washington. July 2001.

[2] Selwin Jakobus Emiel Roon, “Resource dimensioning in a mixed traffic environment”. Tesis, Faculty of Engineering, University of Pretora, September 2004. Documento en pdf.

Anexo B

Simulador NS-2 (Network Simulator-Versión 2)

NS es un simulador orientado a objetos que sirve para simular eventos discretos, desarrollado para ayudar en la investigación de redes telemáticas. Con *NS* se pueden simular una gran variedad de protocolos de la capa de aplicación (Ftp, Http, Cbr, etc.), Transporte (TCP. UDP, RTP, SRM, SCTP), enrutamiento tanto para enlaces multicast como unicast, también diferentes topologías de red: cableadas, inalámbricas satelitales o combinación de estas (cableadas e inalámbricas, cableadas y satelitales), dentro de una simulación se pueden variar gran cantidad de parámetros (tamaño de los paquetes, tiempo de inicio de la simulación, envió de tráfico de manera aleatoria, ancho de banda, retardos, etc.). Dentro de *NS* se tienen módulos para diferenciación de tráfico (DIffserv), gestión de colas en los nodos que reciben el tráfico, simulación de pérdidas dentro de un enlace etc. El simulador también contempla mecanismos correspondientes a la capa de enlace de datos, como el protocolo MAC (Control de acceso al medio) del tipo CSMA/CD.

NS comienza en 1989 como una variante del Simulator REAL Network Simulator, en los últimos años su evolución es notoria, su creación y constante evolución es un esfuerzo conjunto de diversas personas de la UC Berkeley, USC/ISI, LBL y Xerox PARC, también recibe apoyo de DARPA (Defense Advanced Research Projects Agency).

El lenguaje base de este simulador es *C++*, cuya interfaz o cara, con la cual interactúa más el usuario, esta escrita en *OTcl*, a la jerarquía de clases escrita en *C++* se le llama comúnmente jerarquía compilada y a la jerarquía de clases escrita en *OTcl* se le denomina jerarquía interpretada, estas dos jerarquías están muy ligadas una con la otra, desde la perspectiva del usuario existe una correspondencia uno a uno entre la clase de jerarquía compilada y la clase de jerarquía interpretada. La raíz de esta jerarquía es la clase *TclObjet*. Cuando se crea un nuevo objeto simulador dentro del intérprete, este objeto es instanciado dentro del intérprete y se crea una estrecha relación con otro objeto similar en la jerarquía compilada.

Los dos lenguajes dentro de *NS* ayudan a que este simulador sea eficiente en el manejo de grandes volúmenes de datos (bytes, paquetes, cabeceras), de los algoritmos que operan sobre estos datos, del tiempo de respuesta (arranque de la situación, recopilación etc.), *C++* es el encargado de está labor por su rápido arranque. Por otro lado en lo que respecta a la variación de parámetros, configuraciones y exploración de escenarios, se hace por medio de OTcl ya que es ideal para la creación de los scripts [1].

NS trabaja conjuntamente con otras herramientas gráficas tales como *NAM*, *GNUplot*, *Xgraph*, también se puede incluir dentro de los scripts código escrito en otros lenguajes como *Perl* y *Awk*.

Por ejemplo en el siguiente script se hace uso de varios de estos lenguajes.
En general, la mayoría de simulaciones que se van ha realizar en *NS* inician con el comando

set ns [new Simulator]

Por medio de este se crea una instancia, la cual nos permite crear un objeto simulador.

```
set tl_f [open data1.tr w]
set traza [open traza1.nam w]
$ns namtrace-all $traza
```

Con los anteriores comandos se crean dos clases de archivos con el fin de guardar en ellos todos los eventos que se dan durante la simulación, estos son abiertos para escritura, aquí también se determina la traza de eventos de tipo *nam* en *ns*.

```
$ns color 0 red
$ns color 1 blue
set n0 [$ns node]
$n0 color "red"
$n0 label "Fuente"
$n0 label-color "black"
set n1 [$ns node]
$n1 color "blue"
$n1 label "Destino"
$n1 label-color "black"
```

El bloque de código de arriba crea los nodos, cada uno de los cuales posee un color especifico, luego se les coloca una etiqueta para su respectiva diferenciación. Para que los nodos no queden aislados se crea un enlace o interconexión de la siguiente manera:

```
$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link-op $n0 $n1 color "green"
$ns duplex-link-op $n0 $n1 label "Pareto"
$ns queue-limit $n0 $n1 2000
```

Se presentan diferentes opciones, aquí se define un enlace full duplex entre el nodo cero y el nodo uno, con un ancho de banda de 10Mbps, un retardo de 10ms, una cola del tipo

DropTail (FIFO), posterior a esto se le asigna un color al enlace, una etiqueta de distinción y el número de paquetes que pueden estar en la cola.

Ya se definieron los nodos y el enlace con sus respectivas características, pero esto no tiene mucha importancia si no existe tráfico entre ellos, para poder generar tráfico se debe definir un agente, el cual se encarga de representar los puntos finales en donde los paquetes de nivel de red inician o terminan. En este caso se crea un agente del tipo RTP, el cual es atado al nodo fuente *n0*, nodo en el que se origina el tráfico.

```
set udp0 [new Agent/RTP]
$ns attach-agent $n0 $udp0
```

En NS también se debe definir el tipo de aplicación que se va ha trasportar, y se hace de la siguiente manera:

```
set p0 [new Application/Traffic/Pareto]
$p0 set packetSize_ 210
$p0 set burst_time_ 500ms
$p0 set idle_time_ 1.5s
$p0 set rate_ 84k
$p0 set shape_ 1.1
$p0 attach-agent $udp0
```

Aquí se define que la aplicación responde a la distribución de Pareto, a los parámetros involucrados se les puede realizar una configuración personalizada. Para que la aplicación pueda ser transmitida se debe enlazar al agente creado anteriormente.

Como se menciono previamente, los agentes se crean en los puntos extremos, con las siguientes líneas de código se crea un agente *sink* (de tipo LossMonitor) siendo el encargado de recibir los paquetes enviados, este agente *sink* es enlazado al nodo receptor (*nodo1*).

set sink0 [new Agent/LossMonitor]
$ns attach-agent $n1 $sink0

Teniendo los agentes acoplados a los respectivos nodos, solo falta conectar estos agentes para poder que comience la circulación de paquetes. Esto se hace con el siguiente comando:

$ns connect $udp0 $sink0

Los archivos ejecutables en *NS* admiten una serie de procedimientos, entre estos, se crea uno cuyo objetivo es el del cierre correcto de los archivos abiertos al principio, dentro del código interno de este se puede llamar a herramientas como *nam*, *awk* etc. En el procedimiento presentado a continuación se ejecutan las herramientas gráficas *nam* y *xgraph*.

```
proc finish {} {
        global tl_f
        #cierre de los archivos de salida
        close $tl_f
        #llamado de XGraph y nam para el despliegue de resultados
         exec nam traza1.nam &
        exec xgraph data1.tr -geometry 800x400 &
        exit 0
}
```

En las simulaciones de redes de telecomunicaciones, es de interés poder recolectar información acerca del comportamiento de los paquetes cuando están en una cola. El procedimiento *record* calcula el ancho de banda y lo escribe dentro de un archivo.

proc record {} {

```
    global sink0 tl_f
    #consigue una instancia para el simulador
    set ns [Simulator instance]
    #el procedimiento es llamado cada 0.02 segundos.
    set time 0.02
    #forma en que sink recibe el tráfico (en bytes)
    set bw0 [$sink0 set bytes_]
    #ajuste al tiempo actual..
    set now [$ns now]
    #Calcula el ancho de banda en (Mbps) y lo escribe dentro de un archive.
    puts $tl_f "$now [expr $bw0/$time*8/1000000]"
    #receteo de la variable bytes_ a cero
    $sink0 set bytes_ 0
    #recargo y llamado al procedimiento.
    $ns at [expr $now+$time] "record"
}
```

Por ultimo, dentro del *script* se deben estipular los tiempos de lanzamientos de los procedimientos si los hay, el tiempo en que comienza la trasmisión de paquetes y el tiempo de parada para que pueda ser ejecutado por *ns*, además de que es indispensable arrancar *ns* por medio de *ns run*.

```
$ns at 0.01 "record"
$ns at 0.02 "$p0 start"
$ns at 300.0 "finish"
$ns run
```

Nam y *xgraph* son herramientas muy utilizadas dentro de *ns,* motivo para realizar una breve explicación de las mismas.

NAM (Animador de Red)

Nam es una herramienta de animación basada en *Tcl/TK* para la visualización de las trazas y paquetes de datos de tráfico real. *Nam* esta habilitado para animar[1] en conjunto un gran número de datos leídos, se puede extender para visualizar un gran número de redes en diferentes situaciones. Bajo este contexto *nam* esta diseñado para la lectura de eventos y realizar una animación simple para una traza de archivos grande. *Nam* realiza el manejo de grandes cantidades de datos guardando una mínima cantidad de estos en memoria, los eventos son guardados dentro de un archivo y son releídos desde este archivo cuando sea necesario [2].

El primer paso para el uso de *nam* es la creación de un archivo de trazas. Este archivo contiene la información topológica de la red a simular (nodos, enlaces, trazas de paquetes, etc.), usualmente el archivo de trazas es generado por *NS* durante una simulación. Cuando el archivo de trazas es generado, este es leído y animado por *nam*. Al inicio *nam* puede leer el archivo de trazas, crear la topología, y colocarla en una ventana que es desplegada, en donde se presenta un esquema inicial de la red, también se realiza un ajuste del tiempo de simulación a cero. A través de esta interfaz de usuario *nam* proporciona control sobre muchos aspectos de la simulación. En la figura b.1 se puede observar una ventana de *nam*, la cual tiene una topología de 2 nodos que intercambian información.

[1] Dotar de movimiento cosas inanimadas.

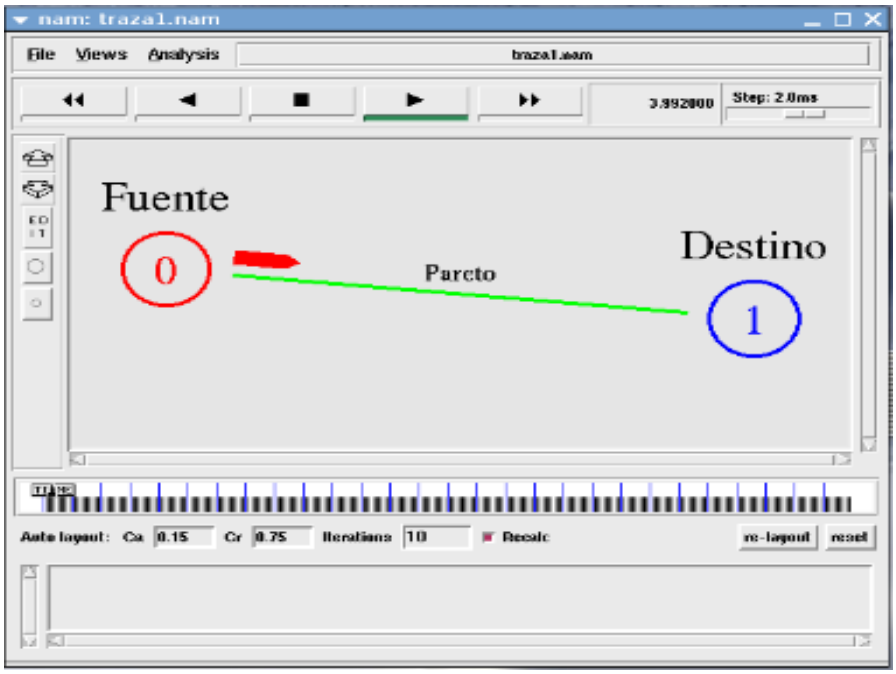

Figura b.1 Nodos intercambiando Tráfico de Pareto.

Trazas en Nam: en *nam* se pueden hacer muchos cambios a los diversos componentes que se tienen dentro de una determinada topología de red, se le puede cambiar el color a los nodos, a los enlaces, cambiar de color a los paquetes que circulan por la red, se puede ver el sistema de encolamiento etc. De todas estas alternativas, la más relevante dentro de una simulación es la información del comportamiento de los paquetes internamente en la topología, desde el nodo fuente hasta el nodo destino. Todo lo anterior en una simulación de *NS* queda consignado en un archivo con extensión *nam*. Este archivo contiene una serie de líneas de las cuales la de mayor importancia es la que contiene las trazas de los paquetes. Esta tiene el siguiente formato:

<type> -t <time> - e <extent> -s <source id> -d <destination id> -c <conv> -i <id>

Donde *<type>* es una de las siguientes opciones:

h salto: al inicio de la transmisión de un paquete desde un nodo fuente a un nodo destino este puede ser enviado al próximo salto.

r recepción : el paquete finaliza la transmisión y puede ser recibido por el destino.

d descarte: el paquete es descartado en la cola de un enlace en particular.

+ entrada en cola: el paquete entra en cola.

- salida de la cola: el paquete sale desde el lado izquierdo de un nodo fuente a un nodo destino.

Las otras banderas significan lo siguiente:

-t <time> tiempo en que ocurre el evento.

-s <source id> identificador del nodo origen.

-d <destination id> identificador del nodo destino.

-p <pkt-tipe> se puede ver el nombre descriptivo del tipo de paquete.

-e <extent> tamaño del paquete en bytes.

-c <conv> conversión del *id* a flujo *id* de la sesión.

-i <id> identificador del paquete en la comunicación.

-a <attr> atributo del paquete transportado. Puede ser usado como identificador de dirección.

-x <src-na-pa> <dst-na-pa> <seq> <flags> <sname> toma las trazas de *NS* y da información del nodo fuente, nodo destino, dirección de puertos, número de secuencia, banderas y el tipo de mensaje.

-n <sequence number> número de secuencia.

En la figura b.2 se muestra un ejemplo de una traza generada entre el nodo fuente y el nodo destino presentados en la figura b.1.

```
V -t * -v 1.0a5 -a 0
A -t * -n 1 -p 0 -o 0xffffffff -c 31 -a 1
A -t * -h 1 -m 2147483647 -s 0
n -t * -a 0 -s 0 -S UP -v circle -c black -i black
n -t * -a 1 -s 1 -S UP -v circle -c black -i black
l -t * -s 0 -d 1 -S UP -r 1000000 -D 0.01 -c black
+ -t 0.272559173608639 -s 0 -d 1 -p pareto -e 210 -c 0 -i 0 -a 0 -x {0.0 1.0 1 ------- null}
- -t 0.272559173608639 -s 0 -d 1 -p pareto -e 210 -c 0 -i 0 -a 0 -x {0.0 1.0 1 ------- null}
h -t 0.272559173608639 -s 0 -d 1 -p pareto -e 210 -c 0 -i 0 -a 0 -x {0.0 1.0 -1 ------- null
r -t 0.284239173608639 -s 0 -d 1 -p pareto -e 210 -c 0 -i 0 -a 0 -x {0.0 1.0 1 ------- null}
+ -t 0.292559173608639 -s 0 -d 1 -p pareto -e 210 -c 0 -i 1 -a 0 -x {0.0 1.0 2 ------- null}
- -t 0.292559173608639 -s 0 -d 1 -p pareto -e 210 -c 0 -i 1 -a 0 -x {0.0 1.0 2 ------- null}
h -t 0.292559173608639 -s 0 -d 1 -p pareto -e 210 -c 0 -i 1 -a 0 -x {0.0 1.0 -1 ------- null
r -t 0.30423917360864 -s 0 -d 1 -p pareto -e 210 -c 0 -i 1 -a 0 -x {0.0 1.0 2 ------- null}
+ -t 0.312559173608639 -s 0 -d 1 -p pareto -e 210 -c 0 -i 2 -a 0 -x {0.0 1.0 3 ------- null}
- -t 0.312559173608639 -s 0 -d 1 -p pareto -e 210 -c 0 -i 2 -a 0 -x {0.0 1.0 3 ------- null}
h -t 0.312559173608639 -s 0 -d 1 -p pareto -e 210 -c 0 -i 2 -a 0 -x {0.0 1.0 -1 ------- null
r -t 0.32423917360864 -s 0 -d 1 -p pareto -e 210 -c 0 -i 2 -a 0 -x {0.0 1.0 3 ------- null}
+ -t 0.33255917360864 -s 0 -d 1 -p pareto -e 210 -c 0 -i 3 -a 0 -x {0.0 1.0 4 ------- null}
- -t 0.33255917360864 -s 0 -d 1 -p pareto -e 210 -c 0 -i 3 -a 0 -x {0.0 1.0 4 ------- null}
h -t 0.33255917360864 -s 0 -d 1 -p pareto -e 210 -c 0 -i 3 -a 0 -x {0.0 1.0 -1 ------- null}
r -t 0.34423917360864 -s 0 -d 1 -p pareto -e 210 -c 0 -i 3 -a 0 -x {0.0 1.0 4 ------- null}
+ -t 0.35255917360864 -s 0 -d 1 -p pareto -e 210 -c 0 -i 4 -a 0 -x {0.0 1.0 5 ------- null}
```

Figura b.2 Traza generada por dos nodos.

En la figura anterior las primeras líneas dan información sobre la versión de la herramienta *Nam*, en donde el script presenta una versión de *Nam 1.0 a 5 -a 0*, también se ofrece información sobre la jerarquía de direccionamiento *(A –t * -n 1 –p 0 –o 0xffffffff -c 31 –a 1)*, seguidamente se tiene todo lo relacionado con el estado del nodo, la dirección del nodo fuente es cero *(-a 0)*, el identificador del nodo es cero *(-s 0)*, el estado del nodo es activo *(-S UP)*, la forma del nodo es circular *(-v circle)*, el color del nodo es negro *(-c black)*, el color de la etiqueta es negro *(-i black)*.

La información sobre el estado del enlace (la letra *l* al comienzo), dice que esta arriba todo el tiempo, la comunicación esta echa entre el nodo cero y el nodo uno, el ancho de banda es de *10Mbps*, el retardo es de *10ms*, por último se observa el comportamiento del tráfico que circula por la red. Por ejemplo:

r -t 0.34423917360864 -s 0 -d 1 -p pareto -e 210 -c 0 -i 3 -a 0 -x {0.0 1.0 4 ------- null}

Nos dice que el paquete fue recibido por el nodo uno en el tiempo *0.34423917360864*, el paquete es de tipo pareto, su tamaño es de 210 bytes, pertenece al flujo número 3 *(-i 3)* etc. [1].

Xgraph.

Xgraph es una herramienta que permite graficar datos (x-y) de propósito general, la cual admite una ampliación de la figura representada o de una porción de la misma. Se pueden graficar datos de diversos archivos dentro de la misma gráfica, se puede hacer el manejo del tamaño de los datos y del número de datos que contienen estos archivos.

Con *Xgraph* se puede exportar e importar datos hacia o desde diferentes tipos de formatos (wysiwyg PostScript, PDF, procesadores de texto etc.) para su respectivo análisis. También se pueden hacer diversas gráficas en una figura, estas están diferenciadas con colores para su mejor compresión. En la figura b.3 se presenta un ejemplo de una figura graficada con *xgraph*, en donde se muestra el tráfico de tres fuentes de pareto con su respectivo color. Más información sobre *xgraph* se puede conseguir en [3].

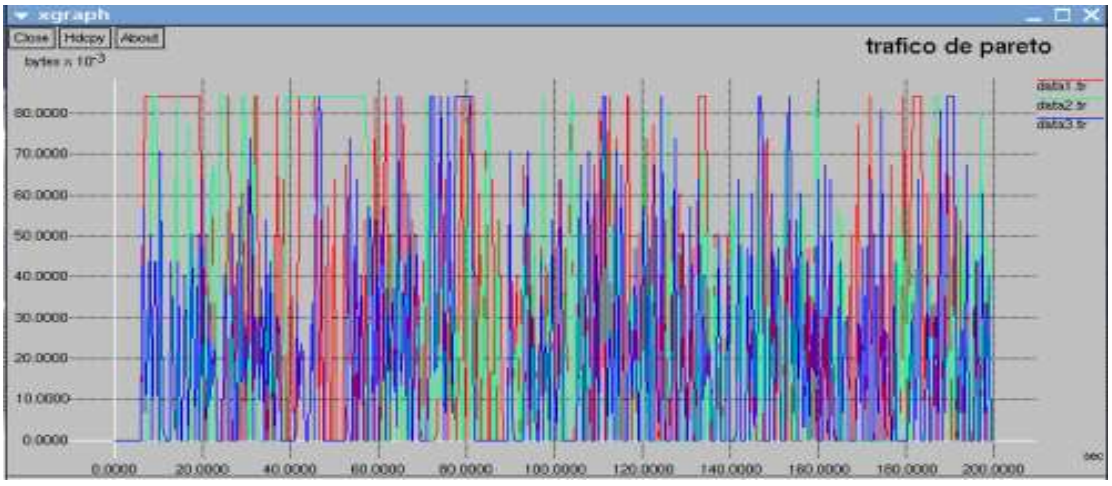

Figura b.3 Tráfico generado por tres fuentes nodos.

Teniendo en cuenta lo poderoso que resulta ser *NS-2*. Se realizó una simulación de una red NGN. Su generación responde a la distribución de Pareto. La cual se tomo como base para la caracterización del tráfico voz IP en redes reales.

Simulación de Tráfico de Voz con NS-2

En el capitulo 2 se hizo mención de que el comportamiento del tráfico en las redes de paquetes (Ethernet, WAN) [4] [5] [6] [7] es auto-similar o de dependencia de largo rango, también se menciono que la estructura de autosimilitud puede ser reducida a un modelo de simple fuentes ON/OFF (trenes de paquetes). Tanto los periodos ON como los OFF poseen características que los diferencian, respondiendo a distribuciones de cola pesada y de varianza infinita [7]. Los modelos tradicionales son caracterizados por medio de la distribución exponencial pero el modelo ON/OFF no se ajusta a este modelo, por este motivo se necesitan modelos que simulen colas pesadas. La distribución de Pareto es ampliamente utilizada con este fin.

En NS-2 se puede realizar generación de tráfico que se ajuste a la distribución de Pareto. POO_TRAFFIC genera tráfico de acuerdo a la distribución de Pareto ON/OFF, en donde los periodos ON y OFF responden a esta distribución [1]. En el periodo ON los paquetes son transmitidos a una velocidad constante, mientras que en los periodos OFF no se realiza transmisión de paquetes. Estas fuentes son utilizadas para la generación de tráfico agregado que exhibe LRD (Dependencia de Largo Rango). El tráfico agregado de VoIP se puede simular ajustando diversos parámetros dentro del simulador, esto con el fin de comprobar si hay presencia del fenómeno de autosimilitud en estas redes. Internamente en *NS-2*, pareto ON/OFF esta contenido dentro de la clase *Otcl* (Application/Traffic/Pareto), los parámetros que se pueden configurar son los siguientes:

*packetSise_* para la generación de paquetes de tamaño constante.
*burst_ time_* tiempo promedio que utiliza el generador para el periodo “on”.
*idle_time_* tiempo promedio que utiliza el generador para el periodo “off”.
*rate_* velocidad de envió durante el tiempo “on”.
*shape_* parámetro de forma para la distribución de pareto.

El tráfico de pareto en *NS-2* maneja internamente una implementación de la distribución de pareto que en general es como sigue:

Cálculos iniciales:
interval = packetSize_ * 8 / rate_
burstlen = burst_time_ / interval (en paquetes para un packetSize_ dado)
burstlen determina el número de paquetes que podría ser generado durante el periodo ON.

Durante cada ciclo ON/OFF, dos variables aleatorias (independientes) de Pareto son calculadas:
next_burstlen: paquetes que pueden ser transmitidos en el próximo periodo de ráfaga (periodo ON).
next_idle_time es la longitud en segundos del periodo desocupado (periodo ON).

Algoritmo para Generación de Tráfico de Pareto:

Una breve visión de los parámetros y el funcionamiento de este algoritmo se dan enseguida.

Paso 1: Se realiza el cálculo *de next_burstlen* el cual se hace con los valores de la longitud promedio del periodo ON (*burst_ time_*) y el parámetro de forma de la distribución de pareto (*shape_*).

Paso 2: Envió de todos los paquetes *next_burstlen* calculados.

Paso 3: Se realiza el cálculo de *next_idle_time* el cual se hace con los valores de la longitud promedio del periodo OFF (*idle_time_*) y el parámetro de forma de la distribución de pareto (*shape_).*

Paso 4: Esperar a que pase el tiempo *next_idle_time* y volver al paso 1.

El generador de tráfico de Pareto hace uso de la distribución de pareto (2.5.1) de la siguiente manera:

Recordando que α es el parámetro de forma, κ es el parámetro de escala y que la media viene dada por:

$$E(X) = \bar{X} = \begin{cases} \dfrac{\alpha k}{\alpha - 1} & \alpha > 1 \\ \infty, & \alpha \leq 1 \end{cases} \tag{B.1}$$

Se tiene:

$$burstlen \;\; = E(X) = {\kappa_1 * \alpha}\big/{(\alpha - 1)} \tag{B.2}$$

$$idle\_time\_ \; = E(Y) = {\kappa_2 * \alpha}\big/{(\alpha - 1)} \tag{B.3}$$

Por consiguiente:

$$\kappa_1 = burstlen * {\alpha - 1}\big/{\alpha} \tag{B.4}$$

$$\kappa_2 = idle\_time\_ * {\alpha - 1}\big/{\alpha} \tag{B.5}$$

NS-2 tiene un generador de números aleatorios que están de acuerdo con la ley de Pareto, los cuales reciben un parámetro de escala y un parámetro de forma, por eso se tiene:

double pareto (double scale, double shape)

Donde el generador de Pareto necesita calcular *next_burstlen* esto es:

int *next_burstlen* = int (pareto (b1, a) + 0.5);
/* *next_burstlen* al menos podría tener un paquete */
If (next_burstlen == 0) *next_burstlen* = 1;

Cuando el generador de Pareto necesita calcular *next_idle_time*, lo hace de la siguiente forma:

double *next_idle_time* **=** pareto $(\kappa_{2,}\alpha)$

Teniendo presente que en las redes circulan diferentes clases de tráfico, en el simulador se realizaron diferentes acercamientos a las redes reales, en uno de estos se le integro el modulo Diffserv disponible en *NS-2* para la gestión de tráfico.
La arquitectura Diffserv dentro de *ns* esta compuesta por cinco módulos: uno para la funcionalidad básica de enrutamiento Diffserv, uno para cada una de los enrutadores de frontera y de núcleo, uno basado en el encolamiento de RED y otro para las políticas (monitoreo) [8].

Modulo dsRED: aquí se realiza la definición de la clase *dsREDQueue,* de la cual se derivan las clases *edgeQueue* y *coreQueue*, en esta clase se implementan todos los parámetros que son comunes para los enrutadores (routers) *Diffserv* de frontera y de núcleo. La estructura de encolamiento esta compuesta por 4 colas físicas, cada una de las cuales contiene 3 colas virtuales de la clase RED, con niveles de precedencia para cada una. Cada cola física corresponde a una clase de tráfico, a la combinación de una cola y el número de precedencia se le asigna a un codepoint (preferencia de descarte). El codepoint especifica cierto nivel de servicio. Se puede dar el caso de que no se utilicen todas las colas físicas y virtuales, por medio de algunos comandos dentro de los script de simulación se pueden ajustar estos parámetros. El mapeo de un codepoint

correspondiente a una cola particular y nivel de precedencia, se hace por medio de la tabla PHB que se encuentra inmersa dentro de la clase *dsREDQueue*. Esta tabla posee tres campos los cuales son: *Codepoint*, *Class* (Cola física). *Precedente* (Cola virtual).

Modulo para el enrutador de Frontera (Edge): se utiliza este modulo para la implementación de *Diffserv* en los enrutadores de borde. Está definido en la clase *edgeQueue* que simula todas las funcionalidades del enrutador de borde o frontera. Entre sus funcionalidades se encuentran: mantenimiento de las colas físicas y virtuales de acuerdo a sus parámetros, marcado de paquetes con *codepoints*, monitoreo del tráfico agregado. La clase *Edge* posee una instancia de la clase *Policy* que maneja toda la creación y ejecución de políticas.

Modulo para el enrutador de Núcleo (Core): emula un enrutador de núcleo dentro de la arquitectura *Diffserv*. Trabaja llevando el flujo que le mandan los enrutadores de borde. Los paquetes son enviados de acuerdo a la marcación echa en los enrutadores de borde. Este envío se hace de acuerdo a los *codepoint*, por medio de los enrutadores de núcleo se puede dar prioridades al tráfico que circula por ellos.

Modulo basado en el encolamiento RED: la clase *redQueue* como una cola física esta compuesta de nodos virtuales. En esta clase se define una cola física que contiene tres colas virtuales, en la configuración de cada cola virtual perteneciente a una cola física se le pueden colocar diferentes parámetros. El fin principal de esta clase es habilitar la diferenciación por medio de la definición de las colas virtuales.

Modulo para Monitoreo de políticas (policy): la clase *Policy*, maneja la creación, manipulación y la ejecución de políticas en los enrutadores de frontera. Una política

determina el tratamiento que se le debe dar al tráfico agregado en los enrutadores de frontera. Los nodos de frontera utilizan las políticas para marcar con un codepoint los paquetes.

Una política se establece entre un nodo fuente y un nodo destino, todos los flujos correspondientes a un par fuente destino son tratados como un simple tráfico agregado. En cada política se puede definir el tipo de política, velocidad y otros parámetros.

En conclusión el modulo *Diffserv* en *NS-2* permite:

- Implementar múltiples colas físicas RED a lo largo de un simple enlace.
- Simulación de múltiples colas virtuales en cada cola física, independientes parámetros pueden ser configurados para cada cola virtual.
- La habilidad de colocar un paquete en una cola física y virtual determinada, dependiendo de las especificaciones colocados por el usuario.

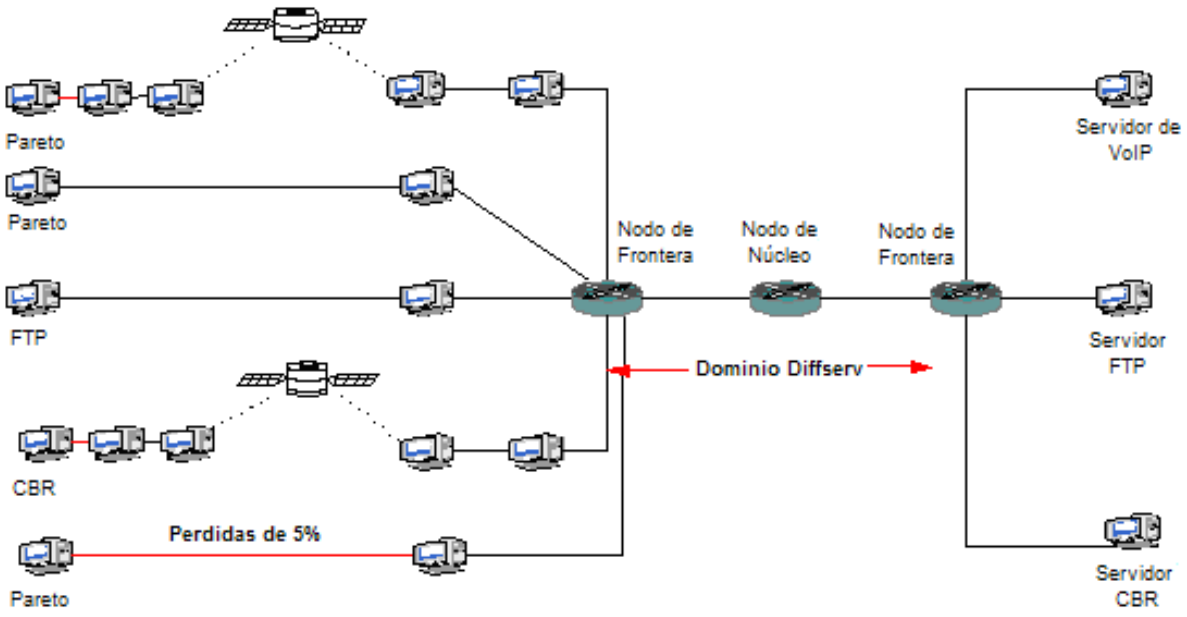

Figura b.4 Red NGN simulada con NS-2.

En la figura b.4 se observa el tipo de topología que se monto dentro de *NS-2* para simular una red NGN por donde circula en tráfico de VoIP. En este gráfico varios tipos de tráfico se encuentran presentes en el núcleo de la red simulada, por medio de una serie de comandos *awk* dentro de la simulación, se obtiene específicamente el tráfico de Pareto, el cual es pasado luego como para parámetro a los estimadores utilizados para calcular el parámetro de Hurst. En todas las simulaciones con *NS-2* el parámetro de forma puede ser variado, esto con el fin de saber coma afecta este parámetro el cálculo numérico de *H*. En la tabla b.1 se obtienen valores de *H* para diferentes valores del parámetro de forma α.

Parámetro α	**No de Muestras**	**Varianza-Tiempo**	**Periodograma**
2	300.000	0.5390	0.5378
1.7	300.000	0.5438	0.5324
1.4	300.000	0.5367	0.5191

Tabla b.1 Parámetro de Hurst calculado a partir de la variación de α **.**

En la tabla b.1 se muestra que al variar el parámetro de forma α también existe una variación en el valor que arrojan los estimadores para el parámetro de autosimilitud *H*. En las figuras b.5 y b.6 se muestran las gráficas arrojadas en Matlab por los estimadores de Varianza-Tiempo y Periodograma respectivamente para α = 1.7.

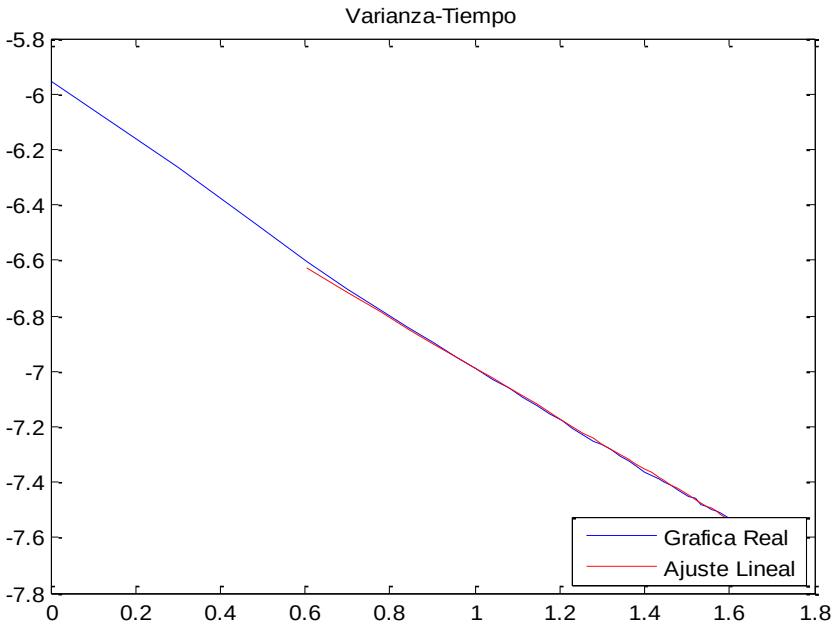

Figura b.5 Gráfica Varianza-Tiempo para α =1.7.

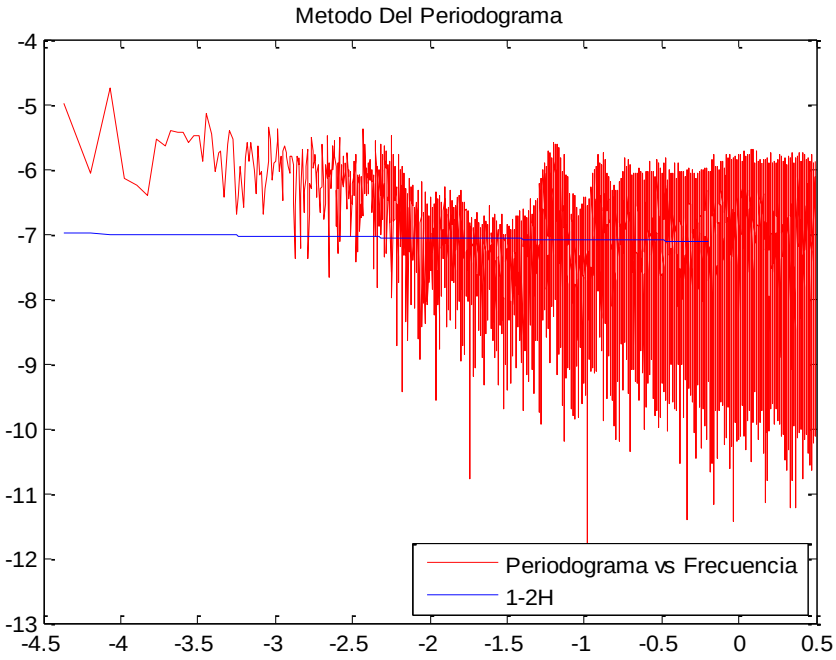

Figura b.6 Gráfica del Periodograma para α =1.7.

La gráfica b.7 muestra el comportamiento del tráfico de voz simulado por medio de *NS-2*, se observa el número de paquetes por segundo recibidos por el servidor de Pareto. Este tráfico es generado por varias fuentes de Pareto.

El tráfico es sacado del núcleo de la red simulada, por donde también circulan paquetes de tráfico Ftp y Cbr.

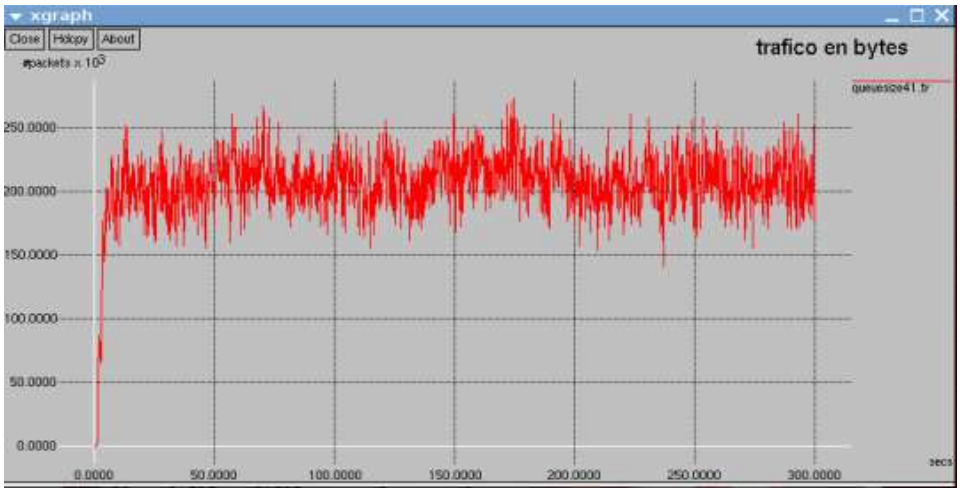

Figura b.7 Gráfica de Número de paquetes Vs Tiempo.

REFERENCIAS

[1] The VINT project is a joint effort by people from UC Berkeley, USC/ISI, LBL, and Xerox PARC. “The ns Manual (formerly ns notes and Documentation)”. Agust 8, 2007. Documento en pdf.

[2] Network Animator (NAM): http://www.isi.edu/nsnam/nam/. Página web.

[3] XGraph: http://jean-luc.ncsa.uiuc.edu/Codes/xgraph. Página web.

[4] Walter Williger, Murad S. Taqqu, Robert Sherman and Daniel V. Wilson. Self-Similarity Through High.Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level. Documento en pdf.

[5] Walter Willinger, Vern Paxson and Murad S. Taqqu. Self-similarity and Heavy Tails: Structural Modeling of Network Traffic. Documento en pdf.

[6] Will E. Leland Bellcore, Murad S. Taqqu Department of Mathematics Boston University, Walter Willinger Bellcore, Daniel V. Wilson Bellcore. “On the Self-Similar Nature of Ethernet Traffic”. 1993.

[7] Kihong Park, Network Systems Lab, Department of Computer Sciences, Purdue University, Walter Willinger, Information Sciences Research Center, AT&T Labs Research, Florham Park. “ SelfSimilar Network Traffic: An Overview”. Documento en pdf.

[8] Peter Pieda. Jeremy Ethridge, Mandeep Baines, Farhan Shallwani. “A Network Simulator Differentiated Services Implementation Open IP”. Nortel Networks. July 26 2000. Documento en pdf.

Anexo C

Calidad de servicio en redes IP

Algunas Técnicas para la provisión de calidad de servicio en redes IP

C.1 Reservación de recursos: la reservación de recursos básicamente se refiere al modelo de servicios integrados (IntServ) el cual apunta a la inclusión de dos clasificaciones para el tráfico: servicio garantizado y servicio de carga controlada [2]. Se propone dentro de esta técnica que los recursos deben ser expresamente gestionados para cumplir con los requerimientos de las aplicaciones, esto implica que la reservación de recursos y el control de admisión son claves para asegurar una calidad de servicio. El servicio Garantizado [3] básicamente requiere que el retardo sea fijo de extremo a extremo, para ello se tienen una serie de elementos que vigilan los diferentes flujos con el fin de asegurar un ancho de banda específico para cada servicio, evitando que se presenten pérdidas en las colas. Aplicaciones que necesitan que el tiempo de llegada de los paquetes sea constante pueden hacer uso de este servicio.

El servicio de carga controlada [4] esta orientado a ofrecer un comportamiento visiblemente aproximando al que se presentaría bajo una red que tiene sus recursos libres, con alta probabilidad de que los paquetes dentro de la red lleguen al otro extremo sin que se presenten perdidas significativas, el retardo experimentado por los paquetes en

transito, en un alto porcentaje puede estar por debajo de los requerimientos mininos para la llegada satisfactoria del paquete. Para satisfacer el servicio de carga controlada se implementan una serie de técnicas entre las que se encuentran algoritmos de sheduling y de control de admisión, permitiendo el aumento de la eficiencia en la reservación de los recursos.

El Tráfico Intserv básicamente es un servicio por flujos, de acuerdo a las solicitudes por parte de los servicios, cada dispositivo dentro de la red debe reservar recursos para un flujo en particular, también se necesita que los enrutadores dentro del modelo IntServ tengan implementado funcionalidades adicionales para el soporte de QoS (Calidad de servicio) [2]. El protocolo para la señalización dentro de Intserv es RSVP (Resource Reservation Protocol), el cual es usado por un nodo para poder solicitar una calidad de servicio especifica a la red para los datos de una aplicación o flujo de datos en particular. RSVP es también usado por los enrutadores para el envió de solicitudes de calidad de servicio a todos los nodos a lo largo del trayecto con el fin de reservar los recursos suficientes. Un flujo es admitido o rechazado dependiendo de los recursos disponibles a lo largo de todo el trayecto de la red, RSVP utiliza un estado “Soft” para gestión de los recursos en los enrutadores y este puede ser refrescado periódicamente por el nodo (host) final.

IntServ es una arquitectura que presenta significativos avances para el mejoramiento de la calidad de servicio, el crecimiento constante del número de aplicaciones y de usuarios que hacen uso del servicio de Internet aumenta el flujo de datos dentro del núcleo de la red, motivo por el cual los mensajes de reservación de recursos aumentarían, haciendo que la carga y el ancho de banda para la transmisión de datos disminuya debido a la gran cantidad de mensajes de señalización entre enrutadores, este es el principal problema

para que la Arquitectura Intserv no sea escalable, se han propuesto otras arquitecturas para mejorar la escalabilidad tales como Diffserv y MPLS.

C.2 Servicios Diferenciados (Diffserv) En esta arquitectura básicamente se hace el acondicionamiento y clasificación del tráfico entrante en diferentes niveles de servicio, en lo posible en el entorno de la red con el fin de reducir la complejidad, seguidamente se aplica un comportamiento agregado para todos los flujos de una determinada clase de servicio, el cual es identificado por un simple codepoint DS. Este codepoint sirve para el envío de los paquetes dentro del núcleo de la red, cumpliendo con el llamado comportamiento por saltos [5].

Un dominio de servicios diferenciados esta compuesto por un grupo de nodos DS los cuales operan con una política de servicios común, Los nodos DS de frontera son los encargados de marcar y acondicionar el tráfico seleccionando un PHB o un grupo de PHB`s soportados dentro del dominio, los nodos dentro del domino DS seleccionan el comportamiento de envío de los paquetes basados en el codepoint DS. El valor que es soportado por cada uno de los PHB es conseguido siguiendo el codepoint recomendado para el mapeo a PHB o mapeo local a la medida del campo (DS field). Dentro del dominio DS todos los nodos deben soportar diferenciación de servicios, de lo contrario, si hay un nodo que no soporte DS lleva consigo a la degradación de la calidad ofrecida.

En [6] se realiza la definición del campo de servicios diferenciados (DS field) para la cabeceras de Ipv4 e Ipv6. En Ipv4 el campo comprometido en este proceso de reemplazo es el octeto TOS y para Ipv6 el campo es el octeto Clase de Tráfico. Seis bits del campo DS son usados como un codepoint (DSCP) para seleccionar el PHB (Per-Hop Behavior)

que decidirá que comportamiento debe ser aplicado al tráfico cuando es encaminado en el núcleo de la red. Dos bits (CU) del campo DS están actualmente sin usar, reservados para uso futuro, el valor de estos dos bits son ignorados por los nodos pertenecientes al dominio de servicios diferenciados cuando determinan el comportamiento entre saltos para el paquete recibido. La estructura del campo DS es la siguiente. Ver figura C.1:

DSCP: codepoint para diferenciación de servicios
CU: Actualmente no usado

Figura C.1: Estructura del Campo DS.

El administrador en un dominio DS es el encargado de asegurar los recursos que son provistos o reservados para cumplir con el SLA (acuerdo de nivel de servicio) celebrado entre el proveedor de servicios de Internet (ISP) y los clientes.

Dos clases de nodos se encuentran presentes en un dominio DS con diferentes tipos de funciones. La primera clase de nodos que se puede distinguir son los nodos de frontera los cuales se encuentran entre un dominio DS y los dominios que no son DS, la principal función de los nodos de frontera es la de ejecutar funciones de acondicionamiento de tráfico, como se define en los TCA (Traffic Conditioning Agreement) entre el dominio DS al que pertenecen y el dominio al cual están conectados.

Los nodos de borde actúan como nodos de entrada y salida para los diferentes tipos de tráfico, en la función de nodo de entrada se debe asegurar que el tráfico entrante se

comporta de acuerdo a un determinado TCA acordado entre los dominios. Como nodo de salida debe realizar funciones de envío de paquetes al dominio directamente conectado, en otras palabras, realiza acondicionamiento de tráfico.

La otra clase de nodos son los llamados nodos Interiores, los cuales realizan funciones de acondicionamiento de tráfico limitado, tal como el remarcado del codepoint DS. En la figura C.2 se puede visualizar un esquema de un dominio Diffserv.

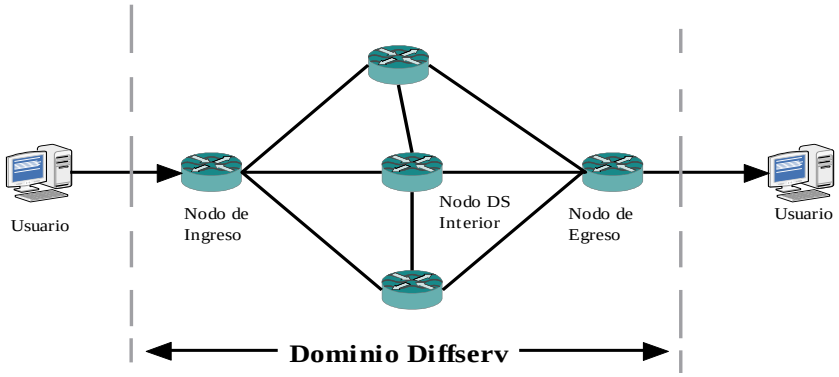

Figura C.2 Dominio Diffserv.

Dentro de los conceptos que se dan internamente en la arquitectura de servicios diferenciados, se encuentra el concepto de Región de Servicios diferenciados, la cual esta compuesta por uno o más dominios DS, además es capaz se soportar diferenciación de servicios a lo largo del trayecto, entre estas se pueden dar diferentes grupos de PHB, la interconexión de los diferentes dominios se puede hacer por medio de SLA, los cuales definen específicamente un TCA para el transito del tráfico, pero también es posible que a lo largo de una Región de servicios diferenciados se pueda adoptar una política de aprovisionamiento de servicio común, se puede soportar un conjunto de PHB común y mapeo de los codepoint comunes, logrando con esto una menor complejidad y evitando la necesidad de acondicionamiento del tráfico entre los dominios DS.

La clasificación y el acondicionamiento de tráfico se refiere a la forma en que se realiza la clasificación y marcado de los paquetes de acuerdo a una serie de reglas establecidas en los SLA`s, los cuales dan perfiles para clases especificas de tráfico, los TCA entre los dominios se derivan de los SLA's. Básicamente en el acondicionamiento del tráfico se hace la vigilancia, marcado y modelado para asegurar que el tráfico entrante en el dominio DS esté de acuerdo con las reglas especificadas en los TCA.

El clasificador de los paquetes escoge una porción de la cabecera de los paquetes para realizar su selección. Se definen dos tipos de clasificadores:

1. Clasificador de Comportamiento Agregado BA (Behavior Aggregate) el cual realiza una clasificación de los paquetes basándose únicamente en codepoints DS.

2. El Multi-Campos MF (Multi-Field) clasifica los paquetes basado en una combinación de los valores de uno a mas campos de la cabecera, tales como dirección fuente, dirección destino, campo DS, identificador del protocolo (protocol ID), número de puerto fuente y número de puerto destino, y otra información tal como la interfaz de entrada.

El clasificador realiza funciones de vigilancia que miden la tasa a la que llegan los paquetes para cada flujo.

Un código que identifica el tipo de servicio es escrito dentro de la cabecera IP por parte de un marcador, esto se hace de acuerdo al SLA contratado, cada PHB sirve como identificador del tipo de servicio determinando un perfil de tráfico soportado por el dominio, los perfiles especifican las propiedades temporales de una corriente de tráfico seleccionada por el clasificador. Diferentes acciones de acondicionamiento pueden ser aplicadas a los paquetes, se puede dar prioridad de descarte de unos paquetes frente a otros en caso de congestión.

Un acondicionador de tráfico puede contener los siguientes elementos: medidor, modelador, marcador, despachador. El medidor es usado para medir un flujo de tráfico de acuerdo con un perfil de tráfico, y de esta manera tomar una decisión sobre cada paquete en particular realizando una acción frente a esto, las acciones pueden ser: marcado, modelado o descarte. En la figura C.3 se muestra el diagrama en bloques de un clasificador y acondicionador de tráfico.

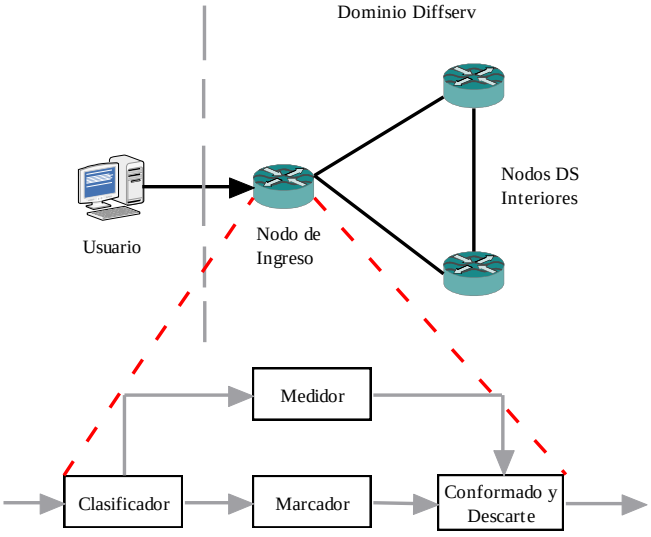

Figura C3: Clasificador y acondicionador de tráfico.

El procedimiento por saltos (PHB Per-Hop Behavior) es una descripción del comportamiento observado externamente para un nodo DS aplicado a un DS agregado con una conducta particular.

La distinción de comportamientos se puede observar cuando múltiples comportamientos agregados compiten por el buffer y ancho de banda en un nodo. El PHB realiza la mediación para la reservación de recursos para los comportamientos agregados, pueden ser medidos bajo una variedad de condiciones de competencia, pudiendo garantizar anchos de banda durante un intervalo de tiempo razonable.

Los PHB pueden ser especificados en relación a la prioridad relativa de los recursos sobre otros PHBs, o en términos de las características de tráfico observadas respectivamente (retardo, perdidas), los PHB son implementados en los nodos por medio de mecanismos de gestión de buffers y mecanismos de ordenación de paquetes, además están definidos en términos de las características más relevantes de las políticas de aprovisionamiento del servicio.

El PHB esta dividido en tres clases de servicio: Expedited Forwarding (EF), Assured Forwarding (AF) y Best-Effort (BE), en los servicio de Internet se refiere a servicio premiun, assured y Best-Effort respectivamente.

El servicio Premiun (Expedited Forwarding (EF)) [7] intenta proporcionar un servicio que sea capaz de ofrecer bajas perdidas, bajo retardo y bajo jitter a los usuarios, esta clase de PHB es apropiada para el transporte de VoIP. La causa principal de los retardos es el encolamiento en los equipos tales como: enrutadores y switches, el retardo y la variación del retardo son minimizados, cuando el retardo en el encolamiento es minimizado, el PHB EF proporciona un PHB en el cual se realiza una marcación apropiada de los paquetes para que el encolamiento sea corto o nulo, ya que el encolamiento es mínimo las pérdidas de paquetes es menor. Para asegurar que el tiempo en las colas para los paquetes EF es corto, la interfaz de salida debe tratar los paquetes de forma más rápida que la llegada de los mismos a la interfaz, independiente de los demás tipos de tráfico no EF. Todo esto con el fin de que los usuarios que contratan este servicio crean que se trata de un enlace virtual punto a punto, en [1] se describen las especificaciones definidas para el PHB con paquetes EF cuantificando la velocidad de servicio para el envió sobre cualquier intervalo de tiempo.

El grupo de envió Asegurado (Assured Forwarding) PHB sirve para asegurar el envió de paquetes IP de acuerdo a las prioridades dadas por las diferentes clases de AF definidas en [8].

En un dominio Diffserv el grupo AF se divide en cuatro clases, en donde cada clase es un nodo DS.

Reserva recursos de envió (espacio en el buffer, ancho de banda), previo acuerdo entre el proveedor de servicios y el cliente, se ofrece determinado servicio asignando un AF a los paquetes IP que van a ser trasportados, dentro de cada clase AF los paquetes son marcados con uno o tres posibles grados de descarte, en caso de congestión la precedencia de descarte indicara la prioridad dentro de las clases AF de un paquete frente otro. Los paquetes IP de una clase AF deben ser enviados independientes de los paquetes de otras clases AF, un nodo DS no debe agregar conjuntamente dos clases AF.

Para el AF de un grupo PHB no se especifican requerimientos de tiempos de envió asociados con el envió de paquetes AF, se debe tener en cuenta que los Nodos DS no deben realizar reordenación de los paquetes pertenecientes a un micro flujo. a pesar de que se presente descarte de paquetes.

En la tabla C.1 se pueden observar los puntos de código (codepoints) recomendados para las cuatro clases de AF con sus respectivas precedencias de descarte.

	Clase 1	Clase 2	Clase 3	Clase 4
Descarte Bajo	001010	010010	011010	100010
Descarte Medio	001100	010100	011100	100100
Descarte Alto	001110	010110	011110	100110

Tabla C.1 Clases de Servicio grupo AF PHB.

La clase de servicio Besf-Effort (Mejor-Esfuerzo) es el servicio tradicional prestado por Internet, el cual no maneja una calidad de servicio mayor a las diferentes clases AF en un dominio de servicios diferenciados.

Después de la marcación de los paquetes con un determinado PHB se envía a la cola de los nodos frontera, dicha cola debe contar con mecanismos de prioridad para el tratamiento de los paquetes.

La arquitectura de servicios diferenciados es una solución escalable para proporcionar calidad de servicio en las llamadas redes de próxima generación. Una posible dificultad en esta arquitectura se puede presentar cuando en el núcleo de la red confluyen muchos servicios que necesitan tener la misma calidad de servicio, para ayudar a solucionar este problema se puede utilizar la tecnología MPLS.

La Conmutación de Etiquetas Multiprotocolo (MPLS - Multi-Protocol Label Switching).

MPLS es uno de los protocolos que esta creciendo en popularidad, su principal meta es aumentar la velocidad de procesamiento de paquetes, está propuesto para ser transparente a las limitaciones de los protocolos que sirven para enrutar los paquetes, puede ser implementado en multiplicidad de redes, MPLS es una tecnología por medio de la cual se da soporte para ofrecer ingeniería de tráfico y la implementación de VPN (Virtual Prívate Networks) con una mayor facilidad y escalabilidad. Anterior a MPLS se tenían diferentes soluciones propietarias con el fin de mejorar la velocidad de encaminamiento de los paquetes. (Ej. IP switching, Tag switching) como un consenso de estas propuestas surgió el protocolo de Conmutación de Etiquetas Multiprotocolo (MPLS).

MPLS suministra una serie de enlaces virtuales o túneles virtuales que se utilizan para transporte de información a nodos que se encuentran en el borde de la red. La principal característica de la arquitectura MPLS es la de no enviar los paquetes por medio del direccionamiento IP. En lugar de esto se utilizan etiquetas de igual tamaño que sirven para la conmutación de los paquetes de acuerdo a estas etiquetas [9].

Tradicionalmente los paquetes cuando se encuentran en los enrutadores son enviados a otro enrutador. Para el envió de los paquetes se analiza la cabecera del paquete y de esta manera se toma la decisión junto con los algoritmos de encaminamiento, este proceso de análisis de la cabecera se lleva a cabo en cada enrutador para cada paquete, los enrutadores en general escogen la ruta más corta para enviar estos paquetes, esto ocasiona que unos enlaces estén mas sobrecargados que otros, lo cual puede llevar a congestión aunque no se utilicen los enlaces alternativos. En MPLS los paquetes son clasificados de acuerdo a las FEC (Forwarding Equivalence Class), las FEC se dan a un conjunto de paquetes IP que poseen características comunes, y que se reenvían de la misma manera.

Cuando un paquete entra a un dominio MPLS la asignación de la FEC solo es realizada por el enrutador de frontera y es codificada como una etiqueta, así, cada que se realiza el envió de un paquete la etiqueta es enviada con el, antes de ser enviados los paquetes son etiquetados, después de esto en cada salto no se hace análisis de la cabecera del nivel de red, la etiqueta se utiliza como índice en la tabla donde se especifica el próximo salto y la nueva etiqueta. Antes del envió de los paquetes al próximo salto la etiqueta vieja es removida y sustituida por una nueva etiqueta.

Una etiqueta es un identificador con asignación local, de longitud fija y corta que se utiliza para identificar una clase de las FEC. La etiqueta que se pone a un paquete determinado representa la clase FEC asignada al paquete.

En MPLS a los enrutadores que soportan conmutación de etiquetas se les conoce como enrutadores de Conmutación de Etiquetas (LSR – Label Switching Router), a grandes rasgos MPLS opera de la siguiente manera figura C.4:

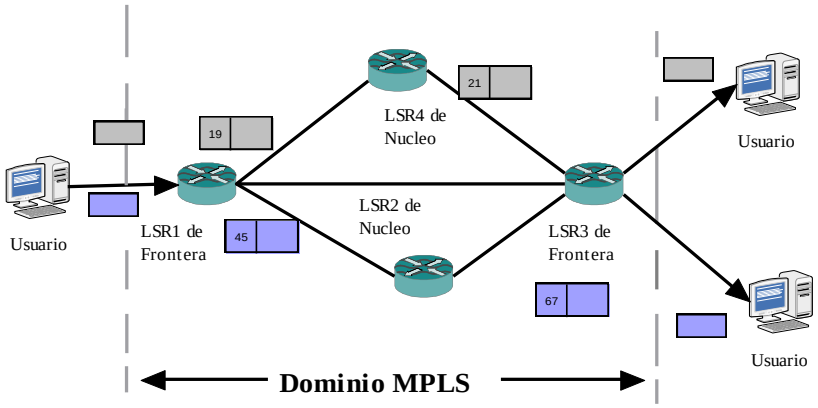

Figura C.4: Red Conmutada MPLS.

La ruta que siguen los datos se define por la transición de los valores de la etiqueta a medida que la etiqueta es cambiada en los LSR. El camino completo se determina por el valor inicial de la etiqueta, a ese camino se le denomina Trayecto Conmutado de Etiquetas (Label Switching Path).

Un paquete IP entrante al LSR 1 (Edge-LSR) se clasifica con una FEC equivalente, se le asigna un trayecto determinado y una etiqueta, el envió del paquete se realiza por una interfaz determinada. El LSR 2 (Core-LSR) recibe este paquete realiza un análisis del conjunto (interfaz de entrada, etiqueta), estos valores son reemplazados por (interfaz de salida, nueva etiqueta) para su envió. El LSR 3 (Edge-LSR) actúa como enrutador de

salida, elimina la etiqueta y además entrega el paquete al nodo destino por medio de la utilización de encaminamiento de nivel tres.

Observar que el formato exacto de una etiqueta y como se añade al paquete depende de la tecnología de enlace de nivel dos, usada en la red MPLS. Por ejemplo, una etiqueta podría corresponder a un VPI/VCI de ATM, un DLCI de Frame Relay. Para otros tipos de nivel 2 (tales como Ethernet y PPP) la etiqueta MPLS se añade al paquete de datos entre las cabeceras del nivel dos y tres.

MPLS hace uso de protocolos por medio de los cuales un enrutador LSR informa de las ligaduras etiqueta/FEC que se han hecho, también se encarga de las negociaciones de los dos puertos de distribución de etiquetas para aprender las posibilidades MPLS de otro LSR.

En MPLS no se define un solo protocolo para realizar las funciones anteriormente mencionadas, sino que hay múltiples protocolos, entre estos se puede mencionar MPLS-BGP, MPLS-LPD, RSVP.

En MPLS también se permite el apilamiento de etiquetas, las cuales son organizadas de forma LIFO (last-in, first-out), el procesado está basado siempre en la etiqueta de más arriba de la pila, sin considerar la utilización de las otras etiquetas que estén en la pila.

REFERENCIAS

[1]. Maurice David Woernhard. “Generating Synthetic VoIP Traffic for Analyzing Redundant OpenBSD-Firewalls”. Tesis de Maestría. Universidad de Oslo. Departamento de Informática. Mayo 23, 2006. Documento en pdf.

[2] R. Braden. ISI. D, Clark MIT. S. Shenker. Xerox PARC “Integrated Services in the Internet Architecture”. An Overview RFC 1633. June 1994.

[3] S Shenker Xerox. C. Partridge BBN. R. Guerin. IBM. “Specification of Guaranteed Quality of Service“. RFC 2212. September 1997.

[4] J. Wroclawski. MIT LCS. “Specification of the Controlled-load Network Element Service”. RFC 2211. Septiembre 1997.

[5] S .Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss. “An Architecture for Differentiated Services”. RFC 2475. Torrent Networking Technologies, EMC Corporation, Sun Microsystems. Nortel UK. Bell Labs Lucent Technologies. Lucent Technologies. Diciembre 1998”

[6] K. Nichols, S. Blake, F. Baker, D. Black. “Definition of the Differentiated Services Field

(DS Field) in the IPv4 and IPv6 Headers". RFC 2474. Cisco Systems, Torrent Networking Technologies, Cisco Systems, EMC Corporation. Diciembre 1998.

[7] B. Davie, A Charny, INC. J.C.R Bennet, K. Benson. Tellabs. J.Y. Le Boudec EPFL. W. Courtney TRW. S. Davari. PMC-Sierra. V. Firoiu. D. Stiliadis. "An Expedited Forwarding PHB (Per-Hop Behavior)". RFC 3246. Cisco Systems, Motorola, Nortel Networks, Lucent Technologies. Marzo 2002.

[8] J. Heinanen, Telia. Finland, F. Baker. W [1] . Weiss. J Wroclawski [2] . MIT LCS "Assured Forwarding PHB Group". RFC 2597. Cisco Systems [1] , Lucent Tecnologies [2] . June 1999.

[9] E. Rosen [1] . A. Viswanathan [2] . R. Callon [3] . "Multiprotocol Label Switching Architecture". RFC 3031. Cisco Systems, Inc [1] , Force 10 Networks, Inc [2] , Juniper Networks. Inc [3] . Enero 2001.

Anexo D

Métodos de Estimación del Parámetro de Autosimilitud.

Existen diferentes métodos para determinar si una serie temporal dada de datos es auto-similar o no. En este anexo se abordan una serie de métodos que permiten estimar el parámetro de autosimilitud *H*, para así establecer si hay presencia de dependencia de largo rango en las trazas de tráfico.

D.1 Varianza-Tiempo

El estimador varianza – tiempo, propuesto por Cox y Smith en 1953 [1, 2], está basado en la propiedad de los procesos auto-similares en que la varianza de la muestra de las medias converge mas lentamente a cero que m^{-1} [3, 4].

$$Var(X^{(m)}) = Var(X)\, m^{2H-2} = a\, m^{2H-2} \qquad \text{(D.1.1)}$$

Donde $a = Var(X) > 0$ y $X^{(m)}$ es el proceso obtenido por la agregación del proceso original X definido en la ecuación (2.7.3.5). La representación gráfica de la varianza – tiempo es obtenida graficando $\log(Var(X^{(m)}))$ vs. $\log(m)$, con los puntos resultantes en el plano y utilizando mínimos cuadrados se ajusta una simple recta, una estimación del parámetro de Hurst esta dada por $H = 1 - \beta/2$, donde β $(0 < \beta < 1)$ es la pendiente de la recta resultante. La gráfica varianza – tiempo necesita tres pasos para estimar H [3].

(i) Para diferentes enteros m en el rango $2 \le m \le n/2$, y un número suficiente de (q) sub-series de longitud m, calcular las medias de las muestras $\overline{X}_1^{(m)}, \overline{X}_2^{(m)}, \ldots\ldots, \overline{X}_i^{(m)}$ y la media total $\overline{X}^{(m)}$, donde

$$\overline{X}^{(m)} = \frac{1}{i}\sum_{j=1}^{i} \overline{X}_j^{(m)} \qquad \text{(D.1.2)}$$

(ii) Para cada m, calcular la muestra de la varianza de las medias $\overline{X}_j^{(m)}$, $j = 1, \ldots, i$:

$$Var(X^{(m)}) = \frac{1}{1-i}\sum (\overline{X}_j^{(m)} - \overline{X}^{(m)})^2 \qquad \text{(D.1.3)}$$

(iii) Graficar $\log(Var(X^{(m)}))$ *vs.* $\log(m)$.

En la práctica, se asume que ambos n y m son grandes, el largo de cada grupo y el número de grupos. Si las secuencias no tienen dependencia de largo rango y varianza finita, la estimación de H es *0.5* y la pendiente de la línea adaptada debería ser *-1*. Algunos puntos muy por debajo y por encima de la representación gráfica son usados para adaptar la línea por medio de mínimos cuadrados. En otras palabras

- ✓ Para m pequeña, la estimación de las varianzas son basadas en la estructura de correlación de corto plazo. Los efectos de corto rango (SRD) pueden distorsionar la estimación de H. Estas estimaciones simplemente no son usadas en el procedimiento de adaptación.
- ✓ Para m grande, las estimaciones de la varianza son basadas en solo pocas observaciones y por consiguiente no son confiables. Para evitar deteriorar la estimación de H, estas estimaciones simplemente no son usadas en el procedimiento de adaptación.

D.2 Método del Periodograma

El método del periodograma estima el parámetro de Hurst por ajuste de una línea recta en el dominio espectral. Este se basa en la observación de la densidad espectral, la densidad espectral del Movimiento Browniano Fraccional esta dada por:

$$f(\lambda) = 2\sin(\pi H)\Gamma(2H+1)(1-\cos\lambda)\left[|\lambda|^{-2H-1} + B(\lambda, H)\right] \qquad \text{(D.2.1)}$$

Con

$$B(\lambda, H) = \sum_{j=1}^{\infty}\left\{(2\pi j + \lambda)^{-2H-1} + (2\pi j - \lambda)^{-2H-1}\right\} \qquad \text{(D.2.2)}$$

su comportamiento es igual a $cf|\lambda|^{1-2H}$ para $|\lambda| \to 0$.

La estimación de la densidad espectral para el periodograma esta dada por:

$$I(\lambda) = \frac{1}{2\pi N}\left|\sum_{n=0}^{N-1} x[n]e^{j\lambda n}\right|^2 \qquad \text{(D.2.3)}$$

donde λ es la frecuencia, N es la longitud de la serie, y X es la serie temporal. En el caso de varianza finita, $I(\lambda)$ es un estimador de la densidad espectral de X, una serie con dependencia de largo rango puede tener una densidad espectral proporcional a $|\lambda|^{1-2H}$ para frecuencias cercanas al origen. De este modo una regresión $\log I(\lambda)$ frente a $\log(\lambda)$ proporciona un estimado del valor de *H*. Consiguiendo una gráfica de log-log para el periodograma *vs.* la frecuencia, se obtiene una línea recta con pendiente de *1-2H*. En la práctica, para el cálculo se utilizan únicamente las frecuencias bajas, solo si se comprenden las frecuencias cercanas a cero.

D.3 Método de Varianza Agregada

Considere la serie agregada (D.3.1) obtenida de dividir una serie dada de longitud *N* en bloques de longitud *m*, y promediando las series sobre cada bloque [1], [3].

$$X^{(m)}(k) = \frac{1}{m} \sum_{i=(k-1)m+1}^{km} X_i, k = 1,2,..., [N/m]. \qquad \text{(D.3.1)}$$

A partir de la serie agregada se puede calcular su varianza muestral:

$$VarX^{(m)} = \frac{1}{N/m} \sum_{k=1}^{N/m} (X^{(m)}(k) - \overline{X})^2 \qquad \text{(D.3.2)}$$

El escalamiento de la serie $X^{(m)}$ es igual a m^{H-1}, esto si la serie es Gaussiana, o al menos la varianza es finita, La varianza muestral puede ser asintóticamente proporcional a m^{2H-2} para N/m y *m* grandes.

Para valores sucesivos de *m*, la varianza muestral de la serie agregada es dibujada vs. *m* en una gráfica logarítmica. El resultado podría ser una línea recta con pendiente de *2H-2*. En la práctica, la pendiente es estimada mediante regresión lineal con los puntos obtenidos. Se asume que tanto N/m y N son grandes. Esto asegura que la longitud de cada bloque, así como el número de bloques sea grande. Si la serie no tiene dependencia de largo rango y la varianza es finita, entonces *H=0.5* y la pendiente de la línea podría ajustarse a *-1*. En la práctica los puntos que están muy por debajo o muy altos en la figura no son tenidos en cuenta para el ajuste de línea por regresión lineal. Efectivamente, si el extremo bajo de la figura es usado, los efectos de corta dependencia pueden distorsionar la estimación de H y en el extremo muy alto de la figura existen pocos bloques para obtener una estimación confiable de la varianza. En la figura d1 se esboza la gráfica obtenida y el valor de H para muestras de trafico de voz IP.

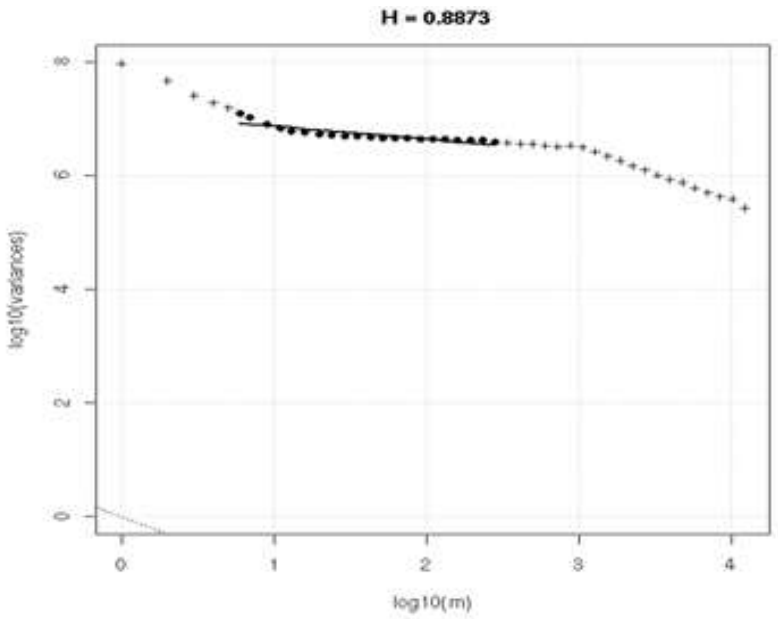

Figura d.1 Gráfica generada por medio del método de Varianza Agregada.

D.4 Método de momentos Absolutos

Consideremos la serie agregada (D.4.1) obtenida de dividir una serie dada (asumimos que puede ser FGN) de longitud *N* en bloques de longitud m, y promediamos la serie sobre cada bloque.

$$X^{(m)}(k) = \frac{1}{m}\sum_{i=(k-1)m+1}^{km} X_i, k = 1,2,...,[N/m]. \qquad \text{(D.4.1)}$$

Nosotros tomamos n momentos absolutos de esta serie

$$AM_n^{(m)} = \frac{1}{N/m}\sum_{k=1}^{N/m} | X^{(m)}(k) - \overline{X} |^n. \qquad \text{(D.4.2)}$$

Donde $X^{(m)}$ esta definido como en la ecuación (D.2.1) y X es el promedio total de la serie. La serie agregada $X^{(m)}$ se comporta asintóticamente como $Cm^{n(H-1)}$ para un m grande, y así $AM_n^{(m)}$ es proporcional a $m^{n(H-1)}$.

Este método se usa típicamente para $n=1$ (promedio absoluto). Para $n=2$, se reduce al método de varianza agregada [2]. Para valores sucesivos de m, la muestra para los momentos absolutos para las series agregadas es trazada en una gráfica log-log para m. El resultado podría ser una línea recta con pendiente *n(H-1)*.

Se debe asumir que tanto N como N/m son grandes. Esto asegura que la longitud del bloque y el número de bloques sea grande. Si la serie no tiene dependencia de largo rango y varianza finita, entonces *H=0.5*, y la pendiente puede caer a razón de $-n/2$. En la práctica, los puntos extremos muy bajos y altos no son usados para trazar la línea. De hecho, efectos de rango corto pueden distorsionar la estimación de H si el extremo bajo de la figura es usado, y en el extremo muy alto de la traza hay pocos bloques para obtener un estimado confiable de $AM_n^{(m)}$. La figura d2 es obtenida por medio de este método, también ahí aparece el valor estimado para H.

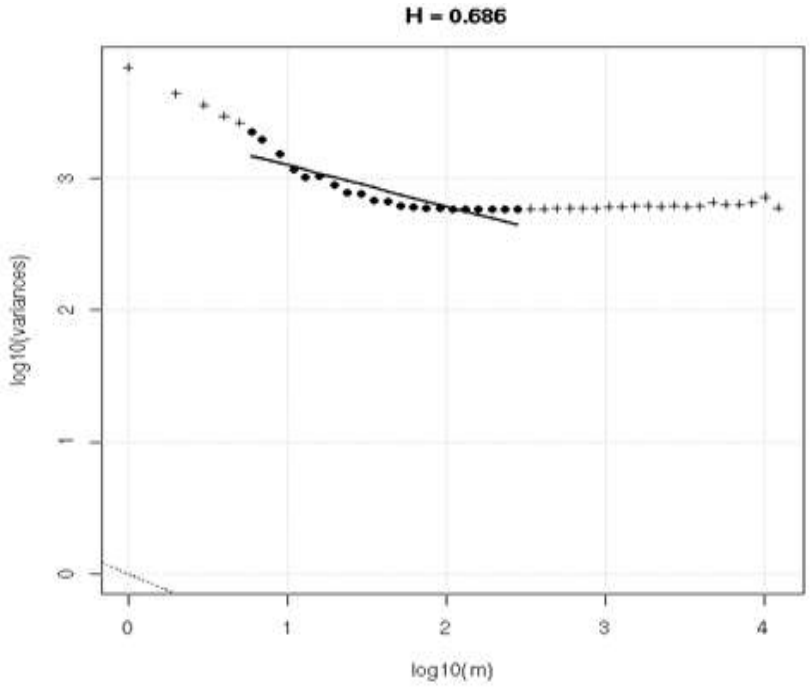

Figura d.2 Gráfica generada por medio del método de Momentos Absolutos

D.5 Método de Dimensión Fractal

Una técnica sugerida por Higuchi, es muy similar al método de momentos absolutos con $n=1$, este usa

$$L(m)=\frac{N-1}{m^3}\sum_{i=1}^{m}[\frac{N-1}{m}]^{-1}\sum_{k=1}^{[(N-i)/m]}|\sum_{j=i+(k-1)m+1}^{i+km}X(j)|. \qquad \text{(D.5.1)}$$

donde N es la longitud de la serie temporal, m es equivalente al tamaño de un bloque y [ ] denota la función entera más grande. Para series con dependencia de largo rango, $EL(m)-Cm^{H-2}$ esencialmente la diferencia con el método de momentos absolutos $(n=1)$, la línea es usada como una ventana deslizante para computar la serie agregada, en lugar de usar bloques intercalados, pero este método es mucho mas intensivo [1]. Mientras que esta modificación puede resultar en un incremento de la exactitud de la series temporales cortas. Esto parece no ser una ventaja usando las series de longitud grande. En la figura d.3 se puede ver el resultado gráfico para este método.

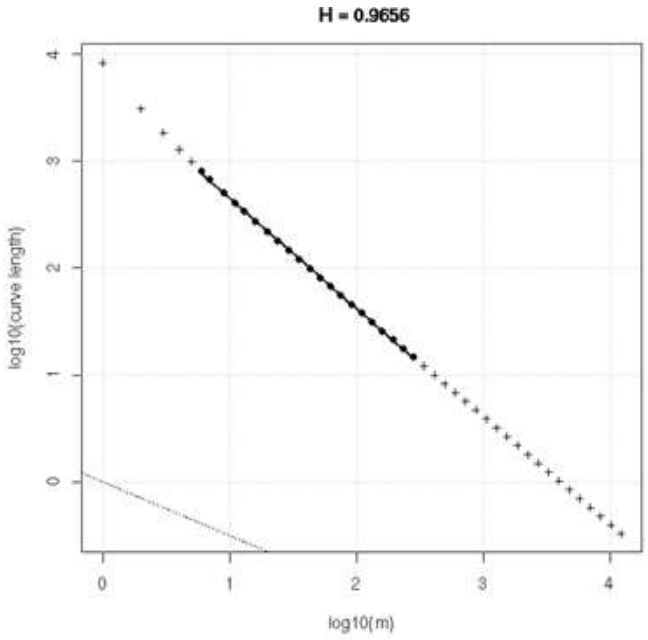

Figura d.3 Gráfica generada por medio del método de Higuchi.

D.6 Método de R/S

Este método esta basado en la obtención y estudio estadístico denominado rango ajustado de reescalado, o más habitualmente, estadístico R/S [2], [3].

*Definición: Estadístico R/*S, para una serie temporal $\{Xk : k = 1,2..., N\}$ con media muestral $\overline{X}(N)$ y varianza muestral $S^2(N)$, el estadístico R/S viene dado por el cociente $R(n)/S(N)$,

donde

$$R(N) = \max(0, W_1, W_2, ..., W_n) - \min(0, W_1, W_2, ..., W_n) \quad \text{(D.6.1)}$$

$$W_k = \sum_{i=1}^{k} X_i - k\overline{X}(N) \quad 1 \le k \le N \quad \text{(D.6.2)}$$

Para los modelos tradicionales, la media de esté estadístico se comporta típicamente según:

$$E[R(N)/S(N)] \approx an^{0.5} \text{ cuando } n \to \infty \quad \text{(D.6.3)}$$

Sin embargo, para los procesos auto-similares el estadístico se comporta asintóticamente de la forma:

$$E[R(N)/S(N)] - C_H n^H \text{ cuando } n \to \infty \quad 1/2 < H < 1 \quad \text{(D.6.4)}$$

Donde C_H es una constante positiva, finita, que no depende de n. Concretamente Hurst encontró que muchos de los registros históricos se comportan así para un valor típico de H en torno a *0.7*.

Para determinar H mediante el estadístico R/S se procede de la siguiente manera. Se divide la serie en K bloques de tamaño $[N/K]$. Posteriormente se halla

$R(t_i,n)/S(t_i,n)$, comenzando en los puntos $t_i=(i-1)N \quad i=1,2,...$ $R(t_i,n)$, reemplazando W_k por $W_{ti+k}-W_{ti}$, y $S^2(t_i,n)$ es la varianza muestral de $X_{ti+1,}X_{ti+2},...,X_{ti+n.}$ Debe cumplirse que $ti+n\le N$, de forma que $n<N/K$ se obtienen K valores de R/S, mientras que, a medida que n crece hacia N, el número de valores va disminuyendo hasta un único valor cuando $n\ge N-N/K$.

Eligiendo valores de n logarítmicamente equi-espaciados (comenzando con n en torno a 10), se representa $\log[N(k_i,n)/S(k_i,n)$ frente a $\log(n)$, y se obtienen varios puntos para cada n. El parámetro H puede estimarse directamente con el valor de la pendiente de la recta de regresión que mejor se ajusta a los puntos de la gráfica. Ya que cualquier presencia de SRD en las series temporales de lugar típicamente a una zona de transitorio en la zona inferior de la gráfica, y la zona superior contiene pocos puntos para que la estimación sea fiable, en la práctica la regresión solo se realiza para los puntos ubicados en la zona intermedia. Un resultado gráfico se muestra en la figura d.4

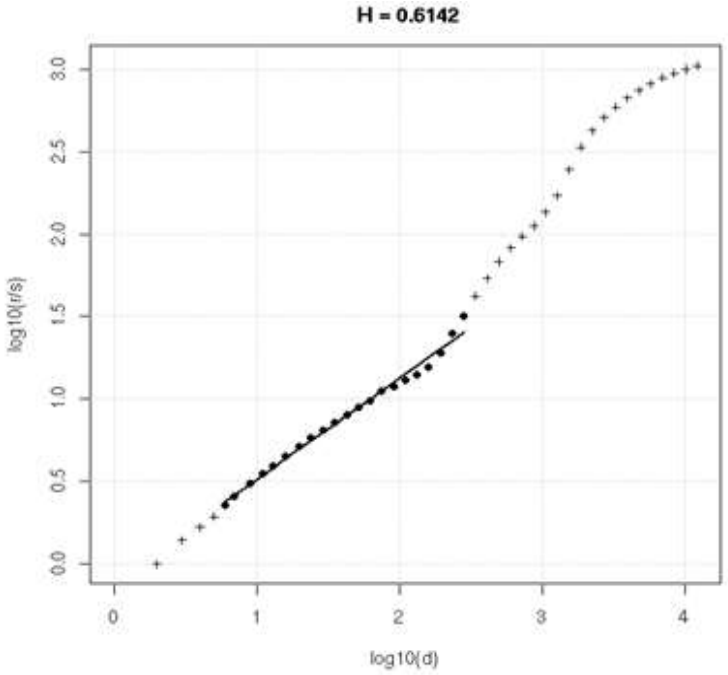

Figura d.4 Gráfica generada por medio del método de R/S.

REFERENCIAS

[1] Ton Dieker. “Simulation of fractional Brownian motion”. Tesis de maestría Departamento de Ciencias Matemáticas Universidad de Twente The Netherlands. Año 2002.

[2] Izzeldin Ibrahim Mohamed Abdelaizz “Modeling And Performance Evaluation Of Self-Similar Behavior of Mpeg-4 Video Trafffic Generators. Universiti Teknologi Malaysia 2006.

[3] Richard G. Clegg. “The Statistics of Dynamic Networks” Tesis de Doctorado. University of York. Department of Mathematics. Junio 2004.