UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

CONFIGURACIÓN DE SAP ERP PARA UN MODELO DE INDUSTRIA DE CONSUMO MASIVO (CPG)

Por:
JULIO DE ABREU MOLINA

Realizado con la asesoría de:
KENYER DOMINGUEZ

PASANTÍA LARGA
Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero en Computación

Sartenejas, Septiembre del 2013

UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

ACTA FINAL PASANTÍA LARGA

CONFIGURACIÓN DE SAP ERP PARA UN MODELO DE INDUSTRIA DE CONSUMO MASIVO (CPG)

Presentado por:
JULIO DE ABREU MOLINA

Esta Pasantía Larga ha sido aprobado por el siguiente jurado examinador:

KENYER DOMINGUEZ

PROFESOR 2

ANA CECILIA GARCIA REVERÓN

Sartenejas, FECHA (dd/mm/aa)

Resumen

AQUÍ VA EL CONTENIDO DEL RESUMEN.

Como parte de este sistema, se planteó la evaluación de un método de detección de ataques de denegación de servicio mediante la estimación del parametro de Hurst utilizando el mecanismo de ventanas deslizantes. Los métodos se incorporaron en una herramienta de línea de comando versátil capaz de obtener series de tiempo apartir de una traza de datos tomados de la red y de graficar las estimaciones del parámetro de Hurst. La herramienta fue evaluada mediante el uso del algoritmo de Paxson con respecto a su precisión y con trazas de los escenarios de ataques de denegación de servicio producidas en el laboratorio Lincoln del *Massachusetts Institute of Technology*, durante la evaluación dirigida por la *Defense Avadanced Research Projects Agency*, con respecto a su capacidad de detección de ataques. Los resultados muestran que el método de detección se puede implementar para su uso en tiempo real y se puede mejorar con respecto a la detección de ataques.

DEDICATORIA

Agradecimientos

AQUÍ VAN LOS AGRADECIMIENTOS.

Índice general

Índice de cuadros

Índice de figuras

Introducción

AQUÍ VA EL CONTENIDO DE ANTECEDENTES.

AQUÍ VA EL CONTENIDO DE LA JUSTIFICACIÓN.

En la Universidad de Tohoku de Sendai, Japón, en el laboratorio Kinoshita de la facultad de Ciencias de la Computación, se tiene un *framework* con el cual han creado un *Active Information Resource* (AIR) - *Network Management System* (NMS) [KIAK04] para gestionar las redes de computadoras. Mediante el lenguaje de programación orientado a agentes basado en Java, DASH (*Repository-based Agent Framework*) y su ambiente de dise no interactivo IDEA (*Interactive Design Environment*), la idea final es que la red y los elementos que la componen se pueda auto-gestionar mediante los agentes instalados en los elementos [USKS02], sin la necesidad de tener un administrador. El objectivo es identificar y solucionar los problemas de la red en tiempo real [AAK04] [KAIK07].

AQUÍ VA EL CONTENIDO DEL PLANTEAMIENTO DEL PROBLEMA.

En los últimos a nos ha habido un gran interés por determinar si los nuevos modelos basados en la autosimilaridad y la dependencia de largo alcance pueden ser utilizados para la detección de anomalías en las redes de datos [DDHT08]. Entre las anomalías más destacadas se encuentran las de ataques de denegación de servicio y su ampliación: el llamado ataque distribuido de denegación de servicio [Li06] [XLLH04].

En el trabajo de [Li06] se utilizó, con buenos resultados, el cálculo del grado de autosimilaridad para detectar anomalías en la red. Se realizó una implementación para un *framework* de autogestión de redes desarrollado en el laboratorio Kinoshita de la Universidad de Tohoku. Como se explica en la próxima sección la idea del proyecto de grado fue mejorar esta

implementación.

El objetivo principal del proyecto es la de la implementación de una solución *SAP SD* en una empresa de bebidas de consumo masivo *SSA CPG* para gestionar los procedimientos comerciales.

1. Adquirir los conocimientos de la Metodología Ascendant SAP y el funcionamiento del Sistema SAP en la funcionalidad Ventas y Distribución (SD).

2. Analizar los procesos de Ventas y Distribución con las funcionalidades SAP que los soportan. Adquirir los conocimientos generales del negocio, procesos y visión global de la solución SAP a implantar.

3. Configurar los campos del sistema a implantar para el módulo SD acorde a las necesidades del cliente, hacer los desarrollos necesarios y probar las funcionalidades SAP ERP SD que soportan los procesos comerciales.

4. Preparar los datos y el sistema, realizar la demostración y documentar los resultados correspondientes al módulo SD, para comprobar el funcionamiento y configuración del módulo de manera individual y en relación con otros módulos implicados del sistema.

La implementación deberá tomar como entrada una traza creada por la librería `libpcap`[1] y poder crear fácilmente una serie de tiempo para la estimación del parámetro de Hurst, haciendo flexible la estimación con diferentes series de tiempo creadas a partir de la misma traza. Como salida, el programa debe graficar los resultados de las estimaciones del parámetro

[1]Esta librería provee una interfaz de alto nivel para la captura y manipulación de los paquetes a bajo nivel pudiendo manipular todos los paquetes, incluidos aquellos destinados a otros nodos de red. La librería se encuentra disponible en `http://www.tcpdump.org`.

de Hurst, incluyendo el cambio del parámetro en el tiempo, para así poder verificar de forma rápida los resultados.

Una vez validada la estimación del parámetro de Hurst, se deberá verificar si esta estimación puede ser utilizada para detectar las anomalías en tiempo real creadas por un ataque distribuido de denegación de servicio en las trazas de red.

Capítulo 1

Marco teórico

AQUÍ VA EL CONTENIDO DEL MARCO TEÓRICO.

Los conceptos más importantes sobre las series de tiempo, el parámetro de Hurst y su uso para la detección de ataques de denegación de servicio se presentan en este capítulo.

La sección 1.1 presenta las definiciones básicas para entender los procesos autosimilares y sus propiedades. La sección 1.2 describe los métodos para estimar el parámetro de Hurst, mientras que la sección 1.3 describe los problemas inherentes con su estimación. La última sección define los ataques de denegación de servicio y cómo estos afectan la autosimilaridad de las redes de datos.

1.1. Procesos autosimilares

Para explicar el uso del parámetro de Hurst para la detección de anomalías de red, debemos primero presentar algunas definiciones básicas. Entre ellas están las series de tiempo, los procesos autosimilares, los procesos estacionarios y los procesos agregados. Todos los conceptos descritos fueron tomados de [PW00], [SSO07] y [WWT03].

1.1.1. Definiciones

Definición 1. Una **serie de tiempo** es una secuencia de datos, observaciones o valores, medidos en determinados momentos del tiempo, ordenados cronológicamente y, normalmente, espaciados entre sí de manera uniforme.

Un proceso aleatorio en tiempo discreto o la serie de tiempo $X(k)$, $k \in Z$ es interpretada como el volumen de tráfico (medido en paquetes, bytes o bits) hasta el momento k.

Definición 2. Consideremos que un proceso con valores reales $\{X(t),\, t \in R\}$ tiene **incrementos estacionarios** si para todo $h \in R$:

$$\{X(t+h) - X(h),\, t \in R\} \stackrel{d}{=} \{X(t) - X(0),\, t \in R\} \tag{1.1}$$

donde $\stackrel{d}{=}$ denota la igualdad de las distribuciones finito-dimensionales.

La secuencia de incrementos para $\{X(t),\, t \in R\}$ en tiempo discreto se define como $Y_k = X(k+1) - X(k)$, $k \in Z$.

Definición 3. Para la simulación de tráfico, el proceso $X(k)$ es considerado **estacionario en el más amplio sentido**, aplicando la restricción de que la función de covarianza $R(k_1, k_2) = M[(X(k_1) - \mu)(X(k_2) - \mu)]$ es invariante con respecto al desplazamiento.

La definición 3 implica que $R(k_1, k_2) = R(k_1 + m, k_2 + m)$ para cualquier $k_1, k_2, m \in Z$. Siendo M la operación de promedio, se supone que los dos primeros momentos $\mu = M[X(k)]$, $\sigma^2 = M[X(k) - m^2]$ existen y son finitos para cualquier $k \in Z$. Suponemos por conveniencia que $\mu = 0$. Como en la condición estacionaria $R(k_1, k_2) = R(k_1 - k_2, 0)$, la función de covarianza será designada como $R(k)$ y el factor de correlación será $r(k) = R(k)/R(0) = R(k)/\sigma^2$.

Definición 4. El proceso de valores reales $\{X(t),\, t \in R\}$ es **autosimilar** con el exponente autosimilar, $H > 0$ (as-H) si, para cualquier $a > 0$

$$\{X(at),\, t \in R\} \stackrel{d}{=} \{a^H X(t),\, t \in R\} \tag{1.2}$$

Teorema 1. *Si* $\{X(t),\, 0 < t < \infty\}$ *es un proceso as-H, el proceso*

$$Y(t) = e^{-tH} X(e^t),\ -\infty < t < \infty \tag{1.3}$$

es estacionario. A la inversa, si el proceso $\{Y(t),\ -\infty < t < \infty\}$ *es estacionario, el proceso*

$$X(t) = t^H Y(\ln t),\ 0 < t < \infty \tag{1.4}$$

es un proceso as-H.

Desde el punto de vista de implementación práctica aquellos con incrementos estacionarios son de especial interés, ya que éstos conllevan a secuencias estacionarias con un comportamiento especial.

Definimos entonces un proceso autosimilar con parámetro H e incrementos estacionarios (asie-H).

Lema 1. *Asumimos que* $\{X(t),\, t \in R\}$ *es el proceso asie-H con varianza infinita. Entonces* $0 < H \leq 1$, $X(0) = 0$ *la covarianza está definida la expresión:*

$$R(t_1, t_2) = \frac{1}{2}\{|t_1|^{2H} + |t_2|^{2H} - |t_1 - t_2|^{2H}\sigma_x^2\} \tag{1.5}$$

Si $X(t)$ es un proceso asie-H con varianza finita, entonces se da que $0 < H < 1$. En el rango $0 < H < 0{,}5$ el proceso incremental es de dependencia de corto alcance. En el rango

$0{,}5 < H < 1$ el proceso incremental es de dependencia de largo alcance. La dependencia de corto y largo plazo se describirá en mayor detalle en la sección 1.1.3. Para efectos de este documento y en los modelos basados en autosimilaridad el rango $0{,}5 < H < 1$ es el importante. En este rango el proceso incremental $Y(t)$, que se define como:

$$Y_k = X(k) - X(k-1) \ , \ k \in Z \tag{1.6}$$

tiene un factor de correlación de la siguiente forma:

$$r(k) = \frac{1}{2}[(k+1)^{2H} - 2k^{2H} + (k-1)^{2H}] \tag{1.7}$$

Definición 5. Sea $Y = Y_i$, $i \in Z$ un proceso estacionario con la función de covarianza $R(k)$. La **serie de tiempo m-agregada** $Y^{(m)}$ está definida promediando los intervalos no solapados con tama no m de la serie original y substituyendo su promedio en cada intervalo por la fórmula:

$$Y_k^{(m)} = \frac{1}{m^H} \sum_{i=(k-1)m+1}^{km} Y_i \ , \ k \in Z \ , \ 0 < H < 1 \tag{1.8}$$

y designando la función de covarianza correspondiente a $R^{(m)}(k)$.

Definición 6. El proceso discreto aleatorio $\{Y_k, \ k \in Z\}$ es **estrictamente autosimilar** en el más amplio sentido (exactamente autosimilar de segundo orden) con parámetro de Hurst $(\frac{1}{2} < H < 1)$ si

$$R(k) = \frac{\sigma^2}{2}[(k+1)^{2H} - 2k^{2H} + (k-1)^{2H}], \ \forall k \geq 1 \tag{1.9}$$

Definición 7. El proceso $X(t)$ es un **proceso asintóticamente autosimilar con parámetro autosimilar** H si

$$\lim_{m\to\infty} R^{(m)}(k) = R(k),\ \forall m \geq 1 \tag{1.10}$$

1.1.2. Propiedades

Las siguientes propiedades de los procesos autosimilares son equivalentes:

- **Función de covarianza que decrece de forma hiperbólica**: Esta toma la forma

$$R(k) \cong k^{(2H-2)} L(t),\ k \to \infty \tag{1.11}$$

donde $L(t)$ cumple que

$$lim_{t\to\infty} L(tx)/L(t) = 1,\ \forall x > 0 \tag{1.12}$$

Entonces se puede notar que la función de covarianza no se puede sumar y la serie formada por lo valores secuenciales de la función divergen en

$$\sum_{k} R(k) = \infty \tag{1.13}$$

Esta suma infinita implica que aunque los $R(k)$ individualmente son peque nos en el infinito, su efecto acumulativo es sumamente importante. Se describe esta propiedad con mayor detalle en la sección 1.1.3.

- **La varianza muestral del proceso agregado decrece más lentamente que la magnitud y es inversamente proporcial al tama no de la muestra**: Si la nueva secuencia de tiempo $\{X_i^{(m)}; i = 1, 2 \ldots\}$ obtenida por promediando los intervalos no

solapados con tama no m de la serie original $\{X_i; i = 1, 2 \ldots\}$ y substituyendo su promedio en cada intervalo la varianza seguirá la regla

$$\sigma^2(X^{(m)}) \sim m^{(2H-2)},\ m \to \infty \tag{1.14}$$

1.1.3. Dependencia de corto y largo alcance

Esta sección explica el significado del exponente de Hurst H y sus valores limitantes, para entender mejor el significado de la dependencia de corto y largo alcance.

Definición 8. El proceso $\{Y_i,\ i \in Z\}$ es llamado **estacionario con dependencia de largo alcance** si existe un $c_r > 0$ y un número real $\alpha \in (0; 1)$, $\alpha = 2 - 2H$ tal que

$$\lim_{k\to\infty} \frac{r(k)}{c_r k^{-\alpha}} = 1 \tag{1.15}$$

Definición 9. El proceso $\{Y_i,\ i \in Z\}$ es llamado **estacionario con dependencia de corto alcance** si existe un $0 < c_0 < 1$ tal que

$$\lim_{k\to\infty} \frac{r(k)}{c_0^k} = 1 \tag{1.16}$$

Dado el comportamiento asintótico del factor $r(k)$, con la ayuda de la expansión de la serie de Taylor en:

$$r(k) = H(2H - 1)k^{2H-2} + O(k^{2H-2}) \text{ mientras } k \to \infty \tag{1.17}$$

La utilización de la la definición 8 muestra que las correlaciones tienen la siguiente propiedad:

$$\sum_{k=-\infty}^{\infty} r(k) = \infty \tag{1.18}$$

Por lo tanto, si la $r(k)$ de un proceso estacionario cumple esta propiedad, este es un proceso estacionario con dependencia de largo alcance.

Análogamente, utilizando la definición 9, existe un $p \in R$ tal que:

$$\sum_{k=-\infty}^{\infty} r(k) = p < \infty \tag{1.19}$$

1.2. Métodos para la estimación del parámetro de Hurst

En la práctica, comprobar la autosimilaridad y estimar el parámetro de Hurst son problemas complicados (ver sección 1.3). Sin embargo, existen varios métodos que permiten detectar si una serie de tiempo es autosimilar. Los métodos se definen en el dominio de tiempo [LTWW93] [BP05] y el de la frecuencia [BP05] [PW00] [SSO07]. En este documento sólo se describen los métodos implementados en la herramienta, los cuales están definidos en el dominio de tiempo debido a la naturaleza de los datos a utilizar. Otros métodos interesantes tanto en el dominio de tiempo como en el de frecuencia puede consultarse en [SSO07].

1.2.1. Estadístico R/S

El método del estadístico R/S, abreviado en este documento como *R/S*, obtiene un estimador muy conocido y descrito en gran detalle en [LTWW93]. Teniendo una serie de tiempo X_k de tama no N, para calcular el estadístico R/S se debe obtener un $t_j = jn$ para $4 \leq n \leq N$ con $0 \leq j \leq N/n$. Se calcula entonces el cociente $R(t_j, n)/S(t_j, n)$ como se muestan en las

ecuaciones (1.20), (1.21) y (1.22). Los puntos resultantes deben ser graficados en un gráfico $log - log$ y luego se aplica el método de mínimos cuadrados para obtener la pendiente que es el estimador del parámetro de Hurst.

$$\overline{X_n} = \frac{\sum_{i=t_j}^{t_j+n} X_i}{n} \tag{1.20}$$

$$\begin{aligned} R(t_j, n) &= \text{máx}(W_0(t_j)...W_n(t_j)) - \text{mín}(W_0(t_j)...W_n(t_j)) \qquad (1.21) \\ W_0(t_j) &= 0 \\ W_k(t_j) &= \sum_{i=t_j}^{t_j+k} X_i - k\overline{X_n} \end{aligned}$$

$$S(n) = \sqrt{\frac{\sum_{i=0}^{n} (X_i - \overline{X_n})^2}{n-1}} \tag{1.22}$$

La figura 1.1 muestra un ejemplo. En esta figura se muestra la graficación de los puntos resultantes de calcular $R(t_j, n)/S(t_j, n)$ con ejes $\log - \log$. La pendiente resultante, que es el estimador del parámetro Hurst, debe tener un valor entre $0{,}5 < H < 1$. En la figura 1.1 se grafican las pendientes 0,5 y 1, entre cuales se debe encontrar la pendiente resultante.

En el cálculo del estimador de Hurst usando el estadístico R/S, parte del costo del algoritmo está en el cálculo de $W_k(t_j)$ debido a las repetidas sumas de los mismo términos. En [ITU+07] se presenta una forma de mejorar el tiempo de estimación creando una memoria temporal para guardar las sumas intermedias.

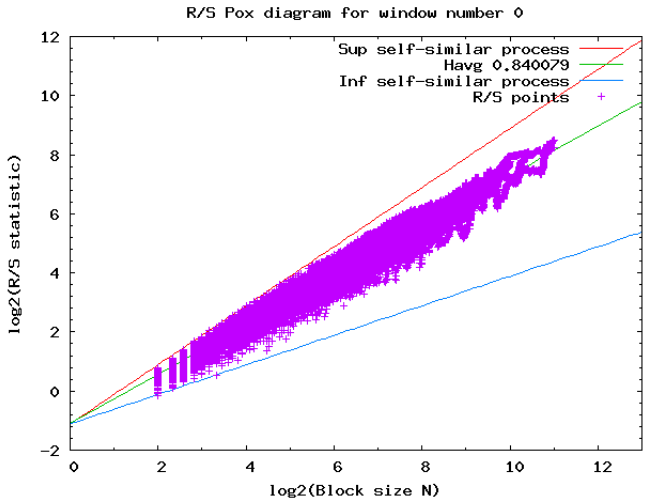

Figura 1.1: Gráfica del estadístico R/S

1.2.2. Gráficas varianza-tiempo

El método de gráficas varianza-tiempo, abreviado en este documento como *Var-Tpo*, descrito completamente en [HZ97], es un método que utiliza el gráfico de $\log(Var(X_k^{(m)}))$ contra $\log(m)$ y luego el método de mínimos cuadrados con los puntos resultantes para obtener una pendiente que es el estimador β. El estimador β se convierte en un estimador del parámetro de Hurst mediante la fórmula $H = 1 - \frac{\beta}{2}$.

La varianza de la serie agregada se obtiene de la siguiente forma. Para una serie de tiempo X_k de tama no N con $2 \leq m \leq \frac{N}{2}$ y un q suficientemente grande de subseries de $\{X_k^{(m)} : k = 1, ..., q\}$ obtenemos la media de todas las subseries mediante la ecuación (1.23) y calculamos la varianza de cada m particular mediante la ecuación (1.24).

$$\overline{X} = q^{-1} \sum_{k=1}^{q} X_k^{(m)} \tag{1.23}$$

$$Var(X_k^{(m)}) = (q-1)^{-1} \sum_{k=1}^{q} [X_k^{(m)} - \overline{X}]^2 \tag{1.24}$$

La figura 1.2 muestra un ejemplo de la estimación del parámetro de Hurst utilizando el método de gráficas de varianza-tiempo. En esta figura se muestran los puntos resultantes graficados sobre ejes $\log - \log$ La pendiente resultante debe encontrarse entre las pendientes $-1 < \beta < 0$, las cuales se muestran en la figura.

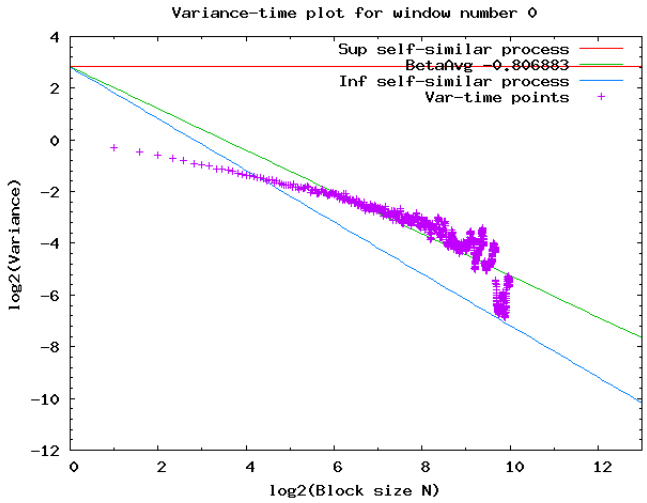

Figura 1.2: Gráfica de varianza-tiempo

1.2.3. Varianza modificada de Allan

El método de la varianza modificada de Allan, denominado *MAVAR* por su nombre en Inglés *Modified Allan Variance*, es un método descrito en [BP05] que demostró tener resultados muy positivos al ser usado para la estimación del parámetro Hurst.

Para una serie de tiempo X_k de tama no N, para $4 < n \leq \lfloor N/3 \rfloor$ se calcula la ecuación principal (1.25) con la ecuación (1.26). Los puntos resultantes son graficados en una gráfica $\log - \log$ y con los puntos se utiliza el método de mínimos cuadrados. Lo que se obtiene es una pendiente que es el estimador μ para el parámetro de Hurst. μ está relacionado con

el parámetro de Hurst mediante la fórmula $H = \frac{\mu}{2} + 2$. La ecuación (1.26) se utiliza para minimizar el número de cálculos obteniendo un tiempo de corrida $\sim O(N^2)$ [Bre02].

$$\text{Mod } \sigma_y^2(nc) = \frac{1}{2n^4c^2(N-3n+1)} \sum_{j=1}^{N-3n+1} A_j^2(n) \tag{1.25}$$

$$\begin{aligned} A_1(n) &= \sum_{i=1}^{n} (X_{2n+i} - 2X_{n+i} + X_i) \\ A_{j+1}(n) &= A_j(n) + (X_{3n+j} - 3X_{2n+j} + 3X_{n+j} - X_j) \end{aligned} \tag{1.26}$$

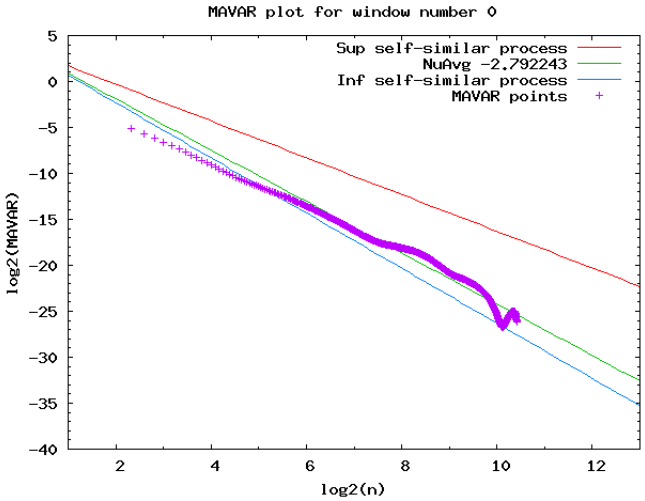

Figura 1.3: Gráfica de varianza modificada de Allan

La figura 1.3 muestra un ejemplo de la estimación del parámetro de Hurst utilizando el método de varianza modificada de Allan. En la figura se muestran los puntos resultantes de calcular la ecuación (1.25) y graficar los puntos sobre ejes $\log - \log$. La pendiente resultante, el estimador μ, debe encontrarse entre las pendientes $-3 < \mu < -2$, las cuales se muestran en la figura.

1.3. Problemas con la estimación del parámetro de Hurst

Normalmente lo que se intenta hacer es buscar en los datos características propias de procesos autosimilares o procesos autosimilares con dependencia de largo alcance. Debido a que las pruebas para verificar estas características dependen de valores como el tama no de la muestra, el intervalo de medición e incluso el método de estimación utilizado, es más razonable hablar de una estructura autosimilar para una escala y rango particular [KFR02] [MVN97].

Capítulo 2

Planteamiento del problema

AQUÍ VA EL DESARROLLO DEL PLANTEAMIENTO DEL PROBLEMA.

Para verificar si los métodos para detección de ataques de denegación de servicio mediante el uso del parámetro de Hurst, descritos en la sección son utilizables en tiempo real necesitamos implementarlos e incluirlos en una herramienta que además maneje las trazas y sea capaz de graficar los resultados.

Los requerimientos funcionales y no funcionales para la creación del programa, son descritos en las sección 2.1.

2.1. Requerimientos

Los requerimientos de la herramienta o software a implementar se mencionan a continuación:

- Por posibles cuestiones legales, la implementación de la solución debe ser de código libre para poder, luego de su validación, ser modificados y los módulos e incluidos en el AIR-NMS del laboratorio Kinoshita de la Universidad de Tohoku.

- La herramienta de línea de comando debe desarrollarse en el lenguaje `C`, ya que se quiere utilizar la librería `libpcap` para manipular las trazas de red como fuese necesario.

- Con el uso de la libería `libpcap` se debe extraer todo la información posible sobre los tiempos de llegada, cantidad y tama no de los paquetes de distintos protocolos que componen TCP/IP en la traza que alimenta el programa.

- Debido a que algunos métodos de estimación del parámetro de Hurst utilizan gráficas, la herramienta debe tener capacidades gráficas. El software debe también poder graficar el cambio del parámetro en el tiempo cuando se usa el mecanismo de ventanas deslizantes, y la serie de tiempo creada a partir de la velocidad de captura (c).

- El programa debe ser flexible. Esto incluye el hacer todos los aspectos importantes de la estimación parametrizables, tales como la velocidad de captura (c), el tama no de la ventana (w) y el tama no de la ventana deslizante (s).

- El programa debe incluir al menos 3 métodos para la estimación del parámetro de Hurst. Dichos métodos fueron escogidos desde un principio por el laboratorio Kinoshita y los criterios de selección se exponen a continuación. Ellos ya tenían una implementación del estadístico R/S y querían seguir teniendo este método como alternativa. Debido a los artículos [WY08] y [XLLH04] dónde se hace un análisis exhaustivo de la estimación del parámetro de Hurst mediante las gráficas varianza-tiempo y su comportamiento en ataques de denegación de servicio era también razonable su implementación. Por último, el método de la varianza modificada de Allan fue seleccionado ya que hoy en día es uno de los métodos más novedosos utilizados para la estimación del parámetro de Hurst y se ha demostrado que es bastante preciso, especialmente con muestras peque nas [BP05].

- Por tratarse de un prototipo, todas sus funciones serían para el análisis de forma *offline*. Los resultados darían una idea del comportamiento en un ambiente real.

Capítulo 3

Resultados

AQUI VA EL CONTENIDO DE RESULTADOS.

Los resultados que se quieren evaluar están divididos en tres partes: Primero, es necesario verificar la calidad de estimación del parámetro de Hurst usando los métodos implementados, segundo, es importante medir el tiempo necesario para cada estimación y, por último, es importante comprobar si se pueden detectar ataques de denegación de servicio mediante la variación del parámetro y el uso del mecanismo de ventanas deslizantes.

La descripción del computador utilizado para ejecutar las pruebas se presenta en la sección 3.1. Los archivos utilizados y los resultados de la verificación de la estimación del parámetro de Hurst y del tiempo necesario para la estimación se presentan en la sección 3.2. Estos resultandos muestran qué tan preciso es la implementación de los métodos y cuánto tiempo en promedio necesita la herramienta para estimar el valor del parámetro de Hurst en una ventana. La metodología para la evaluación de la detección de ataques de denegación se muestra en la sección. Por otro lado en la sección se muestran los resultados de utilizar hilos de ejecución para observar el cambio del parámetro de Hurst en el tiempo en modo *offline* con trazas producidas por `tcpdump`.

3.1. Máquina de prueba

Todas las pruebas fueron realizadas con el computador cuyas características aparecen en el cuadro 3.1. La herramienta `d2Hgr` fue compilada sobre esa máquina, exclusivamente con optimización gcc de nivel 2, que optimiza el binario resultante de la compilación con respecto a bloques repetitivos y cálculos punto flotante.

CPU	Althon 64 X2 4200+
RAM	4 GB
Distribución	Slackware 13.0 x86_64
Kernel Linux	2.6.29.6
Versión gcc	4.3.3

Cuadro 3.1: Computador usado para las pruebas

3.2. Validación de los estimados

Se tenía que probar que las implementaciones de los 3 métodos producían estimaciones razonables del parámetro de Hurst para saber de antemano que tan confiable podrian ser la estimaciones cuando se usasen junto con el mecanismo de ventanas delizantes. Para la validación de los métodos implementados se probaron los mismos con datos producidos por el algoritmo de Paxson [Pax95].

El algoritmo de Paxson permite crear una serie de tiempo pseudo-aleatoria en el dominio del tiempo de forma rápida, y permite especificar el parámetro de Hurst a ser simulado y el número de datos que debe tener la serie de tiempo resultante, manteniendo la media en 0 y la varianza en 1, con valores uniformemente distribuidos en el intervalo $[0, 2\pi]$. Mediante el algoritmo se puede simular un proceso estacionario autosimilar con dependencia de largo alcance [Pax95].

La implementación del algoritmo de Paxson utilizada es la que viene en el paquete `fArma`

de R. La validación contó con 6 tama nos diferentes para la serie de tiempo X_k generada por el algoritmo de Paxson. Estos tama nos fueron $\{1024, 2048, 4096, 8192, 16384, 32768\}$. Por cada tama no se generaron 10 series de tiempo X_k para cada uno de los 9 valores del parámetro de Hurst que se quería simular. Estos valores fueron $\{0{,}55, 0{,}60, 0{,}65, 0{,}70, 0{,}75, 0{,}80, 0{,}85, 0{,}90, 0{,}95\}$.

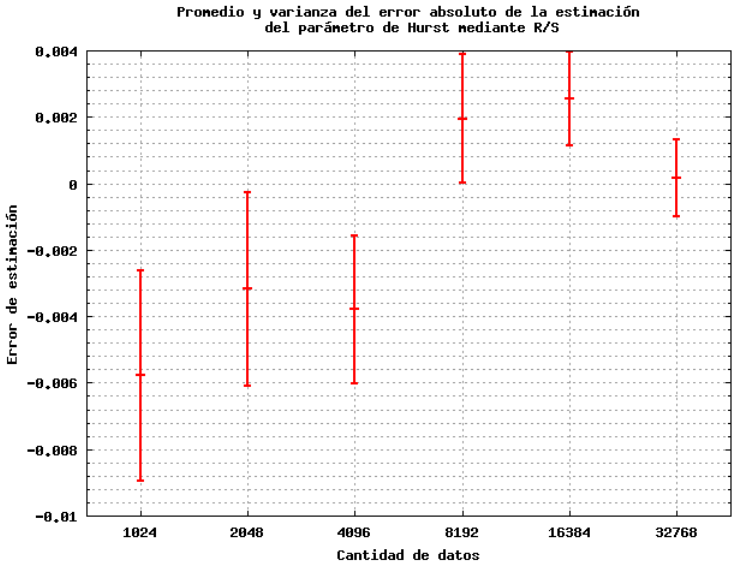

Figura 3.1: Promedio y varianza del error absoluto de la estimación del parámetro de Hurst para el método estadístico R/S

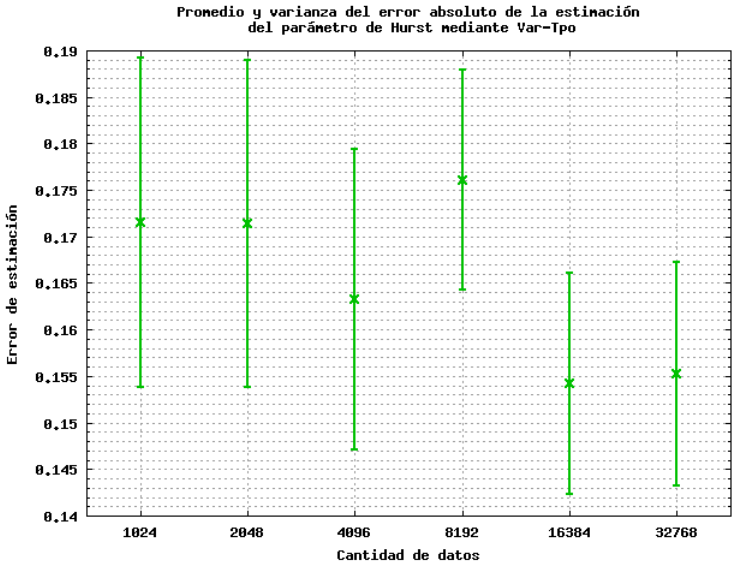

Figura 3.2: Promedio y varianza del error absoluto de la estimación del parámetro de Hurst para el método de gráficas de varianza-tiempo

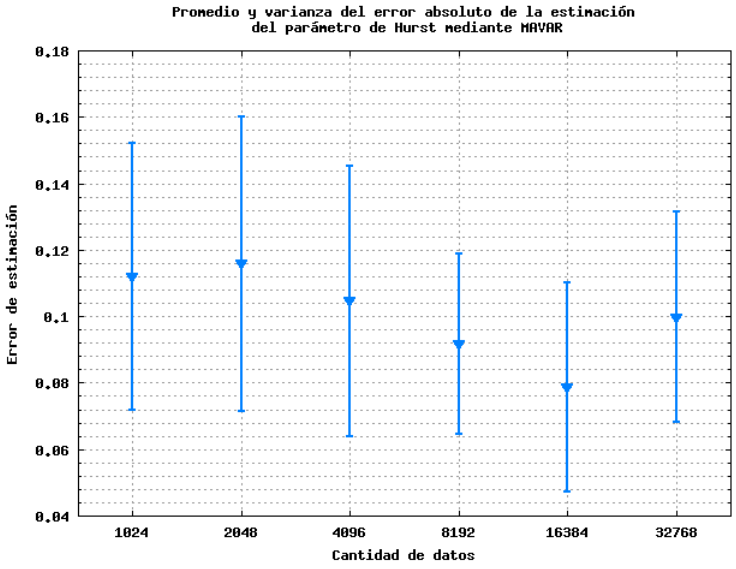

Figura 3.3: Promedio y varianza del error absoluto de la estimación del parámetro de Hurst para el método de varianza modificada de Allan

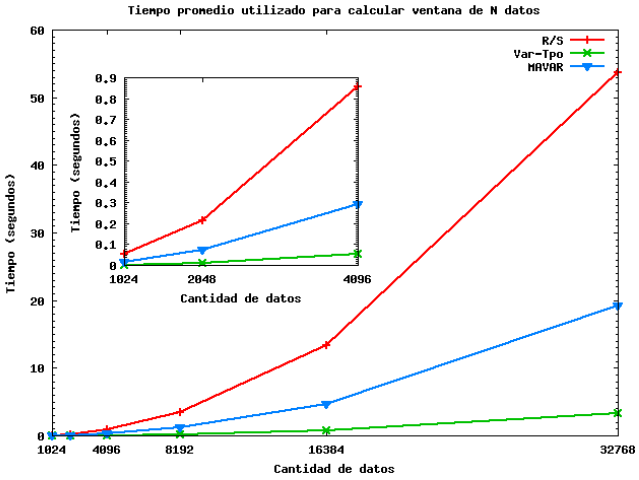

Figura 3.4: Tiempo promedio utilizado para estimar una ventana de 1024 a 32768 datos para los métodos implementados

Observando las figuras 3.1, 3.2 y 3.3, lo que se puede evidenciar es que, en general, el promedio del error absoluto de las estimaciones del parámetro de Hurst de los métodos mejoran y la varianza disminuye cuando se toman en cuenta un mayor número de datos para la ventana de estimación. Sin embargo, para los dos métodos que dependen de cálculos con la varianza el promedio del error absoluto de la estimación, incluso con el mayor número de datos es mayor a 0,1. El método del estadístico R/S es el método que se comporta mejora ya que tiene, en general, un error absoluto muy cerca a 0 con una varianza mínima.

De acuerdo a [WY08] si queremos analizar los datos en tiempo real para una red se tendría que hacer una estimación como mínimo cada segundo, por lo que una estimación no puede tardar más de un segundo en realizarse. Esto es seriamente limitante ya que cada método tiene un tiempo de ejecución muy diferente. Observando la figura 3.4, con esta restricción la única posibilidad que se tiene es limitar al método del estadístico R/S a 4096 datos, al método de gráficas varianza-tiempo a 16834 datos y al método de varianza modificada de Allan a 4096 datos. Dado que para comparar los métodos es importante usar la misma cantidad de datos, decidimos trabajar con 4096 datos. El limitar el número de datos va a afectar la estimación

del parámetro de Hurst directamente como se muestra en las figuras 3.1, 3.2 y 3.3.

A pesar de estos problemas, seguimos adelante con el proyecto porque no queremos una sola estimación puntual y precisa de la traza, sino detectar su variación, por lo que optamos por un buen balance entre el tiempo de ejecución y el número de datos a utilizar [WY08].

Capítulo 4

Conclusiones y recomendaciones

AQUÍ VAN LAS CONCLUSIONES Y RECOMENDACIONES.

Como resultado de este proyecto se logró implementar una herramienta de línea de comando que pudiese crear series de tiempo a partir de trazas producidas por `tcpdump` para estimar el parámetro de Hurst y graficar sus resultados. La estimación se hizo mediante el mecanismo de ventana deslizante, limitando el tama no de datos para cada ventana de forma de probar si de esta forma, utilizando las técnicas descritas ,se podían obtener estimaciones del parámetro de Hurst en tiempo real que permitieran detectar ataques de denegación de servicio.

Los resultados muestran que aunque la herramienta implementada logra crear una serie de tiempo que puede estimar el parámetro de Hurst con un error acceptable y puede graficar sus resultados, su detección de ataques de denegación de servicio en tiempo real es mejorable. Si bien se logra detectar cambios drásticos, en algunos casos cerca de la región del ataque, la cantidad de falsos positivos y errores hace su uso casi imposible.

Para seguir trabajando en la detección de ataques de denegación de servicio usando el parámetro de Hurst se presentan las siguientes recomendaciones:

- Emplear algún mecanismo de muestreo de los valores de n utilizados en la estimación del parámetro de Hurst en una ventana con todos los métodos para mejorar la precisión y posiblemente disminuir el tiempo necesario para una estimación:

  En la herramienta implementada, no se usó ningún mecanismo de muestreo en la estimación del parámetro de Hurst, implementando los métodos como fueron escritos en el capítulo 1. Esto significa que se tomaron en cuenta todos los tama nos n viables para la creación de las gráficas $\log - \log$. Parte de la lentitud de los métodos en dar un resultado está en tomar en cuenta todos los n posibles. Por otro lado, de acuerdo con [BP05] usando el método de varianza modificada de Allan, el tomar todos los n acercándose a $N/3$ comienza a afectar seriamente la estimación. Sin embargo, en el artículo no se presenta una forma determinística que describa hasta que punto se debería tomar en cuenta los n para una estimación.

- Encontrar algún método de utilizar la información obtenida en la estimación de una ventana de tal forma que los cálculos hechos para la primera ventana aceleren la estimación de la siguente ventana [WY08].

- Al poder estimar una ventana en menor tiempo también sería recomendable aumentar el número de datos utilizados para una estimación. Cómo no se quiere aumentar el tiempo de traza real w usado en una estimación, esto implicaría usar un c más peque no [HDTI00].

- En la estimación de una ventana, si se logra encontrar una forma de dividir el cálculo de tal forma de poder aplicar una paralelización, el uso de más de un hilo de ejecución para estimación de la ventana podría ser probado [HDTI00].

- En lugar de utilizar un único tama no de s en el mecanismo de ventana deslizante, se podría usar un valor flexible a los cambios de la red, de tal forma que si el parámetro

de Hurst cambia repentinamente entre dos ventanas, se acorte s con tal de verificar si en realidad hay una sucesión de paquetes uniformemente espaciados que puedan ser parte un ataque de denegación de servicio, minimizando la cantidad de falsos positivos [WY08].

Bibliografía

[AAK04] Sameera Abar, Toru Abe, and Tetsuo Kinoshita. A next generation knowledge management system architecture. In *AINA (2)*, pages 191–195, 2004.

[BKM96] Sabyasachi Basu, Steve Klivansky, and Amarnath Mukherjee. Time series models for internet traffic, 1996.

[BP05] Stefano Bregni and Luca Primerano. Using the modified allan variance for accurate estimation of the hurst parameter of long-range dependent traffic. *CoRR*, abs/cs/0510006, 2005.

[Bre02] Stefano Bregni. *Synchronization of Digital Telecommunications Networks*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[CB96] Mark E. Crovella and Azer Bestavros. Self-similarity in world wide web traffic evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5:835–846, 1996.

[DDHT08] R. Dobrescu, M. Dobrescu, D. Hossu, and S. Taralunga. Using internet traffic self-similarity for detection of network anomalies. In *11th International Conference on Optimization of Electrical and Electronic Equipment, 2008. OPTIM 2008*, pages 81–86, 2008.

[FP01] Sally Floyd and Vern Paxson. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 9:392–403, 2001.

[Gon03] Fengmin Gong. Deciphering detection techniques: Part 3 denial of service detection. *White paper, McAfee Network Security Technologies Group and Network Associates*, 2003.

[HDTI00] T. Hagiwara, H. Doi, Hideki Tode, and Hiromasa Ikeda. High-speed calculation method of the hurst parameter based on real traffic. In *LCN*, pages 662–669, 2000.

[HZ97] Oliver Yang H.F. Zhang, Y.T. Shu. Estimation of hurst parameter by variance-time plots. *IEEE*, 0-7803-3905-3, 1997.

[ITU+07] Ryuji Igarashi, Akinori Takahashi, Hiroshi Ueda, Yutaka Nasuno, Yukio Iwaya, Masato Sakata, and Tetsuo Kinoshita. A proposal for real time hurst parameter derivation. *IEEJ Transactions on Electronics, Information and Systems*, 127(6):968–969, 2007.

[JMR04] David Dittrich Jelena Mirkovic, Sven Dietrich and Peter Reiher. *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice Hall PTR, 2004.

[KAIK07] Susumu Konno, Sameera Abar, Yukio Iwaya, and Tetsuo Kinoshita. Effectiveness of autonomous network monitoring based on intelligent-agent-mediated status information. In *IEA/AIE*, pages 1078–1087, 2007.

[KFR02] Thomas Karagiannis, Michalis Faloutsos, and Rudolf H. Riedi. Long-range dependence: Now you see it, now you don't! In *In IEEE GLOBECOM, Global Internet Symposium*, page 2002, 2002.

[KIAK04] Susumu Konno, Yukio Iwaya, Toru Abe, and Tetsuo Kinoshita. Design of network management support system based on active information resource. *Advanced Information Networking and Applications, International Conference on*, 1:102, 2004.

[Kle76] L. Kleinrock. *Queueing Systems, Volume 2: Computer Applications*. Wiley, 1976.

[Li06] Ming Li. Change trend of averaged hurst parameter of traffic under ddos flood attacks. *Computers & Security*, 25(3):213–220, 2006.

[LTWW93] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of ethernet traffic, 1993.

[LW91] Will E. Leland and Daniel V. Wilson. High time-resolution measurement and analysis of lan traffic: Implications for lan interconnection. In *INFOCOM*, pages 1360–1366, 1991.

[Man82] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman, August 1982.

[MM02] Richard Mortier and C Richard Mortier. Internet traffic engineering, 2002.

[MVN97] Sándor Molnár, Attila Vidács, and Arne A. Nilsson. Bottlenecks on the way towards fractal characterization of network traffic: Estimation and interpretation of the hurst parameter, 1997.

[MVS01] David Moore, Geoffrey Voelker, and Stefan Savage. Inferring internet denial-of-service activity. In *In Proceedings of the 10th Usenix Security Symposium*, pages 9–22, 2001.

[Odl01] Andrew Odlyzko. Internet growth: Myth and reality, use and abuse. *Journal of Computer Resource Management*, 102:23–27, 2001.

[Pax95] Vern Paxson. Fast approximation of self-similar network traffic. Technical report, Lawrence Berkeley Laboratory and EECS Division, University of California, 1995.

[PF95] Vern Paxson and Sally Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3:226–244, 1995.

[PW00] Kihong Park and Walter Willinger. *Self-Similar Network Traffic and Performance Evaluation*. John Wiley & Sons, Inc., New York, NY, USA, 2000.

[SSO07] Oleg Sheluhin, Sergey Smolskiy, and Andrew Osin. *Self-Similar Processes in Telecommunications*. John Wiley & Sons, Inc., New York, NY, USA, 2007.

[SV01] Biplab Sikdar and Kenneth S. Vastola. The effect of tcp on the self-similarity of network traffic. In *Proc. of the 35th Conf. on Information Sciences and Systems*, pages 21–23, 2001.

[TIU+13] Akinori Takahashi, Ryuji Igarashi, Hiroshi Ueda, Yutaka Nasuno, Yukio Iwaya, and Tetsuo Kinoshita. Online network observation to detect traffic anomaly. *IEICE technical report*, 107(221):57–62, 20070913.

[USKS02] Takahiro Uchiya, Takuo Suganuma, Tetsuo Kinoshita, and Norio Shiratori. An architecture of active agent repository for dynamic networking. In *AAMAS '02: Proceedings of the first international joint conference on autonomous agents and multiagent systems*, pages 1266–1267, New York, NY, USA, 2002. ACM.

[WP98] Walter Willinger and Vern Paxson. Where mathematics meets the internet. *Notices of the American Mathematical Society*, 45:961–970, 1998.

[WWT03] Rolf H. Riedi Walter Willinger, Vern Paxson and Murad S. Taqqu. Long-Range Dependence and Data Network Traffic. In G. Oppenheim P. Doukhan and M. Taqqu, editors, *Theory and Applications of Long-Range Dependence*, pages 373–407. Birkäuser, Berlin, 2003.

[WY08] Jiangtao Wang and Geng Yang. An intelligent method for real-time detection of ddos attack based on fuzzy logic. *Journal of Electronics (China)*, 25(4):511–518, 2008.

[XLLH04] Y. Xiang, Y. Lin, W.L. Lei, and S.J. Huang. Detecting ddos attack based on network self-similarity. *IEE Proceedings - Communications*, 151(3):292–295, 2004.

Apéndice A

Información adicional de `d2Hgr`

AQUÍ VA EL CONTENIDO DE LOS APÉNDICES.

Esta parte del apéndice contiene el manual del usuario de la herramienta implementada, la descripción de sus archivos y el código fuente de algunos módulos de posible interés.

A.1. Requerimientos de *software* y *hardware*

El programa resultante debería poder ser utilizado sobre cualquier maquina con más de 32 MB libres de RAM. Aunque no hay limitaciones para el procesador a usar, entre más nuevo el procesador y mayor número de núcleos tenga, más rápido hará los calculos la herramienta. En el caso de la memoria, entre más tenga disponible la herramienta, mayor cantidad de datos podrá utilizar en los cálculos.

- Para su compilación, el programa requiere una versión actualizada de gcc, el compilador `C` GNU. Se ha utilizado varias versiones para su compilación por lo cualquier versión mayor a la 4,2 debería funcionar.
- La libería `libpcap` es necesaria para darle las funcionalidades de manipulación de trazas

`tcpdump` al programa. A partir de la versión 0,8 se encuentran todas las funciones que utiliza el programa.

- La compilación de la herramienta se hace más fácil con `make`, programa GNU. Cualquier versión mayor a 3,6 no debería dar problemas.

- El programa `gnuplot` es indispensable para poder graficar los resultados creados por el programa. La versión más utilizada durante su desarrollo fue la 4,2 por lo que es la que se recomienda.

- La graficación sólo se puede hacer sobre un sistema operativo bajo el estándar POSIX[1] con la herramienta gnuplot instalada. Esto se debe a una necesidad de la interfaz gnuplot en ANSI C que utiliza un "pipe" tipo POSIX para comunicarse directamente con el programa gnuplot instalado en la máquina. Sin embargo, dentro de las opciones del programa se puede pasar la información obtenida en la estimación a archivos de texto para su posterior análisis.

- La librería pthreads da las funciones necesarias para utilizar hilos de ejecución en el programa.

A.2. Ayuda de la línea de comando

La herramienta toma como parámetro principal un archivo `tcpdump` o un CSV de una serie de tiempo pseudo-aleatoria generada con la herramienta R. Aparte se puede escoger una velocidad de captura, una ventana, una ventana deslizante, filtrar paquetes, delimitar corridas, correr con varios hilos de ejecución, además de estimar el parámetro de Hurst con

[1] "Portable Operating System Interface [for Unix]"son una familia de estándares de llamadas al sistema operativo definidos por la IEEE y especificados formalmente en el IEEE 1003. La gran mayoría de las distribuciones GNU/Linux siguen los estándares aunque no están oficialmente certificados.

los tres métodos mencionados. Se puede también obtener el cambio del parámetro de Hurst en el tiempo o graficar la estimación de una ventana en particular. La ayuda de la línea de comando se muestra abajo:

```
Usage: ./d2Hgr -[f|g] file -[i|nrlvxmu] [OPTIONS]

Estimate and graph Hurst parameter calculations from a file.

   -f file   Specifies a tcpdump dump file.
   -g file   Specifies a CSV file with a simulated network traffic stream.
             This option cannot be used with the time based or tcpdump
             options since the stream is simulated and is static in it's
             definition.
   -i        Prints only the protocol statistic information available from the
             tcpdump file.
   -n        Tells the program if you wish to graph the packets per delta time
             graph. Affected by -p, -o, -d, -c, -b -e, -y flags.
   -r        Tells the program to calculate the Hurst parameter changes over
             time by use of the R/S statistic. Affected by -p, -o, -d, -c, -w,
             -s, -j, -t, -b, -e, -y flags.
   -l num    Tells the program to graph the Pox diagram that creates the Hurst
             data point for a given window number between 0 and Datapoints
             designated by num. Affected by -p, -o, -d, -c, -w, -s, -j, -b,
             -e, -y flags.
   -v        Tells the program to calculate the Hurst parameter changes over
```

```
            time by using the Variance-time plot technique. Affected by -p,
            -o, -d, -w, -s, -j, -t, -b, -e, -y flags.
  -x num    Tells the program to graph the Variance-time plot that estimates
            the Hurst parameter for a given window number designated by num.
            Affected by -p, -o, -d, -c, -w, -s, -j, -b, -e, -y flags.
  -m        Tells the program to calculate the Hurst parameter changes over
            time by use of the Modified Allan Variance. Affected by -p, -o,
            -d, -c, -w, -s, -j, -t, -b, -e, -y flags.
  -u num    Tells the program to graph the Modified Allan Variance that
            estimates the Hurst parameter for a given window number designated
            by num. Affected by -p, -o, -d, -c, -w, -s, -j, -t, -b, -e, -y
            flags.

OPTIONS:
  -p proto  Specifies which protocol to be taken into account when doing
            calculations. Default is all.
            Implemented protocols are:
             ip tcp udp icmp sctp ftp ssh telnet smtp dns dhcp http pop3 ntp
             imap snmp ldap https smtps ldaps imaps pop3s nfs squid
  -o dir    Output directory for results. Default is current directory.
  -d        Tells the program if the results should be placed in text files
            for later use. This option doesn't graph. Useful for using this
            program where gnuplot is not available.
  -y        Tells the program to graph and print the resulting data files.
  -a        Tells the program not to graph or create data files.
```

```
-c sec    Specify a packet capture speed. This makes the program not look
          for one. Example: 0.01 = 0.01 seconds.
-w sec    Window time size. This makes the program not look for one.
          Example: 60 = 60 seconds.
-s sec    Slide time size. This makes the program not look for one.
          Example: 1 = 1 seconds.
-j num    Tells the program to use log base num for all the calculations of
          blocks for the R/S statistic, Variance-time plot and Modified Allan
          Variance. Default is 2.
-t num    Does Hurst calculation using threads. If 0 is placed, then the
          default number of threads (4) is used.
-b sec    Tells the program from which second in the time data to begin the
          calculation. Default is 0s.
-e sec    Tells the program how many seconds after the beginning point to
          include in the calculation. Default is the whole tcpdump time.
-k num    Average Hurst parameter change with which to try to detect attacks.
-K num    Standard deviation of Hurst parameter change with which to detect
          attacks.
-q num    Average Hurst parameter value with which to try to detect attacks.
-Q num    Standard deviation of Hurst parameter value with which to detect
          attacks.
-h        Print this help.
```

A.3. Archivos del programa

El programa de C llamado `d2Hgr`, consiste de 18 archivos de código fuente que incluye 9 archivos de cabecera. Los archivos contienen la siguiente información:

- **config.h**: Cabecera de funciones para parsear las opciones de linea de comando.
- **config.c**: Implementación de las funciones que parsean las opciones de la linea de comando.
- **d2Hgr.h**: Archivo que contiene las definiciones de las funciones para leer los archivos producidos por `tcpdump`.
- **d2Hgr.c**: Implementación de las funciones para leer la información del archivo producido por `tcpdump` y poder obtener la información necesaria para su posterior análisis.
- **externvars.h**: Archivo que contiene todas las estructuras especiales para el uso del programa.
- **filter.h**: Archivo que contiene la definición de las funciones para construir la expresión de filtro de libpcap para usar durante la corrida del programa.
- **filter.c**: Archivo que contiene la implementación de todas las funciones de filter.h.
- **flows.h**: Cabecera de funciones para contabilizar los flujos de IPv4.
- **flows.c**: Archivo que contiene la definición de las funciones para contabilizar los flujos de IPv4.
- **gnuplot_i.h**: Cabecera para el archivo gnuplot_i.c que define las funciones de la interfaz gnuplot para que el programa pueda graficar los resultados. Esta es una versión modificada de la interfaz ANSI C de N. Devillard para gnuplot.

- **gnuplot_i.c**: Implementación de las funciones de N. Devillard para su interfaz de ANSI C con gnuplot.
- **graph.h**: Archivo que contiene la definición de las funciones para graficar los resultados.
- **graph.c**: Archivo que contiene la implementación de las funciones para graficar los resultados.
- **hurst.h**: Archivo que contiene las definiciones de los métodos para aproximar el parámetro de Hurst.
- **hurst.c**: Archivos que contiene las implementaciones de los métodos para aproximar el parámetro de Hurst y los métodos para extraer la información sobre los paquetes una vez leidos por las funciones de d2Hgr.h.
- **main.c**: El archivo que contiene el main del programa.
- **detect.h**: El archivo que contiene las definiciones de los métodos para la detección de ataques de denegación de servicio.
- **detect.c**: El archivo que contiene las implementaciones de los métodos para la detección de ataques de denegación de servicio.

A.4. Creación de *Xtdata*

```
1 // Reset j for next cycle.
2 float j = 0.0;
3 // Counter to know in which delta_time section we are on the xtdata
4 // array.
5 int arrcount = 0;
6 // Temporal storage for the number of packets seen.
```

```
7  int numpktseen = 0;
8
9  // While the added time is less than the total time and we haven't
10 // run out of packets, run this cycle to see how many packets we see
11 // per delta_time section.
12 while (arrcount < datapoints && timecnt < packets &&
13         j <= time_int ) {
14
15     // Add the next timedata to j and pass to the next packet time
16     // diff.
17     j += timedata[timecnt++];
18
19     // We are seeing one new packet, so add it to our temporal
20     // storage.
21     numpktseen++;
22
23     if (j >= (arrcount+1)*(delta_time)) {
24         // If we've surpassed one of the delta_times, fill the
25         // information for the delta_time section.
26         xtdata[arrcount++] = numpktseen -1;
27
28         // Boolean variables
29         short int passed = 0;
30         short int carry = 1;
31
32         // Since we're not sure if the packet jumps through too
33         // many sections, we check for it, filling each
34         // sectioned passed with 0, except the first one which has
35         // at least one.
36         while (arrcount < datapoints &&
37                 j > (arrcount)*(delta_time)) {
```

```
38              if (passed == 0) {
39                  xtdata[arrcount++] = 1;
40                  passed = 1;
41                  carry = 0;
42              } else {
43                  xtdata[arrcount++] = 0;
44              }
45          }
46
47          // Finally we reset the temporal storage for the next cycle.
48          if (carry == 1) {
49              numpktseen = 1;
50          } else {
51              numpktseen = 0;
52          }
53      }
54  }
```

Código fuente A.1: Creación de *Xtdata*