Facultad de Informática
Universidad Nacional de La Plata

-

Modelización de tráfico en redes IP utilizando los "Polinomios Potenciales de Grado uno en cada variable". Estudio del modelo y Aplicaciones

-

Tesis presentada para obtener el grado de

Magíster en Redes de Datos

Director: Luis Marrone

Co-director: Jorge Nanclares

Autor: Ulises M. A. Rapallini

Agosto 2010

Índice general

Índice de figuras

Resumen

El **Potential Polynomial of Degree 1 in Each Variable** o **P1P** es una teoría reciente que resuelve un problema variacional complejo, los P1P se utilizaron para análisis de datos masivos en las áreas de Aprendizaje en Maquinas y Dataminig con resultados positivos, la tesis presenta el uso de los P1P en el modelado tráfico en redes IP, detalla la teoría de los P1P con un enfoque en redes para su posterior aplicación. Los dos modelos construidos, uno en el nivel de red y trasporte, y dos en el nivel de aplicación, demuestran que los P1P son una alternativa de modelado con la propiedad de ajustase a las características de tráfico particular. La tesis proporciona métodos de aplicación y construcción de P1P's, también brinda una forma de reducir la cantidad de variables para modelos complejos.

El metamodelo y el estudio de los P1P con la teoría de la información son el inicio de nuevos trabajos de investigación para el uso en aplicaciones y para la propia teoría de P1P.

La evaluación de la representatividad del modelo matemático de un sistema en relación a la realidad es un aspecto importante para obtener buenas estimaciones, realizamos una comparación entre un P1P y la Teoría de Colas para modelar tráfico de correo electrónico.

Organización de la tesis

La tesis está organizada en partes estructurando al documento para una mejor lectura, comprensión y utilización. Se puede obviar la parte **Fundamentos Teóricos** si el lector esta familiarizado con la Teoría de colas y la teoría de la información, sin embargo el capítulo **Polinomios Potenciales de Grado Uno en cada variable** trata la reciente teoría de los P1P y es recomendable su lectura.

El documento esta estructurado de la siguiente forma:

1. INTRODUCCIÓN: La introducción expone las motivaciones que llevaron al uso de los P1P para el estudio de tráfico IP, el estado del arte, la importancia de los P1P como nueva herramienta alternativa para el estudio de tráfico en Redes y describe el largo camino que posiblemente tiene el análisis de tráfico.

2. FUNDAMENTOS TEÓRICOS: Esta parte presenta los capítulos:

   a) TEORIAS: Éste capítulo es un resumen de la Teoría de colas y la Teoría de la información, para las dos Teorías se describe lo necesario utilizado en la tesis.

   b) POLINOMIO POTENCIAL DE GRADO UNO EN CADA VARIABLE: Este capítulo describe la teoría de los P1P's con un enfoque en las problemáticas de redes.

3. APLICACIÓN DE P1P AL MODELADO DE TRÁFICO: éste capitulo tiene una introducción sobre como utilizar los P1P para modelar tráfico, dos modelos de tráfico en distintos niveles TCP/IP y un metamodelo P1P.

4. ESTUDIO DEL MODELO P1P, COMPARACIONES: aquí estudiamos el modelo P1P con la teoría de la información intentando determinar las capacidades y generando disparadores para nuevos estudios. En la segunda parte modelamos tráfico SMTP con análisis de colas y P1P para realizar una comparación y evaluar aspectos de aplicación de los modelos.

5. CONCLUSIONES: detallamos las conclusiones de la tesis y los trabajos de investigación futuros.

6. APÉNDICES: se anexan los programas en la herramienta *Mathematica* y los programas de software desarrollados para la tesis.

Parte I

Introducción

Estado del arte y motivación

2.1. Medición y caracterización del tráfico

La medición y caracterización de tráfico en redes son aspectos claves para el modelado, en los últimos 20 años se realizaron importantes avances a través de la medición y análisis del volumen de tráfico en las redes, los requerimientos de los usuarios evolucionan y aparecen nuevos desafíos como: medidas y análisis de tráfico en entornos de Internet, análisis e implicancias de características de tráfico en LAN, modelos de tráfico en redes de alta velocidad, y otros. [9]. El uso de los P1P para modelar tráfico con el propósito de estudiar o analizar nuevos requerimientos depende exclusivamente de los datos asociados al requerimiento. Por ejemplo supongamos que en cierta red aparece un nuevo requerimiento nombrado *soporte para telepresencia*, el nuevo requerimiento implica analizar si los recursos existentes de ancho de banda, buffers, velocidad de tratamiento de los paquetes, el uso de la red, el tráfico existente, y otras variables, soportan la telepresencia; para generar un modelo P1P capturamos tráfico particular de telepresencia, identificamos las variables de interés, y calculamos los coeficientes del P1P. Para cualquier sistema los P1P intentarán determinar las características del tráfico capturado, esta propiedad los hace atractivos para encontrar modelos ajustados al entorno, se construyen a partir de datos observados y generan un modelo matemático específico del sistema tratado, la captura de tráfico se puede realizar en entornos Internet (por ejemplo en un Servidor de Correo, un servidor DNS), en una red LAN, en ambientes Inalámbricos, u otros.

2.2. Modelado y Estimación de tráfico

El modelado y la estimación de tráfico en redes son una herramienta que se utiliza para: análisis de performance, diseño de redes con requerimientos específicos, estimación de recursos, reserva de recursos, realización de ajustes, caracterización de tráfico, entre otras aplicaciones. Las características de "tráfico estadístico" traen un conjunto de problemas para modelar el tráfico de forma efectiva porque involucra variables de difícil determinación. La realización de un modelo adecuado necesita detectar las características de tráfico y estimar el tráfico a futuro basándose en el tráfico observado [3]. Los P1P tienen la propiedad de intentar determinar patrones en las muestras de datos estadísticos, en la teoría y en los casos de aplicación los P1P demostraron modelar razonablemente bien casos de análisis de datos masivos. Los datos de captura de paquetes IP tienen implícito características particulares (o patrones) de tráfico en la red, el uso de los P1P para la estimación o detección de características de tráfico IP es análogo al uso para la detección de patrones en datos. Los modelos P1P surgieron en un proyecto de investigación en el área de inteligencia artificial, el autor de la tesis publico como co-autor la teoría de los P1P en Octubre de 2007, inicialmente se uso para tratar problemas de aprendizaje en máquinas y posteriormente para análisis de información en bases de datos.

Algunos de los aspectos específicos afectados por un modelo de tráfico son: reserva de recursos de memoria y procesamiento, algoritmos de encolado de paquetes en routers, tamaño de las memorias intermedias (buffers), detección temprana de la congestión, y clasificación de tráfico para brindar calidad de servicio, también los usuarios de la red requieren poder administrar las prestaciones y decidir como y cuando usar el ancho de banda disponible. Estos entre otros factores muestran la importancia del modelado de tráfico y el uso de nuevas

metodologías. Para cada uno de estos problemas podemos identificar un sistema con variables, medir sus valores en el tiempo con captura de tráfico (Variables aleatorias de tiempo fijo $X(t)$) y construir un P1P.

2.3. Análisis de tráfico

Para finalizar este capitulo describimos algunos ejemplos de análisis de tráfico que demuestran la importancia de continuar con el estudio. El trabajo de Jeffrey R. Spirn [8] en 1982 mostró que hay fenómenos relacionados con el diseño de redes que no están bien manejados por los modelos de colas clásicos, su desempeño en las predicciones es optimista. Otro ejemplo es la publicación de Will E. Leland, Murad S. Taqqu, Walter Willinger, y Daniel V. Wilson, [10] en 1994 sobre auto-similitud de tráfico en redes Ethernet, mostró que los modelos clásicos no pueden advertir este comportamiento y tiene graves influencias para el control, análisis y diseño en redes de alta velocidad. Otras razones por las cuales el modelado de tráfico tiene relevancia son: el crecimiento indiscriminado de usuarios en la red Internet y la creciente cantidad de aplicaciones distribuidas que corren sobre redes LAN, WAN e Internet generando tráfico y utilizando una parte importante del ancho de banda disponible. La combinación de estos factores genera un ambiente propicio para medir e investigar el tráfico con diversas teorías, el análisis de tráfico con P1P es un método inédito que proporciona importantes características frente a otros como la adaptación a casos particulares, basado en la muestra de datos reales, enfoque sistémico, y fácil construcción.

El análisis de tráfico es un desafío en redes de datos porque son sistemas complejos, los modelos más precisos son los construidos a partir de paquetes por nivel, en redes de gran tamaño tienen como principal inconveniente los recursos computacionales requeridos y la comprensión de cómo los parámetros seleccionados afectan a la performance de la red [13]. Modelos recientes como el trabajo "Modeling Communication Networks With Hybrid Systems" en Junio 2007 [13] reducen la complejidad realizando aproximaciones continuas de variables (por ejemplo el tamaño de un buffer, el tamaño de la ventana de congestión TCP) , las aproximaciones continuas de variables permiten capturar la dinámica de los fenómenos transitorios, mientras que los modelos estadísticos que utilizan el promedio de variables discretas en cortos períodos de tiempo no pueden advertir estos fenómenos.

El análisis de series de tiempo y la autocorrelación (auto co varianza) son una importante herramienta para descubrir relaciones entre los datos analizados. El tráfico con tasa de bit variable como el video y el audio tienen características de auto correlación que son difíciles de modelar, una alternativa de modelado es la metodología TES [14] usada para obtener funciones de auto correlación en los datos. Uno de los problemas más importantes que presenta la teoría de colas aplicada al análisis de performance de redes es la determinación de la distribución de arribo, como respuesta este desafío aparecen diferentes métodos como QTES [15] que se puede utilizar para detectar la autocorrelación de los datos de entrada. La utilización de estas técnicas involucra complejidad. La utilización de los P1P implica la definición de un sistema, la determinación de las variables en juego, la medida de las variables y la construcción del modelo, el modelo construido se ajustará al sistema definido y proporcionara un modelo de menor entropía o de menor incertidumbre respecto del interior del sistema. Luego de entrenar el modelo P1P con los datos de entrada y salida, éste tiene la capacidad de estimar resultados con valores supuestos de variables para un periodo de tiempo t fijo, o generar un vector con valores supuestos de variables en los tiempos $t_1, t_2, t_3, ..., t_n$ y estimar los resultados $r_1, r_2, r_3, ..., r_n$ en un solo paso.

2.4. Modelos estadísticos de ajuste de curvas y modelos de tráfico

Uno de los problema clásicos de la ingeniería es observar un sistema cuya función interna es desconocida y necesitar obtener dicha función para el estudio del sistema. Existen técnicas para aproximar la función o intentar descubrirla analizando y relacionando los datos de entrada y las salidas generadas, luego de obtenida la aproximación se puede: suponer casos extremos, suponer valores de entradas extremos y tomar los valores de salida, analizar el comportamiento del sistema en esos casos críticos, determinar valores limites, reservar recursos para casos especiales y descartar la utilización de ese sistema para otros casos no aptos.

Podemos decir que el caso de estudio de tráfico es un problema de modelado de similares características al nombrado anteriormente, el proceso general consiste en analizar los datos de entrada y salida, armar un modelo y estudiar el sistema con el modelo obtenido.

Un problema más genérico donde se consideran valores de entrada y salida de cualquier tipo se trata en el área de aprendizaje en Maquinas, Aprendizaje Estadístico o tradicionalmente llamado métodos de ajuste, el problema de fundamento consiste en: dados un conjunto de valores cualesquiera, encontrar una curva o función que se ajuste lo mejor posible a todos los valores. Existe una amplia cantidad de métodos para tratar este problema, el lector puede encontrar en el libro [18] algunos métodos de aprendizaje con un enfoque estadístico y en el libro [17] donde se trata el aprendizaje estadístico y métodos de regresión.

En esta tesis utilizamos los P1P como método para modelar datos de tráfico en una red IP donde los valores son enteros positivos acotados y los sistemas parecen estar cerca de los sistemas lineales, consisten básicamente en obtener el valor de salida a partir de la suma de los valores de entrada.

Parte II

Fundamentos Teóricos

Teorías utilizadas en la tesis

El objetivo principal de esta parte es describir en detalle la teoría de los P1P y repasar los fundamentos teóricos utilizados en la tesis, el núcleo de la tesis es el modelado con P1P y los fundamentos de la Teoría de los P1P.

Tanto en la teoría de los P1P como en el estudio sobre los P1P se utilizan conceptos de la teoría de la información, es conveniente tener presentes algunos conceptos como *Información* y *Entropía*.

Las problemáticas de redes suelen estudiarse con la Teoría de colas, en la tesis la utilizamos para una aplicación de tráfico SMTP y comparar los resultados con un modelo P1P, por esto se realiza una breve descripción de la Teoría de colas abordando un sistema de colas simple.

Al modelar tráfico es importante determinar si el modelo obtenido conduce a resultados representativos de la realidad y si las teorías utilizadas para el modelado son adecuadas, por ejemplo si se determina que el tráfico en estudio es autosimilar, entonces los modelos de colas no son representativos de lo que ocurre realmente y los resultados tienen un grado de error no aceptable, es necesario utilizar otros modelos o adaptar los conocidos. La determinación de uno u otro método para el modelado implica evaluar cual de los métodos resulta mejor, se podrían considerar como aspectos importantes: la complejidad del método, la determinación de las variables del modelo, la versatilidad del modelo para su uso en distintos sistemas, y quizás el más importante, obtener resultados en las estimaciones lo mas cercano a la realidad. Más adelante modelamos un caso particular con teoría de colas y P1P, con el propósito de evaluar los aspectos mencionados anteriormente.

La teoría de los P1P es una alternativa nueva para el modelado de tráfico, surge la necesidad de medir la capacidad de los P1P para representar el sistema y proporcionar resultados estimados cercanos a la realidad. Los modelos de aprendizaje como los P1P tratan de memorizar la información implícita en los datos. La teoría de la información tiene relevancia en comunicaciones porque proporciona una forma de medir la información trasmitida, la medida de la información es fundamental para evaluar si cierta cantidad de información puede o no ser trasmitida por un canal de comunicaciones. Medir la información de una muestra de datos también puede ser útil para verificar si un modelo es capaz de memorizar una cantidad de información.

Si el lector esta familiarizado con la Teoría de colas y la Teoría de la información puede obviar el próximo capitulo nombrado *Teorías*.

Teorías

4.1. Teoría de Colas

4.1.1. Redes y teoría de colas

El proceso de transmisión de datos en una red requiere de recursos compartidos, pueden existir varios procesos de transmisión accediendo a recursos de forma concurrente y no sincronizada, esto provoca falta de recursos y consecuencias como la congestión y/o pérdida de paquetes. Cuando los recursos no son suficientes los paquetes se almacenan temporalmente en buffers hasta poder ser atendidos, el uso de colas (buffers) mejora los problemas de perdida de paquetes y de solicitudes de nuevas transmisiones.

Algunos problemas particulares que se estudian con la teoría de colas son:

- Análisis de Rendimiento y dimensionamiento de los buffers en los enlaces,
- Planificación de la capacidad de la red para conectar diferentes nodos,
- Evaluación de la performance de diferentes protocolos que se utilizan para que diferentes usuarios compitan por los mismos recursos.

4.1.2. Sistema de colas simple

Un sistema de colas simple consiste básicamente en un Servidor que atiende solicitudes, las solicitudes entran en una cola de espera hasta poder ser atendidas por el servidor. La gráfica a continuación muestra los elementos de un sistema de colas simple:

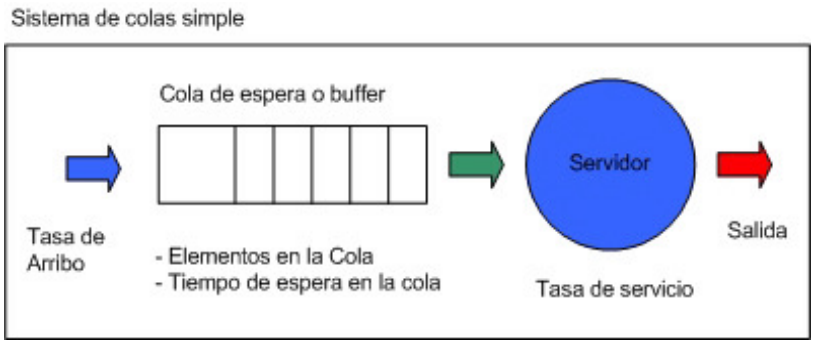

Figura 4.1: Elementos de un sistema de cola simple.

Un sistema de colas simple utilizado para Redes considera los siguientes aspectos para su construcción:

- Tiempo de llegada entre los mensajes
- Tiempo de atención del mensaje o servicio
- Número de mensajes en el buffer (o cola)
- Características de ordenación de mensajes en el buffer (o cola de espera)
- Tipo de población, finita o infinita.

Tiempo de llegada entre los mensajes

Los elementos llegan al sistema cada un cierto tiempo, la *tasa de arribo* al sistema es λ. El patrón de llegadas de los mensajes está especificado por el tiempo entre llegadas (el tiempo inter-arribo), se toma como tiempo promedio de arribo al valor esperado de los tiempos inter-arribo $E[A] = T_a$, la tasa de arribo es entonces $\lambda = \frac{1}{T_a}$

Para la aplicación en la tesis, un mensaje o elemento es un PDU (Protocol data Unit, Ej. Paquete IP) que llega al sistema y espera ser procesado.

Tiempo de atención o Servicio

Los mensajes se procesan en un cierto tiempo t, el patrón de servicio está especificado por el tiempo de servicio, que es el tiempo que le toma a un servidor atender a un mensaje. Se toma como tiempo de servicio al valor esperado de los tiempos de atención $E[B] = T_b$. Un servidor puede atender a un mensaje o podría atender a varios mensajes a la vez, el caso más sencillo es cuando un servidor atiende a aun solo mensaje. La tasa de atención queda definida por $\mu = \frac{1}{T_b}$.

Número de mensajes en el buffer

Es el número de mensajes que entran en el buffer y esperan hasta ser atendidos, en la realidad todos los buffers son finitos, la teoría de colas considera los dos casos, buffer infinito y finito. En redes en muy importante considerar este número porque cuando un buffer esta completo los mensajes se rechazan y no pueden ser atendidos.

Características de ordenamiento de los mensajes en el buffer

La característica de ordenamiento es una regla para seleccionar mensajes en el buffer y ser atendido. Una de las disciplinas más utilizadas es la denominada "First In First Out", FIFO, en la cual los primeros que llegan serán los primeros en salir; otra disciplina es la denominada "Last In First Out", LIFO, en la cual los últimos en llegar serán los primeros en salir. Existen otras disciplinas denominadas al azar y de prioridad, sin embargo para este trabajo se utilizará únicamente la disciplina de servicio FIFO.

Población

Los elementos llegan al sistema desde una población, para que no se altere el régimen de llegadas suponemos que la población es muy grande y se puede considerar infinita, cuando se utiliza la teoría de colas para modelar problemas de redes se puede considerar a la población infinita.

4.1.3. Notación de Kendall

Para describir estos aspectos D. G. Kendall en 1953 utilizo la siguiente notación

$$A/B/C/D/E/F$$

La letra A describe el proceso de llegada de los mensajes, la B representa la estadísticamente de la duración del servicio, por ejemplo si B=M el tiempo de servicio tiene una distribución exponencial. C indica el número de servidores del sistema de colas. D es la capacidad de servidor, el número de elementos en la cola del servidor esperando a ser atendidos incluyendo la solicitud que se esta atendiendo, E es el tipo de población y finalmente F es el ordenamiento en la cola para atender a las solicitudes.

Para la aplicación en la tesis usamos un sistema de colas $M/M/1/k/infinito/FIFO$ un sistema donde el régimen de llegadas y salidas tiene distribución exponencial, un solo servidor, un buffer limitado a una cantidad finita k, población infinita y planificación FIFO en la cola.

4.1.4. Modelo de colas M/M/1/k

Utilizamos la siguiente notación para las fórmulas del modelo de colas:

$$\begin{aligned}
\text{Tiempo promedio de interarribo } E[A] &= T_a \\
\text{Tasa de arribo } \lambda &= \frac{1}{T_a} \\
\text{Tiempo promedio de servicio } E[B] &= T_b \\
\text{Tasa de servicio } \mu &= \frac{1}{T_b} \\
\text{Utilización } \rho &= \frac{\lambda}{\mu} \\
\text{Rendimiento del sistema de colas (Throughput) } \lambda &= \rho\mu \\
\text{Número de elementos en el sistema } K_s &= \frac{\rho}{1-\rho} \\
\text{Tiempo de respuesta del sistema } T_s &= \frac{\frac{1}{\mu}}{1-\rho} \\
\text{Tiempo de espera en el sistema } W_s &= \frac{\frac{\rho}{\mu}}{1-\rho} \\
\text{Número de elementos en la cola } q &= \frac{\rho^2}{1-\rho}
\end{aligned}$$

para un sistema de cola finita tenemos que:

$$\text{Número de elementos en el sistema } N_s = \begin{cases} \frac{a}{1-a} - \frac{k+1}{1-a^{k+1}}a^{k+1}; a \neq 1 \\ \frac{k}{2}; a = 1 \end{cases}$$

donde $a = \frac{\lambda}{\mu}$ y $k =$ límite de la cola.
y para cualquier valor de a

$$\begin{aligned}
\text{Número de elementos en servicio es } L_{se} &= 1 - p_0 \\
\text{Probabilidad de que no existan elementos } p_0 &= \frac{1-\frac{\lambda}{\mu}}{1-(\frac{\lambda}{\mu})^{k+1}} \\
\text{Número de elementos en la cola } L_q &= N_s - L_{se}
\end{aligned}$$

4.2. Teoría de la información

4.2.1. Introducción a la teoría de la información

El concepto que analiza la Teoría de la información es la medida de la información, trata la información como una unidad técnica de medida. Desde este enfoque la información en una red de comunicaciones se refiere a los mensajes que salen de una fuente (trasmisor) y llegan a un destino.

Un destino en una red podría ser: un router a la espera de paquetes para conmutar, una placa de red Ethernet a la espera de una trama, una aplicación a la espera de un dato, o cualquier dispositivo o sistema que reciba mensajes. Si nos concentramos en los mensajes que pueden llegar a un destino determinado, y para reservar recursos quisiéramos calcular la cantidad de información que recibe ese destino, tenemos que asignar una probabilidad de llega a cada posible mensaje.

Por ejemplo, consideremos una interfaz i_1 en un router, y supongamos predecir los mensajes:

- 1 - llegan paquetes a destinos desconocidos
- 2 - llegan paquetes ICMP
- 3 - llegan 20 paquetes por segundo con destino al puerto 80
- 4 - llegan 100 paquetes por segundo con segmentos UDP al puerto 53

Los dos primeros mensajes no brindan información relevante porque a un router llegan paquetes a destinos desconocidos y también llegan paquetes ICMP, el tercero y cuarto mensaje brindan más información y según las predicciones 3 y 4 podríamos reservar memoria en los buffers para atender esa cantidad de paquetes. Podemos observar que las probabilidades de que ocurran 1 y 2 son mayores a la que ocurra 3 y 4, la teoría de la información deduce que cuanto menor sea la probabilidad de ocurrencia de un mensaje, más información contendrá.

La medida de la información estaría relacionada con la incertidumbre que existe sobre cuales y cuantos serán los paquetes que lleguen a la interfaz, del mismo modo está relacionada con la probabilidad de ocurrencia de los mensajes, y sería una función de la probabilidad.

4.2.2. Medida de la Información

Para un mensaje x_i definimos la probabilidad de que ocurra como $P(x_i)$, Shannon definió la medida de la información como una función logarítmica:

$$I_i = \log_b \frac{1}{P(x_i)} = -\log_b P(x_i)$$

donde b es la base logarítmica.

La autoinformación I_i del mensaje x_i depende solamente de la probabilidad $P(x_i)$ y no de su interpretación real. El uso de la función logaritmo proporciona propiedades particulares a la medida de la información:

- La autoinformación I_i de un mensaje x_i , disminuye cuando aumenta la probabilidad $P(x_i)$, cuando tenemos certeza de que un suceso x_i ocurrirá, su probabilidad $P(x_i)$ es uno y la autoinformación es cero.
- como la $0 \geq P(x_i) \geq 1$ la autoinformación $I_i \geq 0$; cuando $P(x_i) \to 1$ la autoinformación $I_i \to 0$; si tenemos dos mensajes x_i y y_j y sus probabilidades $P(x_i) > P(y_j)$ entonces $I_i < I_j$
- si se obtienen dos mensajes independientes x_i y y_j, la información que brindan en conjunto es $I_{ij} = \log_b \frac{1}{P(x_i y_j)} = \log_b \frac{1}{P(x_i)P(y_j)} = \log_b \frac{1}{P(x_i)} + \log_b \frac{1}{P(y_j)} = I_i + I_j$

4.2.3. Unidad de información

Cuando se realizan mediciones, los valores obtenidos se expresan en alguna unidad, la teoría de Shannon trata la información como un concepto medible y define la unidad de información especificando la base del logaritmo, el estándar de la teoría define la base en dos y la unidad de información como **bit**, debemos aclarar la diferencia entre bit (binary digit) y un **bit de información**, un bit en el concepto de la teoría de la información puede tener valores de probabilidad no equivalentes y por tanto aparecen cantidades de bits no enteras.

Si consideramos un **bit** como dígito binario (un digito que toma dos valores posibles), el universo de mensajes posibles es $U = \{0, 1\}$ y las probabilidades de cada mensaje son 0,5, la información que llega a un receptor cuando el mensaje es un digito binario se puede calcular como: dos valores posibles con igual probabilidad $P(x = 0) = 0{,}5$ y $P(x = 1) = 0{,}5$ por lo tanto la cantidad de información para cualquiera de los dos casos es $\log_2 \frac{1}{0{,}5} = 1bit$, o $\log_b \frac{1}{0{,}5} = 1bit$

4.2.4. Entropía

Otro concepto importante que utilizamos en este trabajo es el de **Entropía**, la entropía es el valor esperado de información. Consideremos como una fuente de información un nodo generando paquetes por segundo clasificados según algún criterio de interés, por ejemplo cantidad de paquetes por segundo con destino al puerto x, el conjunto de todos los posibles paquetes a recibir esta formado por $U = \{x_1, x_2, x_3, ...x_n\}$ este conjunto tienen un limite inferior x_1 y superior x_n, desde el punto de vista del receptor cada x_i tiene una probabilidad $P(x_i)$ de ocurrir (que pueda llegar), si suponemos que las cantidades son estadísticamente independientes (cualquier mensaje puede llegar sin depender de otro), la sumatoria de las probabilidades en el universo considerado es uno

$$\sum_{i=1}^{n} P(x_i) = 1$$

En la teoría de la información se define fuente discreta sin memoria, una fuente donde las probabilidades son constantes en el tiempo, las probabilidades de dos símbolos sucesivos son independientes y una tasa promedio de x mensajes por segundo. Para un destino (un router, una interfaz, una aplicación, etc.) que recibe paquetes por segundo, la red es una fuente discreta sin memoria. La cantidad de mensajes generados durante un tiempo arbitrario es una variable aleatoria discreta I con valores posibles $I_1, I_2, ...I_n$ y la información esperada por mensaje es entonces

$$\begin{aligned} H(X) &= \sum_{i=1}^{n} P(x_i) I_i \\ H(X) &= \sum_{i=1}^{n} P(x_i) \log_2 \frac{1}{P(x_i)} [bit/mensaje] \end{aligned}$$

4.2.5. Análisis de tráfico y teoría de la información

Los dos conceptos anteriores pueden utilizarse para calcular la información de una fuente F y calcular si es posible que el modelo soporte representar o almacenar la cantidad de información de la fuente, la idea básica es poder determinar si un modelo de tráfico como los P1P son capaces de aprender del tráfico la cantidad de información que llega por un canal.

Polinomios Potenciales de grado uno en cada Variable

5.1. Introducción

Esta teoría surgió de estudios en el área de IA, fue publicada en Octubre del 2007 como co-autor, aun no aparece en la bibliografía sobre modelos ni se ha utilizado para análisis de tráfico en redes IP, su aplicación al modelado de redes IP es inédita, esta tesis es la primera publicación sobre el uso de los P1P en redes.

Se describe a continuación los fundamentos de la teoría de los P1P enfocada en la utilización para el modelado de tráfico en redes. El lector puede optar por las publicaciones originales: *Harmonic Theory and Machine Learning* con referencia [1] y *Harmonic Functions On The Closed Cube: An Application To Learning Theory* referenciada en [2].

Un P1P modela el problema de Entrada/Salida clásico de ingeniería, dado un sistema cuya función interna es desconocida, aproximar la función relacionando los valores de entrada y salida del sistema. El proceso general consiste en analizar los datos de entrada y salida, armar un modelo y estudiar el sistema con el modelo obtenido.

Un problema similar se trata en el área de aprendizaje en Máquinas, Aprendizaje estadístico o tradicionalmente llamado métodos de ajuste,

5.2. Formulación del problema de modelado

Formularemos el problema de modelado tal como se hizo en el texto original de los P1P, más adelante lo relacionamos con el análisis de tráfico en redes.

Tenemos un sistema cuya función interna es f , el sistema se alimenta de entradas binarias $\{0,1\}^n$ y produce como resultado valores en el conjunto $\{0,1\}$, $f : \{0,1\}^n \to \{0,1\}$ el problema consiste en obtener o deducir la función f basándonos en los datos de entrada y salida, luego suponer casos de valores de entrada que no estén en la muestra y generalizar o predecir las salidas correspondientes. Por ejemplo, supongamos tener un sistema que acepta valores de entrada binarios y produce salidas con resultados binarios, esto se puede describir de la siguiente forma: se toman muestras de una función binaria:

$$\begin{aligned} f &: \{0,1\}^n \to \{0,1\} \\ f(m_1) &: 0,1,1,0,0,0,0 \to 0 \\ f(m_2) &: 1,1,1,0,0,1,0 \to 0 \\ f(m_3) &: 0,0,0,1,1,1,1 \to 1 \end{aligned}$$

donde las m_i son las muestras binarias, y tenemos que predecir, por ejemplo:

$$f(m_x) : 0,1,0,0,0,0,1 \to ?$$

Se puede pensar en que el total de los valores de entrada (todas las posibilidades) forman un cubo n-dimensional en el espacio R^n y que cada valor corresponde a un vértice. El problema consiste entonces en dados algunos valores conocidos de entrada (algunos vértices), requerimos inferir los valores de los otros vértices para completar el cubo y obtener de esta forma la función del sistema.

5.3. Problemática del modelado de funciones

La teoría de los P1P surgió en el área de estudio de Aprendizaje en Maquinas (ML Machine Learning) donde el problema principal es intentar obtener una función a partir de muestras de datos. Un proceso que aprende recibe datos desde el exterior, los procesa e intenta reconocer patrones en las muestras, de esta forma se dice que la función que sintetiza el proceso es el resultado de aprender o deducir de esos datos patrones, o características, para luego reconocerlos.

Mas allá de la terminología utilizada en ML el problema de obtener un modelo matemático a partir de datos observados, es análogo para otras áreas de estudio, también para el caso de análisis de tráfico en redes. Por ejemplo, podría interesarnos obtener una función $Congestion(X_1, ..., X_n)$ para predecir si la red estará congestionada con determinados valores extremos de tráfico, este modelo se construye con las variables de tráfico relacionadas con la congestión en el contexto de una red particular. Otro ejemplo podría ser obtener una función $Buffer(X_1, ..., X_k)$ que nos dé el valor estimado de un buffer en una interfaz con el objetivo de no perder paquetes, también en este caso el modelo utilizaría variables de tráfico que afecta al buffer en cuestión. Se pueden buscar más ejemplos que se ajusten al problema general de **encontrar la mejor función para el sistema**.

El problema general consiste en encontrar la mejor función que asocie las variables $X_1, ..., X_n$ de manera que los resultados para valores que no aparecen en la muestra sean lo más cercanos a la realidad, desde el punto de vista de la información significa que el modelo nos proporcione la mayor información posible sobre el caso extremo, o que reduzca la incertidumbre lo mejor posible, o que nos proporcione la mayor entropía posible.

En física existe un problema análogo que consiste en calcular la función f que proporcione la menor energía potencial para puntos conocidos, tenemos un conjunto de puntos aislados en el espacio R^n y queremos encontrar la función con la menor energía potencial E, dicho de otra forma, buscar una función que represente el sistema de puntos donde las distancias entre ellos es la menor, matemáticamente es un problema variacional con la siguiente ecuación:

$$E = \min_f \int_0^1 ... \int_0^1 \sqrt{\sum_{i=1}^{n} \left(\frac{\partial f}{\partial X_i}\right)^2} dX_1 ... dX_n$$

La termodinámica estudió el proceso de transferencia del calor de un elemento a otro, el problema de encontrar la función del flujo de calor al pasar de un punto a otro, es análogo al problema general que describimos anteriormente, en este caso hay un proceso natural que encuentra la función de mayor entropía. El modelo matemático del flujo de calor obtenido tiene la característica de **aplanar** los valores límites y llegar a los valores mínimos absolutos, este proceso no agrega información, sino que transfiere información hasta llegar a un estado de equilibrio donde se alcanza el estado de mínima energía potencial y un máximo de entropía.

Heurísticamente podemos decir que la información de todo el dominio D, incluidos los valores de la frontera, se mantienen desde el estado inicial hasta el estado estable final, llegando a una función sin máximos o mínimos locales. La información total se mantiene desde el estado inicial hasta el final.

Consideremos una ecuación de flujo de calor n-dimensional:

$$\frac{\partial T(X_{1,...,}X_n, t)}{\partial t} = \sum_{i=1}^{n} \frac{\partial^2 T(X_{1,...,}X_n, t)}{\partial X_i^2}$$

donde los vértices del cubo en R^n tienen una configuración representada por una función $T(X_1, X_2, ..., X_n, t)$, siendo el estado inicial para $t = 0$ y considerando como estado fijo $T(X_1, X_2, ..., X_n, 0)$ con todos los valores de los vértices pertenecientes a la función T.

Luego del proceso de transferencia del calor cuando $t \to \infty$, el estado será de equilibrio y obtendremos el valor de la función T. Una vez obtenida la función T podemos predecir los valores de T en los vértices.

El límite del proceso es la función armónica, esto es, una función dos veces derivable que cumple con la ecuación de Laplace

$$\sum_{i=1}^{n} \frac{\partial^2 T(X_{1,...,}X_n, 0)}{\partial X_i^2} = 0$$

La propiedad más importante de la solución proporcionada por el proceso de transferencia del calor es que resuelve el siguiente problema variacional:

Encontrar la función f (en nuestro caso T) que minimice la ecuación de la energía potencial E en los puntos conocidos que forman el cubo de datos (formado por las muestras medidas):

$$E = \min_f \int_0^1 ... \int_0^1 \sqrt{\sum_{i=1}^{n} \left(\frac{\partial f}{\partial X_i}\right)^2} dX_1 ... dX_n$$

Estas ecuaciones se originaron en la termodinámica y por esto se mantuvo la sintaxis de esta área, pero como se puede observar, una función como $Congestion(X_1, ..., X_n)$ o $Buffer(X_1, ..., X_k)$ tienen las mismas características que $T(X_1, X_2, ..., X_n, t)$, veamos que las variables de la función congestión y buffer $X_i(t)$ representan el estado en el tiempo t de cada uno de los tráficos asociados al problema y que el tiempo t es el mismo para todas las variables por lo tanto podemos expresar las funciones como $Congestion(X_1, ..., X_n, t)$ o $Buffer(X_1, ..., X_k, t)$.

5.4. Polinomio Potencial de grado uno en cada variable

Definición:

Un P1P (Potential Polynomial of Degree 1 in Each Variable) se define como una cantidad finita de monomios sobre las variables:

$$X_1, ..., X_n$$

donde cada variable tiene un valor real y a lo sumo de grado 1.

La formula más general de un P1P es:

$$c + \sum c_i X_i \ + \sum c_{ij} X_i X_j + \sum c_{ijk} X_i X_j X_k \ + ... + c_{1,2...n} X_1 X_2 ... X_n$$

donde:

- cada termino incluye las variables con sub-índices $i, j, ..., k, ...n$ son distintos y sin repetición.
- los coeficientes $\{c, c_i, c_{ij}, c_{ijk}, ..., c_{123..n}\}$ son valores reales
- los sub-índices de los coeficientes pueden ser cualquiera

5.5. Solución al problema de modelado de funciones

El trabajo original plantea los P1P booleanos como una solución, a continuación describimos los P1P booleanos y posteriormente una analogía con los modelos de tráfico en redes.

5.5.1. Modelos PIP

Un caso especial booleano puro donde los monomios están compuestos por variables X_i o sus negaciones boobleanas $1 - X_j$, donde los coeficientes toman valores cero o uno (0 o 1) tiene la forma

$$P(X_1, X_2, .., X_i, .., X_n)$$

este P1P con monomios booleanos es similar a la forma normal disyuntiva en el calculo de predicados, las variables booleanas asegura que el polinomio tiene todas las variables de grado uno. Los polinomios P son funciones en el espacio R^n cuyas derivadas son:

$$\frac{\partial P}{\partial X_i} = Q_i(X_1, X_2, ..., X_n)$$

Las derivadas Q_i también son P1Ps con variable independiente X_i (X_i no aparece en el nuevo P1P Q_i), esto implica que la derivada segunda es cero. La derivada segunda es cero para todas las variables X_i por lo tanto los P1P cumplen con la ecuación de Laplace y entonces son funciones armónicas en R^n.

$$\frac{\partial Q_i}{\partial X_i} = \frac{\partial^2 P}{\partial X_i^2} = 0$$

Consideremos nuevamente la ecuación del flujo del calor n-dimensional:

$$\frac{\partial T(X_{1,...,}X_n, t)}{\partial t} = \sum \frac{\partial^2 T(X_{1,...,}X_n, t)}{\partial X_i^2}$$

Siendo la función $T(X_{1,...,}X_n, 0)$ la distribución inicial con temperatura fija $t = 0$, es decir, el estado inicial de los vértices del cubo en R^n. Durante el proceso de transferencia de calor, la temperatura fluirá gradualmente hasta converger al estado de equilibrio, manteniendo fijos los datos (las muestras) inicialmente llega a un estado de equilibrio sobre los bordes, y luego hacia el interior del cubo también manteniendo fijos los datos.

La función resultante cuando $t \to \infty$ es única y resuelve el **problema de Dirichlet's** que consiste en encontrar una función armónica en un dominio R^n que tome valores conocidos (los datos o vértices de nuestro caso) sobre el contorno del dominio.

5.5.2. Analogía con un modelo de tráfico

Una analogía al modelado de tráfico de redes IP es: podemos pensar en una función $Net(X_{1,...,}X_n, t)$ que representa algún sistema particular de la red en el tiempo t (congestión, buffer, otro), las variables $X_{1,...,}X_n$ son matrices columnas con los k valores de las capturas de tráfico, para una variable genérica

$$X_i = \begin{pmatrix} X_{i1} \\ X_{i2} \\ . \\ X_{ik} \end{pmatrix}$$

la función es

$$Net(\begin{pmatrix} X_{11} \\ X_{12} \\ . \\ X_{1k} \end{pmatrix}, ..., \begin{pmatrix} X_{n1} \\ X_{n2} \\ . \\ X_{nk} \end{pmatrix}, t)$$

esta función inicial tiene todos los valores de la muestra de tráfico para el tiempo $t = 0$, la función es la primer aproximación al sistema Net, podemos pensar en una analogía entre el proceso de trasferencia del calor y el proceso de encontrar un modelo para el tráfico, en nuestra analogía el flujo del calor es análogo a la cantidad

de información, se estudia la información que proporciona la función Net en el tiempo t (recordemos que la función Net es un modelo sobre algún aspecto de la red).

Desde el punto de vista de la Teoría de la información, utilizando la función Net para estimar el estado del sistema con tráfico $X_1 = x_1, .., X_n = x_n$, la función proporciona un valor con cierta incertidumbre en el tiempo t, es decir que existe una probabilidad $p(v_i)$ de que el valor estimado v_i este cercano a la realidad.

De forma heurística podemos decir que cuando la función Net pasa por un proceso similar al de transferencia del calor, pero en este caso se transfiere información desde el estado inicial hasta el final, y el tiempo $t \rightarrow \infty$, la incertidumbre disminuye hasta su valor óptimo (mínimo), significa que la entropía del sistema es máxima, y la probabilidad $p(v_i)$ de que el valor v_i (estimado por la función Net) este cercano a la realidad es máxima.

5.5.3. Demostración

Explicamos en los siguientes párrafos la demostración original de que los P1P son una solución de la ecuación potencial (una solución al problema de encontrar la mejor función en un cubo R^n). Según la teoría de los P1P sólo hay una solución para f armónica que resuelve el problema de Dirichlet en el cubo cerrado, con valores de f en sus vértices.

Tenemos un número finito de valores de una función desconocida f, las variables de la función son $X_1, X_2, ..., X_n$, y los valores de las muestra son $f_i(\ X_{1i}, X_{2i}, ..., X_{ni})$ estos valores son parte de un subconjunto de vértices en un cubo en R^n, el problema consiste en encontrar una función f en el cubo R^n, y expresarlo como un P1P sobre los datos de muestra.

Se trata de entrenar una función (el P1P) con valores de muestra para que ajuste sus coeficientes y que esta función responda al modelo requerido.

Tenemos entonces variables que identifican las muestras:

$$X_{1i}, X_{2i}, ..., X_{ni}(i = 1, ..., m)$$

variables booleanas que identifican m vértices del cubo $n - dimensional$, y sea

$$f_i(\ X_{1i}, X_{2i}, ..., X_{ni})$$

los valores dados de la función f.

Si definimos una matriz M_k con las k variables de los P1P y construimos un sistema de ecuaciones de la forma

$$M_k * \lambda = f$$

podemos obtener la función f. M_k es una matriz con los valores de las variables (más adelante explicamos el formato de esta matriz), $\lambda = \begin{pmatrix} \lambda_1 \\ . \\ . \\ \lambda_n \end{pmatrix}$ es una columna de coeficientes y $f = \begin{pmatrix} f_1 \\ . \\ . \\ f_n \end{pmatrix}$ es una columna con los valores de la función. Se trata entonces de definir el sistema de ecuaciones de manera tal que tenga solución única, para ello tenemos que encontrar una matriz M_k cuyo determinante $\det(M_k) \neq 0$ sea distinto de cero, y entonces el sistema tendrá solución única.

Definimos una matriz M_k con las siguientes características:

- Tenemos la matriz M_c donde las filas son todos los vértices del cubo R^k, un vértice esta definido por las combinaciones de las k variables donde todos los índices son distintos, son todas las $2^k - uplas$ de la forma $(1, X_{1i}, X_{2i}, .., X_{ni}, X_{1i}X_{2i}, ..., X_{ni-1}X_{ni}, X_{1i}X_{2i}X_{3i}, ..., X_{1i}X_{2i}..X_{ki})$ que son vértices del cubo. Esta matriz es de orden $2^k x 2^{2^k}$, ya que la cantidad de combinaciones de 2^k elementos es 2^{2^k}, a partir de la matriz M_c construimos la matriz M_k de manera que su determinante sea distinto de cero, las siguientes condiciones hacen que el $\det(M_k) \neq 0$:

- de orden 2^k $(2^k x 2^k)$
- las columnas son $1, X_1, X_2, .., X_n, X_1X_2, ..., X_{n-1}X_n, X_1X_2X_3, ..., X_1X_2..X_k$
- seleccionamos la primera fila de manera que todas las variables sean cero $(1, 0, 0, ..., 0)$
- seleccionamos la primera columna de manera que sean todos valores unos $\begin{pmatrix} 1 \\ 1 \\ 1 \\ . \\ . \\ . \\ 1 \end{pmatrix}$.

Ejemplo para k=1

si $k = 1$ tenemos una variable X_1,

- la Matriz M_c tiene orden $2^1 x 2^{2^1} = 2x4$, $M_c = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$,

luego construimos la matriz $M_{k=1}$ con las condiciones definidas:

- orden $2^k = 2^1 = 2$ (una matriz de $2x2$)
- las columnas son $1, X_1$ (solo $1, X_1$ porque hay 2 columnas)
- la primer fila $(1, 0)$
- la primer columna $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$

la matriz M_1 (indicamos una referencia a las filas como Fx, $x = 1, 2$, y a las variables) es entonces:

$$\begin{array}{cc} fila \setminus variables & \begin{array}{cc} 1 & X_1 \end{array} \\ \begin{array}{c} F1 \\ F2 \end{array} & \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \end{array}$$

si reemplazamos los 1 por el encabezado de cada columna tenemos $M_1 = \begin{pmatrix} 1 & 0 \\ 1 & X_1 \end{pmatrix}$

Ejemplo para $k = 2$

si $k = 2$, tenemos dos variables X_1 y X_2

- la matriz M_c tiene orden $4x16$, indicamos una referencia a las filas como Fx, $x = 1, 2.,16$, y las variables en las columnas, más adelante las usamos para la construcción de M_k, la matriz es:

$$\begin{array}{c} \text{fila} \setminus \text{var} \\ F1 \\ F2 \\ F3 \\ F4 \\ F5 \\ F6 \\ F7 \\ F8 \\ F9 \\ F10 \\ F11 \\ F12 \\ F13 \\ F14 \\ F15 \\ F16 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & X_1 & X_2 & X_1X_2 \end{array} \\ \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 1 & 0 & 0 & 1 \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ 1 & 0 & 1 & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{array} \right) \end{array}$$

$$M_c = \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 1 & 0 & 0 & 1 \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ 1 & 0 & 1 & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{array} \right)$$

construimos la matriz $M_{k=2}$ con las condiciones:

- orden $2^k = 2^2 = 4$ (una matriz de $4x4$)
- las columnas son $1, X_1, X_2, X_1X_2$ (solo $1, X_1, X_2, X_1X_2$ porque hay 4 columnas)
- la primer fila debe ser $(1, 0, 0, 0)$
- la primer columna debe ser $\left(\begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \end{array} \right)$

Seleccionamos las filas de forma que cumplan con las condiciones para que el $det(M_2) \neq 0$, seleccionamos las filas marcadas en negrita y la matriz M_2 queda de la siguiente forma:

$$\begin{array}{c} \text{fila} \setminus \text{var} \\ F9 \\ F13 \\ F11 \\ F16 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & X_1 & X_2 & X_2X_2 \end{array} \\ \left(\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{array} \right) \end{array} , \; M_2 = \left(\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 1 & X_1 & 0 & 0 \\ 1 & 0 & X_2 & 0 \\ 1 & X_1X_2 & X_1X_2 & X_1X_2 \end{array} \right)$$

Observamos que la matriz M_2 esta compuesta por la M_1 y podemos generalizar la construcción de la matriz M_k definiéndola de forma recursiva como:

$$\begin{aligned} M_1 &= \left(\begin{array}{cc} 1 & 0 \\ 1 & 1 \end{array} \right) \\ M_2 &= \left(\begin{array}{cc} M_1 & 0 * M_1 \\ M_1 & 1 * M_1 \end{array} \right) \\ M_k &= \left(\begin{array}{cc} M_{k-1} & 0 * M_{k-1} \\ M_{k-1} & 1 * M_{k-1} \end{array} \right) \end{aligned}$$

Generamos la matriz M_{k+1} con las columnas anteriores más las nuevas:

$$X_{k+1} * \; [\, 1, X_1, .., X_1X_2...X_k]$$

entonces tenemos que:

$$M_{k+1} = \left(\begin{array}{cc} M_k & 0 * M_k \\ M_k & 1 * M_k \end{array} \right) = \left(\begin{array}{cc} M_k & 0 \\ M_k & M_k \end{array} \right)$$

El $\det(M_{k+1}) = \det(M_k)$ por la hipótesis de inducción el $\det(M_k)$ no es cero, por lo tanto $\det(M_{k+1})$ no es cero.

Según el resultado anterior, las ecuaciones lineales:

$$M_n * \left(\begin{array}{c} \lambda_1 \\ \lambda_2 \\ . \\ . \\ \lambda_{2^n} \end{array} \right) = \left(\begin{array}{c} f_1 \\ f_2 \\ . \\ . \\ f_{2^n} \end{array} \right)$$

para cualquier f_i es compatible determinado y tiene exactamente una solución dada por:

$$M * \lambda = f$$

o

$$f = \lambda_1 + \lambda_2 X_1 + ... + \lambda_{2^n} X_1X_2....X_n$$

es la única solución en el cubo n-dimensional.

Parte III

Aplicación de P1P al modelado de tráfico

Modelado de tráfico con P1P

El objetivo de este capítulo es mostrar la forma en la que utilizamos los P1P para modelar tráfico, en las próximas secciones analizamos aspectos teóricos para comprender el uso de los P1P y en la sección **Datos de entrada** se muestra un ejemplo ilustrativo.

6.1. Introducción

Muchos de los problemas de las redes se estudian y se analizan con un modelo de tráfico para luego estimar el valor de alguna variable crítica V_c, una manera de armar un modelo es realizar un experimento, medir el estado de las variables y los resultados para luego construir una ecuación matemática que permita obtener valores estimados de V_c.

6.2. Enfoque de caja negra

El tráfico que circula por una red atraviesa diferentes componentes afectando su comportamiento, los resultados varían según las variables consideradas y el tipo de tráfico involucrado en el problema. La aplicación de los P1P al análisis de tráfico consiste en modelar el problema con un enfoque de Entradas->Caja Negra->Salida, esto es, identificar el sistema donde se focaliza el problema, las variables de entrada que lo afectan y los resultados que generan estas variables (ver figura). La problemática es intentar descubrir el interior de la caja negra a partir de los datos de entrada y salida y construir una función que asocie las muestras de las variables de entrada con los resultados de salida.

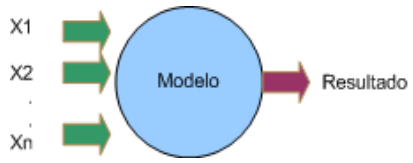

Figura 6.1: Enfoque de caja negra

En el gráfico, $X_1, X_2, ..., X_n$ son entradas, variables aleatorias $X_i(t)$ de tiempo fijo que afectan al sistema, y *Resultado*, que llamaremos $S(t)$, es la salida del sistema, el *Modelo* es entonces una función $f : R^n \rightarrow R^1$ que asocia un vector de variables $v(X_1, X_2, ..., X_n)$ de entrada con una salida S, la función relaciona vectores en el espacio R^n con resultados en el espacio R^1, la ecuación general es:

$$S(t) = f(X_1(t), X_2(t), ..., X_n(t))$$

Para modelar tráfico identificamos las variables $X_i(t)$ como cantidad de PDU (Protocol Data Unit) por unidad de tiempo, por ejemplo paquetes por segundo, segmentos por segundo, u otros, y el resultado $S(t)$ es algún

aspecto de interés relacionado con las variables $X_i(t)$. El tiempo t es el mismo para todas las variables y podemos escribir la ecuación como

$$S = f(X_1, X_2, ..., X_n, t)$$

La función f es desconocida y es la función que hay que aproximar midiendo los estados de las variables y observando el resultado en diferentes tiempos, en la próxima sección se indica como realizar una aproximación de la caja negra con un P1P.

6.3. Función para la caja negra

La teoría de los P1P propone construir una función polinómica que asocia las variables $X_i(t)$ de entrada con la salida $S(t)$, el P1P es una aproximación al interior de la caja negra, lo cual permite realizar estimaciones de valores de salida del sistema suponiendo entradas particulares.

Para un sistema con dos entradas X_1, X_2 y salida S un modelo P1P es una función

$$S = P1P(X_1, X_2)$$

y es por definición:

$$S = c_0 + c_1 * X_1 + c_2 * X_2 + c_3 * X_1 * X_2$$

donde las c_i son coeficientes.

El modelo no esta completo sin los valores de los coeficientes, los coeficientes se calculan utilizando las muestras obtenidas experimentalmente y con una regresión no lineal. Las muestras de datos de tráfico son grandes por lo que se requiere de alguna herramienta para calcular la regresión. En esta tesis la regresión la calculamos con la ayuda del programa *Mathematica*, que cuenta con un paquete de funciones para estadística (paquete **Statistic**), la función que utilizamos es **NonLinearRegress(datos, modelo, variables, parámetros),** cuyos parámetros son 4:

- 1 - Los **datos** de la muestra, están representados como una matriz donde las columnas son los valores de las variables, la columna final son los valores de la función.
- 2 - El **modelo**, para este caso es el polinomio P1P.
- 3 - Un vector con las **variables** del modelo, las variables del modelo P1P.
- 4 - Un vector con los coeficientes (o **parámetros**) del modelo, los coeficientes del P1P

Si queremos resolver los coeficientes del P1P anterior, entonces el programa Mathematica seria el siguiente:

```
NonlinearRegress[ datos, c + c1 * x1 + c2 * x2 + c3 * x1 * x2, {x1, x2}, {c, c1, c2, c3}]
```

la variable **datos** es una matriz con los valores de captura de tráfico, más adelante se explica como construirla.

La ejecución de esta función da como resultado los valores de los coeficientes que al reemplazarlos en el polinomio obtenemos el modelo P1P completo. En la siguiente sección se explica en detalle el resultado y las opciones de esta función.

6.4. Datos de entrada y salida

Veamos con un ejemplo completo como calculamos los coeficientes del P1P.

Después de identificar el problema y las variables asociada, debemos obtener una muestra de valores de variables de entrada y salida de la caja negra, esta es la etapa de captura de tráfico y determinación de los valores de las variables.

Por ejemplo, supongamos que se observa congestión en la red, el tráfico circula por un dispositivo D bajo nuestra administración y queremos determinar los estados críticos donde el tráfico genera congestión para reservar recursos y evitar la perdida de paquetes en nuestro nodo. Experimentalmente se realiza una captura de tráfico y se observa tráfico de video streaming, voz IP, audio de radio, tráfico de WWW, Correo, FTP y otros, observamos también el buffers de salida con las variaciones que se generan en los distintos tiempos medidos. Un esquema de caja negra para este problema es el siguiente:

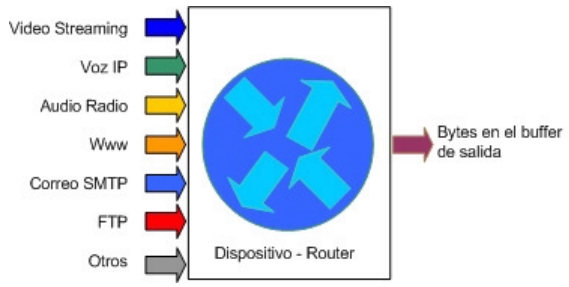

Figura 6.2: Enfoque de caja negra para la congestión en un router

Identificamos las variables de entrada como la cantidad de **paquetes por segundo** y en cada paquete identificamos segmentos TCP donde el puerto de origen y/o destino identifica alguno de los servicios nombrados más arriba. De esta forma las variables de entrada son variables aleatorias $X_i(t)$ donde cada valor de la muestra es la cantidad de paquetes en el segundo t_i. En el primer segundo t_1 se podrían encontrar los siguientes valores:

- $X_1(1)$ = video streaming: 60 paquetes
- $X_2(1)$ = voz IP: 40 paquetes
- $X_3(1)$ = audio de radio: 20 paquetes
- $X_4(1)$ = www: 78 paquetes
- $X_5(1)$ = correo SMTP: 25 paquetes
- $X_6(1)$ = FTP: 23 paquetes
- $X_7(1)$ = Otros paquetes: 40 paquetes
- $B(1) = 560340$ *bytes*

el tamaño del buffer B de salida deberá ser como mínimo la suma de los tamaños de todos los paquetes entrantes para el tiempo t_1 de manera que no se pierda ningún paquete, para nuestro caso hipotético la suma de los tamaños de los paquetes podría ser 560340 *bytes*.

Si expresamos los valores de las muestras en forma de vector v la primera muestra de datos para el tiempo $t = 1$ es

$$v_1(X_1(t_1) = 60, X_2(t_1) = 40, X_3(t_1) = 40, X_4(t_1) = 20, X_5(t_1) = 78, X_6(t_1) = 25, X_7(t_1) = 23, B(t_1) = 40)$$

de forma genérica las siguientes muestras para $t = 2, 3, ..., k$ serán:

$$v_i(X_1(t_i), X_2(t_i), X_3(t_i), X_4(t_i), X_5(t_i), X_6(t_i), X_7(t_i), B(t_i))$$

y si formamos una matriz **datos** con todas las muestras tenemos:

$$datos = \begin{pmatrix} X_1(t_1) & X_2(t_1) & X_3(t_1) & X_4(t_1) & X_5(t_1) & X_6(t_1) & X_7(t_1) & B(t_1) \\ X_1(t_2) & X_2(t_2) & X_3(t_2) & X_4(t_2) & X_5(t_2) & X_6(t_2) & X_7(t_2) & B(t_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_1(t_k) & X_2(t_k) & X_3(t_k) & X_4(t_k) & X_5(t_k) & X_6(t_k) & X_7(t_k) & B(t_k) \end{pmatrix}$$

esta matriz es la que se utiliza para calcular los coeficientes del P1P con el programa en Mathematica y la función: `NonlinearRegress`.

Un P1P con 7 variables tiene $2^7 = 128$ términos, no resulta práctico para esta ilustración, en su lugar usamos solo 2 variables que generan un P1P con 4 términos, de manera que la matriz datos queda entonces:

$$datos = \begin{pmatrix} X_1(t_1) & X_2(t_1) & B(t_1) \\ X_1(t_2) & X_2(t_2) & B(t_2) \\ \vdots & \vdots & \vdots \\ X_1(t_k) & X_2(t_k) & B(t_k) \end{pmatrix}$$

y el modelo P1P es

$$S = c_0 + c_1 * X_1 + c_2 * X_2 + c_3 * X_1 * X_2$$

con la matriz y el modelo podemos calcular los coeficientes con la siguiente función del programa Mathematica:

```
NonlinearRegress[ datos,
                  c0 + c1 * x1 + c2*x2 + c3 * x1 * x2,
                  {x1, x2},
                  {c0, c1, c2, c3},
                  RegressionReport->{opciones}]
```

La función *NonlinearRegress* estima los parámetros con un ajuste por mínimos cuadrados, minimizando la suma de los cuadrados de los residuos, retorna múltiples reportes con los resultados especificando opciones en el parámetro *RegressionReport*. En la tesis utilizamos solo los valores de mejor ajuste *BestFitParameters*.

La ejecución de esta función da un resultado similar al de la Figura 6.3 :

Si no se especifican valores en *RegressionReport* por defecto toma la opción

RegressionReport->SummaryReport

que incluye los elementos en el resultado:

- *BestFitParameters:* Retorna una lista de valores para los parámetros
- *ParameterCITable:* Retorna una tabla con: los valores de parámetros, los errores estándar (SE), y los intervalos de confianza (CI)
- *EstimatedVariance:* Retorna la suma de los cuadrados de *FitResidual* dividido por los $n - p$ grados de libertad, donde n es la longitud del conjunto de datos y p es el número de parámetros. (*FitResidual* es la diferencia entre los valores de datos de entrada y el los valores de mejor ajuste.)
- *ANOVATable:* Retorna una tabla de análisis de la varianza para el modelo ajustado.

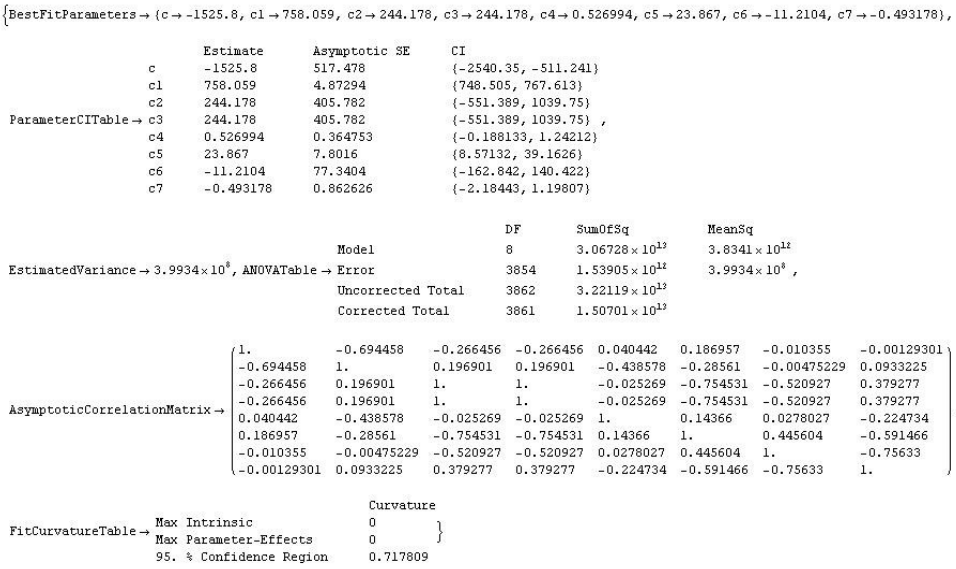

Figura 6.3: Resultado de la función *NonlinearRegress* de Mathematica.

- *AsymptoticCorrelationMatrix:* Retorna la matriz de correlación, esta basada en aproximaciones lineales para el modelo ajustado no lineal.
- *FitCurvatureTable:* Incluye valores máximos de la curvas, incluye el máximo relativo de la curva intrínseca, máximo relativo de la curva con los parámetros, y la curvatura relativa a la región de confianza de la solución de mínimos cuadrados.

6.5. Método guía para construir un modelo con P1P

La construcción de un modelo P1P tiene etapas similares a la construcción de modelos en simulación, adicionalmente existe una etapa de ajuste de coeficientes o aprendizaje, una guía metodológica para la construcción de los modelos P1P puede ser:

- Identificación del problema, o identificación del sistema a modelar.
- Determinación de las variables de entrada y salida del sistema.
- Construcción del P1P de acuerdo a la definición y la cantidad de variables, se obtiene el polinomio P1P con coeficientes sin valor.
- Obtención de muestras para variables de entrada y su resultado asociado.
- Confección de la matriz para el cálculo de los coeficientes
- Calculo o Ajuste de los coeficientes con una regresión no lineal (NonLinearRegress de Mathematica).
- Construcción del modelo completo reemplazando los valores de los coeficientes.
- Utilización del modelo final.

El modelo P1P construido a partir del sistema es un modelo de aprendizaje sobre la muestra de datos, inicialmente se construye el modelo teórico donde los coeficientes aun no tienen valor, luego se calculan los valores de los coeficientes con una regresión no lineal. Luego el modelo que se obtiene ajustó sus coeficientes de acuerdo a la muestra de datos. Heurísticamente podemos decir que cuanto más grande sea la muestra, más representativo será el modelo del sistema considerado. El estudio sobre el tamaño adecuado de la muestra y la representatividad del modelo obtenido no se trata en ésta tesis.

Modelo de tráfico de transporte y red

En esta sección construimos un modelo de tráfico basado en los paquetes y segmentos TCP/IP, se focaliza el problema en la cantidad de bytes a procesar en estos niveles y se usa para estimar tráfico en situaciones supuestas. Los datos de volcado se preprocesan para filtrar el tráfico y generar la matriz de muestras con un formato adecuado para el calculo de los coeficientes del P1P. En el final del trabajo se muestra como realizar estimaciones con el modelo.

7.1. Introducción

Las muestras de datos se tomaron de uno de los enlace a Internet de la red mostrada en la topología de la figura a continuación. La red de la institución educativa cuenta con 4 enlaces, dos con tecnología ADSL utilizados para el tráfico de acceso a Internet desde las PC's en las VANS, y dos enlaces Frame Relay (FR) con un CIR de 100 % utilizados para que proporcionar servicios con IP fija, el primero enlace FR es para acceso a Internet 1 y ofrecer servicios SMTP, DNS, WEB. El segundo es un enlace a la red Internet 2.

Consideremos el ancho de banda disponible para acceder a Internet desde una PC en la LAN en determinadas condiciones, dicho ancho de banda depende de varios factores y resulta difícil estimarlo, realizaremos un modelo de tráfico para estimar la cantidad de bytes en paquetes IP que podrían procesarse en uno de los enlaces ADSL donde realizamos las medidas.

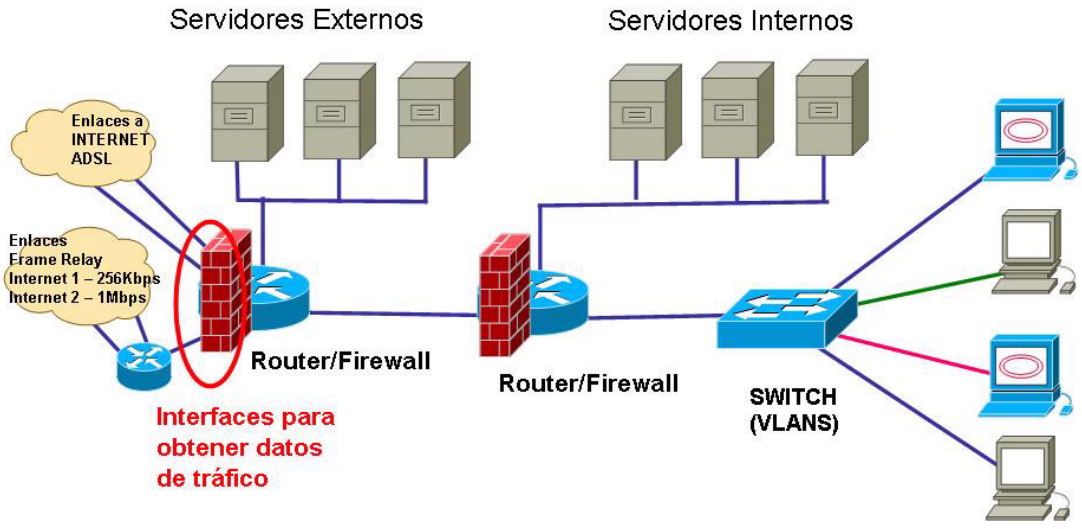

Figura 7.1: Topología de red, enlaces para la medición de tráfico.

7.2. Identificación del sistema

Consideremos el problema de procesamiento de bytes en paquetes IP en una interfaz como un sistema o caja negra que procesa información, las entradas de nuestra caja negra son segmentos TCP, datagramas UDP y otros protocolos como ICMP, este sistema procesa los datos de entrada y como salida genera paquetes IP de diferente tamaño. El siguiente esquema muestra el concepto:

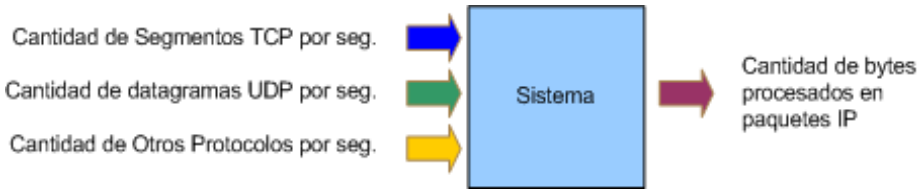

Figura 7.2: Definición del sistema para procesamiento de bytes. Modelo de Trasnporte y Red.

El problema consiste en encontrar un modelo matemático para el sistema a partir de la medida de datos de entrada y la salida, consideraremos para nuestro sistema tres variables aleatorias con tiempo fijo $t = 1s$, entradas X_1, X_2, X_3 y una salida B, cada variable representa lo siguiente:

- X_1: Cantidad de segmentos TCP por segundo.
- X_2: Cantidad de datagramas UDP por segundo.
- X_3: Cantidad de otros protocolos por segundo.
- B: Cantidad de bytes en los paquetes IP por segundo.

7.3. Datos de entrada

Para obtener la muestra de datos utilizamos la herramienta de volcado de paquetes *Tcpdump* de Unix, se realiza un volcando en un archivo de texto para luego con un programa especifico (programado en lenguaje php), analizar y obtener los datos de entrada.

Para el uso de TCPdump seleccionamos opciones particulares para que identifique los paquetes IP, segmentos TCP y datagramas UDP. Corremos TCPdump en una interfaz Fast-Ethernet conectada a un modem-router ADSL con salida a Internet, el comando utilizado fue:

$$\#\texttt{tcpdump} \ -\texttt{n} \ -\texttt{i eth0} \ -\texttt{s 1500} \ -\texttt{vvv} \ > \texttt{10,10,20,10.tcpdump}$$

donde las opciones elegidas indican:

- -i: escuchar en la interfaz eth0,
- -s: volcar el paquete completo Ethernet (-s 1500),
- -vvv: mostrar toda la información
- > `10,10,20,10.tcpdump`: redireccionar los datos a un archivo llamado "10.10.20.10.tcpdump" para su posterior procesamiento.

TCPDump captura datos de tráfico en la interfaz Ethernet y los escribe en un archivo de texto, veamos que contienen las líneas del archivo (las primeras 5 líneas):

1. 18:10:03.000842 IP (tos 0x0, ttl 116, id 9296, offset 0, flags [DF], proto: TCP (6), length: 1472) 200.42.92.5.554 > 10.10.20.10.1108: . 3969520650 :3969522082(1432) ack 209013619 win 64197
2. 18:10:03.001296 IP (tos 0x0, ttl 126, id 7663, offset 0, flags [DF], proto: TCP (6), length: 40) 10.10.20.10.1108 > 200.42.92.5.554: ., cksum 0x08ee (correct), 1:1(0) ack 1432 win 65535
3. 18:10:03.012978 IP (tos 0x0, ttl 116, id 9297, offset 0, flags [DF], proto: TCP (6), length: 1472) 200.42.92.5.554 > 10.10.20.10.1108: . 1432:2864(1432) ack 1 win 64197
4. 18:10:03.024873 IP (tos 0x0, ttl 116, id 9298, offset 0, flags [DF], proto: TCP (6), length: 1472) 200.42.92.5.554 > 10.10.20.10.1108: . 2864:4296(1432) ack 1 win 64197
5. 18:10:03.025327 IP (tos 0x0, ttl 126, id 7664, offset 0, flags [DF], proto: TCP (6), length: 40) 10.10.20.10.1108 > 200.42.92.5.554: ., cksum 0xfdbd (correct), 1:1(0) ack 4296 win 65535

donde cada una de las columnas indica:

- dato de tiempo "18:10:03.001296", tiempo: se utiliza para considerar el tamaño de bytes procesados por segundo.
- dato IP(...): campos del protocolo IP, consideramos los datos de:
  - "proto": protocolo encapsulado en el paquete IP: se cuentan por segundo la cantidad de protocolos TCP, UDP o ICMP.
  - "length": longitud del paquete IP, se suma la cantidad de bytes en los paquetes IP por segundo.

Este archivo contienen los datos crudos del volcado, por cada paquete se indica el tiempo de llegada, necesitamos procesar los datos para obtener los valores de las variables aleatorias por segundo. El resultado del procesamiento es una matriz de la forma:

$$\begin{pmatrix} 67 & 1 & 0 & 56203 \\ 63 & 0 & 2 & 51436 \\ 64 & 2 & 2 & 59032 \\ 141 & 10 & 1 & 115695 \\ 74 & 10 & 1 & 57400 \\ 77 & 4 & 3 & 64768 \\ 61 & 0 & 1 & 54557 \\ 78 & 2 & 0 & 60523 \end{pmatrix}$$

Donde la primera columna es la cantidad X_{1i} de segmentos TCP por segundo, la segunda columna es la cantidad X_{2i} de segmentos UDP por segundo, la tercera columna es la cantidad X_{3i} de protocolos ICMP por segundo, y la cuarta columna es la suma B_i de los tamaños de los paquetes IP por segundo.

El algoritmo para procesar los datos calcula los valores de las muestras para $X_{1i}, X_{2i}, X_{3i}, B_i$ para el mismo tiempo i.

A continuación se muestra el algoritmo básico que se utilizó para preprocesar los datos generados con TCPDump, para obtener mayor claridad en la descripción del algoritmo omitimos controles que se hacen para casos especiales en las líneas de texto del archivo (el algoritmo completo aparece en el apéndice de scripts con el nombre de **Procesamiento de datos para modelo de Transporte y Red**).

```
archivo_datos = leer_archivo_datos("10.10.20.10.tcpdump");
matriz = crear_matriz();
poner_a_cero_contadores(contador_TCP, contador_UDP, contador_OTROS, contador_bytes);
```

```
mientras( archivo_datos no termine){
    linea = leer_linea(archivo_datos);
    segundo= extraer_segundo(linea);
    si (segundo <> segundo_anterior) {
        agrego_a_matriz(matriz,
            {contador_TCP,
             contador_UDP,
             contador_OTROS,
                  contador_bytes} );
        poner_a_cero_contadores(contador_TCP,
                  contador_UDP,
                  contador_OTROS,
                  contador_bytes);
         segundo_anterior = segundo;
    }
    proto = extraer_protocolo(linea);
    en caso de ( proto ){
          es = "TCP" { contador_TCP = contador_TCP + 1; }
          es = "UDP" { contador_UDP = contador_UDP + 1; }
          otro caso { contador_OTROS = contador_OTROS + 1;}
    }
    bytes = extraer_bytes_IP(linea);
    contador_bytes = contador_bytes + bytes;
}
cerrar_archivo();
guardar_matriz_en_archivo(matriz, "TCP_UDP_OTROS_BYTES.datos");
```

El resultado del algoritmo es un archivo en forma de matriz con los valores separados por comas (cada línea es una fila y cada columna esta separada por comas), este formato se generó para leer el archivo desde un programa Mathematica.

Podemos observar las mediciones realizadas graficando los valores de las columnas con la ayuda de la función Plot de Mathematica. Las gráficas muestran las características del tráfico en estudio:

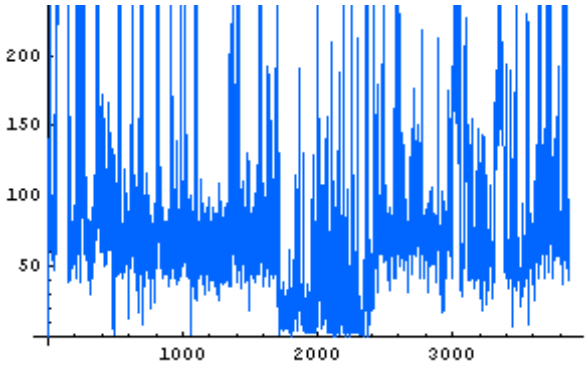

Figura 7.3: Grafico de la cantidad de segmentos TCP por segundo.

El gráfico muestra la cantidad de segmentos TCP contados por segundo, se observa que el tráfico de segmentos TCP es en ráfagas

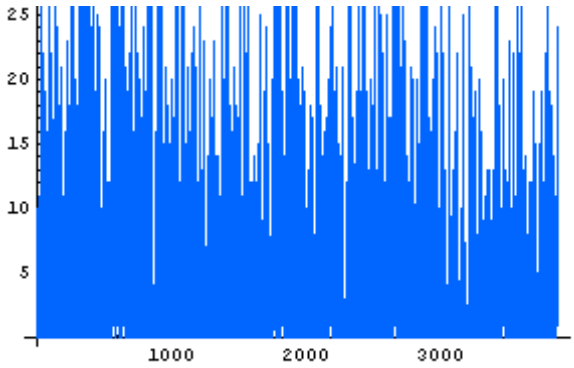

Figura 7.4: Cantidad de datagramas UDP por segundo.

El gráfico muestra la cantidad de datagramas UDP por segundo, se observa que no existe una aparente relación entre los segmentos TCP y los datagramas UDP, la cantidad de datagramas es del orden de 4 veces menor a la cantidad de segmentos TCP.

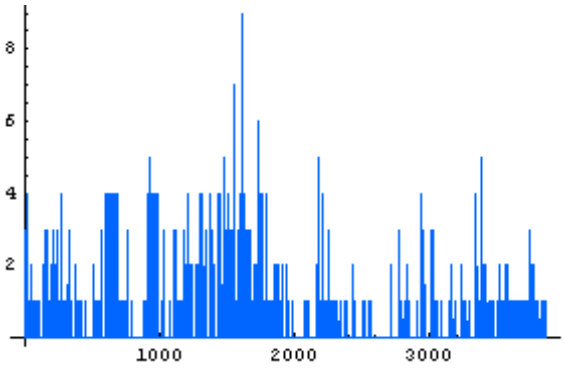

Figura 7.5: Cantidad de otros protocolos por segundo.

El gráfico muestra la cantidad de otros protocolos por segundo, en el volcado se observa que en su mayoría son protocolos ICMP, según los gráficos la cantidad de unidades por segundo es del orden de 1,5 veces menor a la cantidad de datagramas UDP y 5,5 veces menor a la cantidad de segmentos TCP.

La figura 7.6 muestra la cantidad de bytes en los paquetes IP por segundo (suma de los tamaños de los paquetes IP por segundo), se observa que existe una fuerte relación con la cantidad de segmentos TCP.

7.4. Construcción del Modelo P1P

A continuación construiremos el P1P y calculamos los coeficientes del modelo. Llamaremos a la matriz resultante del algoritmo como M, podemos expresarla como un vector de columnas, cada columna corresponde a los valores de las variables, la última a la salida del sistema:

$$M = (X_1, X_2, X_3, B)$$

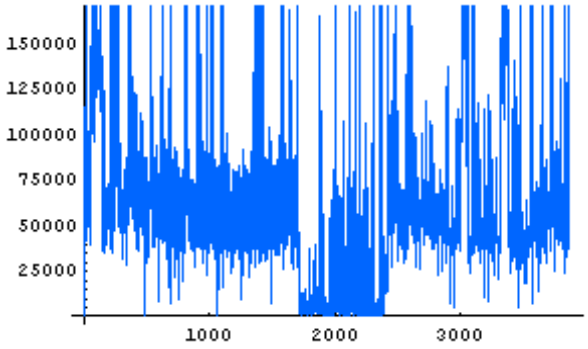

Figura 7.6: Cantidad de bytes en los paquetes IP por segundo.

Una fila F_i de esta matriz esta formada por $F_i(X_{1i}, X_{2i}, X_{3i}, B_i)$ y corresponde a los valores muestreados en el tiempo i.

- X_{ji} es un valor de variable aleatoria, $j = 1.,3$ que son las 3 variables del modelo, $i = 1..n$ es el tiempo en segundos.

- B_i es un vector columna con las sumas de los tamaños en los paquetes IP procesados en el segundo i.

De acuerdo a la fórmula general para construir el P1P (la definición general):

$$c + \sum c_i X_i \ + \sum c_{ij} X_i X_j + \sum c_{ijk} X_i X_j X_k \ + ... + c_{1,2...n} X_1 X_2 ... X_n$$

y la nomenclatura utilizada anteriormente el modelo es:

$$\begin{aligned} B \quad = \quad & c + c_1 * X_1 + c_2 * X_2 + c_3 * X_3 + \\ & c_4 * X_1 * X_2 + c_5 * X_1 * X_3 + \\ & c_6 * X_2 * X_3 + c_7 * X_1 * X_2 * X_3 \end{aligned}$$

donde:

- B : es una matriz columna con los valores de las sumas de los bytes en los paquetes IP.

- $\{c, c_1, c_2, c_3\}$: son coeficientes del polinomio P1P.

- $\{X_1, X_2, X_3\}$: son matrices columna, con los valores de: segmentos TCP (X_1), datagramas UDP (X_2), protocolos ICMP (X_3) por segundo.

Reemplazando los valores de B y las X_j por las i muestras queda el siguiente polinomio P1P:

$$\begin{pmatrix} b1 \\ b2 \\ . \\ bk \\ . \\ bn \end{pmatrix} = c +$$

$$c_1 * \begin{pmatrix} X_{11} \\ X_{12} \\ . \\ X_{1k} \\ . \\ X_{1n} \end{pmatrix} + c_2 * \begin{pmatrix} X_{21} \\ X_{22} \\ . \\ X_{2k} \\ . \\ X_{2n} \end{pmatrix} + c_3 * \begin{pmatrix} X_{31} \\ X_{32} \\ . \\ X_{3k} \\ . \\ X_{3n} \end{pmatrix} +$$

$$c_4 * \begin{pmatrix} X_{11} \\ X_{12} \\ . \\ X_{1k} \\ . \\ X_{1n} \end{pmatrix} * \begin{pmatrix} X_{21} \\ X_{22} \\ . \\ X_{2k} \\ . \\ X_{2n} \end{pmatrix} + c_5 * \begin{pmatrix} X_{11} \\ X_{12} \\ . \\ X_{1k} \\ . \\ X_{1n} \end{pmatrix} * \begin{pmatrix} X_{31} \\ X_{32} \\ . \\ X_{3k} \\ . \\ X_{3n} \end{pmatrix} +$$

$$c_6 * \begin{pmatrix} X_{21} \\ X_{22} \\ . \\ X_{2k} \\ . \\ X_{2n} \end{pmatrix} * \begin{pmatrix} X_{31} \\ X_{32} \\ . \\ X_{3k} \\ . \\ X_{3n} \end{pmatrix} + c_7 * \begin{pmatrix} X_{11} \\ X_{12} \\ . \\ X_{1k} \\ . \\ X_{1n} \end{pmatrix} * \begin{pmatrix} X_{21} \\ X_{22} \\ . \\ X_{2k} \\ . \\ X_{2n} \end{pmatrix} * \begin{pmatrix} X_{31} \\ X_{32} \\ . \\ X_{3k} \\ . \\ X_{3n} \end{pmatrix}$$

tenemos n muestras por variable y 8 incógnitas, 8 ecuaciones con ocho incógnitas: $c, c_1, c_2, c_3, c_4, c_5, c_6, c_7$ que son los parámetros del modelo, utilizando una regresión no lineal se calculan los parámetros y se consigue el modelo final.

Resolviendo la regresión no lineal con un programa en Mathematica se obtiene:

$$\begin{aligned} c &= -1525{,}8 \\ c_1 &= 758{,}059 \\ c_2 &= 244{,}178 \\ c_3 &= 244{,}178 \\ c_4 &= 0{,}526994 \\ c_5 &= 23{,}867 \\ c_6 &= -11{,}2104 \\ c_7 &= -0{,}493178 \end{aligned}$$

el modelo final es entonces:

$$\begin{aligned} B = {} & -1525{,}8 + 758{,}059 * X_1 + 244{,}178 * X_2 + 244{,}178 * X_3 + \\ & 0{,}526994 * X_1 * X_2 + 23{,}867 * X_1 * X_3 + \\ & -11{,}2104 * X_2 * X_3 + -0{,}493178 * X_1 * X_2 * X_3 \end{aligned}$$

Observamos que la variable B y las variables X_i son matrices columnas, o pueden ser escalares. Este polinomio permite estimar un valor de bytes de salida para X_1, X_2, X_3 en un tiempo t_1, o estimar valores de bytes para una matriz de valores de entrada (X_1^i, X_2^i, X_3^i) para una serie continua de tiempos $t_1, t_2, ..., t_k$.

Esta última opción permite realizar gráficas y cálculos para intervalos de tiempo, como resultado obtenemos $bytes\_estimados(b1, b2, ..., bk)$ para un conjunto de valores de $X_1(x11, x12, ..., x1n)$, $X_2(x21, x22, ..., x2n)$, $X_3(x31, x32, ..., x3n)$. En el próximo capítulo realizamos un ejemplo para los dos casos.

7.5. Estimaciones

7.5.1. Estimación de bytes por segundo

Con el modelo de tráfico podemos suponer determinadas cantidades de PDU de transporte y red para estimar la cantidad de bytes en paquetes IP. Supongamos que un cierto nodo de la red realiza una comunicación utilizando el enlace a Internet donde se tomaron los datos y necesitamos estimar la cantidad de bytes a procesar que la comunicación provoca por segundo, supongamos que la comunicación requiere de la siguiente cantidad estimada de segmentos y paquetes:

- la cantidad estimada de segmentos TCP sea 60 (X1=60)
- la cantidad estimada de datagramas UDP sea 2 (X2=2)
- la cantidad estimada de protocolos ICMP 2 (X3=2)

reemplazamos los valores en el modelo y obtenemos:

$$\begin{aligned} B &= -1525{,}8 + 758{,}059 * 60 + 244{,}178 * 2 + 244{,}178 * 2 + \\ &\quad 0{,}526994 * 60 * 2 + 23{,}867 * 60 * 2 + \\ &\quad -11{,}2104 * 2 * 2 + -0{,}493178 * 60 * 2 * 2 \\ B &= 47698{,}5 \\ B &\simeq 47699 \end{aligned}$$

esto indica que la cantidad de bytes que se estima procesar por segundo en el enlace es de $47699 bytes$.

7.5.2. Estimación de bytes en varios segundos

El modelo permite estimar la cantidad de bytes en paquetes IP para varios segundos, por ejemplo supongamos que una comunicación necesitara 6 segundos para completarse y que la cantidad de pdu TCP, UDP y otros en cada segundo es:

	Cantidad de PDU		
Seg	**TCP**	**UDP**	**ICMP**
1	60	2	2
2	64	4	1
3	79	2	1
4	74	1	0
5	60	0	1
6	58	3	2

separando las variables por columnas tenemos:

$$X_1 = \begin{pmatrix} 60 \\ 64 \\ 79 \\ 74 \\ 60 \\ 58 \end{pmatrix}, X_2 = \begin{pmatrix} 2 \\ 4 \\ 2 \\ 1 \\ 0 \\ 3 \end{pmatrix}, X_3 = \begin{pmatrix} 2 \\ 1 \\ 1 \\ 0 \\ 1 \\ 2 \end{pmatrix}$$

reemplazando en el modelo y calculando:

$$\begin{aligned} B &= -1525{,}8 + 758{,}059 * \begin{pmatrix} 60 \\ 64 \\ 79 \\ 74 \\ 60 \\ 58 \end{pmatrix} + 244{,}178 * \begin{pmatrix} 2 \\ 4 \\ 2 \\ 1 \\ 0 \\ 3 \end{pmatrix} + 244{,}178 * \begin{pmatrix} 2 \\ 1 \\ 1 \\ 0 \\ 1 \\ 2 \end{pmatrix} + \\ & 0{,}526994 * \begin{pmatrix} 60 \\ 64 \\ 79 \\ 74 \\ 60 \\ 58 \end{pmatrix} * \begin{pmatrix} 2 \\ 4 \\ 2 \\ 1 \\ 0 \\ 3 \end{pmatrix} + 23{,}867 * \begin{pmatrix} 60 \\ 64 \\ 79 \\ 74 \\ 60 \\ 58 \end{pmatrix} * \begin{pmatrix} 2 \\ 1 \\ 1 \\ 0 \\ 1 \\ 2 \end{pmatrix} + \\ & -11{,}2104 * \begin{pmatrix} 2 \\ 4 \\ 2 \\ 1 \\ 0 \\ 3 \end{pmatrix} * \begin{pmatrix} 2 \\ 1 \\ 1 \\ 0 \\ 1 \\ 2 \end{pmatrix} + -0{,}493178 * \begin{pmatrix} 60 \\ 64 \\ 79 \\ 74 \\ 60 \\ 58 \end{pmatrix} * \begin{pmatrix} 2 \\ 4 \\ 2 \\ 1 \\ 0 \\ 3 \end{pmatrix} * \begin{pmatrix} 2 \\ 1 \\ 1 \\ 0 \\ 1 \\ 2 \end{pmatrix} \\ B &= (47698{,}5, 49702{,}2, 60961{,}8, 54853{,}7, 45633{,}9, 46283{,}9) \\ B &\simeq (47699, 49703, 60962, 54854, 45634, 46284) \end{aligned}$$

el resultado indica que los valores estimados en los 6 segundos consecutivos es:

Seg	**Bytes en IP**
1	47699
2	49703
3	60962
4	54854
5	45634
6	46284

En el apéndice de *Programas Mathematica*, se muestra el programa realizado para los cálculos del modelo de tráfico de trasporte y red, nombrado **Modelo de trasporte y red**.

Modelo de tráfico a nivel de aplicación

8.1. Introducción

Para algunas aplicaciones podría no ser apto estudiar solo el tráfico a nivel de red y transporte, los modelos de tráfico en las distintas capas de TCP/IP son diferentes, incluso para diferentes protocolos de nivel de aplicación son distintos, para este nivel se podría requerir clasificar el tráfico de protocolos específicos como Radio, video streaming u otros. En esta sección realizamos un modelo P1P de tráfico para estimar los bytes por segundo considerando variables a nivel de aplicación. Se considera una variable aleatoria con tiempo fijo por puerto y analizamos los puertos bien conocidos para disminuir la complejidad del modelo. El modelo permite estimar tamaños de paquetes a partir de variables en el nivel de aplicación.

8.2. Análisis de paquetes

Como en el caso anterior utilizamos la herramienta Tcpdump y obtenemos datos de tráfico en una interfaz Ethernet conectada a un enlace ADSL a Internet, por el enlace circula tráfico de unas 200 PC´s de la LAN. El comando que se utilizo es el siguiente:

$$\#\texttt{tcpdump -n -i eth0 -s 1500 -vvv} > \texttt{10,10,20,10.tcpdump}$$

las opciones elegidas son iguales al caso anterior.
El comando genera datos de tráfico y los envía a un archivo, se muestran las primeras líneas del archivo:

1. 18:10:03.000842 IP (tos 0x0, ttl 116, id 9296, offset 0, flags [DF], proto: TCP (6), length: 1472) 200.42.92.5.554 > 10.10.20.10.1108: . 3969520650:3969522082(1432) ack 209013619 win 64197

2. 18:10:03.001296 IP (tos 0x0, ttl 126, id 7663, offset 0, flags [DF], proto: TCP (6), length: 40) 10.10.20.10.1108 > 200.42.92.5.554: ., cksum 0x08ee (correct), 1:1(0) ack 1432 win 65535

3. 18:10:03.012978 IP (tos 0x0, ttl 116, id 9297, offset 0, flags [DF], proto: TCP (6), length: 1472) 200.42.92.5.554 > 10.10.20.10.1108: . 1432:2864(1432) ack 1 win 64197

4. 18:10:03.024873 IP (tos 0x0, ttl 116, id 9298, offset 0, flags [DF], proto: TCP (6), length: 1472) 200.42.92.5.554 > 10.10.20.10.1108: . 2864:4296(1432) ack 1 win 64197

5. 18:10:03.025327 IP (tos 0x0, ttl 126, id 7664, offset 0, flags [DF], proto: TCP (6), length: 40) 10.10.20.10.1108 > 200.42.92.5.554: ., cksum 0xfdbd (correct), 1:1(0) ack 4296 win 65535

a continuación se explican cada uno de los campos y se remarcan en negrita los que se utilizan para clasificar los protocolos en el nivel de aplicación:

- dato de tiempo **18:10:03.001296**, tiempo: se utiliza para procesar los datos por segundo.

- dato del paquete IP(...) indica el valor de cada campo IP
  - tos 0x0: Type Of Services
  - ttl 116: Time to Live "
  - id 9296: identificador del paquete IP
  - offset 0: se utiliza para fragmentación de los paquetes
  - flags [DF]: banderas IP
  - proto: TCP(6): campo que indica el protocolo que encapsula IP
  - **length: 1472**: longitud del paquete en octetos
- **200.42.92.5**.*554* > **10.10.20.10**.*1108*: se indica la IP origen del lado izquierdo al signo > y la IP destino del lado derecho, el puerto origen y destino se indica luego del ultimo punto de cada IP: IP origen.*puerto origen* > Ip destino.*puerto destino*.

8.3. Clasificación por puerto

8.3.1. Descripción

Una vez generado el archivo con los datos del volcado tcpdump, se proceso la información para clasificarla según el puerto. Realizamos un algoritmo para procesar los datos considerando los puertos menores a 1024, para cada paquete analizamos el puerto origen, destino, o puede no tener puerto (ej. algunos paquetes ICMP), los paquetes pueden ser salientes o entrantes de la red, conociendo la dirección IP de la red y analizando los campos de dirección IP origen y destino podemos saber si el paquete es de envío o de recepción, evaluamos los paquetes donde alguno de los puertos (origen o destino) sea menor a 1024. Por ejemplo para la línea 5 del volcado anterior tenemos que el paquete tiene las siguientes direcciones:

$$10{,}10{,}20{,}10{,}1108 > 200{,}42{,}92{,}5{,}554$$

consideramos entonces que el paquete pertenece al puerto 554 que aparece en la dirección destino. Para la línea 6 del volcado tenemos las siguientes direcciones:

$$200{,}42{,}92{,}5{,}554 > 10{,}10{,}20{,}10{,}1108$$

en este caso el paquete pertenece también al puerto 554 según se indica en la dirección origen.

El algoritmo cuenta los paquetes por segundo de acuerdo al puerto, recorre el archivo de volcado y como resultado genera una matriz donde las columnas corresponden a la cantidad de paquetes pertenecientes a un puerto por segundo. El programa completo en php esta en el apéndice nombrado **Procesamiento de datos para el modelo de Aplicación** pero damos a continuación una descripción en seudocódigo y un detalle de las tareas que realiza más abajo.

8.3.2. Algoritmo de clasificación por puerto en seudo-código

```
archivo_datos = leer_archivo_datos("10.10.20.10.tcpdump");
lista_contadores = crear_lista_de_contadores();
matriz = crear_matriz_de_datos();
mientras( archivo_datos no termine){
    linea = leer_linea(archivo_datos);
    segundo = extraer_segundo(linea);
    si (segundo <> segundo_anterior) {
```

```
            agrego_a_matriz_datos(matriz, lista_contadores,contador_bytes );
          poner_a_cero_contadores(lista_contadorres, contador_bytes );
            segundo_anterior = segundo;
      }
       puerto_x = extraer_puerto_de_paquete(linea);
      Incrementar_contador_puerto(puerto_x);
      bytes = extraer_bytes_IP(linea);
      contador_bytes = contador_bytes + bytes;
  }
  cerrar_archivo();
  guardar_matriz_en_archivo(matriz, "analisis_puertos.datos");
```

Tareas que realiza el algoritmo

- Generamos una lista dinámica de contadores para contar los paquetes por segundo de cada puerto, inicialmente esta vacía.
- Generamos una matriz de columnas dinámicas donde se guardaran los datos de paquetes por segundo por puerto.
- Recorremos el archivo de datos y para cada línea realizamos:
  - extraemos la marca de tiempo
  - si la marca no pertenece al segundo actual:
    - cargamos los contadores en la matriz de datos
    - ponemos contadores a cero
    - ponemos el total de bytes a cero
    - actualizamos el segundo
  - sino:
    - extraemos el puerto X del paquete
    - incrementamos el contador del puerto X
    - extraemos el tamaño del paquete
    - sumamos el tamaño del paquete al total

Se obtuvo como resultado que el tráfico involucra paquetes de 11 puertos y paquetes sin puerto, la siguiente tabla muestra una parte de la matriz de datos M_o con los datos obtenidos:

```
38,2,23,2,2,1,0,0,0,0,0,0,59032
32,0,101,1,1,7,0,0,0,0,0,0,115695
40,10,27,1,1,6,0,0,0,0,0,0,57400
38,4,34,3,3,2,0,0,0,0,0,0,64768
40,0,20,1,1,0,0,0,0,0,0,0,54557
38,2,40,0,0,0,0,0,0,0,0,0,60523
38,1,28,2,2,0,0,0,0,0,0,0,53787
40,2,30,1,1,0,0,0,0,0,0,0,58860
40,0,23,0,0,0,0,0,0,0,0,0,52564
40,4,63,1,1,0,0,0,0,0,0,0,85841
40,0,93,0,0,0,0,0,0,0,0,0,90982
40,0,65,4,4,0,0,0,0,0,0,0,76614
```

Las columnas separadas por comas corresponden a las cantidades X_{ij} de paquetes en los puertos y los puertos de cada columna son:

```
554,53,80;sin puerto,25,443,123,160,161,110,162,163,cont_bytes
```

la última columna corresponde a la cantidad de bytes en paquetes IP.

Las Figuras 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 8.10, 8.11, 8.12 muestran la cantidad de paquetes por segundo por puerto, el eje x indica el tiempo en segundos, el eje y indica la cantidad de paquetes.

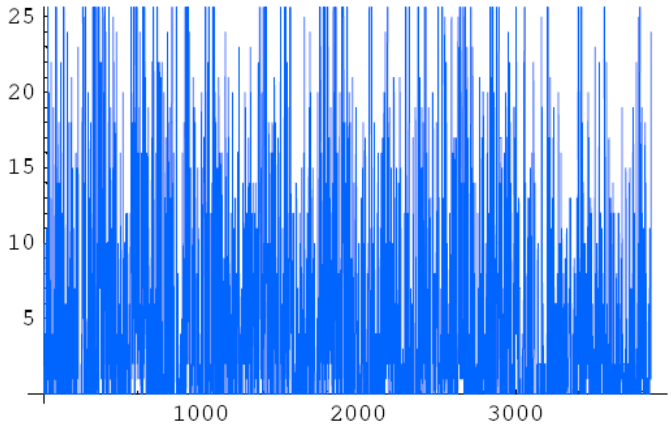

Figura 8.1: Paquetes por segundo al puerto 554 (protocolo RTSP)

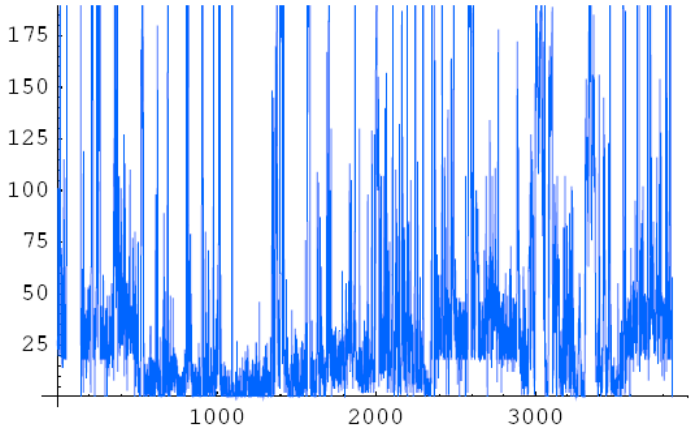

Figura 8.2: Paquetes por segundo pertenecientes al puerto 53 (protocolo DNS).

en las gráficas se observa que el modelo de tráfico para cada puerto es claramente diferente.

Veamos como construir el modelo P1P por puerto.

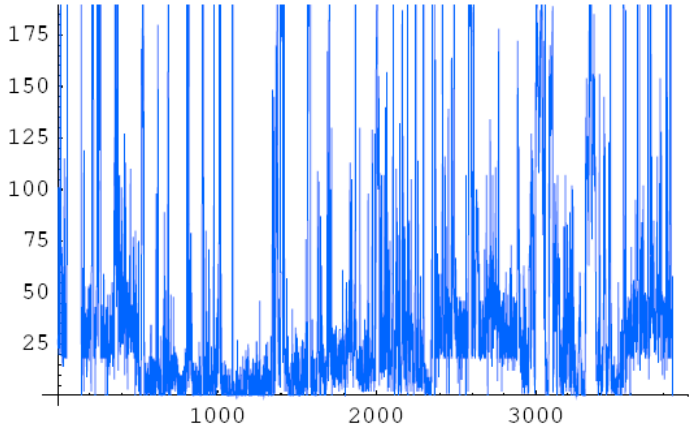

Figura 8.3: Paquetes por segundo al puerto 80, protocolo HTTP.

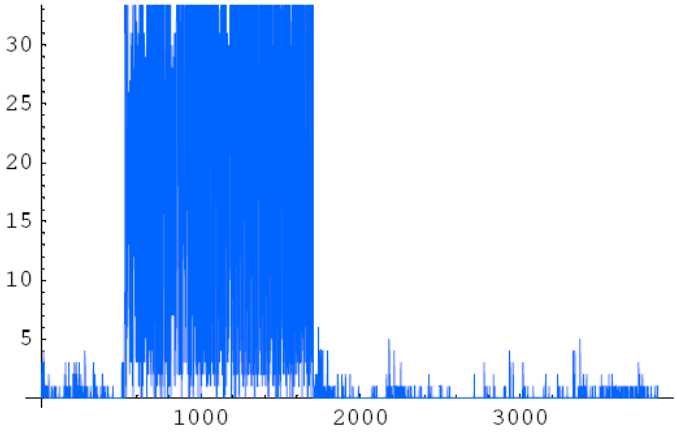

Figura 8.4: Paquetes por segundo, sin puerto.

Del resultado de procesamiento obtenemos una matriz de datos M_o que podemos expresarla en forma teórica como una matriz donde las columnas son las variables:

$$M_o = (X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11}, X_{12}, B)$$

cada X_i corresponde a la columna, y la variable B es la cantidad de bytes procesados.

Si construimos el modelo P1P de acuerdo a la fórmula general

$$c + \sum c_i X_i \ + \sum c_{ij} X_i X_j + \sum c_{ijk} X_i X_j X_k \ + ... + c_{1,2...n} X_1 X_2 ... X_n$$

para todos los términos i, j y k diferentes, obtenemos un modelo de 12 variables formado por $2^{12} = 4096$ términos, construir manualmente un P1P con 4096 términos resulta propenso a errores y es extenso, por esto se utilizó un algoritmo para generarlo de forma automática.

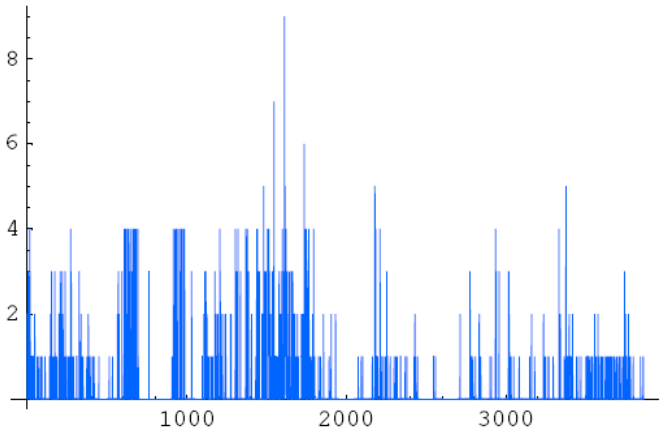

Figura 8.5: Paquetes por segundo pertenecientes al puerto 25 (protocolo SMTP)

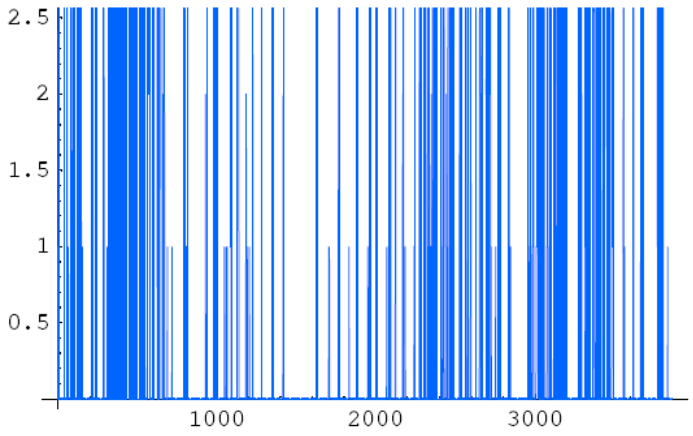

Figura 8.6: Paquetes por segundo al puerto 443, protocolo HTTPS.

8.4. Algoritmos para generar un P1P con n variables.

8.4.1. Método de evaluación de combinaciones

Los primeros términos son una constante y cada una de las variables multiplicada por una constante, los siguientes términos se construyen a partir de la generación de las combinaciones de índices de las variables, por ejemplo si tenemos 2 variables (X_1, X_2) los primeros términos son $c_0 + c_1 * X_1 + c_2 * X_2$ para los demás términos podemos realizar las combinaciones de 2 elementos (i, j) y a partir de ellas armar los términos del P1P:

i	j	Constante	Término
1	1	1	$c_1 * X_1 * X_1$
1	2	2	$c_2 * X_1 * X_2$
2	1	3	$c_3 * X_2 * X_1$
2	2	4	$c_4 * X_2 * X_2$

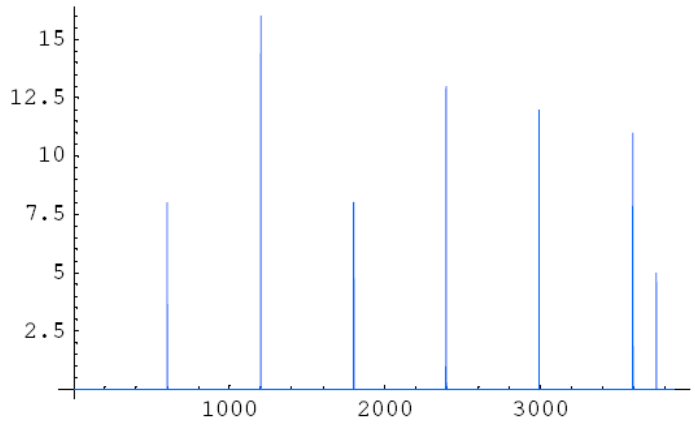

Figura 8.7: Paquetes por segundo al puerto 123 (protocolo NTP)

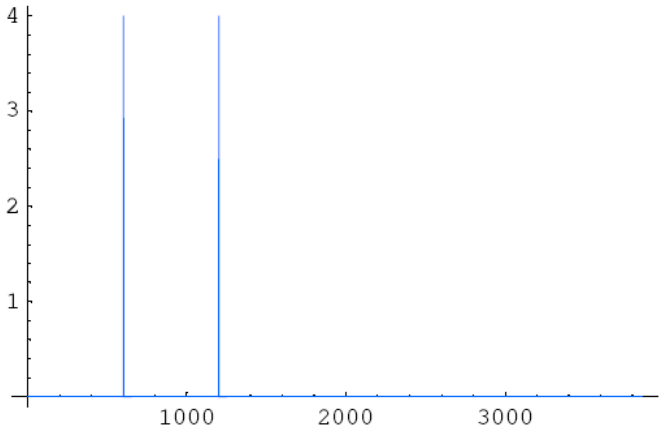

Figura 8.8: Paquetes por segundo al puerto 160 (protocolo SGMP-TRAPS)

utilizando este listado y eliminando los términos donde $i = j$ y donde $i > j$ para suprimir los términos repetidos como (1,2) y (2,1) obtenemos :

i	j	Constante	Término
1	2	2	$c_2 * X_1 * X_2$

finalmente el P1P final es

$$c_0 + c_1 * X_1 + c_2 * X_2 + c_3 * X_1 * X_2$$

8.4.2. Método de las combinaciones binarias

Otra forma de generar los términos sin necesidad de evaluar las condiciones $i = j$ y $i > j$, es utilizar las combinaciones de los números binarios, el procedimiento es el siguiente: tomamos una cantidad de bits igual a la cantidad de variables, generamos las combinaciones y luego reemplazamos los unos por la variable con índice correspondiente a la posición del uno (1), por ejemplo:

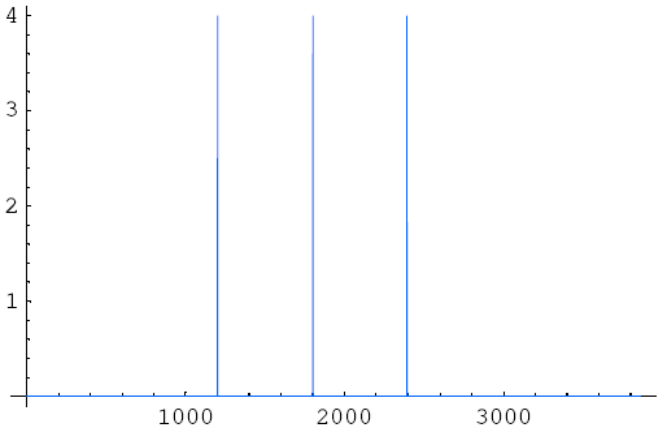

Figura 8.9: Paquetes por segundo al puerto 161, protocolo SNMP.

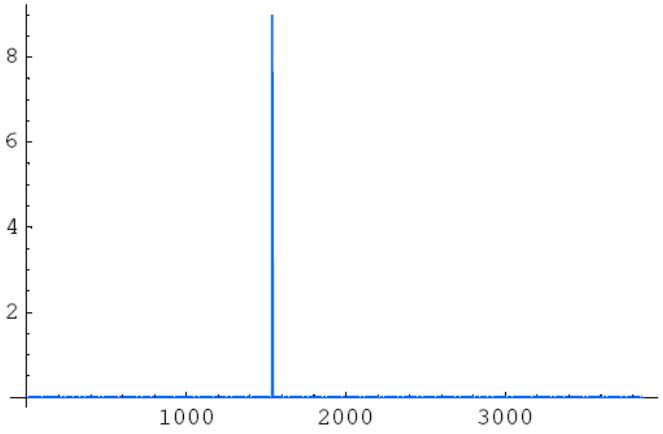

Figura 8.10: Paquetes al puerto 110 (protocolo POP)

consideremos 3 variables X_1, X_2, X_3 realizando las combinaciones de los 3 bits tenemos:

1	2	3	Término
0	0	0	c_0
0	0	1	$c_3 * X_3$
0	1	0	$c_2 * X_2$
0	1	1	$c_{23} * X_2 * X_3$
1	0	0	$c_1 * X_1$
1	0	1	$c_{13} * X_1 * X_3$
1	1	0	$c_{12} * X_1 * X_2$
1	1	1	$c_{123} * X_1 * X_2 * X_3$

luego sumamos los términos:

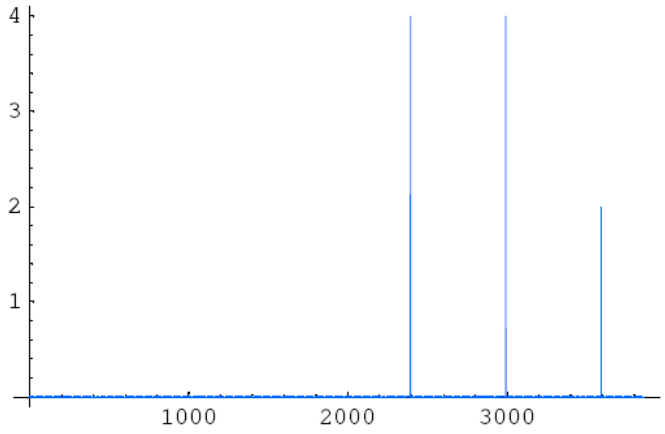

Figura 8.11: Paquetes al puerto 162 (protocolo SNMP-TRAP)

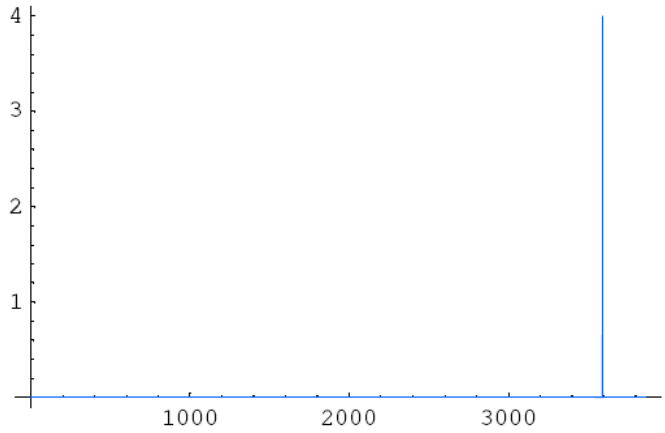

Figura 8.12: Paquetes al puerto 163 (protocolo CMIP-MAN)

$$
\begin{aligned}
P1P \quad = \quad & c_0 + \\
& c_3 * X_3 + \\
& c_2 * X_2 + \\
& c_{23} * X_2 * X_3 + \\
& c_1 * X_1 + \\
& c_{13} * X_1 * X_3 + \\
& c_{12} * X_1 * X_2 + \\
& c_{123} * X_1 * X_2 * X_3
\end{aligned}
$$

podemos ordenar los términos utilizando el índice de los coeficientes y la cantidad de variables de los términos, obtenemos el P1P ordenado:

$$\begin{aligned} P1P &= c_0 + c_1 * X_1 + c_2 * X_2 + c_3 * X_3 + \\ & c_{12} * X_1 * X_2 + c_{13} * X_1 * X_3 + c_{23} * X_2 * X_3 + \\ & c_{123} * X_1 * X_2 * X_3 \end{aligned}$$

El algoritmo para generar el P1P como una cadena de caracteres es el siguiente:

- generar números decimales i_d entre 0 y 2^n
- para cada número decimal i_d generado:
  - pasar i_d a binario y guardarlo en i_b
  - para cada símbolo del número binario i_b armar un termino del P1P como:
    - concatenar la letra c con el número decimal generado i_d
    - concatenar con la cadena anterior:
    - para cada símbolo 1 en el número binario i_d , concatenar una cadena $*X$ con el número correspondiente a la posición del símbolo uno.
  - al terminar la cadena concatenar un signo $+$

a continuación mostramos el código en lenguaje PHP:

```
function genera_P1P($n_var = 0){
   $P1P= ”;
   $const = 1;
   $P1P = ’c0 +<br>’;
   for($i=1;$i<pow(2,$n_var);$i++){
   $combi = str_pad( decbin($i), $n_var, ’0’, STR_PAD_LEFT ) . ’<br>’;
   //echo $combi . ’<br>’;
         $P1P =  $P1P. ’ c’.$const;
         for($j=0;$j<$n_var;$j++){
         $uno = substr($combi,$j,1);
         //echo $uno . ’<br>’;
   if( $uno == ’1’ ){
         $P1P =  $P1P. ’ * X’. trim($j+1);
         }
         }
         $const++;
         $P1P = $P1P .’ + <br>’;
   }
   return $P1P;
 }
```

Con este algoritmo se generan P1P de n variables facilitando la creación de polinomios con muchos términos.

8.5. Reducción de la cantidad de variables del P1P

En el modelado con P1P la cantidad de variables involucradas surge en la definición del sistema a resolver. La cantidad de términos (ct) en el modelo crece exponencialmente con el aumento de la cantidad n de variables. La cantidad de términos de un P1P es $ct(n) = 2^n$. Un problema abierto en la teoría de los P1P es encontrar una forma de reducir las variables justificando cual es importante consideran y cual se pueden despreciar.

En este modelo de aplicación surgió del análisis de paquetes considerar 12 variables (12 puertos) para la construcción del P1P. Computacionalmente es complejo trabajar con $2^{12} = 4096$ términos, por esto es conveniente reconsiderar la relevancia de las variables involucradas, disminuir la cantidad de variables y de esta forma reducir la complejidad al mínimo posible.

Para el modelado de tráfico una selección intuitiva es utilizar las gráficas, en las gráficas anteriores se observa que algunos puertos no son significativos en relación al tráfico total, los puertos 160, 161, 110, 162, 163 registran pocos paquetes en la muestra, sin embargo si los paquetes tienen grandes tamaños puede ser importante considerarlos. A continuación proponemos un método para evaluar y decidir sobre la representatividad de las variables. El método utiliza los mismos datos de la muestra.

8.5.1. Matriz de contribuciones

El método de la Matriz de contribuciones consiste en utilizar la matriz de datos M_o, derivar de ésta una segunda matriz M_{by}, calcular las contribuciones como porcentajes sobre el total y promediarlos. El método tiene las siguientes etapas:

- Calcular por fila en que porcentaje contribuye cada variable al total (porcentaje por fila de cada muestra sobre el total)
- Luego calcular un promedio de los porcentajes para todas las muestras.
- Finalmente seleccionamos las variables con un porcentaje promedio mayor a una cota razonablemente elegida.

Vamos a reducir la cantidad de variables para nuestro modelo de aplicación de 12 variables. Obtenemos una matriz donde las columnas son el tamaño en bytes de los paquetes por segundo pertenecientes a cada puerto, esta matriz se obtuvo con el algoritmo presentado anteriormente (*Algoritmo de clasificación por puerto*) pero cambiando la lista de contadores por una lista de totales. La matriz de bytes por puerto tiene la forma:

$$M_{by} = (B_{p1}, B_{p2}, B_{p3}, B_{p4}, ..., B_{pn}, B)$$

donde cada B_{pi} es una columna que indica los bytes por segundo de los paquetes con cada puerto, cada i corresponde a un puerto determinado, la variable B es la cantidad total de bytes. A continuación mostramos tres filas de la matriz calculada M_{by}, la primer fila indica los puertos a los que pertenecen los bytes, la ultima columna es el total de bytes por fila:

554	**53**	**80**	**sp**	**25**	**443**	**123**	**160**	**161**	**110**	**162**	**163**	**bytes(B)**
35184	180	19468	1152	3000	48	0	0	0	0	0	0	59032
29090	0	81307	576	1500	3222	0	0	0	0	0	0	115695
36720	888	14560	576	1500	3156	0	0	0	0	0	0	57400

Calculamos la contribución de los paquetes por puerto al tamaño total de bytes procesados (B), para cada fila j de la matriz M_{by} calculamos un porcentaje de contribución de los paquetes por puertos, de la forma:

$$c_{pij} = \frac{B_{pij}}{B_j}$$

calculando para todos los puertos se obtiene una matriz M_c con la contribución de los paquetes de cada puerto al total de bytes por segundo. La matriz de contribuciones tiene la forma:

$$M_c = (C_{p1}, C_{p2}, C_{p3}, C_{p4}, ..., C_{pn})$$

donde cada C_{pi} es una columna que indica las contribuciones por segundo de cada puerto, cada i corresponde a un puerto, la variable B es una columna con las cantidades totales de bytes en cada fila. Para cada fila j de la matriz tenemos que la suma de las contribuciones de los puertos es uno:

$$\sum_{k=1}^{n} c_{pkj} = \sum_{k=1}^{n} \frac{B_{pkj}}{B_j} = 1$$

las tres filas en la matriz de contribuciones M_c que corresponden a las tres filas anteriores de la matriz M_{by} son:

554	**53**	**80**	**sp**	**25**	**443**	**123**	**160**	**161**	**110**	**162**	**163**
$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{X}_3$	$\mathbf{X}_4$	$\mathbf{X}_5$	$\mathbf{X}_6$	$\mathbf{X}_7$	$\mathbf{X}_8$	$\mathbf{X}_9$	$\mathbf{X}_{10}$	$\mathbf{X}_{11}$	$\mathbf{X}_{12}$
0,5960	0,0030	*0,3297*	0,0195	*0,0508*	0,0008	0	0	0	0	0	0
0,2514	0	*0,7027*	0,0049	0,0129	0,0278	0	0	0	0	0	0
0,6397	0,0154	*0,2536*	0,0100	0,0261	*0,0549*	0	0	0	0	0	0

en la primer fila hacemos referencia al puerto y a la variable que corresponde, también mostramos solo 4 decimales para visualizar mejor la tabla.

Selección de variables

Como ejemplo ilustrativo consideramos para las tres filas anteriores las variables con un porcentaje mínimo de participación. Seleccionemos las variables con valores mayores a un 5 % (0.05), esto reduce la cantidad de variables en:

- Para la primer fila a 3 variables (X_1, X_3 y X_5),
- Para la segunda fila 2 variables (X_1 y X_3),
- Para la tercer fila también 3 variables (X_1, X_3 y X_6)

Necesitamos seleccionar las variables según todos los datos en la matriz de contribuciones, para esto calculamos el promedio de participación para todas las columnas de la matriz M_c, por columna C_{p1} calculamos:

$$P_{pi} = \frac{\sum_{k=1}^{n} c_{pik}}{n}$$

donde n es el total de muestras.

Calculando los valores para toda la matriz se obtienen las siguientes contribuciones:

Puerto,Variable	**Contribución**
$554, X_1$	**0,483935273209**
$53, X_2$	**0,0292370644889**
$80, X_3$	**0,378299811376**
sp, X_4	**0,0778385216085**
$25, X_5$	0,011429077175
$443, X_6$	**0,0187166186778**
$123, X_7$	0,000188130802427
$160, X_8$	$2{,}54665008143E-6$
$161, X_9$	$4{,}95535965022E-5$
$110, X_{10}$	$3{,}40826350226E-5$
$162, X_{11}$	$8{,}14243900514E-6$
$163, X_{12}$	$2{,}24414674854E-6$

Tomando como criterio seleccionar las variables con contribuciones mayores al $1,5\,\%$, tenemos entonces que los puertos a considerar son 5 $(554, 53, 80, sp, 443)$ y las variables a considerar son 5. El modelo se reduce de 4096 términos a $2^5 = 32$ términos.

8.6. Modelo P1P de aplicación reducido

Armamos el modelo P1P considerando las 5 variables correspondientes a los puertos $(554, 53, 80, sp, 443)$. Construimos una nueva matriz de datos M solo con las columnas de la matriz de datos M_o correspondientes a las variables seleccionadas. Renumeramos las variables de 1 a 5, podemos expresar la matriz de datos M como un vector de columnas

$$M = (X_1, X_2, X_3, X_4, X_5, B)$$

Una fila F_k de esta matriz esta formada por $F_k(X_{1k}, X_{2k}, X_{3k}, X_{4k}, X_{5k}, B_k)$.

- Las X_{ik} son vectores columnas con $i = 1..5$ y $k = 1..n$.
- B_k es un vector columna con las sumas de los tamaños en los paquetes IP procesados en el segundo k.

De acuerdo a la fórmula general para construir el P1P, tenemos $2^5 = 32$ términos, y generando el P1P con el método de las combinaciones binarias tenemos:

$$\begin{aligned} B \;=\; & c0 + c1 * X5 + c2 * X4 + c3 * X4 * X5 + c4 * X3 + c5 * X3 * X5 + c6 * X3 * X4 + \\ & c7 * X3 * X4 * X5 + c8 * X2 + c9 * X2 * X5 + c10 * X2 * X4 + c11 * X2 * X4 * X5 + \\ & c12 * X2 * X3 + c13 * X2 * X3 * X5 + c14 * X2 * X3 * X4 + c15 * X2 * X3 * X4 * X5 + \\ & c16 * X1 + c17 * X1 * X5 + c18 * X1 * X4 + c19 * X1 * X4 * X5 + c20 * X1 * X3 + \\ & c21 * X1 * X3 * X5 + c22 * X1 * X3 * X4 + c23 * X1 * X3 * X4 * X5 + c24 * X1 * X2 + \\ & c25 * X1 * X2 * X5 + c26 * X1 * X2 * X4 + c27 * X1 * X2 * X4 * X5 + \\ & c28 * X1 * X2 * X3 + c29 * X1 * X2 * X3 * X5 + c30 * X1 * X2 * X3 * X4 + \\ & c31 * X1 * X2 * X3 * X4 * X5 \end{aligned}$$

donde:

- B : es una matriz columna con los valores de las sumas de los bytes en los paquetes IP.
- $\{c_o, c_1, c_2, ..., c_{31}\}$: son coeficientes del polinomio P1P.

- $\{X_1, X_2, ..., X_5\}$: son matrices columna, con los valores de los paquetes en los puertos.

Reemplazando los valores de B y las X_j por las i muestras queda el siguiente polinomio P1P:

$$\begin{pmatrix} b1 \\ b2 \\ . \\ bk \\ . \\ bn \end{pmatrix} = c_0 + c_1 * \begin{pmatrix} X_{51} \\ X_{52} \\ . \\ X_{5k} \\ . \\ X_{5n} \end{pmatrix} + ... + c_{31} * \begin{pmatrix} X_{11} \\ X_{12} \\ . \\ X_{1k} \\ . \\ X_{1n} \end{pmatrix} * \begin{pmatrix} X_{21} \\ X_{22} \\ . \\ X_{2k} \\ . \\ X_{2n} \end{pmatrix} * \begin{pmatrix} X_{31} \\ X_{32} \\ . \\ X_{3k} \\ . \\ X_{3n} \end{pmatrix} * \begin{pmatrix} X_{41} \\ X_{42} \\ . \\ X_{4k} \\ . \\ X_{4n} \end{pmatrix} * \begin{pmatrix} X_{51} \\ X_{52} \\ . \\ X_{5k} \\ . \\ X_{5n} \end{pmatrix}$$

tenemos entonces 32 ecuaciones con 32 incógnitas $c_0, ..., c_{31}$; utilizando una regresión no lineal se calculan los parámetros y se consigue el modelo final (el programa *Mathematica* está en *Apéndices* con el nombre **Modelo de Aplicación**).

Resolviendo la regresión no lineal se obtiene:

$$c0 = -10982{,}814770361423;$$
$$c1 = 473{,}9213302845348;$$
$$c2 = 1144{,}3941517453077;$$
$$c3 = 24{,}557846247006697;$$
$$c4 = 784{,}9603752916984;$$
$$c5 = 13{,}904245184349369;$$
$$c6 = 1{,}8665668990245798;$$
$$c7 = -0{,}8085290118174608;$$
$$c8 = 113{,}14482417885597;$$
$$c9 = 44{,}14585619610631;$$
$$c10 = -10{,}607820923314799;$$
$$c11 = -1{,}8244385858594567;$$
$$c12 = -0{,}8333782760414918;$$
$$c13 = -2{,}2180546460278;$$
$$c14 = 0{,}19393113670488074;$$
$$c15 = 0{,}11391877991246326;$$
$$c16 = 1068{,}2335782691548;$$
$$c17 = 10{,}165143802277731;$$
$$c18 = -6{,}210557008079236;$$
$$c19 = -0{,}7907491705171666;$$
$$c20 = -1{,}657139454770654;$$

$$
\begin{aligned}
c21 &= -0{,}49805697194762655;\\
c22 &= 0{,}08924615409005157;\\
c23 &= 0{,}030468937706188835;\\
c24 &= 1{,}280677545467686;\\
c25 &= -1{,}1807614678540275;\\
c26 &= 0{,}18626926722275175;\\
c27 &= 0{,}0575086454333244;\\
c28 &= -0{,}05062294572170121;\\
c29 &= 0{,}06202406745585728;\\
c30 &= -0{,}0018210839332631497;\\
c31 &= -0{,}0026072128388541756;
\end{aligned}
$$

Reemplazando los coeficientes, el modelo final es:

$$
\begin{aligned}
B \;=\; & -10982{,}8 + 1068{,}23X1 + 113{,}145X2 + 1{,}28068X1X2 + 784{,}96X3 - 1{,}65714X1X3 - 0{,}833378X2X3 - \\
& 0{,}0506229X1X2X3 + 1144{,}39X4 - 6{,}21056X1X4 - 10{,}6078X2X4 + 0{,}186269X1X2X4 + \\
& 1{,}86657X3X4 + 0{,}0892462X1X3X4 + 0{,}193931X2X3X4 - 0{,}00182108X1X2X3X4 + 473{,}921X5 + \\
& 10{,}1651X1X5 + 44{,}1459X2X5 - 1{,}18076X1X2X5 + 13{,}9042X3X5 - 0{,}498057X1X3X5 - \\
& 2{,}21805X2X3X5 + 0{,}0620241X1X2X3X5 + 24{,}5578X4X5 - 0{,}790749X1X4X5 - 1{,}82444X2X4X5 + \\
& 0{,}0575086X1X2X4X5 - 0{,}808529X3X4X5 + 0{,}0304689X1X3X4X5 + \\
& 0{,}113919X2X3X4X5 - 0{,}00260721X1X2X3X4X5
\end{aligned}
$$

8.7. Pruebas y gráficas

Para probar si la selección de variables y el modelo es una buena aproximación utilizamos los mismos datos reales de entrenamiento, graficamos el resultado estimado con el P1P y los comparamos con los bytes reales obtenidos de la muestra.

La figura 8.13 muestra los valores estimados de bytes con el P1P, tomando valores de entradas X_1, X_2, X_3, X_4, X_5 reales (de la muestra):

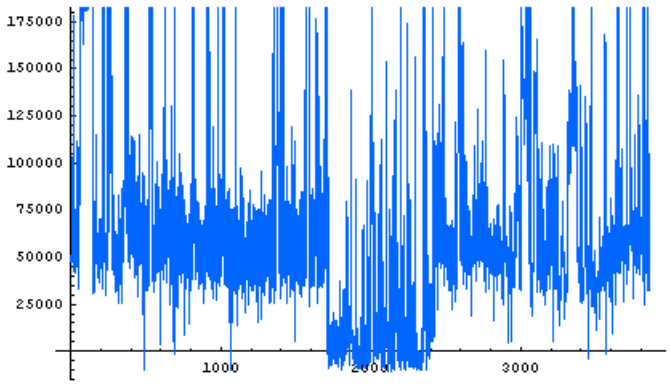

Figura 8.13: Bytes estimados por segundo con el modelo P1P de 5 variables

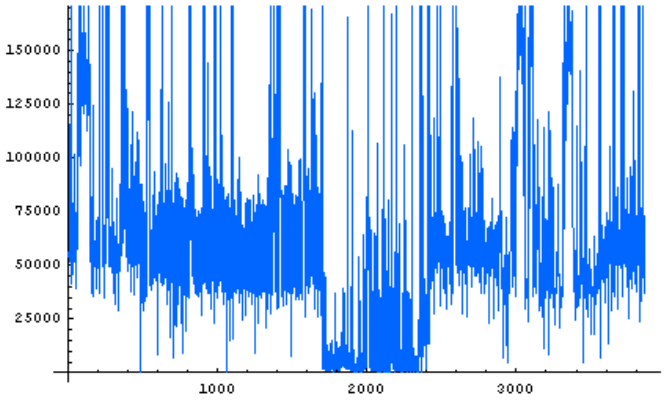

Figura 8.14: Gráfica de los bytes reales obtenidos en la muestra.

La Figura 8.14 muestra los valores de bytes obtenidos en la muestra:

diferencia absoluta entre las dos curvas

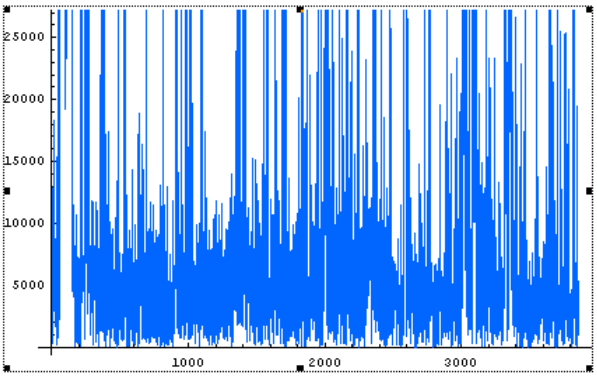

Figura 8.15: Diferencia entre las curvas de bytes estimado y reales.

Si realizamos una comparación entre las gráficas vemos que el P1P reducido mantiene el patrón de tráfico, vemos también que la diferencia entre las dos curvas muestra un error del orden del $25000/150000 = 0{,}166\,67$ aproximadamente un 17 % como máximo.

Metamodelado con P1P

En esta sección damos un marco teórico metodológico para la construcción de un metamodelo P1P.

9.1. Introducción

En los casos anteriores construimos P1Ps para obtener valores estimados de salida cuando las variables de entrada toman valores críticos, con un metamodelo podemos estimar valores de salida para casos donde las variables de entrada son también valores estimados. El metamodelo consiste en construir un P1P con variables que son resultados de otros P1P, la figura a continuación grafica esta idea:

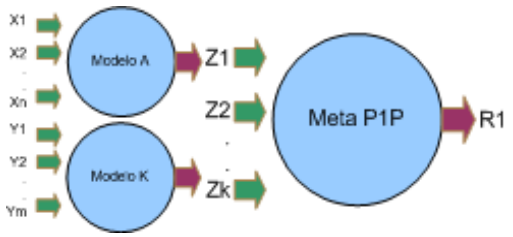

Figura 9.1: Esquema general de un MetaP1P con dos sistemas.

La gráfica muestra un meta P1P con variables de entrada Z_i y salida R_1, las variables Z_i son P1P's con diferentes entradas, se muestra el P1P **Modelo A** con variables de entrada X_i y el P1P **Modelo K** con variables de entrada Y_i.

Un ejemplo sencillo de metamodelo es:

supongamos que construimos dos modelos P1P:

1. $P_1(X_1, X_2, Z_1)$ con 2 variables de entrada X_1, X_2, y salida Z_1
2. $P_2(Y_1, Y_2, Z_2)$ con 2 variables de entrada Y_1, Y_2, y salida Z_2

por definición cada uno de los P1P es:

1. $Z_1 = c_0 + c_1 * X_1 + c_2 * X_2 + c_3 * X_1 * X_2$ donde las c_i son los coeficientes
2. $Z_2 = d_0 + d_1 * Y_1 + d_2 * Y_2 + d_3 * Y_1 * Y_2$ donde las d_i son los coeficientes

El metamodelo construido a partir de estos dos modelos P1P es $M(Z_1, Z_2, R)$,en el cual $Z1$ y Z_2 son las variables de entrada y R la salida, su ecuación es:

$$R = k_0 + k_1 * Z_1 + k_2 * Z_2 + k_3 * Z_1 * Z_2$$

con coeficientes k_i.

Realizar un metamodelo es similar a construir un único modelo considerando todas las variables de entrada, sin embargo el metamodelo da la posibilidad de estudiar los subsistemas de forma independiente y la capacidad de estimar el resultado final con los resultados parciales estimados de los subsistemas que lo componen.

El modelo $M(Z_1, Z_2, R)$ es equivalente a un modelo $p1p(X_1, X_2, Y_1, Y_2, R)$

Veamos que un metamodelo es similar a un P1P considerando todas las variables de entrada.

Si reemplazamos las variables Z_1 y Z_2 por los modelos obtenemos:

$$\begin{aligned} R &= k_0 + k_1 * (c_0 + c_1 * X_1 + c_2 * X_2 + c_3 * X_1 * X_2) + \\ &\quad k_2 * (d_0 + d_1 * Y_1 + d_2 * Y_2 + d_3 * Y_1 * Y_2) + \\ &\quad k_3 * (c_0 + c_1 * X_1 + c_2 * X_2 + c_3 * X_1 * X_2) * (d_0 + d_1 * Y_1 + d_2 * Y_2 + d_3 * Y_1 * Y_2) \end{aligned}$$

aplicando propiedad distributiva en los tres términos principales del P1P tenemos:

$$\begin{aligned} R &= k_0 + (k_1 * c_0 + k_1 * c_1 * X_1 + k_1 * c_2 * X_2 + k_1 * c_3 * X_1 * X_2) + \\ &\quad (k_2 * d_0 + k_2 * d_1 * Y_1 + k_2 * d_2 * Y_2 + k_2 * d_3 * Y_1 * Y_2) + \\ &\quad (k_3 * c_0 * d_0 + k_3 * c_0 * d_1 * Y_1 + k_3 * c_0 * d_2 * Y_2 + k_3 * c_0 * d_3 * Y_1 * Y_2) + \\ &\quad (k_3 * c_1 * X_1 * d_0 + k_3 * c_1 * X_1 * d_1 * Y_1 + k_3 * c_1 * X_1 * d_2 * Y_2 + k_3 * c_1 * X_1 * d_3 * Y_1 * Y_2) + \\ &\quad (k_3 * c_2 * X_2 * d_0 + k_3 * c_2 * X_2 * d_1 * Y_1 + k_3 * c_2 * X_2 * d_2 * Y_2 + k_3 * c_2 * X_2 * d_3 * Y_1 * Y_2) + \\ &\quad (k_3 * c_3 * X_1 * X_2 * d_0 + k_3 * c_3 * X_1 * X_2 * d_1 * Y_1 + k_3 * c_3 * X_1 * X_2 * d_2 * Y_2 + \\ &\quad k_3 * c_3 * X_1 * X_2 * d_3 * Y_1 * Y_2) \end{aligned}$$

Reagrupando los coeficientes:

$$\begin{aligned} R &= k_0 + k_1 * c_0 + k_2 * d_0 + k_3 * c_0 * d_0 + \\ &\quad k_1 * c_1 * X_1 + k_1 * c_2 * X_2 + k_1 * c_3 * X_1 * X_2 + \\ &\quad k_2 * d_1 * Y_1 + k_2 * d_2 * Y_2 + k_2 * d_3 * Y_1 * Y_2 + \\ &\quad k_3 * c_0 * d_1 * Y_1 + k_3 * c_0 * d_2 * Y_2 + k_3 * c_0 * d_3 * Y_1 * Y_2 + \\ &\quad k_3 * c_1 * X_1 * d_0 + k_3 * c_1 * X_1 * d_1 * Y_1 + k_3 * c_1 * X_1 * d_2 * Y_2 + k_3 * c_1 * X_1 * d_3 * Y_1 * Y_2 + \\ &\quad k_3 * c_2 * X_2 * d_0 + k_3 * c_2 * X_2 * d_1 * Y_1 + k_3 * c_2 * X_2 * d_2 * Y_2 + k_3 * c_2 * X_2 * d_3 * Y_1 * Y_2 + \\ &\quad k_3 * c_3 * X_1 * X_2 * d_0 + k_3 * c_3 * X_1 * X_2 * d_1 * Y_1 + k_3 * c_3 * X_1 * X_2 * d_2 * Y_2 + \\ &\quad k_3 * c_3 * X_1 * X_2 * d_3 * Y_1 * Y_2 \end{aligned}$$

sumando los coeficientes en cada termino, renombrando los coeficientes por a_i y reagrupando tenemos:

$$\begin{aligned} R &= a_0 + a_1 * X_1 + a_{10} * X_1 + a_2 * X_2 + a_{14} * X_2 + a_3 * X_1 * X_2 + a_{18} * X_1 * X_2 + \\ &\quad a_4 * Y_1 + a_7 * Y_1 + a_5 * Y_2 + a_8 * Y_2 + a_6 * Y_1 * Y_2 + a_9 * Y_1 * Y_2 + \\ &\quad a_{11} * X_1 * Y_1 + a_{12} * X_1 * Y_2 + a_{13} * X_1 * Y_1 * Y_2 + \\ &\quad a_{15} * X_2 * Y_1 + a_{16} * X_2 * Y_2 + a_{17} * X_2 * Y_1 * Y_2 + \\ &\quad a_{19} * X_1 * X_2 * Y_1 + a_{20} * X_1 * X_2 * Y_2 + a_{21} * X_1 * X_2 * Y_1 * Y_2 \end{aligned}$$

analizando los factores comunes para cada variable y renombrando los coeficientes por b_i tenemos:

$$\begin{aligned} R &= b_0 + b_1 * X_1 + b_2 * X_2 + b_3 * X_1 * X_2 + \\ & b_4 * Y_1 + b_5 * Y_2 + b_6 * Y_1 * Y_2 + \\ & b_7 * X_1 * Y_1 + b_8 * X_1 * Y_2 + b_9 * X_1 * Y_1 * Y_2 + \\ & b_{10} * X_2 * Y_1 + b_{11} * X_2 * Y_2 + b_{12} * X_2 * Y_1 * Y_2 + \\ & b_{13} * X_1 * X_2 * Y_1 + b_{14} * X_1 * X_2 * Y_2 + b_{15} * X_1 * X_2 * Y_1 * Y_2 \end{aligned}$$

el modelo que se obtiene es equivalente a un P1P construido con 4 Variables X_1,X_2,Y_1,Y_2

9.2. Metamodelo y Metodología de construcción

Una vez obtenidos los n modelos componentes con salidas Z_i, $i = 1, 2...n$, podemos construir el metamodelo M compuesto por las salidas Z_i de los sub-sistemas (modelos P1P) tomadas como variables de entrada, obtendremos: $M(Z_1, Z_2, ...Z_n, R)$, la construcción del metamodelo involucra una muestra de las variables de entrada para realizar la regresión y calcular los coeficientes. En la construcción de los modelos anteriores tomamos valores reales para las variables de entrada y con la muestra calculamos la regresión. Para el metamodelo proponemos obtener la muestra a partir de valores estimados de los sub-modelos, un método para la construcción del metamodelo es:

- Construir los sub-modelos:
  - Determinación de las variables de entrada y salidas del sistema.
  - Construcción del P1P con las variables de entrada y salida, como resultado se obtiene el P1P que relaciona coeficientes (aun sin valor), las variables de entrada, la variable de salida.
  - Obtención de muestras de valores para variables de entrada y su resultado asociado.
  - Confección de la matriz para el cálculo de los coeficientes y ajuste de los coeficientes del P1P con una regresión.

Como resultado del punto anterior obtendremos n modelos

$$P1P_i(X_{i_1}, X_{i_2}, ...X_{im}, Z_i), i = 1, 2..n$$

y cada modelo con m_j variables, cada modelo tiene igual o diferente cantidad de entradas. También tendremos las matrices V_i formadas con valores de muestra x_{ij} con las que se calcularon los coeficientes de cada P1P

$$V_i(x_{i1,}x_{i2}, ...x_{i_m}, z_i)$$

- Considerando las n salidas de los sub-modelos como entradas, construimos el metamodelo con los coeficientes aún por calcular. $M(Z_1, Z_2, ...Z_n, R)$.
- Construcción de la Matriz de entrenamiento.
- Calculamos los coeficientes de M utilizando la matriz de valores estimados VE

9.2.1. Calculo de la matriz de entrenamiento para el metamodelo

Cada Matriz $V_i(x_{i1},x_{i2},...x_{i_m},z_i)$ se puede separar en dos matrices:

$$V_i\begin{Bmatrix} V1_i(x_{i1},x_{i2},...x_{i_m}) \\ V2_i(z_i) \end{Bmatrix}$$

$V1$ es la matriz con la muestra de los valores de entrada y $V2$ es la matriz columna con la muestra de la salida, construimos la matriz de entrenamiento del metamodelo con los valores generados por cada P1P al propagar la matriz de muestras $V1_i$ por el P1P $P1P_i(X_{i_1},X_{i_2},...X_{im},Z_i)$, como resultado generamos una matriz

$$Ve(ze_1,ze_2,...,ze_n)$$

Los valores de salida del metamodelo necesarios para entrenar el P1P se pueden obtener de forma directa al tomar valores de muestra de las variables de entrada para los modelos componentes, si ponemos en una matriz columna estos valores tenemos $R(r_i)$. Finalmente podemos entrenar el metamodelo con una regresión y la matriz VE formada por las partes Ve y R :

$$\begin{Bmatrix} Ve(ze_1,ze_2,...,ze_n) \\ R(r_i) \end{Bmatrix} VE(ze_1,ze_2,...,ze_n,r_i)$$

Parte IV

Estudio de los modelos P1P, comparaciones

Análisis con la Teoría de la información

10.1. Capacidades de los P1P

Durante el proceso de aprendizaje los P1P ajustan los valores de sus coeficientes para posteriormente utilizarlos en el procesamiento de información y obtener un resultado. Podemos decir que el uso de los P1P tiene dos etapas, una cuando se entrena el P1P, y dos cuando se utiliza para estimar valores.

El proceso de entrenamiento ocurre cuando se calculan los coeficientes con una regresión, como resultado obtenemos el modelo completo con todos los coeficientes.

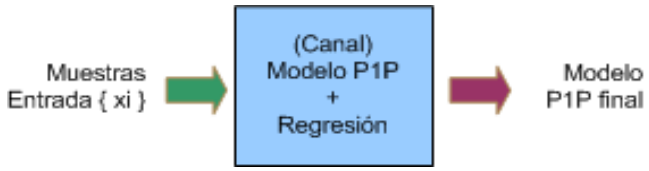

Figura 10.1: Etapa de entrenamiento de los P1P

El proceso de estimación o reconocimiento se puede ver como un proceso de comunicación donde: una entidad envía los datos hacia el canal (el P1P), el P1P genera un mensaje procesando los datos, el mensaje de salida es reconstruido por el P1P y llega al receptor. El P1P estima los valores de salida utilizando los coeficientes que calculó en base a los datos reales.

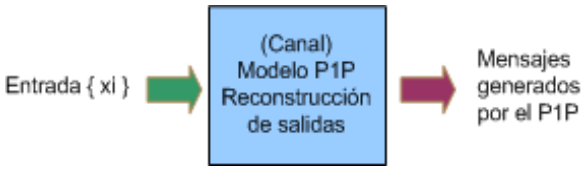

Figura 10.2: Etapa de estimación o funcionamiento de los P1P.

La pregunta que surge de esta visión es: ¿que capacidad tiene este canal?, ¿que capacidad tienen los P1P de almacenamiento de información?, con las respuestas a estas preguntas podríamos calcular la cantidad de información de las muestras y con esa cantidad determinar el modelo P1P adecuado que la soporte.

Cuando se realiza análisis de tráfico podemos calcular las frecuencias de los tamaños de los paquetes, calcular las probabilidades de ocurrencia, posteriormente podemos calcular la cantidad de información que contiene la muestra, y tomar este dato como parámetro para determinar si el P1P es un buen aproximador.

El P1P funciona como fuente de información, recibe los valores de entrada y genera o estima el valor que recibe el receptor.

Analizaremos la posibilidad que tiene un P1P de poder modelar un sistema soportando la cantidad de información proporcionada por la muestra de datos. Este análisis resulto ser más complejo de lo esperado y requiere de un estudio profundo y probado matemáticamente, lo presentamos como un disparador para próximos trabajos de investigación sobre los P1P.

De forma genérica podemos decir que un sistema asocia valores de entrada con valores de salida, o desde el punto de vista de las comunicaciones, la información se envía a un canal y se obtiene una salida, el caso más general de sistema s asociador es

$$s : R^n \rightarrow R^m$$

vectores de entrada en R^n se asocian a vectores de salida en R^m, n y m pueden o no ser iguales.

El problema fundamental consiste en intentar construir la función que asocia las entradas con las salidas de manera que se aproxime lo mejor posible a la función real. El objetivo es estudiar el comportamiento del sistema y evaluar las salidas para casos críticos.

En general los problemas de análisis y estimación relacionados con el de tráfico en redes, consisten en estimar un valor dependiente de varios factores, se construye un modelo con n variables de entrada, $X_1, X_2, X_3, ..., X_n$ y un de resultado Y, es decir un sistema $s : R^n \rightarrow R^1$ que asocia entradas y salidas: $(X_{1i}, X_{2i}, X_{3i}, ..., X_{ni}) \rightarrow Y_i$

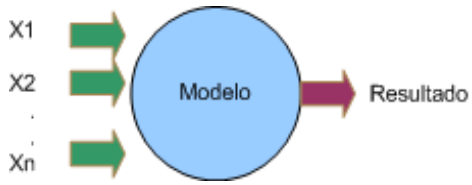

Figura 10.3: Sistema asociador $s : R^n \rightarrow R^1$, asocia $(X_{1i}, X_{2i}, X_{3i}, ..., X_{ni}) \rightarrow Y_i$, $Y_i = resultado$.

10.2. Resultados estimados y resultados reales

Supongamos que luego del proceso de aproximación obtenemos una función $Y = f\ (X_1, X_2, X_3, ..., X_n)$, podemos estimar valores de salida y_e aplicando las entradas X al modelo, los valores de salida obtenidos son aproximaciones del valor real, el valor de salida y_e tiene una probabilidad de ser real.

La información que proporciona el valor de salida y_e depende de la *cercanía* al valor real, la probabilidad $p(y_e)$ de que el valor estimado y_e sea el valor real tienen relación directa con el modelo construido, cuanto mejor sea la aproximación de la función f , la probabilidad $p(y_e)$ se incrementa y la incertidumbre disminuye.

10.2.1. Resultados estimados

La información que nos da un valor estimado y_e es:

$$I = \log_2(\frac{1}{p(y_e)}) = i_{y_e}$$

cuanto mayor es la probabilidad de que y_e sea el valor real, menor es la información que obtenemos (ver figura 10.4).

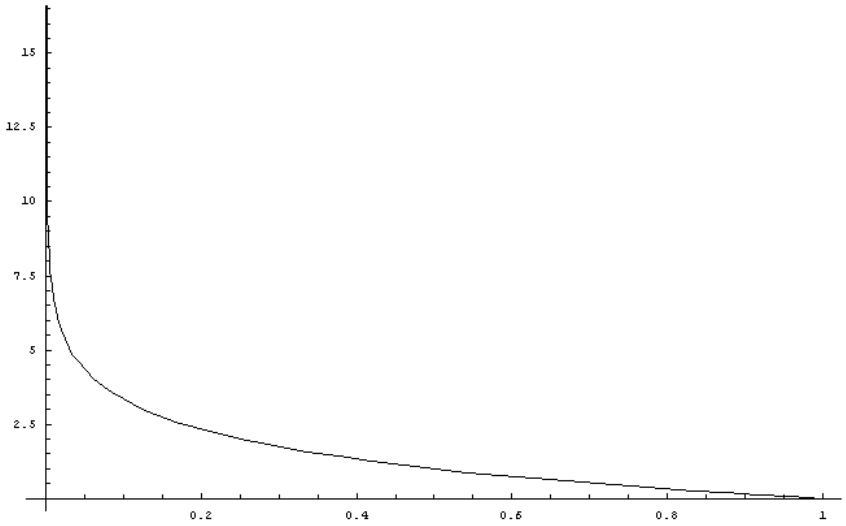

Figura 10.4: Cantidad de Información de y_e según su probabilidad $P(y_e)$

La incertidumbre (o entropía) que proporciona un valor estimado y_e se calcula como:

$$\begin{aligned} H(X) &= \sum_{k=1}^{n} p(x_k).\log_2\left(\frac{1}{p(x_k)}\right) \\ H(X) &= -p(y_e).\log_2(p(y_e)) = h_{y_e} \end{aligned}$$

10.2.2. Resultados reales

La información que tienen un valor real y_r de salida, es 0, su probabilidad es conocida $p(y_r) = 1$

$$\begin{aligned} I &= \log_2(\frac{1}{p(y_r)}) \\ I &= \log_2(\frac{1}{1}) = 0 \end{aligned}$$

la incertidumbre es cero, su entropía es

$$H = -p(y_r).\log_2(p(y_r)) = 0$$

Si encontramos la función exacta del sistema, entonces el sistema proporciona una entropía de cero porque la salida es el valor exacto.

Podemos tomar como parámetro de cercanía entre el valor estimado y el valor real, la entropía que proporcione la función aproximada, *el criterio de minimizar la entropía* durante la construcción de la función de aproximación es crítico para mejorar la estimación.

10.3. Aproximación de funciones minimizando la entropía

A continuación analizaremos como construir una función basados en los datos de entrada, salida, y minimizando la entropía.

10.3.1. Caso 1

Comenzaremos analizando un sistema simple. Supongamos que un sistema recibe como entrada un valor x_i , y se obtiene a la salida un valor constante c para cualquier x_i, el sistema asocia valores x_i con un valor constante c.

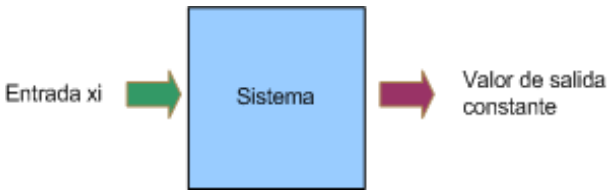

Figura 10.5: Sistema con una variable de entrada y salida constante.

Se puede modelar este sistema como

$$s : x_i \rightarrow cte$$

$$s(X) = c = cte.$$

si tenemos varias dimensiones de entrada tenemos

$$s : x_1, x_2, ..., x_n \rightarrow cte$$

$$s(X_1, X_2, ...X_n) = cte.$$

modelando con un P1P tenemos

$$p1p(X_1, X_2, ...X_n) = c0 + c1 * X_1 + c2 * X_2 + ... + c_n * X_1 * ... * X_n$$

con todos los coeficientes cero, excepto el $c0 = c$

$$p1p(X_1, X_2, ...X_n) = c0$$

Un sistema de este tipo recibe como entrada cualquier valor x_i en R^1 o R^n, y la entropía es cero porque como resultado siempre se obtienen un valor conocido $c0$.

Suponiendo que los valores x_i de entrada son independientes, tenemos que la cantidad de información de cualquier valor x_i en el sistema da como resultado

$$I = \log_2 \left(\frac{1}{p(x_i)} \right) = -\log_2(p(x_i)) = 0$$

porque la probabilidad de que para cualquier valor x_i el resultado sea $c0$ es $p(x_i) = 1$, la entropía del sistema es cero dado que no hay incertidumbre respecto de la salida.

$$\begin{aligned} H(X) &= \sum_{k=1}^{n} p(x_k).\log_2 \left(\frac{1}{p(x_k)} \right) \\ &= -\sum_{k=1}^{n} p(x_k).\log_2(p(x_k)) = 0 \end{aligned}$$

10.3.2. Caso 2

Supongamos tener un sistema que recibe un valor de entrada x_i y como salida un valor y_i, este sistema asocia $s : x_i \rightarrow y_i$. Existen infinitas funciones que asocien dos valores, el problema consiste en encontrar una función que asocie dos valores y que tenga la menor entropía posible, es decir que nos dé la mayor certidumbre posible sobre el valor de salida.

Analicemos la formula de la entropía para buscar un criterio de construcción de función con mínima entropía. La fórmula para calcular la entropía es:

$$H(X) = \sum_{k=1}^{n} p(x_k) * \log_2\left(\frac{1}{p(x_k)}\right)$$

la primer derivada es:

$$\begin{aligned}\frac{\partial H(X)}{\partial x_i} &= \frac{\partial(-\sum_{k=1}^{n} p(x_k) * \log_2(p(x_k)))}{\partial x_i} \\ \frac{\partial H(X)}{\partial x_i} &= -\frac{1}{\ln(2)} - \frac{1}{\ln(2)} * \ln(p(x_i))\end{aligned}$$

si calculamos el punto critico tenemos

$$\begin{aligned}\frac{\partial H(X)}{\partial x_i} &= -\frac{1}{\ln(2)} - \frac{1}{\ln(2)} * \ln(p(x_i)) = 0 \\ \ln(p(x_i)) &= -\frac{\frac{1}{\ln(2)}}{\frac{1}{\ln(2)}} = -1 \\ \ln(p(x_i)) &= \frac{\log_2(p(x_i))}{\log_2(e)} = -1 \\ \log_2(p(x_i)) &= -\log_2(e) \\ p(x_i) &= 2^{-\log_2(e)} = 0{,}367\,88\end{aligned}$$

este valor es un máximo ya que la derivada segunda es negativa

$$\begin{aligned}\frac{\partial^2 H(X)}{\partial^2 x_i} &= \frac{\partial(-(\frac{1}{\ln(2)} + \frac{1}{\ln(2)} * \ln(p(x_i))))}{\partial x_i} \\ \frac{\partial^2 H(X)}{\partial^2 x_i} &= -\frac{\frac{1}{\ln(2)}}{p(x_i)} \\ &= -\frac{1.\,442\,7}{0{,}367\,88} \\ &= -3.\,921\,7 < 0\end{aligned}$$

llamemos a $p(x_i) = 2^{-\log_2(e)} = 0{,}367\,88 = p_{\text{máx}}(x_i)$.

La Figura 10.6 grafica la función de entropía

La entropía aumenta hasta su valor máximo entre $p(x_i) = 0$ y llega a un punto máximo en $p_{\text{máx}}(x_i)$, luego disminuye hacia cero cuando la probabilidad esta entre $p_{\text{máx}}(x_i)$ y $p(x_i) = 1$. Como criterio general podemos observar que cuando las probabilidades de los valores se mantengan superiores a $p_{\text{máx}}(x_i)$ la entropía tiende a minimizarse.

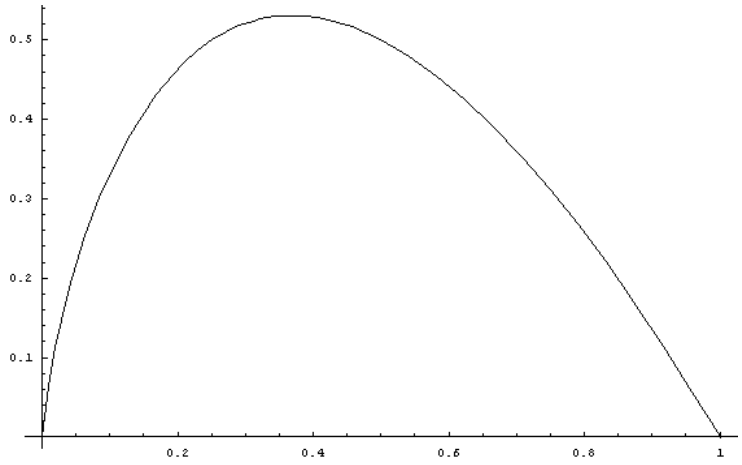

Figura 10.6: Función de Entropía

10.3.3. Construcción del modelo

Sabemos que la función $f(x)$ al menos tiene una variable donde reemplazar la x_i, y calcular la y_i.Supongamos que la función de aproximación f contiene una variable x y que al aplicar la entrada x_i se obtiene $y_i = f(x)$,el resultado y_i tiene una probabilidad $p(x)$ se ser el valor real, la entropía para esa función es entonces $H(x) = p(x) * \log_2\left(\frac{1}{p(x)}\right) = h$, propongamos algunas funciones combinando las operaciones básicas de suma, producto y producto por un escalar:

- a) $y_i = f(x) = x$
- b) $y_i = f(x) = c + x$
- c) $y_i = f(x) = x + x$
- d) $y_i = f(x) = x * x$
- e) $y_i = f(x) = c * x$

La función a) queda descartada porque si fuera la verdadera tendríamos conocimiento total del sistema.
La función c) $x + x = 2 * x$ es similar a la e) $c * x$ con $c = 2$
Las entropías para las demás funciones propuestas son:

- b) $H(x) = p(x) * \log_2\left(\frac{1}{p(x)}\right) = h$
- d) $H(x) = p(x) * \log_2\left(\frac{1}{p(x)}\right) + p(x) * \log_2\left(\frac{1}{p(x)}\right) = 2h$
- e) $H(x) = p(x) * \log_2\left(\frac{1}{p(x)}\right) = h$

La función más simple que asocia dos valores x_k e y_k considerando la menor entropía es

$$s : f(X) = y_i = ci * x_i$$

en una dimensión tenemos asociados dos valores a través de un escalar conocido, suponiendo que los valores x_i son independientes y con probabilidad de aparición $p(x_i)$, tenemos que la entropía del sistema es

$$H(X) = \sum_{k=1}^{1} p(x_k).\log_2\left(\frac{1}{p(x_k)}\right) = h$$

cualquier función que involucre más de una vez al valor de x_i genera una entropía mayor, porque agrega a la sumatoria un término. De forma genérica podemos decir que para asociar dos valores en R^1, se puede construir una función

$$y_i = x_i * ci$$

que contiene a los dos valores x_i e y_i y cuya entropía es la mínima.

Un caso particular es cuando $x_i = 0$, se obtiene como salida $y_i = 0$, para todos los casos de ci es igual, se puede corregir sumando una constante $c0$, y el modelo final es entonces:

$$s : y_i = c0 + ci * x_i$$

la entropía se mantiene constante ya que no se agregan símbolos al modelo.

La cantidad de información que genera este sistema es

$$I = \log_2\left(\frac{1}{p(x_i)}\right) = I_i$$

no necesariamente la probabilidad de los símbolos o valores es equivalente.

el modelo construido es un P1P con una sola variable X_1:

$$p1p(X_1) = c0 + c1 * X_1$$

Interpretación geométrica

Otra interpretación puede ser: supongamos tener dos valores x_i y y_i pertenecientes a un sistema R^1, se pide unirlos de manera que el recorrido entre los dos valores sea mínimo, intuitivamente lo que haríamos es una línea recta que junte los dos valores, geométricamente lo que hacemos es trasladar el primer valor al segundo por el camino más corto.

Si armamos un programa en *Mathematica* para definir un P1P de una variable y graficamos, tenemos:

```
c0 = 10;
c1 = 2;
pip1[x1_] = c0 + c1 * x1;
(* GRAFICO EL P1P *)
g = Plot[pip1[x1], {x1, -100, 100}];
```

Obtenemos:

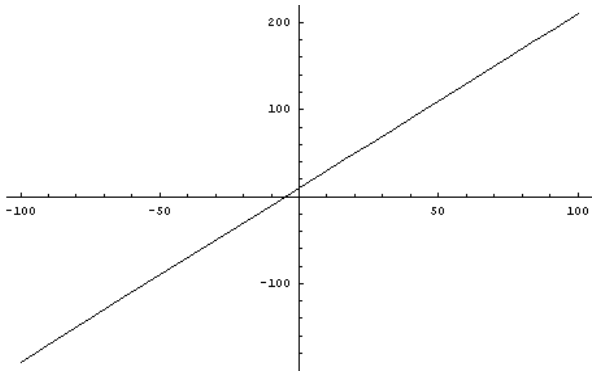

Figura 10.7: Gráfica de un P1P con una sola variable.

10.3.4. Caso 3

Supongamos tener un sistema que recibe dos valores (un punto) como entradas, como resultado se obtiene un valor, una función

$$s : R^2 \rightarrow R^1$$

Tenemos dos variables X_1, X_2 asociadas con una variable de salida Y, podemos pensar que cada variable se asocia con la salida de forma independiente y de forma conjunta. Intentando minimizar la entropía y basado en la función obtenida en caso anterior podemos construir una función que relacione cada variable X con el resultado Y por separado, luego de forma conjunta de manera que la entropía sea la menor posible, tenemos entonces:

- para la variable $X_1 \rightarrow Y$,

$$f1(X_1) = Y = c0_1 + c1 * X_1$$

- para la segunda variable $X_2 \rightarrow Y$,

$$f2(X_2) = Y = c0_2 + c2 * X_2$$

- luego de forma conjunta $(X_1, X_2) \rightarrow Y$, la entropía debe ser la menor posible, las variables aparecen solo 1 vez en la función,

$$f3(X_1, X_2) = Y = c0_3 + c3 * X_1 * X_2$$

Construcción del modelo final

Para relacionar las tres funciones f_1 , f_2 , f_3 las opciones posibles pueden ser multiplicar las funciones

$$f(X_1, X_2, X_3) = f1(X_1) * f2(X_2) * f3(X_3)$$

o sumar las funciones

$$f(X_1, X_2, X_3) = f1(X_1) + f2(X_2) + f3(X_3)$$

si multiplicamos las funciones, las variables aumentan su grado y la entropía aumenta en un valor g_1, si las sumamos se mantiene el grado de las variables en uno y la entropía aumenta en un valor g_2 que es menor que g_1, sumando las tres funciones tenemos:

$$\begin{aligned} f(X_1, X_2, X_3) &= c0 + c1 * X_1 + \\ & \quad c2 * X_2 + \\ & \quad c3 * X_1 * X_2 \end{aligned}$$

donde $c0 = c0_1 + c0_2 + c0_3$ esta función asocia los valores de X_1, X_2 con el resultado y_i de forma independiente para cada variable, y de forma conjunta para las variables con la menor entropía posible.

Esta función es un P1P con tres variables:

$$p1p(X_1, X_2, X_3) = c0 + c1 * X_1 + c2 * X_2 + c3 * X_1 * X_2$$

10.4. Generalización

Podemos generalizar el criterio para construir funciones que asocien n variables X con una de salida Y , de la siguiente forma:

- por cada variable armar la función de menor entropía $Y = c0_i + ci * X_i$
- Tomar de a dos variables sin repetición y armar las funciones de menor entropía $Y = c0_{ij} + cij * X_i X_j$
- Realizar la misma operación incrementando en uno las variables hasta llegar a las n variables:
  - Ej. para 3 variables: $Y = c0_{ijk} + c_{ijk} * X_i X_j X_k$
  - para 4 variable: $Y = c0_{ijkw} + c_{ijkw} * X_i X_j X_k X_w$
- la ultima función tendrá la forma: $Y = c0_{ijk.....n} + c_{ijk.....n} * X_i X_j X_k ....X_n$
- sumar todas las funciones anteriores y simplificar todas las constantes en una única constante $c0$

La función resultante final es un P1P.

Modelo para un Servidor SMTP

11.1. Desripción del problema

Analizamos el tráfico SMTP para un servidor de correo conectado a Internet, la problemática consiste en analizar la cantidad de bytes que procesa un servidor SMTP con un sistema de colas y con un P1P para luego hacer comparaciones entre las dos metodologías. Los aspectos consideramos para el estudio son:

- la velocidad de atención de los paquetes al puerto 25 (SMTP),
- tiempo de llegada de los paquetes al servidor,
- el tamaño de los paquetes al servidor SMTP,

El Servidor SMTP está ubicado en una zona de servidores, detrás de un firewall, desde la Internet el servidor SMTP de ve conectado directamente en una dirección IP pública, la siguiente figura (figura 11.1) muestra la ubicación y la topología de la red.

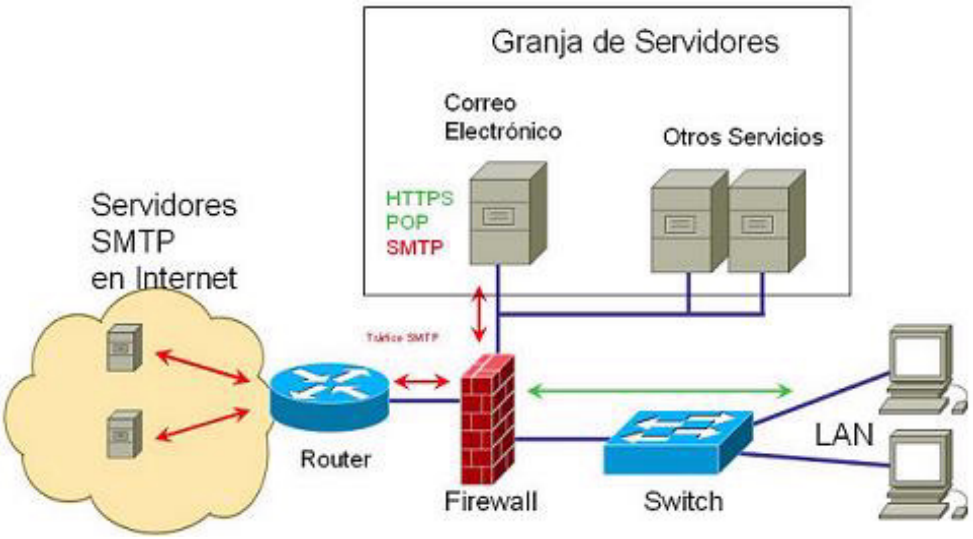

Figura 11.1: Ubicación del servidor SMTP en la granja de servidores, topología de la red.

11.2. Modelo de Colas

Para hacer un modelo de colas necesitamos determinar:

- la función de distribución de llegadas de los paquetes,
- la tasa de llegada de los pdu SMTP
- la tasa de atención de los paquetes
- El tamaño de los paquetes tratados

11.2.1. Consideraciones para las mediciones de tiempos

Experimentalmente resulta complejo medir a nivel de aplicación el tiempo de llegada de cada PDU SMTP y el tiempo de atención de cada pdu. Más específicamente, para medir el tiempo exacto de llegada a la aplicación SMTP deberíamos activar un timmer al llegar el mensaje SMTP, y para calcular el tiempo de atención medir la diferencia entre el tiempo de llagada a la aplicación y el tiempo de salida de la aplicación. La siguiente gráfica describe el proceso de desencapsulación de los protocolos hasta que el mensaje SMTP llega a la aplicación SMTP:

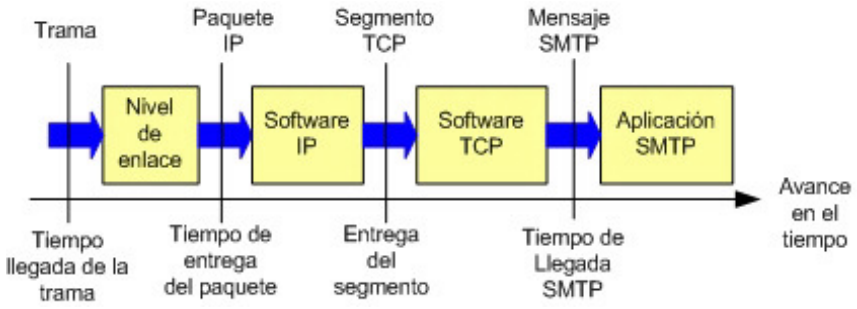

Figura 11.2: Proceso de desencapsulación de los protocolos hasta SMTP.

hay un tiempo de procesamiento en cada nivel de encapsulación, este tiempo es mayor en los niveles donde hay multiplexación como el nivel IP y el nivel TCP. Para obtener el tiempo de llegada mejor aproximado necesitamos abrir el software TCP e incorporar los cambios a tal fin, como generalmente no contamos con el código fuente del software SMTP resulta dificultoso incorporar los cambios. Una opción más práctica es realizar mediciones en la conexión física al servidor y analizar los paquetes con la dirección del servidor SMTP con segmentos TCP hacia y desde el puerto 25. Paquetes con IP destino del servidor y al puerto 25, y paquetes con la IP origen y puerto origen 25 (SMTP).

11.2.2. Tiempo de arribo y tiempo de servicio para el modelo de colas

La siguiente figura muestra la topología física de la granja de servidores. Se utilizan VLANS para las redes de los servidores, una VLAN por Servidor, el firewall es un sistema Linux con software *IPtables* y *Vconfig* para soportar en protocolo VTP de las VLANS, la interfaz lógica en el firewall esta conectada al servidor de correo que tiene la IP: 10.10.10.34/28.

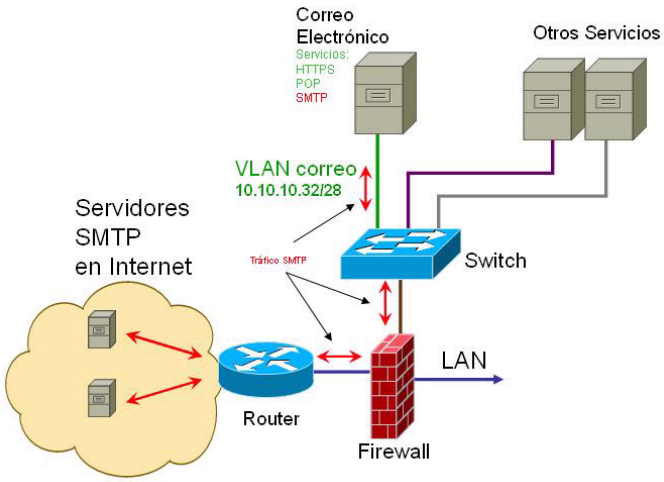

Figura 11.3: Topología fisica de la granja de servidores, VLANS en la DMZ.

Realizamos un volcado TCP (con Tcpdump) sobre la interfaz conectada al servidor SMTP, como el servidor esta aislado en una VLAN todo el tráfico que pasa por esa interfaz es hacia o desde el servidor de correo, los protocolos que aparecen en el volcado son los usados en un servidor de correo, SMTP, DNS, POP, y otros paquetes como ARP, e ICMP.

El proceso para obtener los datos de tiempo entre llegada y tiempo de atención es el siguiente:

- Captura de datos en la interfaz y generación de un archivo
- Procesamos los datos del volcado por línea y extraemos: el tiempo entre paquetes entrantes y el tamaño, el tiempo de los paquetes salientes y el tamaño.
- Para el tráfico entrante y saliente calculamos la cantidad de paquetes, el promedio de los tiempos inter-paquete, y el promedio del tamaño de los paquetes.

El volcado TCP generó un archivo de 91 Mbytes y 463017 líneas de volcado, la captura se inicio a las *15:54:31 hs* y se detuvo a las *17:49:14 hs*, aquí se muestran cuatro líneas del archivo:

1. 15:54:31.598463 IP (tos 0x0, ttl 64, id 4822, offset 0, flags [DF], proto: TCP (6), length: 40) 10.10.10.34.25 > 65.55.90.17.46701: ., cksum 0xc0be (correct), 1:1(0) ack 14560 win 65535
2. 15:54:31.645372 IP (tos 0x0, ttl 43, id 10800, offset 0, flags [DF], proto: TCP (6), length: 1496) 209.85.222.138.64109 > 10.10.10.34.25: . 3505747255:3505748699(1444) ack 3901865552 win 108 <nop,nop,timestamp 1614374141 729964>
3. 15:54:31.645456 IP (tos 0x0, ttl 64, id 42881, offset 0, flags [DF], proto: TCP (6), length: 64) 10.10.10.34.25 > 209.85.222.138.64109: ., cksum 0x2a3b (correct), 1:1(0) ack 4294962964 win 501 <nop,nop,timestamp 730048 1614373820,nop,nop,sack 1 {0:1444}>
4. 15:54:31.646785 IP (tos 0x o0, ttl 109, id 29238, offset 0, flags [DF], proto: TCP (6), length: 46) 83.4.20.214.42922 > 10.10.10.34.25: P, cksum 0xf307 (correct), 1948387900:1948387906(6) ack 3897723760 win 65311

Para el algoritmo de procesamiento del archivo se tomo en cuenta

- Se procesan los datos línea a línea.

- Se procesan solo los paquetes que encapsulan TCP,
- El tiempo interpaquete se calcula restando la marca de tiempo del paquete actual con la marca de tiempo del paquete anterior, se consideran marcas de tiempo diferentes para los paquetes entrantes y salientes.
- Se considera que un paquete es entrante al servidor SMTP si la IP Destino es la 10.10.10.34 y el puerto Destino es el puerto 25
- Se considera que un paquete es saliente del servidor SMTP si la IP Origen es la 10.10.10.34 y el puerto Origen es el puerto 25
- Un paquete que no es ni entrante ni saliente del servidor SMTP, no se procesa.

Aparecerán paquetes con la IP del servidor con otros servicios que utiliza el Correo, por ejemplo los paquetes con IP Origen 10.10.10.34 y puerto Origen distinto al 25 pueden ser paquetes de consulta al DNS (con puerto destino 53) o paquetes de envío de correo a servidores externos con puerto destino 25 y IP publica, el paquete al puerto 53 no se procesa.

El algoritmo esta en el apéndice con el nombre de **Procesamiento de datos para SMTP-Colas**
Por ejemplo,

- Para la línea **1)** se trata de un paquete que encapsula TCP saliente del servidor SMTP, posiblemente se trata de un ACK que forma parte de un establecimiento de conexión.
- La línea **2)** es un paquete entrante al servidor SMTP ya que la IP Origen es pública y la IP Destino es la de nuestro servidor.

si procesamos las 4 líneas obtenemos los siguientes valores

Tipo	**Tiempo entre dos paquetes (en μs)**	**Bytes procesados**
saliente	$31645456 - 31598463 = 46993$	$40 + 64 = 104$
entrante	$31646785 - 31645372 = 1413$	$1496 + 46 = 1542$

En total 4 paquetes, dos entrantes y dos salientes intercalados.

El procesamiento se realiza con un script php que analiza las líneas y calcula la información. El código del script aparece en el Apéndice de Scripts con el nombre **Procesamiento de datos para modelo de colas en un servidor SMTP**.

11.2.3. Tasa de Arribo y Servicio

Al ejecutar el script se procesan 463017 paquetes (o líneas), de los cuales 118520 son entrantes y 110949 son salientes, el resto no son paquetes SMTP, se obtienen los siguientes promedios

Tipo	Cantidad	Total tiempos	Bytes totales	Promedio de tiempo	Promedio de bytes
Entrantes	118520	64154536242 μs	86011828	$T_a = 541297{,}133328$ μs	725,715727303 *bytes*
Salientes	110949	64153951454 μs	5742479	$T_s = 578229{,}199488$ μs	51,7578256676 *bytes*

Calculamos las tasas de arribo y servicio por segundo:

$$\lambda = \frac{1x10^6[\ \mu s]}{T_a} = \frac{1000000}{541297{,}133328} = 1.8474 \text{ paquetes por segundo}$$

y la tasa de servicio:

$$\mu = \frac{1x10^6[\ \mu s]}{T_s} = \frac{1000000}{578229{,}199488} = 1.7294 \text{ paquetes por segundo}$$

tenemos entonces

$$\rho = \frac{\lambda}{\mu} = \frac{1.8474}{1.7294} = 1.068\,2$$

11.2.4. Modelo de colas M/M/1/k

Armemos un modelo de colas para analizar como podría variar el buffer en función de la velocidad de atención y llegadas al sistema. Utilizamos la siguiente notación para las fórmulas del modelo de colas:

$$\begin{aligned}
\text{Tiempo promedio de interarribo } E[A] &= T_a \\
\text{Tasa de arribo } \lambda &= \frac{1}{T_a} \\
\text{Tiempo promedio de servicio } E[B] &= T_b \\
\text{Tasa de servicio } \mu &= \frac{1}{T_b} \\
\text{Utilización } \rho &= \frac{\lambda}{\mu} \\
\text{Rendimiento del sistema de colas (Throughput) } \lambda &= \rho\mu \\
\text{Número de elementos en el sistema } K_s &= \frac{\rho}{1-\rho} \\
\text{Tiempo de respuesta del sistema } T_s &= \frac{\frac{1}{\mu}}{1-\rho} \\
\text{Tiempo de espera en el sistema } W_s &= \frac{\frac{\rho}{\mu}}{1-\rho} \\
\text{Número de elementos en la cola } q &= \frac{\rho^2}{1-\rho}
\end{aligned}$$

y para un sistema de cola finita tenemos que:

$$\text{Número de elementos en el sistema } N_s = \begin{cases} \frac{a}{1-a} - \frac{k+1}{1-a^{k+1}} a^{k+1}; a \neq 1 \\ \frac{k}{2}; a = 1 \end{cases}$$

donde $a = \frac{\lambda}{\mu}$ y $k =$ límite de la cola.
y para cualquier valor de a

$$\begin{aligned}
\text{Número de elementos en servicio es } L_{se} &= 1 - p_0 \\
\text{Probabilidad de que no existan elementos } p_0 &= \frac{1-\frac{\lambda}{\mu}}{1-(\frac{\lambda}{\mu})^{k+1}} \\
\text{Número de elementos en la cola } L_q &= N_s - L_{se}
\end{aligned}$$

11.2.5. Tamaño de los paquetes y buffer

Podemos calcular el tamaño promedio de los paquetes y luego el tamaño de un buffer adecuado para nuestro servidor SMTP. El buffer contiene paquetes a procesar o paquetes procesados listos para ser atendidos o transmitidos.

De acuerdo a los cálculos que hicimos con el *script php*, la cantidad de bytes totales atendidos por el servidor SMTP es de:

$$86011828 + 5742479 = 9.1754 \times 10^7 = 91754000 \; bytes$$

(entrantes+salientes) y la cantidad de paquetes es:

$$118520 + 110949 = 229470 \; paquetes$$

el tamaño promedio de un paquete es entonces de

$$TP_{prom} = \frac{91754000}{229470} = 399.85\ bytes$$

el número de elementos en el sistema es $L_s = \frac{a}{1-a} - \frac{k+1}{1-a^{k+1}}a^{k+1}$, con un buffer limitado a 100 paquetes ($k = 100$) y $a = \rho = 1.0682$, tenemos que:

$$L_s = \frac{1.0682}{1-1.0682} - \frac{100+1}{1-1.0682^{100+1}}1.0682^{100+1} = 85.466\ paquetes$$

el número de elementos en servicio es: $L_{se} = 1 - p_0$ y $p_0 = \frac{1-\frac{\lambda}{\mu}}{1-(\frac{\lambda}{\mu})^{k+1}}$ tenemos entonces que:

$$\begin{aligned} p_0 &= \frac{1-1.0682}{1-(1.0682)^{100+1}} = 8.7182 \times 10^{-5} \\ L_{se} &= 1 - 8.7182 \times 10^{-5} = 0{,}999\,91\ paquetes \end{aligned}$$

el número de elementos en la cola es

$$L_q = N_s - L_{se} = 85.466 - 0{,}999\,91 = 84.466\ paquetes$$

y finalmente el tamaño en bytes del buffer ocupado es

$$\begin{aligned} T_{buff} &= TP_{prom} * 84.466 \\ T_{buff} &= 339{,}85 * 84{,}466 = 28706\ bytes \end{aligned}$$

Pudimos haber supuesto que al tener un $\rho = \frac{\lambda}{\mu} = 1.0250$ que esta próximo a uno, un buffer de cualquier tamaño k tendrá $\frac{k}{2}$ paquetes, un buffer con capacidad de $k = 100$ paquetes tiene un tamaño en bytes de

$$T_{buff} = TP_{prom} * k = 399.85 * 100 = 39985\ bytes$$

aproximadamente $40kbytes$

11.3. Modelo con P1P

11.3.1. Identificación del sistema

El problema que tratamos consiste en estimar la cantidad de bytes que procesa un servidor SMTP. La cantidad de bytes que procesa un servidor SMTP depende de la cantidad de paquetes entrantes, salientes y la velocidad de trasmisión, podemos considerar nuestro servidor SMTP como una caja negra y analizar las entradas y salidas al sistema. Determinar valores de las variables involucradas es complejo ya que a nivel de red y a nivel de transporte hay multiplexación de paquetes y de segmentos, por esto la cantidad de bytes y la velocidad de trasmisión que podemos medir en un volcado a nivel de red corresponde a todos los paquetes de la capa y no solo a los recibidos o trasmitidos al servidor SMTP.(ver Figura 11.4)

Las variables de entrada al sistema son los paquetes hacia al puerto 25 y los paquetes desde el puerto 25. En un volcado TCP podemos medir la cantidad de paquetes dirigidos al puerto 25 y los paquetes salientes del puerto 25, la velocidad de trasmisión en el nivel de enlace es la velocidad de la interfaz, la velocidad de trasmisión a nivel de red depende de la totalidad de los paquetes trasmitidos y recibidos. La velocidad de trasmisión en el nivel de red se puede calcular con la cantidad de bytes en paquetes IP en un segundo.

Nuestro modelo P1P considera:

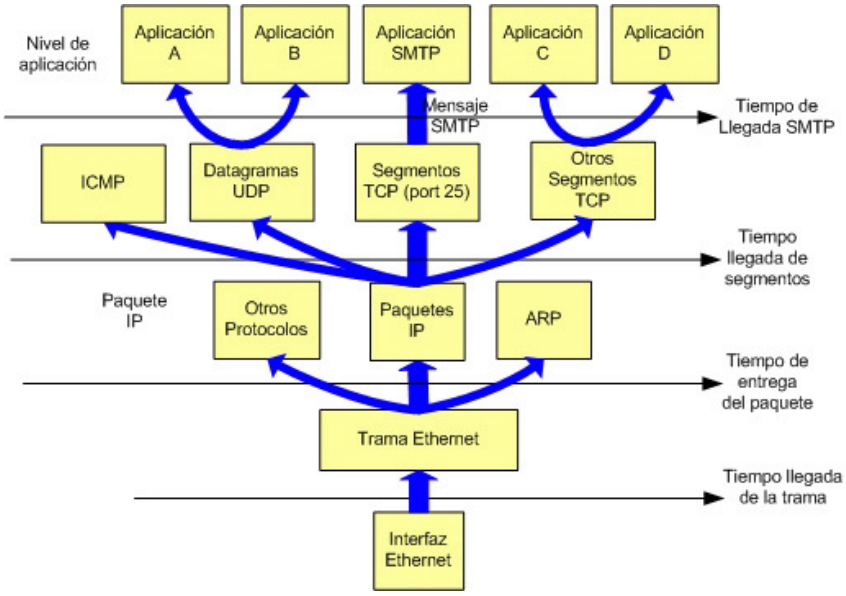

Figura 11.4: Protocolos por niveles, multiplexación y puntos de medida de tiempo.

- El sistema a tratar: estimación de la cantidad de bytes (en paquetes) a procesar en un servidor SMTP.
- Las variables de entrada son:
  - paquetes dirigidos al puerto SMTP,
  - paquetes salientes del servidor SMTP,
  - velocidad de atención de los paquetes a nivel de red y en bps.
- salida: bytes procesados o a procesar por el servidor SMTP.

La figura a continuación muestra el sistema a tratar y sus variables.

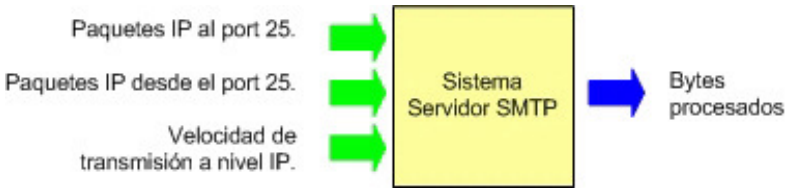

Figura 11.5: Sistema definido para en servidor SMTP.

11.3.2. P1P para el servidor SMTP

Tenemos entonces un modelo P1P de tres variables X_1, X_2, X_3 y una salida B, construimos el modelo:

$$\begin{aligned} B &= c + c_1 * X_1 + c_2 * X_2 + c_3 * X_3 + \\ & \quad c_4 * X_1 * X_2 + c_5 * X_1 * X_3 + \\ & \quad c_6 * X_2 * X_3 + c_7 * X_1 * X_2 * X_3 \end{aligned}$$

donde:
X_1: Paquetes por segundo dirigidos al puerto 25
X_2: Paquetes por segundo salientes del puerto 25
X_3: Velocidad en bps a nivel de red
B: Bytes procesados o a procesar por el servidor SMTP
$c, c_1, c_2, c_3, c_4, c_5, c_6, c_7$: son los coeficientes del P1P que calcularemos

11.3.3. Muestra de datos y entrenamiento del P1P

Utilicemos el archivo de tráfico usado en el trabajo anterior y procesemos los datos requeridos con un script php (**Programa SMTP-P1P** del apéndice de scripts), el resultado del script es un archivo con los siguientes datos:

- paquetes por segundo entrantes,
- paquetes por segundo salientes,
- velocidad de atención,
- y bytes tratados por el servidor SMTP,

nuestro archivo es una tabla organizada en líneas de texto, cada línea separa por comas los campos formando una matriz de 6542 filas y 4 columnas, las primeras filas son:

pk entrantes	**pk salientes**	**velocidad**	**total de bytes**
18	21	144.32	17896
25	28	214.44	24556
31	34	277.184	34406
29	29	222.376	27797

el script cuenta por segundo:

- La cantidad de paquetes entrantes al puerto 25,
- La cantidad de paquetes salientes del puerto 25,
- La velocidad de trasmisión calculada como $vel = \frac{\text{(total de bytes en paquetes IP) } *8}{\text{1 segundo}} = [bps]$,
- y la cantidad de bytes en los paquetes entrantes y salientes.

Una vez obtenida la muestra de datos podemos calcular el valor de los coeficientes con una regresión no lineal, usamos el programa en *Mathematica* para realizar la regresión, (el programa completo está en el apéndice) el valor de los coeficientes se obtiene con las siguientes líneas:

```
(* Carga del modulo de estadística para utilizar la función de regresióon *)
 << Statistics‘NonlinearFit‘
(* borro las variables de la memoria *)
 Clear[smtpdatos, smtpdatos2, pkllegadas, pksalidas, velocidad, bytes];
```

```
Clear[x1, x2, x3, c, c1, c2, c3, c4, c5, c6, c7];
smtpdatos = Import["D:/works/tesis/colas_p1p/smtp_datos_p1p.csv", ÇSV"];

(* Calculo los coeficientes del P1P *)
NonlinearRegress[ smtpdatos,
c + c1 * x1 + c2*x3 + c3 * x3 +
c4 * x1 * x2 + c5 * x1 * x3 +
c6 * x2 * x3 +
c7* x1*x2*x3,
{x1, x2, x3},
{c, c1, c2, c3, c4, c5, c6, c7}]
```

del resultado de la ejecución de la función obtenemos los valores de los coeficientes:

$$\begin{aligned} c &= 850{,}009, \\ c1 &= 736{,}299, \\ c2 &= -1{,}16385, \\ c3 &= -1{,}16385, \\ c4 &= -0{,}676371, \\ c5 &= 0{,}21687, \\ c6 &= -0{,}145403, \\ c7 &= -0{,}0000992072 \end{aligned}$$

y el modelo completo es entonces:

$$\begin{aligned} B = {} & 850{,}009 + 736{,}299 * X_1 + -1{,}16385 * X_2 + -1{,}16385 * X_3 + \\ & -0{,}676371 * X_1 * X_2 + 0{,}21687 * X_1 * X_3 + \\ & -0{,}145403 * X_2 * X_3 + -0{,}0000992072 * X_1 * X_2 * X_3 \end{aligned}$$

Gráficas de los datos obtenidos en la muestra

Grafiquemos los datos de tráfico obtenidos y comparémoslo con los estimados con el modelo P1P, se utilizó un programa en la herramienta Mathematica para realizar las graficas. La figura 11.6 muestra la cantidad de paquetes entrantes al puerto 25 por segundo, la figura 11.7 muestra la cantidad de paquetes salientes del puerto 25 por segundo y la figura 11.8 grafica los bytes por segundo de los paquetes con puerto 25 (entrantes y salientes).

Gráficas de los bytes reales y los bytes estimados

La figura 11.9 muestra los bytes en los paquetes IP hacia el puerto 25 más los bytes en paquetes IP desde el puerto 25, estos datos se utilizaron para entrenar el P1P.

La figura 11.10 muestra la estimación de bytes obtenidas con el modelo P1P. Observamos que el patrón de tráfico en las dos gráficas es similar con lo que podemos concluir que el modelo aprendió las características del tráfico SMTP.

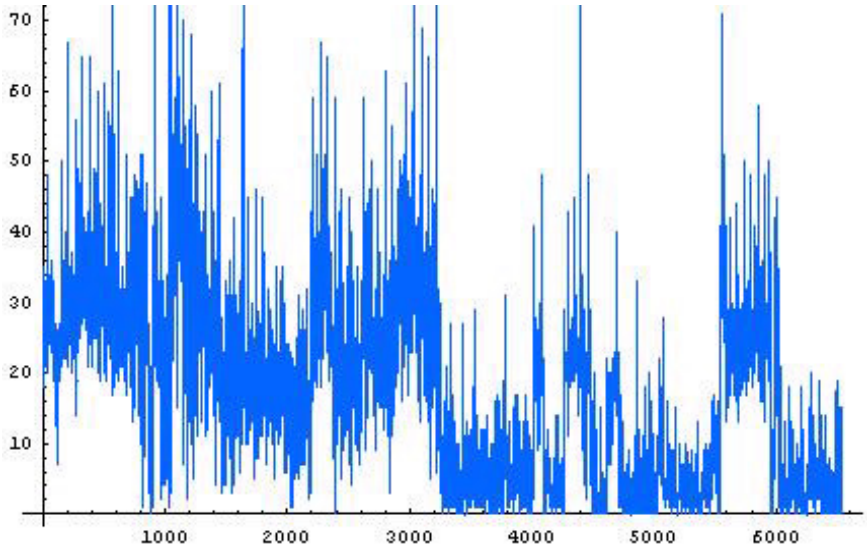

Figura 11.6: Paquetes entrantes al puerto 25

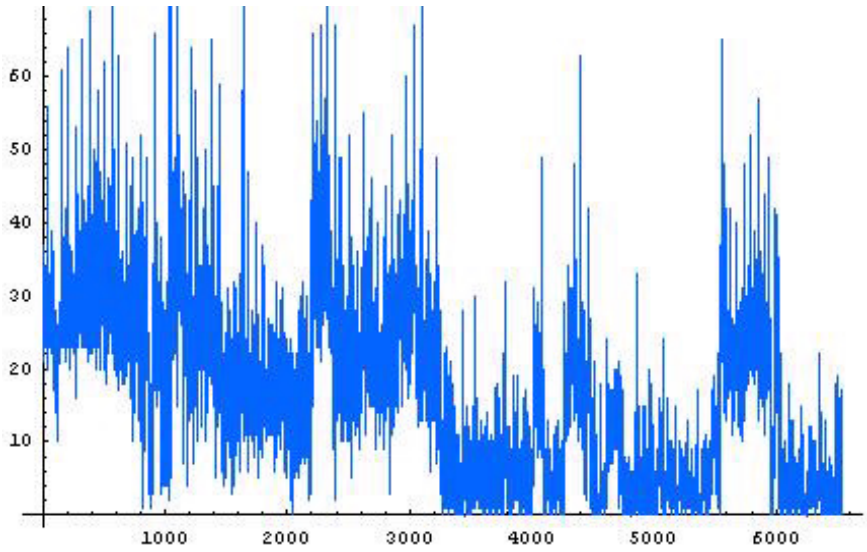

Figura 11.7: Paquetes salientes del puerto 25.

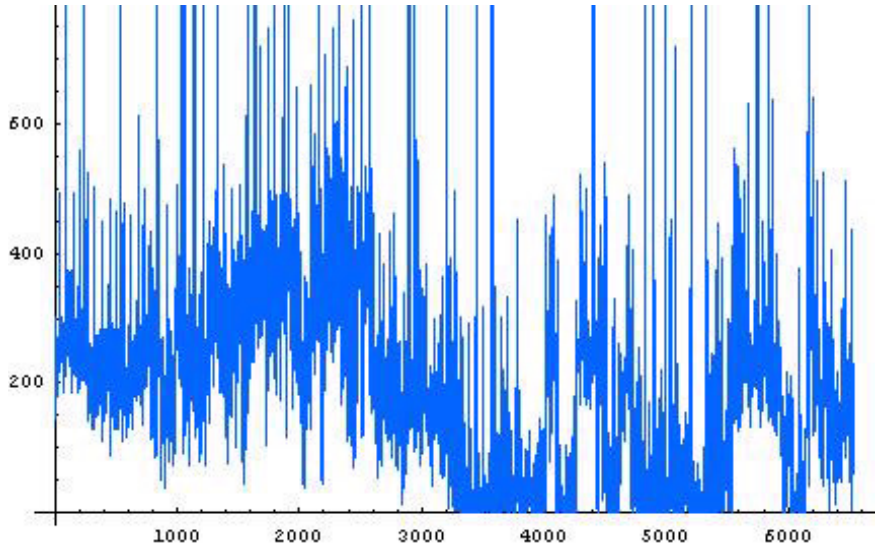

Figura 11.8: Velocidad de transmisión de paquetes a nivel de red en bps.

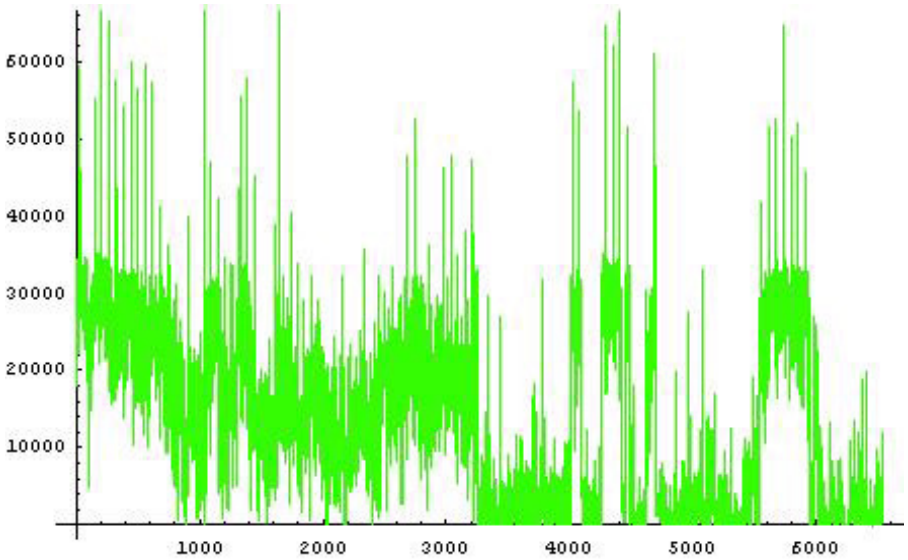

Figura 11.9: Bytes de los paquetes entrantes y salientas al puerto 25.

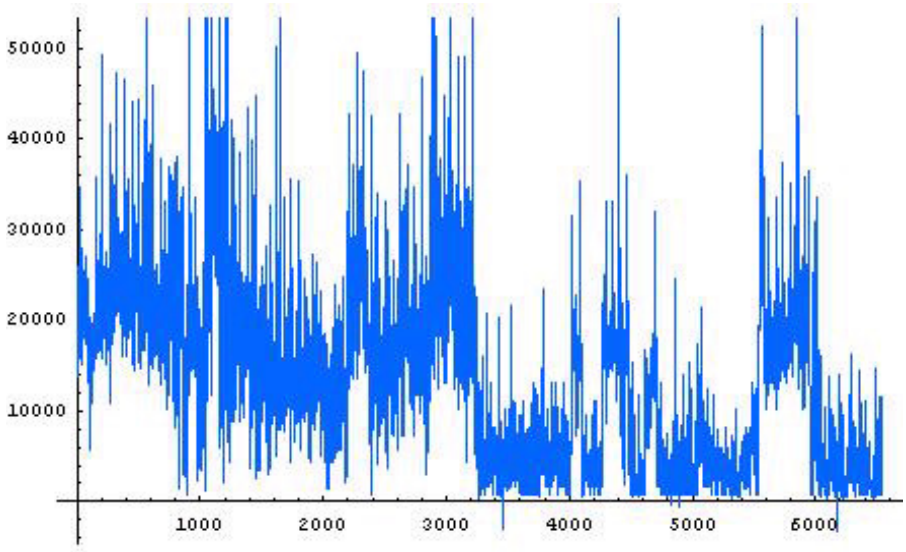

Figura 11.10: Gráfica de bytes estimados (resultado del P1P) con las entradas X_1, X_2, X_3 tomados de la muestra.

Parte V

Conclusiones

Conclusiones

12.1. Conclusiones principales

Los P1P son aplicables al estudio de tráfico estadístico en redes IP con buenos resultados, los modelos P1P se adaptaron a todos los casos tratados aprendiendo las características del tráfico muestreado, queda demostrado que son una alternativa importante para abordar problemáticas donde se requiere estimar factores.

La analogía realizada en la teoría de los P1P, el enfoque sistémico y la facilidad de uso hacen que la aplicabilidad de los P1P a redes sea amplia, esto proporciona caminos a nuevos estudios basados en la caracterización de tráfico de redes.

El estudio del modelo orientado a redes demostró que es necesario continuar avanzando en los aspectos teóricos. Se han propuesto diferentes líneas de trabajo.

Alcanzamos los objetivos propuestos afianzándolos con aportaciones académicas.

12.2. Modelo de transporte y Red

De este primer trabajo se concluye la facilidad de aplicación de los P1P y los aspectos que limitan su utilización. Un P1P tendrá como salida un solo valor, por esto son adecuados para tratar problemáticas donde se requiere estimar un factor de la red dependiente del tráfico. La determinación del sistema a tratar es un aspecto importante para obtener un modelo representativo del problema a solucionar. En este caso construimos un modelo para estimar bytes a procesar en el nivel de red. Puede utilizarse para estimar el ancho de banda requerido a ese nivel.

Una opción es estimar la velocidad de tratamiento de paquetes por segundo utilizando un tamaño de paquete promedio, esta opción puede utilizarse para determinar la capacidad requerida de conmutación en un router.

Un aspecto interesante es que con el P1P podemos estimar una serie de valores en un solo cálculo, con ello se facilita la generación de curvas estimadas para el posterior estudio del comportamiento en forma gráfica.

12.3. Modelo de aplicación

La construcción de un P1P con muchas variables incorpora dos problemas, el primero es la construcción del propio modelo en forma manual, la cantidad de términos del polinomio crece exponencialmente con el número de variables.

Cuando las combinaciones de variables son muchas la construcción manual esta sujeta a errores, es necesario utilizar alguna herramienta que permita su construcción sistemática. El *método de evaluación de combinaciones* se deriva de la definición de los P1P, la implementación con un algoritmo implica generar todas las combinaciones de variables y, para cada una, evaluar las condiciones de la definición de los P1P. Si las combinaciones son mucha el algoritmo demora en el resultado. El *método de las combinaciones binarias* utiliza una tabla con las combinaciones binarias y genera de forma directa los términos del P1P, por esto resulta un algoritmo más rápido y simple.

El segundo problema es la complejidad para computar el modelo, esto lleva a intentar reducir la complejidad disminuyendo el número de variables e intentando mantener buenos resultados. El *método de la matriz de contribuciones* se basa en la reducción de las variables analizando la propia muestra de datos. El resultado de la reducción genero un modelo que logro aprender las características de tráfico aun con 5 variables de las 12 iniciales. Nuevamente en este trabajo aparece la importancia de realizar una buena selección de las variables de entrada y la definición adecuada del sistema a tratar.

12.4. Metamodelo P1P

Un metamodelo con P1P permite estimar valores a partir de estimaciones de sub-sistemas, permite estudiar los sub-sistemas de forma independiente como parte de un metamodelo. El estudio y aplicación de un metamodelo P1P proporciona nuevos caminos de investigación y aplicación de los P1P a problemáticas de redes.

12.5. Análisis de los P1P con la teoría de la información

El análisis de los modelos P1P con la teoría de la información requiere de un estudio más profundo y comprobarlo matemáticamente, el análisis heurístico realizado induce a que los P1P podrían tener una entropía mínima para la construcción de modelos estocásticos.

12.6. Comparaciones

La medición de tiempos de llegadas y atención de mensajes SMTP para modelar un servidor SMTP con Análisis de colas es dificultosa, resulta difícil abrir el software y medir los tiempos. Realizar un modelo de colas basado en el tráfico observable a nivel de red, filtrando los paquetes hacia y desde un servidor, resulta de menor dificultad. Los volcados de paquetes proporcionan el tiempo de cada paquete.

La construcción de un modelo de tráfico para un servidor SMTP con P1P resulto menos compleja que un modelo de colas, el modelo P1P aprendió el patrón de tráfico particular SMTP, mientras que el modelo de colas está basado en casos generales.

Contribución

El principal aporte del trabajo es un nuevo método para modelar y estudiar tráfico en redes IP. Describimos las contribuciones del trabajo en el orden en que aparecen:

1. Se detallo la teoría de los P1P con analogías al estudio de tráfico en redes ampliando los trabajos originales, detallando la demostración, y la construcción de matrices.
2. Generamos un modelo de tráfico basado en el nivel de red y transporte.
3. Generamos un modelo de tráfico de aplicación.
4. Generamos dos métodos para la construccion sistematica de los P1P con muchas variables.
5. Generamos un método para determinar que variables son más representativas del tráfico y reducir la cantidad de variables para construir el P1P.
6. Comenzamos un estudio con la teoría de la información, analizando las capacidades de los P1P de aprender la información de tráfico.
7. Para cada trabajo realizamos programas con la herramienta Mathematica.
8. Realizamos Scripts en php que procesan la información de volcado TCPdump.

Trabajos futuros

El trabajo genero disparadores de futuros trabajos y líneas de investigación. Algunos trabajos futuros podrían ser:

1. Utilización de los P1P para la detección de tráfico autosimilar.
2. Construcción de modelos P1P para la detección de la congestión.
3. Utilización de un modelo P1P de tráfico para calidad de servicio.
4. La aplicación y utilización de un metamodelo P1P para el análisis de tráfico.

Una línea de investigación es:

1. Utilizar modelos de aprendizaje en máquinas (como los P1P) para el modelado de tráfico, este trabajo demuestra que la utilización de técnicas de aprendizaje podría generar aportes significativos para el estudio de tráfico estadístico.

Parte VI

Bibliografía

Bibliografía

[1] Jorge Nanclares, Ulises Rapallini, **Harmonic Theory and Machine Learning**, Journal Of Computer Science & Technologies, publicado en Octubre 2007

[2] O.R. Faure, J. Nanclares, And U. Rapallini, **Harmonic Functions On The Closed Cube: An Application To Learning Theory**, Revista de la "Unión Matemática Argentina" Volumen 48, Número 1, 2007, Páginas 65 a 74.

[3] W. Stallings, **High-Speed Networks and Internets: Performance and Quality of Service**, 2/E, Prentice Hall Copyright: 2002.

[4] Giovanni Giambene, **Queuing Theory And Telecommunications Networks And Applications**, 2005 Springer Science+Business Media, Inc.

[5] Ricardo VECCHIO: **Modelización De Tráfico Auto-Similar Y Evaluación De Sus Efectos En El Tamaño De Las Colas** – ITBA – 2003

[6] Mac Kay, **Information Theory, Inference, and Learning Algorithms**, Cambridge University Press Sep 2003 (http://www.inference.phy.cam.ac.uk/mackay/itprnn/book.html)

[7] B.Ya. Ryabko, A.N. Fionov, V.A. Monarev, and Yu.I. Shokin, **Chapter Using information theory approach to randomness testing**, Book: Computational Science and High Performance Computing II – Springer 2006

[8] Jeffrey R. Spirn, **Network Modeling With Bursty Traffic And Finite Buffer** Space 1982

[9] Carey L. Williamson, **Network Traffic Measurement and Modeling**, Department of Computer Science University of Saskatchewan- 1995

[10] Leland E. Will. Taqqu Murrad S. : **On the Self –Similar Nature of Ethernet Traffic (Extended Versión)**. 1994.

[11] V. Frost and B. Melamed, **Simulating Telecommunications Networks with Traffic Modeling**, IEEE Communications Magazine, Vol. 32, No. 3, pp. 70- 81, March 1994.

[12] ITU–D, Study Group 2 Question 16/2, Handbook **"TELETRAFFIC ENGINEERING"**, 2005

[13] Junsoo Lee, Stephan Bohacek, João P. Hespanha„ and Katia Obraczka, **Modeling Communication Networks With Hybrid Systems**, IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 15, NO. 3, JUNE 2007

[14] Benjamin Melamed, Jon R. Hill **A SURVEY OF TES MODELING APPLICATIONS. SIMULATION**, Vol. 64, No. 6, 353 - 370, 1995.

[15] B. Melamed, Q. Ren and B. Sengupta, **The QTES/PH/1 Queue, Performance Evaluation**, Vol. 26, 1–20, 1996.Tags

[16] Guntel Bolch, Stefan greiner, Hermann de Meer, and kishor S. Trivedi ,**Queueing networks and Markov chains: Modeling and Performance evaluation with computer science applications**, John wiley & sonc, Inc. 1998

[17] Richard A. Berk, **Statistical Learning From a Regression Perspective**, Springer 2008

[18] Trevor Hastie, Robert Tibshirani, Jerome Friedman, **The Elements of Statistical Learning Data Mining, Inference, and Prediction**, Springer 2009

[19] TCPdump, http://www.tcpdump.org/

[20] Mathematica for students, http://www.wolfram.com/products/student/mathforstudents/index.html

[21] PHP, http://www.php.net

Parte VII

Apéndices

Programas Mathematica

15.1. Modelo de trasporte y red

Primera parte del programa

```
* Carga del modulo de estadistica para utilizar la función de regresion *)
<< Statistics‘NonlinearFit‘
(* Borro las variables de la memoria, por las dudas !!!! *)
Clear[datos, datos2, bytes, tcp, udp, otros, bytesproto];
Clear[x1, x2, x3, c, c1, c2, c3, c4, c5, c6, c7];
(*
 DATOS EN EL ARCHIVO
 x1n : segmentos TCP por segundo
 x2n : datagramas UDP por segundo
 x3n : otros protocolos por segundo
 b : bytes en paquetes IP por segundo
 *)
(*
 cargo los datos en una matriz donde las filas son :
 {x11, x21, x31, b1},
 {x12, x22, x32, b2},
 ...
 {x1n, x2n, x3n, bn}
 estan en un archivo csv generado con un script php
 *)
datos = Import["D:/
 works/tesis/mathematica/ejemplo_datos_reales/10.10.20.10_TCP_UDP_OTROS_\
BYTESIP.csv", ÇSV"];
(* separo los datos en vectores para graficas por separado *)
datos2 = Transpose[datos];
tcp = datos2[[1]];
udp = datos2[[2]];
otros = datos2[[3]];
bytesproto = datos2[[4]];
(* grafico los venctores con los datos de PDU y bytes IP *)
ListPlot[bytes, PlotJoined -> True, PlotStyle -> Hue[.6]];
ListPlot[tcp, PlotJoined -> True, PlotStyle -> Hue[.6]];
ListPlot[udp, PlotJoined -> True, PlotStyle -> Hue[.6]];
ListPlot[otros, PlotJoined -> True, PlotStyle -> Hue[.6]];
```

```
ListPlot[bytesproto, PlotJoined -> True, PlotStyle -> Hue[.6]];
 (* El modelo final es
 p1p[x1, x2, x3] = c + c1 * x1 + c2 * x3 + c3 * x3 +
 c4 * x1 * x2 +
 c5 * x1 * x3 +
 c6 * x2 * x3 +
 c7 * x1 * x2 * x3;
 *)
(* CALCULO DE LOS COEFICIENTES P1P con una regresion no lineal. *)
 NonlinearRegress[ datos,
 c + c1 * x1 + c2*x3 + c3 * x3 +
 c4 * x1 * x2 + c5 * x1 * x3 +
 c6 * x2 * x3 +
 c7* x1*x2*x3,
 {x1, x2, x3},
 {c, c1, c2, c3, c4, c5, c6, c7}]
```

Segunda parte del programa

```
(* ej. estimamos la cantidad de bytes IP por segundo para un valor de TCP, \
UDP y OTROS por segundo *)

Clear[c, c1, c2, c3, c4, c5, c6, c7];
c = -1525.7968245050279;
c1 = 758.058985587274;
c2 = 244.1780938878281;
c3 = 244.17809388783107;
c4 = 0.5269939382491858;
c5 = 23.866977788539064;
c6 = -11.210399019150879;
c7 = -0.4931779770254915;
x1 = 60;
x2 = 2;
x3 = 2;
r = c + c1 * x1 + c2*x2 + c3 * x3 + c4 * x1 * x2 + c5 *
 x1 * x3 + c6 * x2 * x3 + c7 * x1 * x2 * x3

(* ej2. estimamos la cantidad de bytes para un vertor de datos TCP,
 UDP, OTROs, consecutivos por segundo *)

Clear[c, c1, c2, c3, c4, c5, c6, c7];
c = -1525.7968245050279;
c1 = 758.058985587274;
c2 = 244.1780938878281;
c3 = 244.17809388783107;
c4 = 0.5269939382491858;
c5 = 23.866977788539064;
c6 = -11.210399019150879;
c7 = -0.4931779770254915;
x1 = {60, 64, 79, 74, 60, 58};
x2 = {2, 4, 2, 1, 0, 3};
```

```
x3 = {2, 1, 1, 0, 1, 2};
r = c + c1 * x1 + c2*x2 + c3 * x3 + c4 * x1 *
 x2 + c5 * x1 * x3 + c6 * x2 * x3 + c7 * x1 * x2 * x3
```

15.2. Modelo de aplicación

Primera parte del programa, calculo de los coeficientes y graficas de tráfico por puerto.

```
(* Carga del modulo de esdistica para utilizar la funcion de regresion *)
<< Statistics‘NonlinearFit‘
(* Borro las variables de la memoria, por las dudas !!!! *)
Clear[datos, datos2, X1, X2, X3, X4, X5, c0, c1, c2, c3, c4, c5, c6, c7, c8, \
c9, c10,

 c11, c12, c13, c14, c15, c16, c17, c18, c19, c20,
 c21, c22, c23, c24, c25, c26,
 c27, c28, c29, c30, c31];
(*
 DATOS EN EL ARCHIVO
 p_554; cantidad de paquetes dirijidos al puerto 554
 p_53; paquetes al port 53
 p_80; paquetes al port 80
 p_sp; paquetes sin puerto
 p_443; paquetes l port 443
 b : bytes en paquetes IP por segundo
 *)
datos = Import["D:/works/tesis/mathematica/ejemplo2_puertos/analisis_ports_\
select.csv", ÇSV"];
(* separo los datos en vectores para graficas por separado *)
datos2 = Transpose[datos];
(* Grafico por separado la cantidad de paquetes por cada puerto *)
cant = Length[ datos2 ];
 For[i = 1, i < cant,
 ListPlot[datos2[[i]], PlotJoined -> True, PlotStyle -> Hue[.6]], i++
 ];

(* El modelo tiene 5 variables, una por puerto seleccionado

 p1p = c0 + c1*X5 + c2*X4 + c3*X4*X5 + c4*X3 + c5*X3*X5 + c6*X3*X4 + \
c7*X3*X4*X5 + c8*X2 + c9*X2*X5 + c10*X2*X4 + c11*X2*X4*X5 + c12*X2*X3 +
 c13*X2*X3*X5 +
 c14*X2*X3*X4 + c15*X2*X3*X4*X5 + c16*X1 + c17*X1*X5 + c18*X1*
 X4 + c19*X1*X4*X5 + c20*X1*X3 + c21*X1*X3*X5 + c22*X1*X3*X4 +
 c23*X1*X3*X4*X5 + c24*X1*X2 + c25*X1*X2*X5 + c26*X1*X2*X4 + c27*
 X1*X2*X4*X5 + c28*X1*X2*X3 + c29*X1*X2*X3*X5 + c30*X1*
 X2*X3*X4 + c31*X1*X2*X3*X4*X5 +;

 CALCULO DE LOS COEFICIENTES P1P con una regresion no lineal.

 lo realizo con dos funciones solo para obtener directamente el modelo
```

```
 con la funcion NolinearFit
 y con todos los parametros con la funcion NolinearRegress *)
NonlinearFit[ datos,
 c0 + c1*X5 + c2*X4 + c3*X4*X5 + c4*X3 + c5*X3*X5 + c6*X3*X4 + c7*X3*X4*X5 + \
c8*X2 + c9*X2*X5 + c10*X2*X4 + c11*X2*X4*X5 + c12*X2*X3 + c13*X2*X3*X5 +
 c14*X2*X3*X4 + c15*X2*X3*X4*X5 + c16*X1 + c17*X1*X5 +
 c18*X1*X4 + c19*X1*X4*X5 + c20*X1*X3 + c21*X1*X3*
 X5 + c22*X1*X3*X4 + c23*X1*X3*X4*X5 + c24*X1*X2 + c25*
 X1*X2*X5 + c26*X1*X2*X4 + c27*X1*X2*X4*X5 + c28*X1*X2*X3 +
 c29*X1*X2*X3*X5 + c30*X1*X2*X3*X4 + c31*X1*X2*X3*X4*X5,
 {X1, X2, X3, X4, X5},
 {c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13, c14, c15,
 c16, c17, c18, c19, c20, c21, c22, c23, c24, c25, c26, c27, c28,
 c29, c30, c31 } ]
NonlinearRegress[ datos,
 c0 + c1*X5 + c2*X4 + c3*X4*X5 + c4*X3 + c5*X3*X5 + c6*X3*X4 + c7*X3*X4*
 X5 + c8*X2 + c9*X2*X5 + c10*X2*X4 + c11*X2*X4*X5 + c12*X2*X3 + c13*X2*X3*
 X5 + c14*X2*X3*X4 + c15*X2*X3*X4*X5 + c16*X1 + c17*X1*X5 + c18*X1*
 X4 + c19*X1*X4*X5 + c20*X1*X3 + c21*X1*X3*X5 + c22*X1*X3*X4 + c23*X1*
 X3*X4*X5 + c24*X1*X2 + c25*X1*X2*X5 + c26*X1*X2*X4 + c27*X1*X2*X4*X5 +
 c28*X1*X2*X3 + c29*X1*X2*X3*X5 + c30*X1*X2*X3*X4 + c31*X1*X2*X3*X4*X5,
 {X1, X2, X3, X4, X5},
 {c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13, c14, c15,
 c16, c17, c18, c19, c20, c21, c22, c23, c24, c25, c26, c27, c28,
 c29, c30, c31 } ]
```

Segunda parte del programa, estimaciones y graficas

```
(* ESTIMACIONES
 cargo los valores de los parametros ci i = 0 a 31
 *)
Clear[b, BY, ERR, X1, X2, X3, X4, X5, c0, c1, c2, c3,
 c4, c5, c6, c7, c8, c9, c10,
c11, c12, c13, c14, c15, c16, c17, c18, c19, c20, c21, c22, c23, c24, c25,
 c26, c27, c28, c29, c30, c31];
c0 = -10982.814770361423;
c1 = 473.9213302845348;
c2 = 1144.3941517453077;
c3 = 24.557846247006697;
c4 = 784.9603752916984;
c5 = 13.904245184349369;
c6 = 1.8665668990245798;
c7 = -0.8085290118174608;
c8 = 113.14482417885597;
c9 = 44.14585619610631;
c10 = -10.607820923314799;
c11 = -1.8244385858594567;
c12 = -0.8333782760414918;
c13 = -2.2180546460278;
c14 = 0.19393113670488074;
c15 = 0.11391877991246326;
```

```
c16 = 1068.2335782691548;
c17 = 10.165143802277731;
c18 = -6.210557008079236;
c19 = -0.7907491705171666;
c20 = -1.657139454770654;
c21 = -0.49805697194762655;
c22 = 0.08924615409005157;
c23 = 0.030468937706188835;
c24 = 1.280677545467686;
c25 = -1.1807614678540275;
c26 = 0.18626926722275175;
c27 = 0.0575086454333244;
c28 = -0.05062294572170121;
c29 = 0.06202406745585728;
c30 = -0.0018210839332631497;
c31 = -0.0026072128388541756;
(* tomo los datos reales para probar el modelo *)
datos = Import["D:/
 works/tesis/mathematica/ejemplo2_puertos/analisis_ports_select.csv", \
ÇSV"];
(* separo los datos en las variables X1, X2, X3, X4, X5, BY *)
datos2 = Transpose[datos];
(* asigno a las variabes del modelo los datos reales *)
X1 = datos2[[1]];
X2 = datos2[[2]];
X3 = datos2[[3]];
X4 = datos2[[4]];
X5 = datos2[[5]];
BY = datos2[[6]]; (* bytes reales *)
(* estimo los valores de bytes con los datos reales y el modelo realizado *)
b = c0 + c1*X5 + c2*X4 + c3*X4*X5 + c4*X3 + c5*X3*X5 + c6*
 X3*X4 + c7*X3*X4*X5 + c8*X2 + c9*X2*X5 + c10*X2*
 X4 + c11*X2*X4*X5 + c12*X2*X3 + c13*X2*X3*X5 + c14*X2*X3*X4 + \
c15*X2*X3*X4*X5 + c16*X1 + c17*X1*X5 + c18*X1*X4 + c19*X1*X4*X5 + c20*
 X1*X3 + c21*X1*X3*X5 + c22*X1*X3*X4 +
 c23*X1*X3*X4*X5 + c24*X1*X2 + c25*X1*
 X2*X5 + c26*X1*X2*X4 + c27*X1*X2*X4*X5 + c28*X1*X2*X3 + c29*X1*
 X2*X3*X5 + c30*X1*X2*X3*X4 + c31*X1*X2*X3*X4*X5;
(* Grafico los bytes estimados
 con el modelo *)
ListPlot[b, PlotJoined -> True, PlotStyle -> \
Hue[.6]]
(* Grafico los bytes reales *)
ListPlot[BY, PlotJoined -> True, PlotStyle -> Hue[.6]]
(* Calculo el error absoluto *)
ERR = Sqrt[(b - BY)(b - BY)];
(* Grafico el error *)
ListPlot[ERR, PlotJoined -> True, PlotStyle -> Hue[.6]]
```

Scripts php

16.1. Procesamiento de datos para modelo de Transporte y Red

```
<?
 $archivo = ”10.10.20.10.tcpdump";
 $resultado = ”10.10.20.10_TCP_UDP_OTROS_BYTESIP.csv";

$fn = fopen($archivo, ”r");

 $f_out = fopen($resultado,”w");
 $seg_ant =”qqq";
 $seg_act =”01";
 $linea = 0;
 $cont_TCP = 0;
 $cont_UDP = 0;
 $cont_OTROS = 1;
 $cont_bytes =0;
 $seg_virtual = 0;
 while( !feof($fn) ) {
 $buffer = fgets($fn, 1024);
        $linea = $linea + 1;
        $arr = explode(’ ’,$buffer);
        $time_stamp = $arr[0];
        $hora = explode(’:’,$time_stamp);
        if(sizeof($hora) ==3 ){
                $seg_aux = explode(’.’,$hora[2]);
                $seg_act = $seg_aux[0];
        }
        if($seg_act <> $seg_ant){
                $seg_virtual++;
                echo ”<br>LIN: ”.$linea."SEG: ”.$seg_virtual.” BYTES: ".
                   $cont_bytes.” TCP: ".$cont_TCP.” UDP: ".$cont_UDP.” OTROS: ".$cont_OTROS;

                /* calculo en tamaño del buffer como:
                    cantidad de bytes / cantidad de protocolos TCP+UDP+OTROS */
                $xx = intval($cont_bytes/($cont_TCP+$cont_UDP+$cont_OTROS));

                /* formato de la linea en el archivo:
                Bytes/ seg, TCP por seg, UDP por Seg, Buffer */
```

```
                $linea_csv = array($cont_TCP, $cont_UDP, $cont_OTROS, $cont_bytes);

                // escribo la linea en el archivo
                fputcsv( $f_out, $linea_csv, ",", "\n");

                // contadores a cero para el proximo segundo
                $cont_TCP = 0;
                $cont_UDP = 0;
                $cont_OTROS = 0;
                $cont_bytes =0;
                $seg_ant = $seg_act;
        }

        $pos = array_search('proto:',$arr);
        //print_r($arr);
        //echo "<br>". $pos;
        if($pos <> FALSE) {
                $suma = 's';
                if(trim($arr[$pos+1]) == "TCP"){ $cont_TCP++; $suma = 'n'; }
                if(trim($arr[$pos+1]) == ÜDP"){ $cont_UDP++; $suma = 'n'; }
                if( $suma == 's' ){ $cont_OTROS++; }
        }
        $pos_leng = array_search('length:',$arr);
        if($pos_leng <> FALSE){
                $bytes = substr($arr[$pos_leng+1],0,strlen($arr[$pos_leng+1]-1));
                $cont_bytes = $cont_bytes + $bytes ;
        }
 }
 fclose($fn);
 fclose($f_out);


?>
```

16.2. Procesamiento de datos para el modelo de Aplicación

```
<?
/*
 analiza el volcado TCP
    Retorna un archivo con
 una matriz donde:
    las primeras columnas estan formadas por los protocolos dentro de IP y sus tamaños
    las demas columnas son la cantidad de mensajes a puertos (por puerto) y sus tamaños
    la ultima columna es la suma de bytes en paquetes IP

    todas las medidas por segundo.

*/
 $archivo = "10.10.20.10.tcpdump";
```

```
$resultado = ”analisis.csv";
$datos_array = analisis_trafico($archivo);
$datos_array[”cont_bytes"] = 0;

print_r($datos_array); echo ”<br>";

$f_out = fopen($resultado,”w");

$linea_csv = array_keys( $datos_array );
fputcsv( $f_out, $linea_csv, ”;", ”\n");
//$linea = 0;
//$seg_virtual = 0;
$seg_ant =”qqq";
$seg_act =”01";
$fn = fopen($archivo, ”r");
while( !feof($fn) ) {
$buffer = fgets($fn, 1024);
       //echo $buffer.”<br>";
       //$linea++;
       //separo los campos y los guardo en un array
       $arr = explode(’ ’,$buffer);
       // obtengo la marca de tiempo y segundo
       $time_stamp = $arr[0];
       $hora = explode(’:’,$time_stamp);
       if(sizeof($hora) == 3 ){
       $seg_aux = explode(’.’,$hora[2]);
       $seg_act = $seg_aux[0];
       }

// si el segundo cambia, gravo los datos en el archivo y pongo a cero los contadores
       if($seg_act <> $seg_ant){
       //$seg_virtual++;
       $linea_csv = array_values($datos_array);
       fputcsv( $f_out, $linea_csv, ”;", ”\n");

           print_r($datos_array); echo ”<br>";

           //pongo e cero los contarores de protocolos, puertos y bytes
           foreach($datos_array as $k=> $v ){
           $datos_array[$k] = 0;
           }
$seg_ant = $seg_act;
       }

       // obtengo longitud del paquete y sumo los bytes
       $pos_leng = array_search(’length:’,$arr);
       $bytes = 0;
       //echo "LENGH".$pos_leng. "<br>";
       if($pos_leng <> FALSE){
```

```
        $bytes = substr($arr[$pos_leng+1],0,strlen($arr[$pos_leng+1]-1));
        $datos_array[çont_bytes"] = $datos_array[çont_bytes"] + $bytes ;
        }

        $pos = array_search('proto:',$arr);
        //print_r($arr);
        //echo "PROTO".$pos."<br>";
        if($pos <> FALSE) {
        $str_proto = $arr[$pos+1];
        $name_proto = "pr_".trim($str_proto);
                    $byte_proto = "by_".trim($str_proto);
                    $pos_p = array_key_exists($name_proto,$datos_array);
        // si el protocolo no esta en el array, agrego el contador para ese proto
        //echo "PROTO_p".$pos_p."<br>";
                    if( $pos_p <> FALSE ) {
        $datos_array[$name_proto] = $datos_array[$name_proto] + 1;
                    $datos_array[$byte_proto] = $datos_array[$byte_proto] + $bytes;
                    }
        }

    //extraigo IP y Port
    $pos_ip = array_search('>',$arr);
           //echo ">IP".$pos_ip ."<br>";
           //echo "<br>IP:". $arr[$pos_ip-1] . "IP:". $arr[$pos_ip+1];
           if( $pos_ip <> FALSE ) {
                       $port = port_ip($arr[$pos_ip-1],$arr[$pos_ip+1]);
                       $name_port = "p_".trim($port);
                       $byte_port = "by_".trim($port);
                       $pos_port = array_key_exists($name_port, $datos_array);
                       //echo "PORT". $pos_port ."<br>";
                       // si el puerto no esta en el array agrego el contador para ese port
                       if( $pos_port <> FALSE ) {
               $datos_array[$name_port] = $datos_array[$name_port] + 1;
                               $datos_array[$byte_port] = $datos_array[$byte_port] + $bytes;
                       }
                       //echo "<br>PORT: ". $port;
    }
    }
    fclose($fn);
    fclose($f_out);
    function analisis_trafico($archivo_nombre){

    $fn1 = fopen($archivo_nombre, "r");
    $proto_array = Array();
    $ports_array = Array();
    while( !feof($fn1) ) {
    $buffer = fgets($fn1, 1024);
                   //separo los campos y los guardo en un array
           $arr = explode(' ',$buffer);
```

```
       $pos = array_search('proto:',$arr);
            if($pos <> FALSE) {
               $str_proto = $arr[$pos+1];
               $name_proto = "pr_".trim($str_proto);
                           $byte_proto = "by_".trim($str_proto);
                           $pos_p = array_key_exists($name_proto,$proto_array);
               // si el protocolo no esta en el array, agrego el contador para ese proto
               if( $pos_p == FALSE ) {
               $proto_array[$name_proto] = 0;
               $proto_array[$byte_proto] = 0;
               }
                   }

    //extraigo IP y Port
    $pos_ip = array_search('>',$arr);
           if( $pos_ip <> FALSE ) {
                               $port = port_ip($arr[$pos_ip-1],$arr[$pos_ip+1]);
                               $name_port = "p_".trim($port);
                               $byte_port = "by_".trim($port);
                               $pos_port = array_key_exists($name_port, $ports_array);
                               // si el puerto no esta en el array agrego el contador para ese
port
                               if( $pos_port == FALSE ) {
                               $ports_array[$name_port] = 0;
                               $ports_array[$byte_port] = 0;
                           }
                   }
           }
    fclose($fn1);
       $analisis = array_merge($proto_array,$ports_array);
    return($analisis);
    }

    function port_ip($ip1, $ip2){
      $puerto = "sp";
        $arr_ip1 = explode(".",$ip1);
      $arr_ip2 = explode(".",$ip2);
      $long = sizeof($arr_ip1);
            if( $long >= 5) {
           $port_ip1 = intval($arr_ip1[4]);
              $port_ip2 = intval(substr($arr_ip2[4],0,strlen($arr_ip2[4])-1)) );
        if( $port_ip1 < 1025 ){
           $puerto = $port_ip1;
            }
            if($port_ip2 < 1025) {
             $puerto = $port_ip2;
          }
       }
       return($puerto);
```

```
 }

?>
```

16.3. Procesamiento para el calculo de la matriz de contribuciones

```
<?
/*
 Modelo de aplicacion
 Matriz de contribuciones.
*/
 $archivo = ”10.10.20.10.tcpdump";
 //$resultado = ”analisis_ports.csv"
 //$resultado = ”analisis_ports_bytes.csv";
 $resultado = ”analisis_ports_bytes_promedio.csv";
 $datos_array = analisis_trafico($archivo);
 $datos_array[”cont_bytes"] = 0;

 //print_r($datos_array); echo "<br>";

 $f_out = fopen($resultado,”w");
 $linea_csv = array_keys( $datos_array );
 fputcsv( $f_out, $linea_csv, ",", "\n");

 echo implode(",",$linea_csv)."<br>";

 //$linea = 0;
 //$seg_virtual = 0;
 $seg_ant =”qqq";
 $seg_act =”01";
 $fn = fopen($archivo, ”r");
 while( !feof($fn) ) {
 $buffer = fgets($fn, 1024);
        //echo $buffer.”<br>";
        //$linea++;
        //separo los campos y los guardo en un array
        $arr = explode(’ ’,$buffer);
        // obtengo la marca de tiempo y segundo
        $time_stamp = $arr[0];
        $hora = explode(’:’,$time_stamp);
        if(sizeof($hora) == 3 ){
        $seg_aux = explode(’.’,$hora[2]);
        $seg_act = $seg_aux[0];
        }

 // si el segundo cambia, grabo los datos en el archivo y pongo a cero los contadores
        if($seg_act <> $seg_ant){
        //$seg_virtual++;
```

```
            $linea_csv = array_values($datos_array);
        /*fputcsv( $f_out, $linea_csv, ",", "\n");
            echo implode(",",$linea_csv)."<br>";*/

            $csv2 = Array();
            $long = sizeof($linea_csv);
            $totalb = $linea_csv[$long-1];
            $j = 0;
            for($i=0; $i < $long - 1; $i=$i+2){
            if($linea_csv[$i] == 0){ $valor = 0;}
                        else { $valor = $linea_csv[$i+1]/$totalb; }
                        $csv2[$j] = $valor;
                        $j++;
            }

            $linea_csv = array_values($csv2);
        fputcsv( $f_out, $linea_csv, ",", "\n");
            echo implode(",",$linea_csv)."<br>";
            //print_r($datos_array); echo "<br>";

            //pongo e cero los contarores de protocolos, puertos y bytes
            foreach($datos_array as $k=> $v ){
            $datos_array[$k] = 0;
            }
    $seg_ant = $seg_act;
        }

        // obtengo longitud del paquete y sumo los bytes
        $pos_leng = array_search('length:',$arr);
        $bytes = 0;
        //echo "LENGH".$pos_leng. "<br>";
        if($pos_leng <> FALSE){
        $bytes = substr($arr[$pos_leng+1],0,strlen($arr[$pos_leng+1]-1));
        $datos_array[çont_bytes"] = $datos_array[çont_bytes"] + $bytes ;
        }

        $pos = array_search('proto:',$arr);
        //print_r($arr);
        //echo "PROTO".$pos."<br>";
        //if($pos <> FALSE) {
        // $str_proto = $arr[$pos+1];
        // $name_proto = "pr_".trim($str_proto);
    //            $byte_proto = "by_".trim($str_proto);
        //            $pos_p = array_key_exists($name_proto,$datos_array);
        // si el protocolo no esta en el array, agrego el contador para ese proto
        //echo "PROTO_p".$pos_p."<br>";
        //            if( $pos_p <> FALSE ) {
        // $datos_array[$name_proto] = $datos_array[$name_proto] + 1;
        //        $datos_array[$byte_proto] = $datos_array[$byte_proto] + $bytes;
```

```
        //            }
        //}

    //extraigo IP y Port
    $pos_ip = array_search('>',$arr);
        //echo ">IP".$pos_ip ."<br>";
        //echo "<br>IP:". $arr[$pos_ip-1] . "IP:". $arr[$pos_ip+1];
        if( $pos_ip <> FALSE ) {
                    $port = port_ip($arr[$pos_ip-1],$arr[$pos_ip+1]);
                    $name_port = "p_".trim($port);
                    $byte_port = "by_".trim($port);
                    $pos_port = array_key_exists($name_port, $datos_array);
                    //echo "PORT". $pos_port ."<br>";
                    // si el puerto no esta en el array agrego el contador para ese port
                    if( $pos_port <> FALSE ) {
            $datos_array[$name_port] = $datos_array[$name_port] + 1;
                            $datos_array[$byte_port] = $datos_array[$byte_port] + $bytes;
                    }
                    //echo "<br>PORT: ". $port;
    }
    }
    fclose($fn);
    fclose($f_out);
    function analisis_trafico($archivo_nombre){

    $fn1 = fopen($archivo_nombre, "r");
    $proto_array = Array();
    $ports_array = Array();
    while( !feof($fn1) ) {
    $buffer = fgets($fn1, 1024);
                //separo los campos y los guardo en un array
        $arr = explode(' ',$buffer);
    $pos = array_search('proto:',$arr);
         if($pos <> FALSE) {
            $str_proto = $arr[$pos+1];
            $name_proto = "pr_".trim($str_proto);
                        $byte_proto = "by_".trim($str_proto);
                        $pos_p = array_key_exists($name_proto,$proto_array);
            // si el protocolo no esta en el array, agrego el contador para ese proto
            if( $pos_p == FALSE ) {
            $proto_array[$name_proto] = 0;
            $proto_array[$byte_proto] = 0;
            }
                }

    //extraigo IP y Port
    $pos_ip = array_search('>',$arr);
        if( $pos_ip <> FALSE ) {
                            $port = port_ip($arr[$pos_ip-1],$arr[$pos_ip+1]);
```

```
                                $name_port = "p_".trim($port);
                                $byte_port = "by_".trim($port);
                                $pos_port = array_key_exists($name_port, $ports_array);
                                // si el puerto no esta en el array agrego el contador para ese
port
                                if( $pos_port == FALSE ) {
                                $ports_array[$name_port] = 0;
                                $ports_array[$byte_port] = 0;
                            }
                        }
                }
    fclose($fn1);
       $analisis = $ports_array; //array_merge($proto_array,$ports_array);
    return($analisis);
    }

    function port_ip($ip1, $ip2){
    $puerto = "sp";
           $arr_ip1 = explode(".",$ip1);
    $arr_ip2 = explode(".",$ip2);
    $long = sizeof($arr_ip1);
           if( $long >= 5) {
    $port_ip1 = intval($arr_ip1[4]);
    $port_ip2 = intval(substr($arr_ip2[4],0,strlen($arr_ip2[4])-1)) );
    if( $port_ip1 < 1025 ){
    $puerto = $port_ip1;
           }
           if($port_ip2 < 1025) {
       $puerto = $port_ip2;
       }
       }
       return($puerto);
    }



   ?>
```

16.4. Procesamiento de datos para SMTP-colas

```
   /*
    Procesamiento del archivo de volcado tcpdump
    para analisis de trafico en servidor SMTP.
    Facultad de informatica - UNLP
    Maestria en Redes de Datos
    Ulises Rapallini
   */
    $archivo = "trafico_DMZ_10.10.10.34_correo.txt";
    $resultado1 = "smtp_llegadas.csv";
```

```
$resultado2 = ”smtp_salidas.csv";
$fn = fopen($archivo, ”r");
$f_out_llegadas = fopen($resultado1,"w");
$f_out_salidas = fopen($resultado2,"w");

// variables para las llegadas
$time_stamp = ”"; //para la marca de tiempo
$time_diff = 0; //para calcular la diferencia entre tiempos
$ta_ant = 000000; //marca de tiempo del arribo anterior
$ta_act = 000001; //marca de tiempo del arribo actual
$total_bytes = 0; //para contar los bytes en los paquetes
$total_time_entrante = 0; //para contar el tiempo en microsegundos
$total_pk_entrante = 0;

// variables para las salidas, atencion del servidor
$time_stamp1 = ”"; //para la marca de tiempo de salida
$time_diff1 = 0; //para calcular la diferencia entre tiempos de atancion
$ts_ant = 000000; //marca de tiempo del salida anterior
$ts_act = 000001; //marca de tiempo del salida actual
$total_bytes1 = 0; //para contar los bytes en los paquetes
$total_time_salida = 0; //para contar el tiempo en microsegundos
$total_pk_salida = 0;
$linea = 0;

while( !feof($fn) ) {

    $buffer = fgets($fn, 1024);
    // cargo en un array la linea, separando en campos por espacios
    $arr = explode(’ ’,$buffer);

    //si el paquete encapsula un segmento TCP
    $pos_proto = array_search(’TCP’,$arr);
    //echo ”<br>"; print_r($arr);

    //echo ”<br>POS:". $pos_ip;
    if( $pos_proto <> FALSE ) {

           $pos_ip = array_search(’>’,$arr);
           $ipO = $arr[$pos_ip-1]; //ip origen
           $ipD = $arr[$pos_ip+1]; //ip destino

     //consulto si es un paquete entrante o saliente
           $t_paquete = port_25( $ipO, $ipD );

           // si el paquete es entrante
           if( $t_paquete == 10 ){

                       //extraigo el campo timestamp
                       $time_stamp = $arr[0];
```

```
                $time_diff = diferencia_tiempo($time_stamp, $ta_ant);

              //sumo los tiempos interpaquetes
                $total_time_entrante = $total_time_entrante + $time_diff;

                //extraigo el campo de longitud del paquete IP
              $pos_leng = array_search('length:',$arr);
              if($pos_leng <> FALSE){

                 //extraigo el tamaño del paquete
                     $bytes_ip = substr($arr[$pos_leng+1], 0,
                                 strlen($arr[$pos_leng+1]-1));

                     //sumo los bytes
                     $total_bytes = $total_bytes + $bytes_ip;
                }

                // cuento la cantidad de paquetes entrantes
                $total_pk_entrante++;

                // guardo la marca de tiempo para calcular la proxima diferencia
              $ta_ant = $time_stamp;

              // formato de la linea en el archivo
                $linea_csv_a = array($time_diff, $bytes_ip);

                //escribo la linea en el archivo
                //fputcsv( $f_out_llegadas, $linea_csv_a, ",", "\n");
                // muestro en la pagina los datos
              //echo "<br>LIN: ".$linea.
                     "TIEMPO ARRIBO: ".$time_diff.
                     "BYTES IP: ".$bytes_ip ;
}

if( $t_paquete == 20) {
        //extraigo el campo timestamp
              $time_stamp1 = $arr[0];
              $time_diff1 = diferencia_tiempo($time_stamp1, $ts_ant);

              //sumo los tiempos interpaquetes
                $total_time_salida = $total_time_salida + $time_diff1;

                //extraigo el campo de longitud del paquete IP
              $pos_leng = array_search('length:',$arr);
              if($pos_leng <> FALSE){

                       //extraigo el tamaño del paquete
                       $bytes_ip1 = substr($arr[$pos_leng+1], 0,
                                           strlen($arr[$pos_leng+1]-1));
```

```
                                //sumo los bytes
                                $total_bytes1 = $total_bytes1 + $bytes_ip1;
                          }

                          // cuento la cantidad de paquetes entrantes
                          $total_pk_salida++;

                          // guardo la marca de tiempo para calcular la proxima diferencia
                        $ts_ant = $time_stamp1;

                        // formato de la linea en el archivo
                          $linea_csv_s = array($time_diff1, $bytes_ip1);

                          //escribo la linea en el archivo
                          //fputcsv( $f_out_llegadas, $linea_csv_a, ",", "\n");
                          // muestro en la pagina los datos
                        //echo "<br>LIN: ".$linea.
                    //         "TIEMPO SALIDA: ".$time_diff1.
                    //         "BYTES IP: ".$bytes_ip1 ;
              }


        }

 $linea++;
}

//muestro en la pagina los totales de entradas
echo "<br>LIN: ".$linea.
       "TIEMPO_TOTAL_llegadas: ". $total_time_entrante.
           "BYTES_TOTAL: ".$total_bytes.
           "PK_ENTRANTES: ". $total_pk_entrante;

  //calculo y muestro los promedios en la pagina, tiempo y tamaño promedio
echo "<br>PROM_TIME_ENTRANTE: ".trim(($total_time_entrante/$total_pk_entrante)).
       "PROM_BYTES_ENTRANTE:". trim($total_bytes/$total_pk_entrante);

 //muestro en la pagina los totales de salidas
echo "<br>LIN: ".$linea.
       "TIEMPO_TOTAL_salidas: ". $total_time_salida.
           "BYTES_TOTAL: ".$total_bytes1.
           "PK_SALIENTES: ".$total_pk_salida;

  //calculo y muestro los promedios en la pagina
echo "<br>PROM_TIME_SALIENTE: ".trim(($total_time_salida/$total_pk_salida)).
       "PROM_BYTES_SAIENTE:". trim($total_bytes1/$total_pk_salida);

fclose($fn);
```

```
fclose($f_out_llegadas);
fclose($f_out_salidas);

// Analiza las dos direcciones IP en formato
// a.b.c.d.port
// Parámetros:
// recibe dos string con direcciones IP
// la $ip1 es la ip origen
// la $ip2 es la ip destino
// Salida :
// 10 - si el paquete es entrante al puerto 25 a la ip 10.10.10.34
// 20 - si el paquete es saliente desde el host 10.10.10.34 y puerto 25
// 30 - en otro caso
function port_25($ip1, $ip2){
         //echo "<br> IP1:". $ip1 . "IP2:".$ip2;
        $salida = 30;
        //pongo en un array las partes del string separadas por puntos
        // para la $ip1 10.10.10.34.25 quedara {10,10,10,34,25}
        $arr_ip1 = explode(".",$ip1);
        // para la $ip2 209.85.222.138.64109: quedará {209,85,222,138,64109:}
        $arr_ip2 = explode(".",$ip2);
        $long = sizeof($arr_ip1);
      if( $long >= 5) {
            $port_ip1 = intval($arr_ip1[4]);
            // quito el ":"del final del string
            $port_ip2 = intval(substr($arr_ip2[4],0,strlen($arr_ip2[4]-1)) );
            //echo "<br> PUERTOS: IP1:". $port_ip1 . "IP2:".$port_ip2;
            // si el puerto origen es el 25, es un paquete saliente
            // del servidor smtp
            if( $port_ip1 == 25 ){
                    $salida = 20;
            }
            // si el puerto de la ip destino es el 25, es un paquete entrante
            //al servidor smtp
            if($port_ip2 == 25) {
                    $salida = 10;
            }
      }
   return($salida);
}

// calcula la diferencia entre dos marcas de tiempo
// retorna la diferencia en microsegundos
function diferencia_tiempo( $t_stamp, $t_ant){
       $micro_act = pasar_microsegundos($t_stamp);
       $micro_ant = pasar_microsegundos($t_ant);
     $diferencia = $micro_act - $micro_ant;
   return($diferencia);
}
```

```
// pasa a microsegundo considerando
// los minutos y los microsegundos
// de la marca de tiempo
function pasar_microsegundos($timestamp){
   $micro = 000000000000;
       $hora_micro = explode('.',$timestamp); // {h:m:s, microsegundos}
       if(sizeof($hora_micro) == 2 ){
               $arr_hora = explode(":",$hora_micro[0]); //{hora, minutos, segundos}
               // calculo s * 1000000 + microsegundos
               $micro = (intval($arr_hora[2]) * 1000000 )+ intval($hora_micro[1]);
               //echo "<br> timestamp:".$timestamp;
               //echo "<br>{h:m:s, microsegundos}:"; print_r($hora_micro);
               //echo "<br> {hora, minutos, segundos}:"; print_r($arr_hora);
               //echo "<br> microseg:".$micro;
       }
       return($micro);
   }
```

16.5. Programa SMTP-P1P

```
<?
/*
 Procesamiento del archivo de volcado tcpdump
 para analisis de trafico en servidor SMTP.
 Datos para el calculo con P1P
    Facultad de informatica - UNLP
 Maestria en Redes de Datos
 Ulises Rapallini
*/
 $archivo = "trafico_DMZ_10.10.10.34_correo.txt";
 $resultado1 = "smtp_datos_p1p.csv";
 $fn = fopen($archivo, "r");
 $f_out_p1p = fopen($resultado1,"w");

 // variables para las llegadas
 $time_stamp = ""; //para la marca de tiempo
 $pk_entrante = 0; //para contar paquetes y bytes por segundo
 $pk_saliente = 0;
 $pk_bytes_entrante = 0;
 $pk_bytes_saliente = 0;
 $total_bytes = 0; // para calcular la velocidad en bps

 $seg_ant = 0; // para controlar los paquetes por segundo
 $seg_act = 0;

 $linea = 0;
 $total_pk_bytes = 0; //para contar los bytes por segundo de los paquetes
 $total_pk_entrante = 0;
```

```
    $total_pk_saliente = 0;
    while( !feof($fn) ) {

    $buffer = fgets($fn, 1024);

           // cargo en un array la linea, separando en campos por espacios
           $arr = explode(' ',$buffer);
           $pos_leng = array_search('length:',$arr);
                       if($pos_leng <> FALSE){
                              //extraigo el tamaño del paquete
                                  $bytes_ip = substr($arr[$pos_leng+1], 0,strlen($arr[$pos_leng+1]-1));
                                  $total_bytes = $total_bytes + $bytes_ip;
                           }


           $time_stamp = $arr[0]; //marca de tiempo
           $seg_act = get_segundos($time_stamp);
           if($seg_act == 0) { $seg_act = $seg_ant; } //cuando la marca de tiempo no esta , dejo
la anterior
           //si el paquete encapsula un segmento TCP
           $pos_proto = array_search('TCP',$arr);
           if( $pos_proto <> FALSE ) {

               $pos_ip = array_search('>',$arr);
               $ipO = $arr[$pos_ip-1]; //ip origen
               $ipD = $arr[$pos_ip+1]; //ip destino

    // consulto si es un paquete entrante o saliente
               $t_paquete = port_25( $ipO, $ipD );

               // si el paquete es entrante
               if( $t_paquete == 10 ){
                           //extraigo el campo de longitud del paquete IP
                       $pos_leng = array_search('length:',$arr);
                       if($pos_leng <> FALSE){
                              //extraigo el tamaño del paquete
                                  $bytes_ip = substr($arr[$pos_leng+1], 0,strlen($arr[$pos_leng+1]-1));
                                  //sumo los bytes
                                  $pk_bytes_entrante = $pk_bytes_entrante + $bytes_ip;
                                  // sumo los bytes totales
                                  $total_pk_bytes = $total_pk_bytes + $bytes_ip;
                           }

                           // cuento la cantidad de paquetes entrantes en el segundo
                           $pk_entrante++;
                           // cuento la cantidad de paquetes totales entrantes
                           $total_pk_entrante++;
           }
```

```
          if( $t_paquete == 20) {
                        //extraigo el campo de longitud del paquete IP
                    $pos_leng = array_search('length:',$arr);
                    if($pos_leng <> FALSE){
                            //extraigo el tamaño del paquete
                                $bytes_ip = substr($arr[$pos_leng+1], 0,strlen($arr[$pos_leng+1]-1));
                                //sumo los bytes en el segundo
                                $pk_bytes_saliente = $pk_bytes_saliente + $bytes_ip;
                                // sumo los bytes totales
                                $total_pk_bytes = $total_pk_bytes + $bytes_ip;
                        }
                        // cuento la cantidad de paquetes salientes en el segundo
                        $pk_saliente++;

                        //sumo la cantidad de paquetes totales
                        $total_pk_saliente++;
                    }
                } //cierre el if , pregunta si es TCP

                if ($seg_act <> $seg_ant){
                        // formato de la linea en el archivo
                        $velocidad_bps = ($total_bytes*8)/1000;
                        $total_bytes_smtp = $total_pk_entrante + $total_pk_saliente;
                        $linea_csv = array($pk_entrante, $pk_saliente, $velocidad_bps, $total_bytes_s
);
                        //escribo la linea en el archivo
                        fputcsv( $f_out_p1p, $linea_csv, ",", "\n");

                        // guardo el segundo actual para controlar el cambio de segundo
                    $seg_ant = $seg_act;
                    //reseteo los contadores
                    $pk_entrante = 0;
                        $pk_saliente = 0;
                        $pk_bytes_entrante = 0;
                        $pk_bytes_saliente = 0;
                        $total_bytes = 0;
            }

    $linea++;
    }

    //muestro en la pagina los totales de entradas
        echo "<br>LIN: ".$linea.
       "PK ENTRANTES TOTAL :". $total_pk_entrante.
               "PK SALIENTE TOTAL :".$total_pk_saliente.
               "PK BYTES TOTAL :". $total_pk_bytes;

    fclose($fn);
    fclose($f_out_p1p);
```

```
// Analiza las dos direcciones IP en formato
// a.b.c.d.port
// Parámetros:
// recibe dos string con direcciones IP
// la $ip1 es la ip origen
// la $ip2 es la ip destino
// Salida :
// 10 - si el paquete es entrante al puerto 25 a la ip 10.10.10.34
   // 20 - si el paquete es saliente desde el host 10.10.10.34 y puerto 25
   // 30 - en otro caso
function port_25($ip1, $ip2){
//echo "<br> IP1:". $ip1 . "IP2:".$ip2;
$salida = 30;
//pongo en un array las partes del string separadas por puntos
// para la $ip1 10.10.10.34.25 quedara {10,10,10,34,25}
$arr_ip1 = explode(".",$ip1);
// para la $ip2 209.85.222.138.64109: quedará {209,85,222,138,64109:}
       $arr_ip2 = explode(".",$ip2);
$long = sizeof($arr_ip1);
       if( $long >= 5) {
$port_ip1 = intval($arr_ip1[4]);
// quito el ":"del final del string
$port_ip2 = intval(substr($arr_ip2[4],0,strlen($arr_ip2[4])-1)) );
//echo "<br> PUERTOS: IP1:". $port_ip1 . "IP2:".$port_ip2;
// si la IP origen es 10.10.10.34.25, es un paquete saliente
               // del servidor smtp, en respuesta a otros servidores
               $pos = strpos($ip1,"10.10.10.34");
               if( $pos !== FALSE){
               //echo "<br> ORIGEN:". $ip1.-". .OK";
            if( $port_ip1 == 25 ){
            $salida = 20;
                   }
       }

       // si la ip destino es 10.10.10.34.25, es un paquete entrante
       //al servidor smtp
       $pos2=strpos($ip2, "10.10.10.34");
       if( $pos2 !== FALSE){
                           //echo "<br> DESTINO:". $ip2.-". .OK";
                           if($port_ip2 == 25) {
               $salida = 10;
               }
   }
   }
   return($salida);
}

// pasa a microsegundo considerando
```

```
// los minutos y los microsegundos
// de la marca de tiempo
function get_segundos($timestamp){
$seg = 000000000000;
       $hora_micro = explode('.',$timestamp); // {h:m:s, microsegundos}
       if(sizeof($hora_micro) == 2 ){
       $arr_hora = explode(":",$hora_micro[0]); //{hora, minutos, segundos}
       // extraigo solo el valor de segundos
              $seg = intval($arr_hora[2]);
       }
   return($seg);
   }

?>
```