Metodologías Cooperativas para la Clasificación del Tráfico en Internet

Federico Rodríguez Teja

Instituto de Computación - Facultad de Ingeniería

Universidad de la República

Supevisor:

Dr. Ing. Eduardo Grampín Castro

Trabajo de

Tesis de Maestría

Diciembre 2009

Deseo dedicar éste trabajo de tesis, a mis hijos Joaquín y Felipe, y compañera de siempre Claudia ...

"Macondo era entonces una aldea de veinte casas de barro y caña brava construidas a la orilla de un río de aguas diáfanas que se precipitaban por un lecho de piedras pulidas, blancas y enormes como huevos prehistóricos. El mundo era tan reciente, que muchas cosas carecían de nombre, y para mencionarlas había que señalarías con el dedo. Todos los años, por el mes de marzo, una familia de gitanos desarrapados plantaba su carpa cerca de la aldea, y con un grande alboroto de pitos y timbales daban a conocer los nuevos inventos. Primero llevaron el imán. Un gitano corpulento, de barba montaraz y manos de gorrión, que se presentó con el nombre de Melquiades, hizo una truculenta demostración pública de lo que él mismo llamaba la octava maravilla de los sabios alquimistas de Macedonia. Fue de casa en casa arrastrando dos lingotes metálicos, y todo el mundo se espantó al ver que los calderos, las pailas, las tenazas y los anafes se caían de su sitio, y las maderas crujían por la desesperación de los clavos y los tornillos tratando de desenclavarse, y aun los objetos perdidos desde hacía mucho tiempo aparecían por donde más se les había buscado, y se arrastraban en desbandada turbulenta detrás de los fierros mágicos de Melquíades."

100 años de soledad, Gabriel García Márquez

Agradecimientos

Deseo agradecer el esfuerzo y dedicación realizado por Eduardo, el grupo de investigación MINA (Network Management - Artificial Intelligence) y el Instituto de Computación de la Facultad de Ingeniería (Universidad de la República), que permitieron realizar la presente investigación.

Abstract

Existen diferentes metodologías para la clasificación del tráfico en Internet, que varían en los niveles de certeza obtenidos, en la clase de tráfico que detectan, así como en su complejidad computacional.

En éste documento se presentan dos novedosas técnicas para la combinación de resultados obtenidos por diferentes mecanismos de clasificación de tráfico, una basada en la asignación de peso por heurística denominada *Combinación con Peso por Heurística* y otra aplicando técnicas Naïve Bayes *Clasificación Bayesiana*.

Las propuestas para la combinación de resultados se realizan en forma genérica, independientemente de las metodologías de clasificación de tráfico aplicadas en las pruebas. Ésto permite su aplicación incluyendo otras heurísticas de clasificación.

Para probar las metodologías de combinación propuestas se desarrolló *trafChar*, un preprocesador del software *open source* Snort. Las metodologías de clasificación de tráfico utilizadas en las pruebas son, *Well Known Ports*, *Análisis de Patrones* y utilizando *Support Vector Machines* técnica de Machine Learning. Además se implementó el entrenamiento de la metodología basada en Machine Learning a partir de información obtenida de la clasificación del tráfico en la red.

El estudio incluye información referente al tiempo de clasificación requerido por las metodologías *Well Known Ports*, *Análisis de Patrones* y *Técnicas de Support Vector Machines*. El tiempo se mide en referencia a la recepción del paquete que permitió clasificarlo, contabilizado desde la llegada del primer paquete del flujo. Se concluye que *Well Known Ports* y *Análisis de Patrones* clasifican en menos de *100 ms*

mientras que la *Técnica de Support Vector Machines* lo hace en más de *100 ms*. Esta información es relevante para el análisis de la posibilidad de clasificación de tráfico *on Line*.

Las pruebas realizadas muestran que ambos mecanismos de combinación de resultados *Combinación con Peso por Heurística* y *Clasificación Bayesiana*, superan en más de 20 % la cantidad de flujos clasificados por las metodologías aisladas con niveles de certeza similares. Las mejoras se fundamentan en que las diferentes metodologías de clasificación obtienen resultados dependiendo del tipo de tráfico a clasificar, y las técnicas de combinación unifican resultados generando una solución global.

Palabras Claves: Network Traffic Classification, Naïves Bayes, Machine Learning, Support Vector Machines, Quality of Service.

Índice general

Índice de figuras

Índice de cuadros

Índice de algoritmos

Acrónimos

ADSL	*Asymmetric Digital Subscriber Line*
AI	*Artificial Intelligence*
CART	*Classification and Regression Trees*
CDF	*Cumulative Distribution Function*
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
FTP	*File Transfer Protocol*
IANA	*Internet Assigned Numbers Authority*
IDS	*Intrusion Detection System*
IP	*Internet Protocol*
ISP	*Internet Service Provider*
ML	*Machine Learning*
OSI	*Open System Interconnection*
P2P	*Peer-to-Peer*
QoS	*Quality of Service*
SVM	*Support Vector Machines*
TCAM	*Ternary Content Addressable Memory*

TCP	*Transmission Control Protocol*
TDG	*Traffic Dispersion Graphs*
UDP	*User Datagram Protocol*

Capítulo 1

Introducción

La clasificación del tráfico en Internet en sus comienzos utilizó técnicas basadas en puertos conocidos o a través del análisis de los datos (*payload*) de los paquetes en la red. Conjuntamente los operadores de red buscado controlar el uso de enlaces proponen catalogar el tráfico para favorecer aplicaciones y desalentar el uso de otras en particular las aplicaciones tipo *Peer-to-Peer* (P2P). Como resultado los programadores de aplicaciones comenzaron a buscar formas para evitar su reconocimiento, utilizando puertos dinámicos y/o el encriptado del tráfico.

A partir de las modificaciones de las aplicaciones, se plantean nuevos mecanismos para clasificar el tráfico basados en elementos de comportamiento o estadísticos que obtienen resultados sin considerar puertos o análisis de los datos.

Existen variadas metodologías que obtienen diferente tasa de éxito en la clasificación y niveles de precisión de resultados. Además el éxito obtenido no es similar para las diferentes clases de tráfico consideradas, teniendo cada metodología una tasa de éxito por tipo de tráfico.

La clasificación del tráfico en función de su aplicación generadora permite a los operadores de red, en primer término contar con información necesaria para planificar las necesidades de la red. En caso de disponer la clasificación *on Line*, posibilita la aplicación de políticas de *Quality of Service* (QoS) y/o de seguridad en la red. Estos elementos permiten a los *Internet Service Provider* (ISP) la mejora de los servicios prestados a sus usuarios y una mejor planificación de los recursos requeridos.

Éste trabajo propone y compara dos procedimientos para la combinación de los resultados obtenidos por diferentes metodologías de clasificación de tráfico, una muy simple asignando pesos a las diferentes heurísticas (*Combinación con Peso por Heurística*) y otra aplicando clasificadores bayesianos (*Clasificación Bayesiana*).

Las propuestas generalizan la combinación de cualquier tipo de metodologías de clasificación, únicamente configurando parámetros en las mismas. Las pruebas se realizaron utilizando tres metodologías conocidas *Well Known Ports*, *Análisis de Patrones* y aplicando *Support Vector Machines (SVM)* una técnica de *Machine Learning* (ML).

Conjuntamente propone un trabajo coordinado entre las diferentes metodologías, buscando la contribución de las mismas para potenciar las diferentes heurísticas de clasificación. En particular se implementa un mecanismo para el entrenamiento de la técnica SVM de ML, en base a la clasificación obtenida de las restantes metodologías aplicadas.

La aplicación de cualquiera de las metodologías de combinación propuestas, *Combinación con Peso por Heurística* y *Clasificación Bayesiana*, aumenta la cantidad y variedad de tráfico clasificado, mejorando la certeza de las metodologías aplicadas en forma individual.

1.1. Organización del documento.

El capítulo 2 presenta el estado del arte en referencia a la clasificación del tráfico en Internet. En el capítulo 3 se exponen los mecanismos para la combinación de metodologías de clasificación propuestos *Combinación con Peso por Heurística* y *Clasificación Bayesiana* y el capítulo 4 presenta la herramienta desarrollada *trafChar* que los implementa. En el capítulo 5 se indican los resultados obtenidos y en el capítulo 6 se exponen las conclusiones y los trabajos futuros propuestos.

Capítulo 2

Metodologías para la Clasificación del Tráfico en Internet

2.1. Introducción

En éste capítulo presentaremos el estado del arte en relación con la clasificación del tráfico en Internet.

Los elementos a clasificar son flujos de datos transmitidos en la red. Un flujo de datos, es un grupo de datagramas, definidos en cierto tiempo y espacio, que tienen la misma identificación. La identificación se realiza a través de la tupla *{protocolo, dir_IP_origen, puerto_origen, dir_IP_destino, puerto_destino}*, donde protocolo es *Transmission Control Protocol* (TCP) o *User Datagram Protocol* (UDP), *dir_IP* identifica los equipos origen y destino, y *puerto* individualiza la aplicación. El flujo contiene datagramas en ambos sentidos, *origen* $\rightarrow$ *destino* y *destino* $\rightarrow$ *origen*, por lo que las tuplas *{prot, dirIP_A, prt_A, dirIP_B, prt_B}* y *{prot, dirIP_B, prt_B, dirIP_A, prt_A}* corresponden al mismo flujo. Ésta definición de flujo contiene el conjunto total de datagramas intercambiado por dos aplicaciones ejecutando en dos nodos de la red.

Las *características* de los flujos se obtienen a través de la visualización/captura de paquetes en la red. Ejemplos de *características* son:

- *Dirección de red* - Dirección de capa red, capa 3 modelo *Open System Interconnection* (OSI), correspondiente al protocolo *Internet Protocol* (IP).

- *Puertos utilizados* - Direcciones de capa 4 modelo OSI (capa transporte), correspondiente a los protocolos TCP o UDP.
- *Tamaño de paquetes* - Indica el tamaño de los paquetes intercambiados en un flujo. Se consideran sobre datos estadísticos como el máximo, mínimo, medias o varianzas.
- *Frecuencia de aparición de paquetes* - Se mide a través del cálculo de tiempo entre datagramas del flujo.
- *Contenido del paquete* - Refiere al análisis de datos transportados por el paquete.

En [MCM+05] se presenta un listado con *características* para la clasificación de flujos.

Clasificación de Tráfico se define como los procedimientos realizados en base a *características* observadas en los flujos en forma pasiva para asignarle una clase de tráfico. La clase de tráfico asignada depende del objetivo de la clasificación.

Como ejemplo un de objetivo de la clasificación del tráfico es la detección de intrusos *Intrusion Detection System* (IDS). Intrusos se definen como intentos no autorizados para acceder, manipular, modificar o destruir información con el objetivo de lograr inutilizar un sistema o transformarlo en poco fiable [GS99]. La metodología utilizada se basa en clasificar el tráfico devolviendo como resultado si corresponde a tráfico aceptado o un ataque. Éstas clasificaciones requieren de un alto grado de especificidad, dado que los ataques siempre buscan parecerse a flujos válidos.

Cada metodología de clasificación de tráfico exige la visualización de información particular de cada flujo. Debe contarse entonces con equipamiento que permita registrar los valores de las *características* que el método necesita y adicionalmente se requieren bases de información con el fin de comparar los valores del flujo en estudio con valores ya conocidos, determinando su tipo de tráfico.

Existen tres tipos de metodologías de clasificación, según los esquemas generales de clasificación utilizados [SAM09]:

- Metodologías Basadas en Búsqueda de Patrones Exactos.

- Metodologías Basadas en Elementos de Comportamiento.
- Metodologías Basados en Elementos Estadísticos.

A continuación presentaremos los tipos de metodologías incluyendo ejemplos concretos de cada una. En [CAI09] pueden encontrarse referencias a publicaciones y artículos sobre metodologías de clasificación de tráfico.

2.2. Metodologías Basadas en Búsqueda de Patrones Exactos

Las metodologías basadas en búsqueda de patrones exactos intentan encontrar elementos particulares y únicos (*patrones*), en los paquetes intercambiados por las aplicaciones. Si se encuentra alguno de éstos *patrones*, se podrá clasificar el flujo como perteneciente a la clase de tráfico que lo contiene.

La unicidad de los *patrones* buscados es un elemento importante. En caso de no ser único no es posible definir una clase, sino que podremos decir que pertenece a alguna de las clases que contiene ese *patrón*.

Las diferentes metodologías de clasificación de tráfico basadas en búsqueda de patrones exactos, se diferencian en el tipo de *patrones* buscados. Estos *patrones* pueden encontrarse en cualquiera de las *características* analizadas para los flujos.

Ejemplos concretos de las presentes metodologías son:

- Metodología Well Known Ports.
- Metodología de Reconocimiento por Patrones de *bit strings*

2.2.1. Metodología Well Known Ports

Well Known Ports es la primera metodología presentada para la clasificación del tráfico en Internet.

Las *características* utilizadas son los puertos de la conexión de capa transporte, considerados tanto para el caso del puerto origen como destino. Buscando un patrón exacto en los mismos se plantea identificar la aplicación generadora del tráfico.

Esta identificación se basa en la asignación de puertos que realiza *Internet Assigned Numbers Authority* (IANA) [IAN09], organización responsable en Internet del gerenciamiento de puertos TCP y UDP para las distintas aplicaciones. El objetivo buscado por IANA es determinar el puerto donde atiende cada aplicación, generando un directorio de puertos conocidos que permita acceder directamente a las aplicaciones que se encuentran activas en un nodo de la red.

2.2.1.1. Algoritmo utilizado por Well Known Ports

El algoritmo aplicado por *Well Known Ports* es presentado en 2.1. El mismo recibe como parámetro el cabezal de un paquete del flujo y devuelve la clase de tráfico de la aplicación que genera el tráfico o *noClasificado* en caso de no poder asignar una clase de tráfico.

Algoritmo 2.1 Algoritmo *Well Known Ports*

Require: pkt
Ensure: $clase\_trafico$, $noClasificado$
  if $pkt.protocol \in \{TCP, UDP\}$ **then**
    if $\{pkt.protocol, pkt.srcPort\} \in WellKnownPorts$ **then**
      return $WellKnownPorts(pkt.protocol, pkt.srcPort)$
    end if
    if $\{pkt.protocol, pkt.dstPort\} \in WellKnownPorts$ **then**
      return $WellKnownPorts(pkt.protocol, dst.srcPort)$
    end if
  end if
  return $noClasificado$

La metodología requiere de una tabla de puertos conocidos (*WellKnownPorts* en el código), conteniendo los siguientes campos:

- *Servicio* - contiene el tipo de aplicación (texto descripción de la aplicación).
- *Protocolo* - contiene el protocolo de capa transporte utilizado por la aplicación, TCP o UDP.
- *Puerto* - contiene el puerto utilizado por la aplicación (entero entre 1 - 65536).

Ésta información en equipos con Sistema Operativo Unix se encuentra en el archivo *services* y presenta el siguiente formato:

```
http      80/tcp    www www-http  # WorldWideWeb HTTP
http      80/udp    www www-http  # HyperText Transfer Protocol
```

El algoritmo busca la existencia de un puerto conocido en el puerto origen y puerto destino.

Los puertos conocidos se utilizan en aplicaciones del tipo cliente/servidor, arquitectura donde un cliente realiza peticiones al servidor obteniendo respuestas del mismo. El puerto conocido es el correspondiente al extremo del servidor, lugar donde se encuentra disponible el servicio. El paquete inspeccionado en el algoritmo es un paquete genérico, por lo tanto no se puede determinar a priori si el mismo se dirige al cliente o al servidor, motivo por el cual se inspeccionan los dos puertos. El funcionamiento correcto del algoritmo presentado exige que el extremo del cliente no utilice puertos conocidos.

Un mecanismo para determinar el puerto del servidor en el caso de servicios orientados a conexión, sobre el protocolo TCP, se fundamenta en el hecho que el nodo cliente es quien inicia la conexión. Dado el mecanismo de conexión de TCP *three-way handshake* [Pos81], una conexión iniciada por el cliente con un paquete con el bit SYN seteado, es contestando por el servidor con los bits SYN y ACK seteados, el resto de los paquetes del flujo no contienen el bit SYN seteado. Por el mecanismo de conexión indicado anteriormente puede entonces inferirse que dado un paquete conteniendo el bit SYN seteado (sin ACK) si el puerto destino es conocido, o considerando un paquete con los bits SYN y ACK seteados si el puerto origen es conocido, se clasificará el flujo.

En el caso de UDP las precisiones presentadas en el párrafo anterior no son aplicables, dado que el protocolo no es orientado a conexión. Para el caso de flujos UDP se considera como el cliente el nodo origen que envió el primer paquete del flujo.

Se entiende que éste agregado presentado en los párrafos anteriores, exige la visualización de los bits de control de TCP por el mecanismo de clasificación lo que dificulta su aplicación. Además la aplicación de la metodología debe realizarse

exclusivamente en el primer o segundo paquete del flujo, donde se realiza el *three-way handshake*. Considerando que no es aceptado que clientes utilicen puertos conocidos se puede concluir que aplicando el algoritmo 2.1 se obtienen mejoras sustanciales en la cantidad de flujos clasificados, sin generar resultados no certeros.

2.2.2. Metodología de Reconocimiento por Patrones de *bit strings*

La metodología de reconocimiento por patrones de *bit strings* realiza la búsqueda en los datos (*payload*) de los paquetes de secuencias de bits/bytes conocidos para las distintas aplicaciones, elementos que las identifican. Los patrones buscados se les denomina *signatures*.

Signatures son patrones únicos que se encuentran asociados a una aplicación. El estudio de las aplicaciones, permiten reconocen los patrones que presentan, generando con éstos una base de información. La entidad clasificadora compara el tráfico contra esta base y a partir de la búsqueda de coincidencias se identifican las aplicaciones. La base de información que contiene los patrones a buscar requiere entonces actualización continua para incluir elementos que identifiquen las nuevas aplicaciones [Cis09].

Dependiendo de la aplicación éstos patrones pueden ubicarse en una ubicación específica (*offset*), o variar su posicionamiento en el paquete. Conocer la ubicación del los patrones existentes en los paquetes facilita la acción de la máquina analizadora y se disminuye la posibilidad de cometer errores. No todas las aplicaciones presentan patrones especiales que permiten identificarlas.

Las *signatures* incluyen además información sobre el protocolo IP y puertos de TCP o UDP. Previo a la búsqueda de coincidencia de patrones se verifica que cumplan las condiciones especificadas sobre el protocolo IP y puertos de capa transporte.

El análisis de *signatures* puede aplicarse *con estado* o *sin estado*.

- En el caso *con estado* el patrón buscado se presentará en una sucesión de elementos que clasifican el flujo. Para reconocer una aplicación se requiere la búsqueda de una sucesión de patrones que debe haber verificado el pasaje

por los estados anteriores. Al reconocer el último patrón se clasificará el flujo.

- Para el caso *sin estado* implica que se busca un patrón independientemente del cumplimiento de otros elementos, el único elemento verificado es la información sobre protocolo IP y puertos de los paquetes inspeccionados.

Ésta metodología consigue una clasificación muy fina. Los resultados permiten determinar la aplicación particular que generó el tráfico.

2.2.2.1. Algoritmos de Reconocimiento de Patrones de *bit strings*

Los mecanismos de reconocimiento de patrones para obtener resultados en tiempo real (*on Line*) exigen la inspección de los datos de los paquetes a la velocidad de los enlaces. El aumento de las velocidades de los enlaces, así como el aumento de la cantidad de patrones a analizar requiere la aplicación de heurísticas que mejoren su desempeño [YKL04].

Los problemas de búsqueda de patrones han sido muy estudiados por la academia y existen diferentes soluciones implementadas en software y hardware.

El algoritmo *Knuth-Morris-Pratt* resuelve la búsqueda de un único patrón. Dado un patrón de largo m la búsqueda en un paquete de n bytes requiere un tiempo de computo de orden $O(m+n)$ [YKL04]. La búsqueda de k patrones por lo tanto será de orden $O(k(m+n))$.

Los problemas planteados para la clasificación del tráfico en Internet, exigen el reconocimiento de un conjunto de patrones. Ésta exigencia plantea la búsqueda de algoritmos que permitan el reconocimiento de múltiples patrones.

El algoritmo *Aho-Corasick* permite reconocer múltiples patrones [AC75]. Para resolverlo implementa una máquina de estados finita, con tres funciones $g(Q,a)$, $f(Q)$ y $out(Q)$ construídas sobre la misma.

Sea como ejemplo el conjunto de patrones $P = \{he, she, his, hers\}$. El algoritmo previo a la búsqueda genera la máquina de estados finita presentado en la figura 2.1.

El algoritmo de búsqueda es presentado en 2.2, que utiliza las funciones $g(Q,a)$ donde Q es un nodo (identificados entre 0 a 9 en la figura) y a es el string de

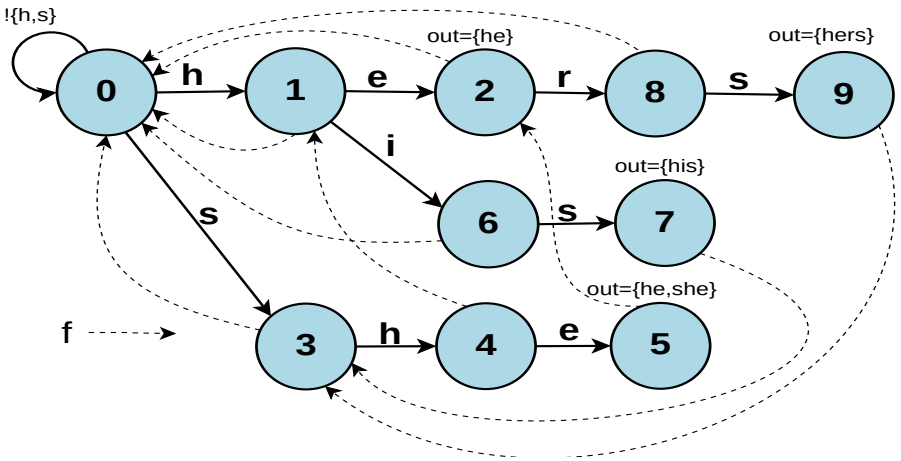


Figura 2.1: Máquina de Estados Finita para búsqueda *Aho-Corasick*

entrada a procesar:

$$g(Q,a) = \begin{cases} V & \text{si existe arista } (Q,V) \text{ rotulada con a} \\ 0 & \text{en caso contrario} \end{cases}$$

La función $f(Q)$ con Q un nodo del árbol, devuelve el nodo correspondiente próximo más largo a partir del camino recorrido (representado con línea punteada en la figura). La función $out(Q)$ con Q un nodo de la máquina de estados devuelve el conjunto de patrones reconocidos por el nodo (representados entre corchetes sobre el nodo en la figura), los nodos donde no se expresa el valor de $out(Q)$ devuelve el conjunto vacío.

El procedimiento para generar la máquina de estados finita y funciones requeridas por el algoritmo se encuentra en [AC75].

El algoritmo *Aho-Corasick* dado un paquete de n bytes presenta un orden de tiempo de búsqueda de todos los patrones configurados de $O(n)$ [YKL04].

Por último existen mecanismos implementados en hardware para el reconocimiento de patrones. *Ternary Content Addressable Memory* (TCAM) es un ejemplo ampliamente utilizado para el procesamiento de los cabezales IP. TCAM es un tipo de memoria que realiza búsquedas en forma paralela a alta velocidad permitiendo el reconocimiento de múltiples patrones simultaneamente [YKL04].

TCAM además permite identificar aplicando expresiones de patrones con comodines (*wildcards*), negaciones o concatenación de los mismos.

Algoritmo 2.2 Algoritmo *Aho-Corascick*

Require: $y \triangleright$ *y cadena a analizar*
Ensure: $R \triangleright$ *conjunto de patrones contenidos en y*
$\triangleright$ *Requiere* $g(Q,a)$, $f(Q)$ *y* $out(Q)$ *generadas a partir de los patrones*
$Q \leftarrow 0$; $\triangleright$ *estado inicial*
$R \leftarrow \{\}$
for $I = 1$ to $largo(y)$ **do**
  while $g(Q, y[I]) = 0$ **do**
    $Q \leftarrow f(Q)$
  end while
  $Q \leftarrow g(Q, y[I])$
  if $out(Q) \neq \{\}$ **then** $\triangleright$ *agrego solución encontrada*
    $R \leftarrow R \cup out(Q)$
  end if
end for
return R

2.2.3. Ventajas y Desventajas de Metodologías Basadas en Patrones Exactos

La clasificación se realiza analizando cada paquete y comparando su información contra directorios de patrones exactos. Esto hace que su aplicación sea simple.

Como contrapartida para su aplicación se requiere de directorios o estudios previos que determinen los patrones exactos que identifican las aplicaciones.

2.2.3.1. Caso Well Known Ports

La clasificación del tráfico basada en puertos conocidos se ha tornado ineficiente. El motivo principal es la aparición de aplicaciones que utilizan puertos seleccionados dinámicamente, sustituyendo a los puertos conocidos. Ésto se debe a la aplicación de filtros por parte de los ISP a puertos conocidos buscando limitar ciertas aplicaciones, por ejemplo las P2P.

El mecanismo utilizado para trabajar con puertos dinámicos se implementa a través de notificar a un nodo *superpeer* el par *{IP,puerto}* donde atiende el

servicio P2P [KBFC04]. El nodo *superpeer* actúa como un servidor para un subconjunto de clientes, mientras los nodos *superpeer* intercambian información entre ellos para brindarla a sus clientes. La información final entregada a los clientes son los pares *{IP,puerto}* donde descargar lo buscado.

Otra dificultad es la existencia de aplicaciones que atienden en puertos utilizadas para otras aplicaciones conocidas. Un ejemplo es el acceso a la aplicación *MSN Messenger*, la que puede realizarse a través de una aplicación nativa que utiliza el puerto TCP conocido 1863 y a través de una página Web (puerto TCP 80) en las redes donde el puerto conocido del MSN se encuentra filtrado o la aplicación *MSN Messenger* no se encuentra disponible.

El bloqueo de puertos puede evitarse fácilmente dado que solamente se requiere de un programa redirector de puertos, que ante la conexión a un puerto en un equipo redirija la misma a otro puerto en otro equipo. Como consecuencia la clasificación realizada por *Well Known Ports* no sería correcta.

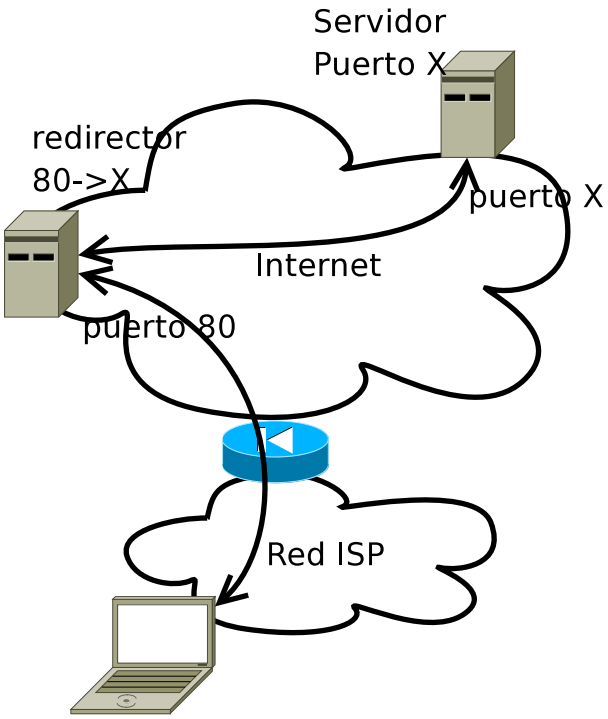


Figura 2.2: Ejemplo de Redirección de Puertos

La figura 2.2 muestra un ejemplo donde se desea acceder al servicio que atiende en el puerto X, y el mismo se encuentra filtrado por la red del ISP. Es razonable que el tráfico *Web* sea admitido sin problemas. En el ejemplo el flujo de la aplicación del *servicio X*, pasa por el router del ISP como tráfico *Web*. Se conecta al equipo que realiza la redirección del puerto 80 de entrada al puerto X. Como puede verse, para la transacción se requiere un nodo que realice las modificaciones de puertos, fuera de la red del ISP.

La utilización de la presente metodología, se encuentra en casos de existencia de *FireWalls*, que permiten el tránsito de tráfico únicamente para puertos predefinidos. Con trucos similares a los presentados, un usuario del interior de la red, logra acceder a servicios no admitidos por el *FireWall*.

La ventaja que presenta la metodología *Well Known Ports* es lo simple de su implementación. Ésta solamente requiere el análisis de un paquete del flujo para poder actuar, característica que no se cumple en la mayoría de las metodologías. Éste hecho permite que en cuanto aparece el primer paquete del flujo ya se obtengan los resultados de la clasificación.

A pesar de las desventajas presentadas en los párrafos anteriores, dada la simpleza y economía de recursos requerida para su implementación y que en aplicaciones *Web*, *Mail* y *Services* presenta buenos resultados, se recomienda su uso. En [KCF+] se concluye que la metodología de puertos conocidos para las aplicaciones más antiguas obtiene muy buenos resultados.

2.2.3.2. Caso de Reconocimiento por Patrones de *bit strings*

La principal ventaja de la metodologías de reconocimiento por patrones de *bit strings* es el nivel de certeza que obtiene.

La aplicación es computacionalmente simple, exige la inspección de una sucesión de paquetes hasta encontrar alguna coincidencia con las *signatures* registradas.

El tiempo requerido para la clasificación está determinado por la ubicación en el flujo del paquete conteniendo la *signature*. Cuanto más al inicio del flujo se encuentre el patrón, menor será el tiempo de clasificación.

La principal desventaja es la imposibilidad de clasificar en aplicaciones que encripten su tráfico, dado que no se obtendrán resultados en la comparación de los patrones.

Otra desventaja es que el análisis de los datos de los paquetes presenta fuertes exigencias de CPU, por lo que los equipos clasificadores presentan requerimientos importantes de hardware para soportar la aplicación.

2.3. Metodologías Basadas en Elementos de Comportamiento

Las metodologías basadas en elementos de comportamiento, refieren a la utilización de heurísticas que permitan a través de la visualización de flujos inferir la clase de aplicación que ejecuta un nodo de la red, en un puerto TCP o UDP.

La idea es analizar el comportamiento "*social*" de los diferentes nodos, infiriendo del mismo elementos que permitan la clasificación.

Presentaremos metodologías que registrando las direcciones IP y puertos TCP o UDP en el tráfico analizado, logran clasificar los distintos flujos [KBFC04; IKF+09; IPF+07].

El uso más extendido de estas metodologías es para identificar flujos de aplicaciones P2P. En base a éstas y otras particularidades se plantean algoritmos los cuales presentan una tasa de aciertos aceptable.

Ejemplos concretos de las presentes metodologías son:

- Análisis en Base a la Existencia de Flujos UDP y TCP.
- Metodología de IP Pairs.
- Clasificación de Tráfico Basada en Grafos.

2.3.1. Análisis en Base a la Existencia de Flujos UDP y TCP

Ésta heurística permite clasificar flujos de tráfico P2P. Es presentada por *Karagiannis, T. et al.* en [KBFC04]. Se basa en la característica de la sesión de

aplicaciones P2P que utilizan protocolos orientados a conexión (TCP) y servicios de datagramas (UDP).

Considerando la dirección IP origen y destino de un flujo, se examina si entre éstos nodos existen flujos de protocolo TCP y UDP. Dada ésta situación posiblemente sea un flujo de una aplicación P2P.

Existen protocolos conocidos que no son de aplicaciones P2P y realizan intercambio de datos con TCP y UDP. Estos son excepciones a ésta heurística, las que se presentan en la tabla 2.1. Esta información fue tomada de [KBFC04], pero la aparición constante de nuevas aplicaciones exige su verificación, dado que estos casos generan errores en la clasificación.

Cuadro 2.1: *Aplicaciones conocidas que utilizan protocolos TCP y UDP.*

Puerto	Aplicación
135,137,139,445	NetBios
53	DNS
123	NTP
500	ISAKMP
554,7070,1755,6970,5000,5001	streaming
7000,7514,6667	IRC
6112,6868,6899	gaming
3531	p2pnetworking.exe

La heurística requiere de un registro de los flujos analizados para realizar la consulta. Esto exige que la heurística implemente la adquisición de información con éste fin.

2.3.1.1. Algoritmo utilizado por Análisis en Base a la Existencia de Flujos UDP y TCP

El estudio se realiza sobre $\mathcal{F}$ un conjunto de flujos registrados F identificados por la 5 tupla $F = \{prot, IP_{orig}, port_{orig}, IP_{dest}, port_{dest}\}$. Cada registro contiene *prot* protocolo TCP o UDP y dos atributos *{IP,port}* que identifican sus extremos.

Algoritmo 2.3 Algoritmo *Análisis en Base a Existencia de Flujos UDP y TCP*

Require: pkt

Ensure: $P2P$, NO_P2P, $noClasificado$

▷ Formato $Tabla\_Flujos$ $\{prot, IP_{orig}, port_{orig}, IP_{dest}, port_{dest}\}$

▷ Formato $Tabla\_PuertosExcepcion$ $\{prot, port\}$

$DF \leftarrow \{pkt.prot, pkt.IP_{orig}, pkt.port_{orig}, pkt.IP_{dest}, pkt.port_{dest}\}$

$insert(Tabla\_Flujos, DF)$ ▷ Inserto datos Flujo

$EXC \leftarrow Tabla\_PuertosExcepcion$

if $(pkt.port_{orig} \in EXC) \vee (pkt.port_{dest} \in EXC)$ **then** ▷ Puerto excepción

  return NO_P2P

end if

$FT \leftarrow Tabla\_Flujos$

$REG\_FLS \leftarrow FT$ *where* $(IP_{orig} = pkt.IP_{orig}, IP_{dest} = pkt.IP_{dest})$

$REG\_FLD \leftarrow FT$ *where* $(IP_{orig} = pkt.IP_{dest}, IP_{dest} = pkt.IP_{orig})$

$REG\_FLOW \leftarrow REG\_FLS \cup REG\_FLD$

if $(TCP \in REG\_FLOW.prot) \wedge (UDP \in REG\_FLOW.prot)$ **then**

  ▷ Es P2P, existen flujos TCP y UDP

  return $P2P$

end if

return $noClasificado$

El pseudocódigo del algoritmo se presenta en 2.3.

Los argumentos requeridos por el algoritmo son los cabezales de un paquete perteneciente al flujo. Del mismo se toman las direcciones IP origen y destino.

La implementación requiere registros de los flujos visualizados en la red (*Tabla_ Flujos*) que contiene el conjunto $\mathcal{F}$ presentado anteriormente. La tabla contiene las 5 tuplas que identifican cada flujo. Además se utiliza una base que presenta las excepciones a la regla (*Tabla_ PuertosExcepcion*), con los valores presentados en la tabla 2.1.

El algoritmo devuelve *NOT_ P2P*, *P2P* o *noClasificado*, según el resultado obtenido.

2.3.2. Metodología IP Pairs

Ésta heurística permite clasificar flujos de tráfico P2P. Es presentada por *Karagiannis, T. et al.* en [KBFC04]. La *característica* utilizada por la metodología *IP Pairs* para clasificar son pares {*IP,puerto*} compuestos de *IP* dirección de capa red y *puerto* identificador de capa transporte. Éstos elementos se denominan *IP Pair*. Como ya se indicó un flujo se identifica por el protocolo y dos *IP pair*. Un *IP pair* se asocia a una aplicación ejecutando en un nodo de la red. La heurística propuesta clasifica éstos elementos y obtiene resultados en la clasificación de flujos P2P.

Las aplicaciones P2P intercambian información utilizando sólo un flujo. Este comportamiento los diferencia de servicios *Web*, donde la sesión involucra varios flujos.

La figura 2.3, muestra la diferencia entre la comunicación con un servidor *Web* y uno sirviendo contenido compartido P2P, las flechas de la figura representan flujos identificados a través de dos *IP pair* origen y destino.

El acceso a una página web, genera diferentes flujos para los elementos que la componen (fotografías, texto y otros componentes). Éstos flujos presentan el mismo *IP pair {IP_ destino, puerto_ destino}* (*puerto_ destino* es 80 para el caso del ejemplo) y difieren en la componente *puerto_ origen* en los *IP pair* del origen.

En el caso de los servicios P2P, se genera un solo flujo de datos. Esta singularidad se encuentra fundamentada en la metodología de publicación de la información en servicios P2P donde existen nodos *superpeer*, que publican en la red un par *{IP,puerto}* donde obtener el contenido buscado.

En la figura 2.3 el *IP pair {IP_ P2P02,C}* que atiende un servicio P2P presenta dos flujos con 2 IP diferentes (IP_cli02,IP_cli03) y dos puertos (D,F), y en el caso del *IP pair {IP_ Web,80}* de un servidor Web, muestra tres flujos con una direcciones IP (IP_cli01) y tres puertos (X,Y,Z). Se desprende entonces que dado un *IP pair* la relación entre direcciones IP y puertos presentes en las conexiones al mismo es mayor para servicios Web que en servicios P2P.

Una vez clasificado un *IP pair {IP,puerto}*, de encontrarse el mismo en un nuevo flujo, como éste identifica la aplicación generadora, se utiliza la clasificación previamente realizada.

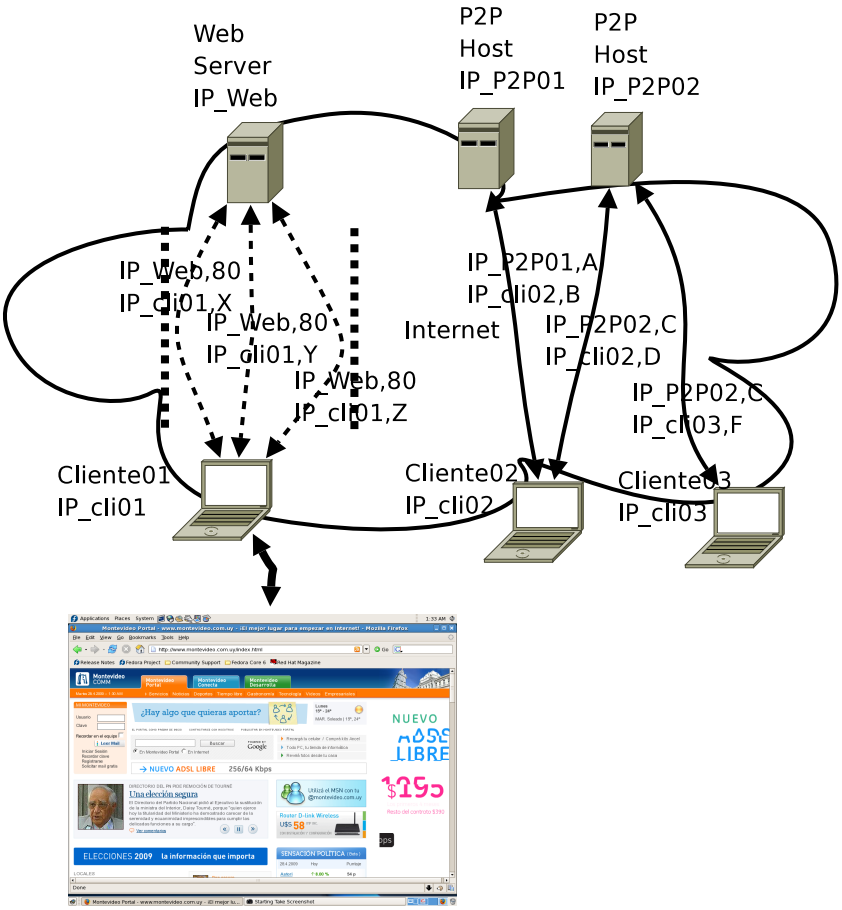


Figura 2.3: Clasificación a través de pares *{IP,puerto}*

2.3.2.1. Algoritmo utilizado por IP Pair

Para la clasificación se utiliza un conjunto $\mathcal{F}$ ya presentado en sección 2.3.1.1. El conjunto $\mathcal{F}$ contiene elementos $F = \{prot, IP_{orig}, port_{orig}, IP_{dest}, port_{dest}\}$ donde F contiene protocolo TCP o UDP y dos *IP pair {IP,puerto}* que identifican sus extremos.

Para el *IP pair {IP,puerto}* se calcula R relación entre el número de direcciones IP diferentes y puertos conectados a éste en cierto período de tiempo (contenidos en $\mathcal{F}$).

$$Port_{dest} = \{f.port_{dest} / f \in \mathcal{F}, f.IP_{orig} = IP, f.port_{orig} = puerto\}$$

$$Port_{orig} = \{f.port_{orig} / f \in \mathcal{F}, f.IP_{dest} = IP, f.port_{dest} = puerto\}$$

$$N_{port} = |\{Port_{orig} \cup Port_{dest}\}|$$

$$IP_{dest} = \{f.IP_{dest} / f \in \mathcal{F}, f.IP_{orig} = IP, f.port_{orig} = puerto\}$$

$$IP_{orig} = \{f.IP_{orig} / f \in \mathcal{F}, f.IP_{dest} = IP, f.port_{dest} = puerto\}$$

$$N_{IP} = |\{IP_{orig} \cup IP_{dest}\}|$$

Sea R entonces:

$$R = \|N_{port} - N_{IP}\| \tag{2.1}$$

El valor R presenta la relación existente entre direcciones IP y puertos, en los *IP pair* del otro extremo del flujo. Entonces si R es cercano a 0 podemos inferir que se trata de un servicio P2P.

El algoritmo que implementa *IP Pairs* se presenta en 2.4.

El procedimiento recibe como argumento el cabezal de un paquete del flujo de donde toma los *IP pair* del flujo. El valor sugerido en [KBFC04] para MAX_P2P es 2 y para MIN_NO_P2P es 10.

El algoritmo realiza el estudio para los dos *IP pair* del flujo. En caso de obtener resultados los retorna, en caso contrario devuelve *noClasificado*. Para la clasificación exige la visualización de un solo paquete del flujo.

2.3.3. Clasificación de Tráfico Basada en Grafos

Ésta metodología se concentra en la clasificación del tráfico generado por aplicaciones P2P. Es presentada por *Iliofotou, M. et al.* en [IKF+09]. Propone la utilización de *Traffic Dispersion Graphs* (TDG) para la clasificación del tráfico. El objetivo es agrupar las aplicaciones en base a información disponible en TDG's, y a partir de la agrupación realizada clasificarlos. Un TDG permite inferir el comportamiento "social" de los flujos que lo componen. Una vez agrupados y clasificados se plantea el estudio de *signatures* que presentan o distinguir elementos estadísticos de los mismos.

Traffic Dispersion Graphs busca agrupar flujos identificados a través de su 5 tupla $\{prot, IP_{orig}, port_{orig}, IP_{dest}, port_{dest}\}$. Sea $\mathcal{F}$ un conjunto de flujos, el TDG asociado al mismo es el grafo dirigido $G(V, E)$, donde los nodos V son el conjunto

Algoritmo 2.4 Algoritmo *IP Pairs*

Require: pkt

Ensure: $P2P$, NO_P2P, $noClasificado$

▷ Formato $Tabla\_Flujos$ $\{prot, IP_{orig}, port_{orig}, IP_{dest}, port_{dest}\}$

▷ Cargo en $Tabla\_Flujos$ datos del flujo

$DF \leftarrow \{pkt.prot, pkt.IP_{orig}, pkt.port_{orig}, pkt.IP_{dest}, pkt.port_{dest}\}$

$insert(Tabla\_Flujos, DF)$ ▷ Inserto datos Flujo

$FT \leftarrow Tabla\_Flujos$

for $\{IP, port\}$ in $(\{pkt.IP_{orig}, pkt.port_{orig}\}, \{pkt.IP_{dest}, pkt.port_{dest}\})$ **do**

▷ REG_IPPS contiene $IP_{dest}, port_{dest}$

$REG\_IPPS \leftarrow FT\ where\ (IP = IP_{orig}, port = port_{orig})$

▷ REG_IPPD contiene $IP_{orig}, port_{orig}$

$REG\_IPPD \leftarrow FT\ where\ (IP = IP_{dest}, port = port_{dest})$

$REG\_IPP \leftarrow REG\_IPPS \cup REG\_IPPD$

$cantPort \leftarrow |REG\_IPP.port|$ ▷ Cantidad de puertos diferentes

$cantIP \leftarrow |REG\_IPP.IP|$ ▷ Cantidad de IP diferentes

if $abs(cantPort - cantIP) < MAX\_P2P$ **then**

▷ Cantidad de puertos similar a cantidad IP, flujo P2P

return $P2P$

end if

if $abs(cantPort - cantIP) > MIN\_NO\_P2P$ **then**

▷ Cantidad de puertos diferente a cantidad IP

return NO_P2P

end if

end for

return $noClasificado$

de direcciones IP IP_{orig} e IP_{dest} de todos los flujos contenidos en $\mathcal{F}$. El conjunto E de aristas esta compuesto por los elementos $\{(u, v)/f \in \mathcal{F} \wedge f.IP_{orig} = u \wedge IP_{dest} = v\}$. Los flujos contienen paquetes en dos direcciones, pero se considera para flujos TCP el sentido en que se establece la conexión (ver *three-way handshake* sección 2.2.2.1), mientras que para los UDP se consideran el sentido del primer paquete analizado.

Para clasificar los grupos de flujos a partir del TDG generado, se utilizan las

siguientes métricas de grafos:

- *Grado de nodos*: presenta información sobre la topología. se calcula en base a $2|E|/|V|$.
- *Direccionalidad*: busca discriminar entre nodos que inician tráfico, los que reciben y los que realizan ambas tareas. Para su cálculo se divide el conjunto V en tres conjuntos disjuntos V_{src} conjunto que origina tráfico, V_{snk} conjunto que recibe tráfico, y V_{ln0} que realiza ambas, se calcula posteriormente $Avg_{inDeg} = 2|E|/|V_{snk}|$ y $Avg_{outDeg} = 2|E|/|V_{src}|$
- *Componentes conectados*: indica el mayor conjunto de nodos débilmente conectados.
- *Distancia*: la menor distancia entre todo los pares de nodos.

A través del estudio de las métricas se determina la aplicación que lo generó.

2.3.3.1. Algoritmo utilizado por Clasificación de Tráfico Basada en Grafos

El algoritmo 2.5 presenta la metodología para agrupar las diferentes aplicaciones en clusters (grupos).

El resultado del algoritmo es el conjunto C que contiene grupos de flujos con la aplicación generadora ($\{c_i, app\}$). De los clusters generados deben extraerse los elementos que permiten la clasificación del tráfico. La clasificación puede realizarse por propiedades estadísticas de los flujos de cada grupo o la búsqueda de *signatures* que permitan identificarlos.

La propuesta presentada en [IKF+09] se enfoca en la clasificación de tráfico de aplicaciones P2P, pero éste puede utilizarse para otras aplicaciones.

2.3.4. Ventajas y Desventajas de Metodologías Basadas en Elementos de Comportamiento

Las ventajas de las presentes heurísticas es que permiten la clasificación del tráfico sin la visualización de los datos contenidos en los paquetes. Esto permite además la clasificación de tráfico encriptado.

Algoritmo 2.5 Algoritmo *Clasificación P2P basada en Grafos*

Require: $FT \triangleright FT$ conjunto de elementos $\{prot, IP_{orig}, port_{orig}, IP_{dest}, port_{dest}\}$
Ensure: $C \triangleright C$ Conjunto de elementos de $\{c_i, app\}$
$\triangleright$ con c_i conjunto de elementos de FT
$\triangleright$ con app identificador de aplicación
$\triangleright$ elimina de FT flujos con aplicación conocida
$FT_{aux} \leftarrow eliminoFlujoClasificados(FT)$
$\triangleright$ agrupo los elementos de FT_{aux} con características similares
$C_{ini} \leftarrow generoClusters(FT_{aux})$
$\triangleright$ fusiono grupos por similitud en conjunto de IP contenidas
$C_{fin} \leftarrow fusionoClusters(C_{ini})$
$C \leftarrow \{\}$
for $c_i \in C_{fin}$ **do**
  $G \leftarrow generoTDG(c_i)$
  $\triangleright$ En función de métrica de G determino applicación
  $app \leftarrow clasifico(G)$
  $C \leftarrow C \cup \{c_i, app\}$
end for
return C

Las heurísticas no están planteadas en base a ningún protocolo P2P específico, dado que clasifican a través de su comportamiento "social". Ante el surgimiento de nuevos protocolos P2P, si éstos se comoportan socialmente en forma similar, éstos métodos seguirán obteniendo resultados sin necesidad de modificaciones.

Estas heurísticas al igual que *Well Known Ports* clasifican con la visualización de un paquete. La diferencia es que requiere generar $\mathcal{F}$ (ver sección 2.3.1.1) el registro de flujos analizados sobre los que se aplica el algoritmo.

El porcentaje de clasificación de las presente metodologías depende directamente de los flujos registrados. Por lo tanto la heurística inicialmente tendrá poco éxito y cuando el conjunto $\mathcal{F}$ sea representativo de los nodos de la red obtendrá buenos resultados.

2.4. Metodologías Basadas en Elementos Estadísticos

Éstas metodologías determinan comportamientos estadísticos de un conjunto de *características* para las diferentes clases de tráfico. Para clasificar un flujo se calculan los valores de éstas *características* determinando según los resultados a que clase de tráfico pertenece.

Agrupando los diferentes flujos por tipo de tráfico se generan subconjuntos de elementos. Éstos grupos se denominan clústers. Los grupos se generan a través de algoritmos de clusterización. Los algoritmos de clusterización se definen como los procedimientos aplicados a un conjunto de elementos para agrupar los similares en el mismo clúster [EMA+07; HTF01]. La partición se realiza en base a la medida de distancia entre los elementos, considerando para determinar la distancia el valor de las *características* de los flujos.

Éstas técnicas forman parte del área del conocimiento *Artificial Intelligence* (AI), y utilizan técnicas conocidas como *Machine Learning* (ML), donde a partir de un conjunto de instancias llamadas de entrenamiento, se generan los valores que presentan las *características* para las diferentes clases de tráfico.

Formalizando éstos conceptos [EMA+07; EMA06], dado un conjunto de flujos $\mathcal{X} = \{X_1, \ldots, X_N\}$. Cada flujo X_i con $1 \leq i \leq N$ se encuentra representado con un vector de atributos que lo describen $X_i = \{x_{i1}, \ldots, x_{id}\}$, donde d es el número de atributos considerados y x_{ih} es el valor de atributo h para el flujo i. El término atributo y *característica* se utiliza en forma análoga y X_i se denomina el vector de *características*. Además se cuenta con un conjunto de clases de tráfico $Y = \{Y_1, \ldots, Y_j\}$ donde j es el número de clases de tráfico, e Y_k con $1 \leq k \leq j$ son clases de tráfico por ejemplo *Web*, *P2P*, *Chat*, *VoIP*, etc.. Dado un vector de atributos X de dimensión d que caracteriza al flujo a clasificar, se le asignará un valor de Y.

Las metodologías de *Machine Learning* (ML) utilizan las N tuplas del conjunto de entrenamiento $\mathcal{X}$ para diseñar la función $f(X) \rightarrow Y$, que permite clasificar nuevos flujos no analizados.

Resumiendo la aplicación de las presentes metodologías divide el espacio en sub-espacios a los que se les asigna una clase de tráfico. El procedimiento de clus-

terización refiere a delimitar los diferentes sub-espacios, agrupando los diferentes elementos según criterios de distancia, buscando que los más cercanos que presentan *características* similares pertenezcan al mismo grupo. La clasificación de un elemento se realiza determinando el sub-espacio al que pertenece y determinando así su clase.

Existen dos clases de técnicas de ML:

- Metodologías ML No Supervisadas.
- Metodologías ML Supervisadas.

Éstas se presentan a continuación incluyendo ejemplos concretos de las mismas.

2.4.1. Metodologías Machine Learning No Supervisadas

Las metodologías de Machine Learning no supervisada (*Unsupervised Learning*) refiere al mecanismo de entrenamiento utilizado para determinar los grupos de elementos (clúster).

La característica de la enseñanza no supervisada es que la muestra de entrenamiento no contienen la clase a la que pertenece cada instancia. La ventaja de los métodos no supervisados, es que ante el surgimiento de nuevas clases, el sistema las reconocerá sin necesidad de nuevas definiciones.

Las metodologías no supervisadas permiten determinar los clusters que se distinguen a partir de la muestra de entrenamiento considerada, pero no determina a que clase de tráfico pertenece cada grupo resultado de la clusterización. Para resolver éste elemento se consideran metodologías semi-supervisadas (*Semi-Supervised Learning*) que resuelven inicialmente los grupos que se desprenden de la muestra de entrenamiento, y posteriormente a cada clúster se le asigna una clase de tráfico en función de una muestra ya clasificada [EMA+07].

Por lo tanto la metodología a aplicar es la siguiente, inicialmente se segmenta el espacio en clusters cada uno identificado con una etiqueta, y posteriormente se asociará a cada etiqueta un clase de tráfico.

Ejemplos de metodologías no supervisadas son:

- Clusterizar mediante *K-Means*

- Clusterizar mediante *Density Based Spatial Clustering of Applications with Noise* (DBSCAN)

2.4.1.1. Clusterizar mediante *K-Means*

El algoritmo *K-Means* permite la subdivisión de un conjunto de elementos en k subconjuntos disjuntos [EAM06]. Es un algoritmo simple y rápido. El particionamiento lo realiza minimizando el error cuadrático (*square-error*).

El cálculo del error cuadrático se presenta en la ecuación 2.2

$$E = \sum_{i=1}^{k} \sum_{j=1}^{N} |dist(x_j, c_i)|^2 \tag{2.2}$$

donde x_j son los elementos a particionar y c_i son los centros de cluster seleccionados. k es el número de clases a subdividir el espacio y N es el tamaño del conjunto de entrenamiento.

E es calculado como la distancia entre cada objeto x_j y el centro de su cluster representado por c_i la media de sus elementos.

Pueden considerarse diferentes formas de medirse la distancia $dist(x, y)$. La distancia más comúnmente utilizada es la Euclidiana [EAM06]. Dados dos vectores $x = \{x_1 \dots x_d\}$ e $y = \{y_1 \dots y_d\}$ la ecuación 2.3 presenta la distancia Euclideana $dist(x, y)$.

$$dist(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \tag{2.3}$$

El algoritmo *K-means* se presenta en 2.6 [EAM06].

El algoritmo de clusterización devuelve $\{c_1 \dots c_k\}$ centros de cluster seleccionados. Dado un elemento $X = \{x_1 \dots x_d\}$ la clasificación se realiza buscando el i que minimiza $dist(X, c_i)$. Calculando $argmin_{i \in [1 \dots k]}(dist(X, c_i))$, se determina la clase a la que pertenece.

Algoritmo 2.6 Algoritmo *K-means*

Require: $\{X_1 \dots X_N\}$, k
Ensure: $\{c_1 \dots c_k\}$
▷ $\{X_1 \dots X_N\}$ contiene muestras de entrenamiento
▷ k cantidad de clusters a generar
$c^{final} \leftarrow generoSolInicial()$ ▷ Genero solución randómica
repeat
$c^{actual} \leftarrow c^{final}$
▷ Determino a que cluster pertenece cada muestra
for $i = 1$ to N **do**
$cluster \leftarrow argmin_{k \in [1 \dots k]}(dist(X_i, c_k^{actual}))$
$S_{cluster} = S_{cluster} \cup X_i$
end for
for $i = 1$ to k **do**
▷ Realizo cálculo de nuevos centros de clusters
$c_i^{actual} \leftarrow \frac{1}{|S_i|} * \sum_{X_j \in S_i} X_j$
end for
until $c^{actual} = c^{final}$
return c^{final}

2.4.1.2. Clusterizar mediante *Density Based Spatial Clustering of Applications with Noise* (DBSCAN)

Los algoritmos del tipo *density-based* buscan determinar zonas de alta densidad de elementos, separados por áreas donde la densidad disminuye [EAM06]. Éstos algoritmos no generan cluster esféricos con centro el representante de la clase, sino que estudian la densidad de las diferentes zonas del espacio para definir los diferentes clusters.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) es un algorimo tipo *density-based*.

DBSCAN recibe dos parámetros ϵ (*eps*) que define la distancia a la que se encuentran los elementos del mismo cluster y m (*minPts*) que define la cantidad mínima de elementos que debe contener cada grupo. Por lo tanto dado dos elementos p y q tal que la distancia entre ellos es menor a ϵ y existen al menos m

elementos a distancia ϵ, entonces p y q pertenecen al mismo grupo de elementos. A dos elementos con ésta condición se les llama *density-reachable*.

Los elementos que no puedan ser considerados dentro de un cluster por no cumplir la condición *density-reachable*, son considerados como ruido (*noise*). Por lo tanto el algoritmo descarta elementos que se encuentran aislados.

El algoritmo DBSCAN se presenta en 2.7 [EAM06].

Una diferencia con *K-means* es el resultado devuelto por ésta metodología. DBSCAN no retorna un conjunto de centros de clusters sino un conjunto de grupos de elementos. Para realizar la clasificación a partir de éstos debe generarse los centros $\{c_1 \dots c_h\}$ con $h = sizeof(S)$. La ecuación 2.4 genera los centros c_i en función de los clusters de elementos C_i generados por le algoritmo.

$$c_i = \frac{1}{|C_i|} * \sum_{X_j \in C_i} X_j \tag{2.4}$$

Dado un elemento $X = \{x_1 \dots x_d\}$, la clasificación se realiza buscando el i que minimiza $dist(X, c_i)$. Calculando $argmin_{k \in [1 \dots h]}(dist(X, c_k))$ con $h = sizeof(S)$, se determina la clase a la que pertenece. La distancia Euclideana (ecuación 2.3) se recomienda para éstos cálculos [EMA+07], pudiendo utilizarse otras distancias.

2.4.2. Metodologías Machine Learning Supervisadas

Las metodologías de *Machine Learning* (ML) supervisadas (*Supervised Learning*) refieren mecanismo de entrenamiento utilizado.

Se define entrenamiento supervisado cuando se incluye la clase de tráfico a la que pertenece cada elemento de la muestra [HTF01].

Formalizando éstos conceptos, dado un conjunto de flujos $\mathcal{X} = \{X_1, \dots, X_N\}$. Cada flujo X_i con $1 \leq i \leq N$ se encuentra representado por el vector de atributos que lo describen y clase de tráfico asignada $X_i = \{x_{i1}, \dots, x_{id}, e_i\}$, donde d es el número de atributos considerados, x_{ih} es el valor de atributo h para el flujo i y e_i es la clase de tráfico asignada al elemento i. Igual que en el caso no supervisado se cuenta con un conjunto de clases de tráfico $Y = \{Y_1, \dots, Y_j\}$ donde j es el número de clases de tráfico. Dado un vector de atributos X de dimensión d que caracteriza al flujo a clasificar, se le asignará un valor de Y. Utilizando las N

Algoritmo 2.7 Algoritmo *DBSCAN*

Require: $\{X_1 \dots X_N\}$, eps, $minPts$
Ensure: S ▷ Conjunto de Clustes C de elementos X_i
  $S \leftarrow \{\}$
  for all X_i *no visitado* **do**
    $visitado(X_i)$ ▷ marco X_i como visitado
    $N_{ini} \leftarrow getNeighbors(X_i, eps)$ ▷ N_{ini} contiene los $X_j / dist(X_i, X_j) < eps$
    if $sizeof(N_{ini}) >= minPts$ **then**
      ▷ X_i es *density-reachable*
      $C \leftarrow \{X_i\}$
      for all $Y \in N_{ini}$ **do**
        if Y *no visitado* **then**
          $visitado(Y)$ ▷ marco Y como visitado
          $N_{aux} \leftarrow getNeighbors(Y, eps)$ ▷ N_{aux} son los $X_j / dist(Y, X_j) < eps$
          if $sizeof(N_{aux}) >= minPts$ **then**
            $N_{ini} \leftarrow N_{ini} \cup N_{aux}$
          end if
        end if
        if Y *no pertenece ningún cluster* **then**
          $C \leftarrow C \cup \{Y\}$
        end if
      end for
      $S \leftarrow S \cup \{C\}$ ▷ Agrego cluster C al conjunto S de clusters
    else
      ▷ X_i no es *density-reachable*
      $noise(X_i)$ ▷ marco X_i como noise
    end if
  end for
  return S

tuplas del conjunto de entrenamiento $\mathfrak{X}$ se diseña la función $f(X) \rightarrow Y$, que permite clasificar nuevos flujos no analizados.

Como en los métodos supervisados cada elemento de la muestra de entrenamiento $\mathfrak{X}$ contiene su clase, los clusters se generan en función de las requer-

imientos de clasificación exigidos por el problema en particular, sin requerir la posterior asignación de una clase a cada cluster obtenido.

Ejemplos de éstas metodologías de ML son:

- Clasificadores Naïve Bayes.
- Clasificadores por *Classification and Regression Trees* (CART).
- Clasificadores por *Support Vector Machines* (SVM).

2.4.2.1. Clasificadores Naïve Bayes

El método de clasificación Naïve Bayes es uno de los más simples que pueden aplicarse a la clasificación de tráfico [MZ05].

Dado un flujo por sus atributos $X = \{x_1, \ldots, x_d\}$, ésta metodología estima la probabilidad de pertenecer a las diferentes clases de tráfico Y_k. El resultado es un vector de probabilidades de tamaño j que contiene para el X proporcionado la probabilidad de pertenecer a cada clase de tráfico Y_k con $1 \leq k \leq j$. El resultado es una función de probabilidad entonces se cumple que la suma de los elementos del vector es 1 ($\sum_{Y_k} Pr[Y_k|X] = 1$).

Naïves Bayes utiliza probabilidades bayesianas condicionales expresadas en la ecuación 2.5:

$$Pr[Y_j|X] = \frac{Pr(Y_j) * F(X|Y_j)}{\sum_{Y_k} Pr(Y_k) * F(X|Y_k)} \tag{2.5}$$

donde $Pr(Y_j)$ es la probabilidad de la clase Y_j y $F(X|Y_j)$ es la función de distribución de probabilidad de X dado Y_j. El denominador permite normalizar la probabilidad a 1, asegurando que $\sum_{Y_k} Pr[Y_k|X] = 1$.

El conjunto de entrenamiento $\mathcal{X} = \{X_1, \ldots, X_N\}$ se utiliza para determinar la función $F(X|Y_k)$ y estimar las probabilidades $Pr(Y_j)$. Los flujos a clasificar presentados con $X = \{x_1, \ldots, x_d\}$ valor de d *características*, se asumen independientes lo que permite su cálculo en función de productos de probabilidades. El estudio de la función $F(X|Y_k)$ se realiza calculando la distribución empírica a partir del conjunto de entrenamiento.

El algoritmo aplicado para clasificar se presenta en 2.8 y simplemente aplica la fórmula 2.5 para cada clase de tráfico.

Algoritmo 2.8 Algoritmo *Naïves Bayes*

Require: $X \triangleright$ Con $X = \{x_1 \dots x_d\}$

Ensure: $\{p_1 \dots p_j\} \triangleright$ Probabilidad p_i de pertenecer a clase i

$\triangleright$ Se cuenta con $F(X|Y_k)$ con $1 \leq k \leq j$

$\triangleright$ Se cuenta con $Pr(Y_k)$ con $1 \leq k \leq j$

$Pr_{den} \leftarrow \sum_{Y_k} Pr(Y_k) * F(X|Y_k)$

for i in $[1 \dots j]$ **do**

$\triangleright$ Calculo p_i

$p_i = \frac{Pr(Y_i) * F(X|Y_i)}{Pr_{den}}$

end for

return $\{p_1 \dots p_j\}$

Para determinar la clase de tráfico resultante del análisis se debe buscar la que presenta mayor probabilidad. Por lo tanto la clase asignada está dada por la función $argmax_{i \in [1 \dots j]}(p_i)$.

2.4.2.2. Clasificadores por *Classification and Regression Trees* (CART)

Este método implementa un árbol de decisión en función del valor de los diferentes atributos de entrada. La construcción del árbol permite clasificar en forma rápida, llevadas a la práctica por medio de un conjunto de condiciones *if-then* [LD00].

Para la construcción de los mismos se cuenta con un conjunto de entrenamiento con el valor obtenido para las diferentes *características* y su clasificación resultado ($\mathcal{X} = \{X_1, \dots, X_N\}$ con $X_i = \{x_{i1}, \dots, x_{id}, e_i\}$).

Sea el siguiente ejemplo [WF05]. Un campo de golf desea determinar los requerimientos de personal de staff. Para ésto estudia el comportamiento de los usuarios según el pronóstico del tiempo, tomando las siguientes variables:

- *Pronóstico* con resultados soleado, nublado y lluvioso.
- *Temperatura* con resultados caluroso, templado y freso.
- *Humedad* con resultados alta o normal.
- *Ventoso* con resultado si o no.

Se toman los parámetros del tiempo obtenidos durante 14 días y se agrega el resultado obtenido (*juega*, *no juega*) para éstos días analizados, los mismos se presentan en la tabla 2.2.

Cuadro 2.2: *Datos utilizados para armar el Árbol de Decisión*

Pronóstico	*Temperatura*	*Humedad*	*Ventoso*	*Resultado*
Soleado	Caluroso	Alta	No	No Juega
Soleado	Caluroso	Alta	Si	No Juega
Nublado	Caluroso	Alta	No	Juega
Lluvioso	Templado	Alta	No	Juega
Lluvioso	Fresco	Normal	No	Juega
Lluvioso	Fresco	Normal	Si	No Juega
Nublado	Fresco	Normal	Si	Juega
Soleado	Templado	Alta	No	No Juega
Soleado	Fresco	Normal	No	Juega
Lluvioso	Templado	Normal	No	Juega
Soleado	Templado	Normal	Si	Juega
Nublado	Templado	Alta	Si	Juega
Nublado	Caluroso	Normal	No	Juega
Lluvioso	Templado	Alta	Si	No Juega
Lluvioso	Templado	Alta	Si	No Juega

En la figura 2.4 se presenta un posible árbol de decisión para los datos presentados. Del mismo se desprenden por ejemplo que cuando está nublado se tiene una asistencia importante y que los días lluviosos si no está ventoso también habrá asistencia.

Un elemento a destacar es que éstos árboles de decisión en ocasiones como la presentada no requieren la utilización de todas las *características* analizadas. Además los resultados no siempre son absolutos, y pueden presentar resultados probabilísticos donde se obtiene la probabilidad de ocurrencia de cada posible evento.

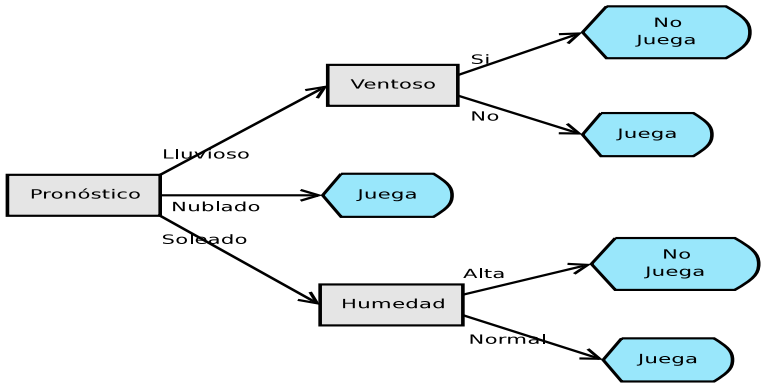


Figura 2.4: *Ejemplo de Árbol de Decisión*

El algoritmo utilizado por CART para generar el árbol de decisión se presenta en 2.9 [Lew00; HTF01].

La clasificación se realiza aplicando la secuencia *if-then* a partir del árbol obtenido T_{fin}, donde las condiciones están determinadas por el valor de las característica expresada en $T.caracteristica$ con el valor expresado en $T.condicion$ y devolviendo el resultado de la última hoja alcanzada.

2.4.2.3. Clasificadores por *Support Vector Machines* (SVM)

La metodología *Support Vector Machines* (SVM) se basa en resolver la clasificación de espacios *linealmente separables*. Un espacio de dimensión n se dice *linealmente separable* si puede separarse con un hiperplano de dimensión $n-1$. Para el caso de un espacio de dos dimensiones se debe cumplir que puede ser separado con una línea recta. De ésta forma se logra separar dos clases de elementos.

SVM resuelve el hiperplano con máximo margen (*maximum margin hyperplane*) [WF05]. Este hiperplano es el que cumple con la condición de tener la mayor distancia entre las clases (ver figura 2.5). Para determinar el hiperplano con máximo margen se consideran los polígonos convexos que contienen todos los elementos de la clase. Al ser linealmente separables, los polígonos convexos de las clases no se superponen. Para el caso presentado en la figura 2.5 de dos dimensiones, el hiperplano con máximo margen es la bisectriz perpendicular de la linea menor, que conecta los polígonos.

Algoritmo 2.9 Algoritmo *CART*

Require: $\{X_1 \dots X_N\}$ ▷ N instancias de entrenamiento
Ensure: T ▷ Árbol binario de decisión
  ▷ Formato $X_i = \{x_{i1}, \dots, x_{id}, e_i\}$
  $T \leftarrow \{\}; T_{fin} \leftarrow T; FIN \leftarrow false$
  while *not FIN* **do**
    ▷ Selecciono para *características* no utilizadas en T la de mayor ganancia
    $T.caracteristica \leftarrow ganancia(T)$
    for all $N_{aux} \leftarrow split(T)$ **do** ▷ Particiono por $T.caracteristica$
      if $e_i = e_j \forall X_i, X_j \in N$ **then** ▷ Pertenecen misma clase
        ▷ Creo hijo HOJA T con condicion de $split(T)$
        $creoHijo(T, HOJA, N_{aux}.condicion)$
      else
        if $caracteristicas(T) < CANTIDAD\_CARACTERISTICAS$ **then**
        ▷ Quedan *características* para procesar en T
          ▷ Creo hijo INTERIOR bajo T con condicion de $split(T)$
          $creoHijo(T, INTERIOR, N_{aux}.condicion)$
        else
          ▷ Creo HOJA bajo T con condicion de $split(T)$
          $creoHijo(T, HOJA, N_{aux}.condicion)$
        end if
      end if
    end for
    $T \leftarrow next(T)$ ▷ Devuelve proximo sub-arbol a procesar
    if $T = \{\}$ **then**
      ▷ Si todos los nodo INTERIOR procesados
      $FIN \leftarrow true$
    end if
  end while
  return T_{fin}

A los elementos que se encuentran más cercanos al hiperplano con máximo margen se denominan *support vectors*. Conocidos éstos elementos ya puede determinar el hiperplano, independientemente de los restantes elementos de la clase.

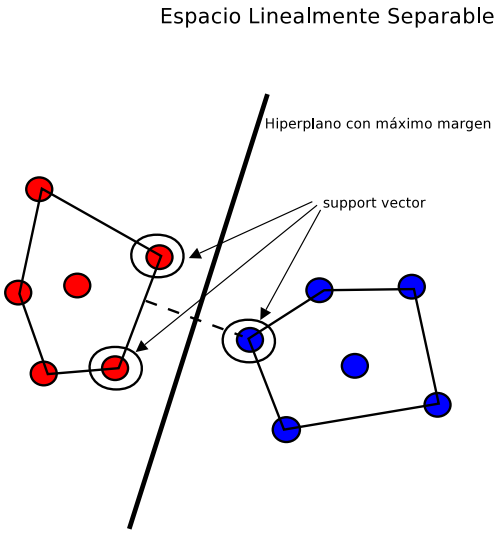


Figura 2.5: Esquema de funcionamiento de SVM

Para resolver la búsqueda de los *support vectors* y determinar el hiperplano con máximo margen, se utilizan las técnicas de problemas de optimización llamadas *quadratic optimization*.

El hiperplano se calcula a través de una función $\mathfrak{f}$ donde $\mathfrak{f} : X \subseteq \mathbb{R}^d \rightarrow \mathbb{R}$ con d la dimensión del espacio de entrada [CST00]. La clasificación para el caso de una entrada $x = (x_1, \cdots, x_d)$, estará dada por el signo de $\mathfrak{f}(x)$, $sgn(\mathfrak{f}(x))$. La ecuación 2.6 presenta la solución genérica de la función $\mathfrak{f}(x)$.

$$\mathfrak{f}(x) = \langle w \cdot x \rangle + b = \sum_{i=0}^{d} w_i * x_i + b \tag{2.6}$$

En la instancia de entrenamiento se determinarán los valores de w y b. La operación $\langle w \cdot x \rangle$ se conoce como producto escalar (*dot product*). Éstos valores se presentan en función de los *support vectors*.

La solución presentada resuelve el problema para espacios linealmente separables. Para poder aplicar SVM a espacios no lineales se plantea utilizar una función que los convierta a linealmente separables. Sea $x = (x_1, \cdots, x_d)$ los valores de las características consideradas, donde se propone entonces realizar la siguientes transformación:

$$x = (x_1, \cdots, x_d) \longmapsto \phi(x) = (\phi_1(x), \cdots, \phi_D(x))$$

La función ϕ transforma el espacio de entrada X a un nuevo espacio F (*feature space*) donde $F = \{\phi(x) \mid x \in X\}$ con D dimensión de F.

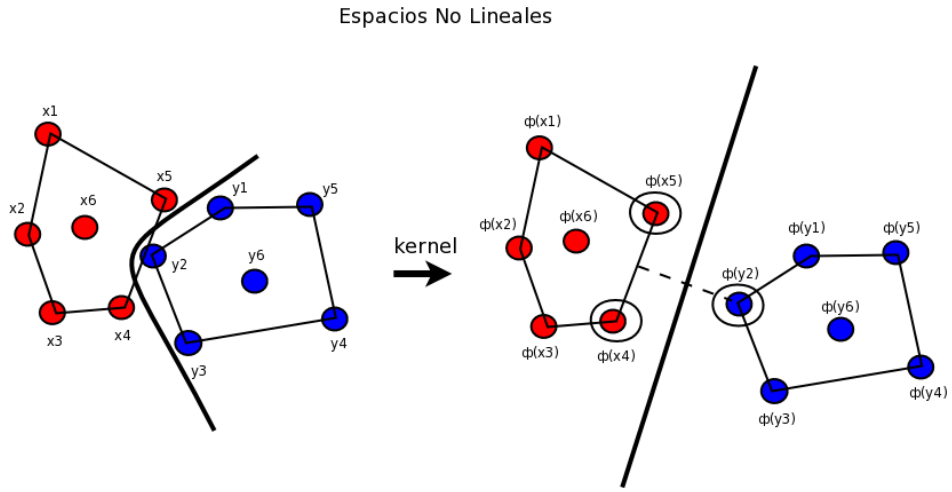


Figura 2.6: SVM en Espacios No Lineales

Los pasos a seguir en éstos casos son (ver figura 2.6):

- Transformar el espacio de entrada X al espacio caracterizado F.
- Aplicar la clasificación SVM sobre el espacio F resultado.

Para realizar los dos pasos en forma directa se plantea la función kernel K. Ésta función se define como la que para todo $x, z \in X$,

$$K(x, z) = \langle \phi(x) \cdot \phi(z) \rangle$$

donde ϕ es una función que transforma el espacio de entrada en X a uno caracterizado F.

Existen funciones de Kernel estandar ya definidas:

- Linear:

$$K(x, z) = \langle x \cdot z \rangle \tag{2.7}$$

- Polynomial:

$$K(x, z) = (\langle x \cdot z \rangle)^d \tag{2.8}$$

- Radial Basis Function (RBF):

$$K(x, z) = exp(-\gamma ||x - z||^2), \gamma > 0 \tag{2.9}$$

donde, γ y d son parámetros del kernel.

La solución presentada permite resolver la clasificación en presencia de dos clases. Para extender la propuesta a clasificación en más de dos clases (*multi-class*) existen dos procedimientos conocidos como *one-against-one* y *one-against-all*.

- *One-against-one* dadas k clases, se construyen $k * (k-1)/2$ clasificadores de dos clases (todos los clasificadores de dos clases posibles). La decisión se realiza a través de una metodología de votación donde se determinan las clases resultado para los $k*(k-1)/2$ clasificadores y la clase con más votos es el resultado final.

- *One-against-all* dadas k clases, para cada clase se genera un clasificador considerando los elementos que pertenecen a la misma y los que pertenecen a las restantes. La clasificación se realiza determinando cuál es de los k clasificadores identificó una clase específica y devolviendo la misma.

2.4.3. Ventajas y Desventajas de Metodologías Basadas en Elementos Estadísticos

Una ventaja presentada por éstas metodologías, es que considerando *características* que no involucren los datos contenidos en los paquetes, permite clasificar flujos sin necesidad de analizar los datos. Esto permite su utilización en tráfico encriptado.

Como desventaja puede considerarse que presentan la necesidad de contar con un conjunto de entrenamiento, por lo que se debe disponer de flujos ya clasificados en forma correcta.

Otra complejidad presentada es la necesidad de estudiar las *características* a utilizar para la clasificación. El estudio requiere analizar los resultados que se obtienen para los atributos disponibles. Para generar el modelo de clasificación se requieren pruebas considerando diferentes atributos y el análisis comparativo de los resultados obtenidos. Esta tarea es difícil y debe realizarse previamente a la puesta en producción del mecanismo de clasificación.

El valor de las diferentes *características* consideradas puede involucrar el cálculo de valores medios o varianzas y requerir el procesamiento de un número de paquetes del flujo para obtenerse, lo que genera que la clasificación exige un tiempo no despreciable.

2.5. Clasificación Aplicando Múltiples Metodologías

La clasificación aplicando múltiples metodologías, plantea la utilización de diferentes mecanismos de clasificación y en base a los resultados obtenidos devolver un resultado final. El objetivo buscado en la combinación de los resultados es aumentar la certeza y el volumen de tráfico clasificado.

En base a técnicas de *Machine Learning* se presentan mecanismos para la combinación de los resultados obtenidos por diferentes metodologías de clasificación. Éstos mecanismos son conocidos como *Bagging* (*Bootstrap Aggregating*), *Boosting* y *Stacking* (*Stacked generalization*) [WF05].

Bagging y *boosting* aplican un mecanismo de votación en base a los resultados obtenidos por los diferentes mecanismos de clasificación a combinar. *Bagging* asigna a los mecanismos el mismo peso relativo, mientras que *boosting* asigna un peso en función de los resultados que presenta cada uno.

Stacking plantea dos niveles de clasificación, el nivel 0 donde refiere a los resultados obtenidos por los mecanismos a combinar y el nivel 1 donde se modela un resultado a partir de los resultados obtenidos en el nivel 0.

Ejemplo de una metodología de combinación es presentada por *Szab et al.* [SSO07]. La metodología se basa en un mecanismo complejo de decisión, que clasifica las diferentes aplicaciones que interactúan en la red.

Algoritmo 2.10 Algoritmo *Clasificación Aplicando Múltiples Metodologías*

Require: $\{resut_{SA}, result_{WKP}, result_{HEUR}\}$
Ensure: $clase\_trafico$, $noClasificado$
▷ $resut_{SA}$ resultado Análisis de Patrones
▷ $result_{WKP}$ resultado WKP
▷ $result_{HEUR}$ resultado Heurísticas
if $cantMetodologias(resut_{SA}, result_{WKP}, result_{HEUR}) = 1$ **then**
▷ Clasificado por una sola metodología
▷ Devuelvo único resultado obtenido
return $getClase(resut_{SA}, result_{WKP}, result_{HEUR})$
end if
if $cantMetodologias(resut_{SA}, result_{WKP}, result_{HEUR}) > 1$ **then**
▷ Clasificado por más de una metodología
if $resut_{SA} = result_{WKP}$ **then**
▷ Devuelvo $resut_{SA}$
return $resut_{SA}$
end if
if $resut_{SA} = result_{HEUR}$ **then**
▷ Devuelvo $resut_{SA}$
return $resut_{SA}$
end if
if $result_{WKP} = result_{HEUR}$ **then**
▷ Devuelvo $resut_{WKP}$
return $resut_{WKP}$
end if
end if
return $noClasificado$

La metodología presentada clasifica en base a los resultados obtenidos por las metodologías *Well Known Ports*, *Análisis de Patrones* en los datos y *heurísticas* basadas en la búsqueda de patrones relacionados con elementos estadísticos del tráfico analizado y de la utilización de puertos de capa transporte.

En función de los resultados obtenidos en las 3 metodologías aplicadas, se propone una metodología de combinación de resultados. Ésta combinación se fun-

damenta en elementos empíricos sobre la certeza de las diferentes metodologías.

El algoritmo aplicado se presenta en 2.10.

La idea propuesta presentada se fundamenta en la certeza de las diferentes metodologías aplicadas. Del mismo se desprende que *Análisis de Patrones* es el mecanismos con mayor nivel de certeza, seguido por *Well Known Ports*, y la metodología menos certera son las *heurísticas* propuestas [SSO07].

2.5.1. Ventajas y Desventajas Clasificación Aplicando Múltiples Metodologías

Dado que las metodologías presentadas en las secciones anteriores del capítulo tienen resultados de éxito diferente dependiendo de la clase de tráfico analizada, la principal ventaja que presenta la aplicación de múltiples metodologías es la posibilidad de unir los resultados logrando un resultado final global mejor en referencia al volumen de tráfico clasificado y certeza del mismo.

La principal desventaja encontrada es que las heurísticas enunciadas se proponen sobre metodologías específicas concretas y su extensión con la aplicación de nuevas metodologías no es simple.

No se encuentran casi propuestas en el ambiente académico de las presentes metodologías, lo que genera un desafío importante.

Capítulo 3

Clasificación del Tráfico Basado en Combinación de Metodologías

3.1. Introducción

En éste capítulo se presentan dos propuestas para la combinación de resultados de heurísticas de clasificación de tráfico.

El objetivo buscado en la combinación de los resultados de las diferentes metodologías de clasificación es aumentar el porcentaje de tráfico clasificado respecto a los resultados obtenidos por las metodologías de clasificación aisladas y mejorar su nivel de certeza.

Se proponen dos mecanismos:

- *Combinación con Peso por Heurística* basada en la asignación de pesos por heurística de clasificación de tráfico y determinando una clasificación global al flujo.
- *Clasificación Bayesiana* donde se aplican técnicas Naïve Bayes para la combinación de los resultados.

Se incluye en el presente capítulo conceptos previos requeridos y las metodologías de combinación propuestas.

3.2. Mecanismos de Clasificación Determinísticos y Probabilísticos

Los resultados obtenidos por los mecanismos de clasificación, se pueden agrupar en dos categorías, *determinísticos* y *probabilísticos* (también denominados clasificación *hard* y *soft*) [MZ05]:

- *Determinísticos.* Los mecanismos determinísticos a través del estudio de la distancia entre los diferentes elementos generan clases delimitadas por fronteras. Todos los elementos que se encuentren en su interior se consideran pertenecientes a ésta clase. El resultado obtenido de una clasificación determinística será la clase a la que pertenece el elemento.

- *Probabilísticos.* Los mecanismos probabilísticos clasifican un elemento devolviendo la probabilidad de que éste pertenezca a cada una de las clases definidas. El resultado obtenido en éste caso será un vector de probabilidades de dimensión la cantidad de clases existentes, indicando la probabilidad de pertenecer a cada clase. El resultado es una función de probabilidad entonces se cumple que la suma de las probabilidades de pertenecer a todas las clases es igual a 1.

A modo de ejemplo, dado un flujo a clasificar el resultado para el caso *determinísitico* sería la clase a la que pertenece el flujo, por ejemplo *Mail*, mientras que para el caso *probabilístico* se presentará con una probabilidad de 0,8 pertenece a la clase *Mail*, 0,1 a la clase *Web* y 0,1 a la clase *Bulk*.

Con el objetivo de formalizar éstas definiciones, consideramos n el número de clases de tráfico definidas. En el caso *determinístico* el resultado de la clasificación es un valor k con $-1 \leq k \leq (n-1)$, donde -1 indica no estar clasificado y en caso $0 \leq k \leq (n-1)$ indica la clase resultado. Para el caso *probabilístico* será un vector P de dimensión $1 \times (n+1)$, con $P = \{p_0, \cdots, p_n\}$, donde p_0 indica la probabilidad de no haber sido clasificado, y p_i con $1 \leq i \leq n$ la probabilidad de pertenecer a la clase $i-1$. P es una función de probabilidad por lo tanto se cumple que $\sum_{i=0}^{n} p_i = 1$.

Considerando que dependiendo de la metodología de clasificación utilizada el resultado puede ser *determinístico* o *probabilístico*, se plantean los siguientes mecanismos de conversión.

Para la conversión *determinístico* a *probabilístico* se utilizará la ecuación 3.1 para el cálculo de p_i, con $0 \leq i \leq n$, donde k es el resultado determinístico de la clasificación y μ el nivel de certeza de la metodología aplicada.

$$p_i = \begin{cases} 1-\mu & \text{si } i = k+1 \\ \mu/n & \text{en caso contrario} \end{cases} \tag{3.1}$$

El parámetro μ permite asignar un nivel de certeza al resultado presentado. El mismo tiene un valor en el rango $[0, 1]$. Valores pequeños de μ determinan mayor nivel de certeza.

La conversión *determinístico* a *probabilístico* propuesta asigna a la clase k la probabilidad mayor según el nivel de certeza μ y asigna equiprobabilidad a las clases restantes. Existen otras posibilidades para la conversión. La propuesta se fundamenta en que es fácilmente generalizable a diferentes mecanismos de clasificación y para su aplicación solamente se requiere disponer del nivel de certeza de la metodología aplicada.

Para la conversión *probabilístico* a *determinístico* se busca el elemento con mayor probabilidad. Por lo tanto dado $P = \{p_0, \cdots, p_n\}$ el resultado determinístico k estará dado por la ecuación 3.2.

$$k = (argmax_{i \in [0 \cdot n]}(p_i) - 1) \tag{3.2}$$

La conversión *probabilístico* a *determinístico* se fundamenta en el concepto intuitivo de asignar la clase que presenta mayor probabilidad.

3.3. Mecanismos para Combinación de Resultados

El problema a resolver es el siguiente, dado un flujo con los resultados obtenidos para las diferentes metodologías de clasificación aisladas aplicadas al mismo, asignarle una clase de tráfico.

Asimismo se plantea priorizar en la combinación presentada los siguientes criterios:

- Aumentar la cantidad de flujos clasificados, lo que implica utilizar toda la información generada por los mecanismos de clasificación aplicados. Éste objetivo pretende que todos los flujos clasificados por alguna de las metodologías de clasificación, obtengan resultado en la combinación.

- Obtener resultados con el mayor nivel de certeza. Definimos *certeza* como:

$$certeza = \frac{\#clasificados\_correctamente}{\#total\_clasificados}$$

  El concepto de certeza considerado se sustenta en determinar la relación entre los flujos correctamente clasificados sobre el total del los flujos clasificados.

En relación con el nivel de certeza debe destacarse que determinar el valor $\#clasificados\_correctamente$ es un elemento complejo, dado que no se cuenta con trazas de tráfico que presenten la clase de tráfico para cada flujo. Esto se debe a que dependiendo del objetivo de la clasificación se determina las clases de tráfico a utilizar. La opción tomada por diferentes investigadores es comparar mecanismos entre sí, para las diferentes trazas de tráfico disponibles para evaluar resultados.

Para el cálculo debe considerarse que alguna de las metodologías aplicadas puede no tener éxito en la clasificación, donde para el caso *determinístico* será representado con -1 y en el caso *probabilístico* será con p_0 el mayor valor del vector de probabilidades.

Se evaluarán dos opciones para clasificar los flujos según las categorizaciones encontradas:

- Metodología de *Combinación con Peso por Heurística.*

- Metodología de *Clasificación Bayesiana.*

Las metodologías se presentan a continuación. Con el fin de generalizar los resultados, para formalizar la presentación consideraremos la aplicación de m

mecanismos de clasificación y n clases de tráfico. Además se consideraran los resultados requeridos por las metodologías de combinación en formato *determinístico* o *probabilístico*. En caso de aplicar una metodología que no devuelva el resultado en el formato requerido, se utilizarán los procedimientos de conversión presentados en la sección 3.2.

3.4. Metodología de *Combinación con Peso por Heurística*

En primera instancia se propone una metodología simple para la combinación de resultados, asignando un peso relativo γ_i a cada una de las m metodologías de clasificación aplicadas. Se implementa una combinación lineal convexa, por lo tanto realizando la suma de probabilidades con peso se realiza la clasificación. Ésta metodología recibe los resultados de los m mecanismos de clasificación de entrada en formato *probabilístico*, y devuelve sus resultados también en formato *probabilístico*.

El ejemplo presentado en la sección 2.5 plantea la asignación de los resultados basados en la certeza de cada metodología aplicada (*Análisis de Patrones*, *Well Known Ports* y *Heurísticas*). La metodología de combinación aquí propuesta generaliza este concepto. Se consideran resultados *probabilísticos* de los mecanismos de clasificación aplicados. Esto permite que además de considerar el peso asignado a cada mecanismo γ_i se considere la probabilidad asignada por el mecanismo p_{ij} para cada clase de tráfico.

El fundamento de la propuesta de *Combinación con Peso por Heurística* se sustenta en que conociendo la certeza de cada una de las metodologías de clasificación aplicadas, permite su aplicación en forma muy simple. De obtener buenos resultados, será un mecanismo de combinación muy atractivo.

3.4.1. Cálculo para *Combinación con Peso por Heurística*

Sea el vector $\mathbf{P^i}$ con $1 \leq i \leq m$ el resultado *probabilístico* de la metodología i de clasificación. Su contenido es $\mathbf{P^i} = \{p_{i0}, \cdots, p_{in}\}$ donde n es la cantidad de

clases de tráfico definidas y p_{ij} es la probabilidad de pertenecer a la clase $j-1$ según la aplicación de la metodología de clasificación i.

La ecuación 3.3 presenta el vector probabilístico $\mathbf{Pr^{Heur}}$ asignado al flujo por el mecanismo de *Combinación con Peso por Heurística*.

$$\mathbf{Pr^{Heur}} = \gamma_1 \mathbf{P^1} + \cdots + \gamma_m \mathbf{P^m} \tag{3.3}$$

donde se cumple,

$$\sum_{i=1}^{m} \gamma_i = 1$$

Puede verificarse que $\mathbf{Pr^{Heur}}$ es una función de probabilidad, por lo que la suma de las probabilidades parciales es 1.

Para verificarlo sea $\mathbf{Pr^{Heur}} = \{p_0^{Heur}, \cdots, p_n^{Heur}\}$ un vector de dimensión $1 \times (n+1)$ resultado de la metodología de *Combinación con Peso por Heurística*. Se debe cumplir que la suma de sus $n+1$ componentes es 1.

$$\sum_{j=0}^{n} p_j^{Heur} = 1$$

Demostración:

$$\sum_{j=0}^{n} p_j^{Heur} = \sum_{j=1}^{n} (\gamma_1 p_{1j} + \cdots + \gamma_m p_{mj})$$

$$\sum_{j=1}^{n} (\gamma_1 p_{1j} + \cdots + \gamma_m p_{mj}) = \gamma_1 \sum_{j=1}^{n} p_{1j} + \cdots + \gamma_m \sum_{j=1}^{n} p_{mj}$$

como

$$\sum_{j=1}^{n} p_{ij} = 1, \forall i$$

por la definición de $\mathbf{P^i}$ se cumple

$$\gamma_1 \sum_{j=1}^{n} p_{1j} + \cdots + \gamma_m \sum_{j=1}^{n} p_{mj} = \gamma_1 + \cdots + \gamma_m$$

que como se definieron los pesos γ_i queda demostrado.

$$\gamma_1 + \cdots + \gamma_m = \sum_{i=1}^{m} \gamma_i = 1$$

Se deben tener las siguientes consideraciones para los valores faltantes (*missing values*). Éstos valores corresponden a las metodologías que no clasifican con éxito y no devuelven resultado. Como la metodología de combinación se aplica sobre resultados *probabilísticos*, los valores faltantes corresponden a vectores de probabilidades $\mathbf{P^i} = \{p_{i0}, \cdots, p_{in}\}$ donde se cumple$-1 = (argmax_{j\in[0\cdot n]}(p_{ij}) - 1)$ según lo presentado en 3.2. En la combinación de resultados debe buscarse que la aparición de valores p_{i0} no genere un valor que anule la clasificación realizada por otra metodología. Para ésto al convertir el valor *determinístico* -1 en *probabilístico*, se considera un valor de certeza μ_{unclf} alto, entonces en el caso de no clasificado retorna una probabilidad p_{i0} baja cumpliendo $-1 = (argmax_{j\in[0\cdot n]}(p_{ij}) - 1)$. El valor μ_{unclf} propuesto debe cumplir la siguiente condición:

$$\mu_{unclf} > \frac{1}{m}$$

siendo m la cantidad de heurísticas aplicadas. Éste valor permitirá que los valores faltantes no incidan en las clasificaciones obtenidas por las demás heurísticas.

3.4.2. Aplicación de *Combinación con Peso por Heurística*

Como se indicó en los fundamentos la metodología *Combinación con Peso por Heurística* en situaciones donde se aplican diferentes metodologías, permite la acumulación de sus resultados.

Como existen diferentes metodologías de clasificación que se focalizan en cierto tipo de tráfico, el aplicar ésta metodología de combinación aumentará los resultados obtenidos, acumulando los resultados parciales de las diferentes metodologías.

Se busca por medio del peso asignado a cada una de las metodologías aisladas, acentuar la certeza del resultado global final, en función de la certeza de cada mecanismo. El mecanismo igualmente no permite la consideración de certeza por tipo de tráfico, lo que se entiende es una limitante del mismo.

Para su aplicación debe considerarse además el valor probabilístico asignado a los resultados de no clasificación, buscando que no anulen resultados obtenidos.

3.5. Metodología de *Clasificación Bayesiana*

Clasificación Bayesiana propone para la combinación de resultados la aplicación de un *clasificador bayesiano* en base a la evidencia obtenida a través de los resultados de las metodologías individuales aplicadas. Las técnicas bayesianas en problemas como la detección de correo *spam*, son antecedentes que permiten inferir la obtención de buenos resultados.

La utilización de *Clasificación Bayesiana* además resuelve problemas de errores sistemáticos en las metodologías aplicadas. Éstos errores representan casos donde las metodologías aisladas confunden tipos de tráfico.

La metodología *Combinación con Peso por Heurística* propuesta anteriormente exige el conocimiento del nivel de certeza de los diferentes mecanismos de clasificación aplicados. Ésta propuesta permite que en base a los registros de la clasificación realizada se devuelvan resultados en función de los niveles de certeza para las diferentes metodologías. Adicionalmente los niveles mencionados presentarán valores relativos a la clase de tráfico analizada, lo que brinda mejoras a la propuesta anterior realizada.

Un *clasificador bayesiano* se define como un clasificador probabilístico que aplica el *Teorema de Bayes* (ver [WF05]). A continuación formalizaremos los conceptos utilizados. Dada una hipótesis H y una evidencia E la aplicación del *Teorema de Bayes* corresponde a la aplicación de la ecuación 3.4.

$$Pr[H|E] = \frac{Pr[H] * Pr[E|H]}{Pr[E]} \tag{3.4}$$

La evidencia E utilizada en nuestro estudio son los resultados *determinísticos* de las heurísticas aplicadas (uno por cada heurística), por lo que $E = \{E_1 \cdots E_m\}$, siendo E_i con $1 \leq i \leq m$ el resultado determinístico de la metodología de clasificación i. Si se consideran E_i como sucesos independientes, se puede extender la fórmula 3.4 de la siguiente manera:

$$Pr[H|E_1 \cdots E_m] = \frac{Pr[H] * \prod_{i=1}^{m} (Pr[E_i|H])}{\prod_{i=1}^{m} (Pr[E_i])} \tag{3.5}$$

Debe examinarse al afirmación realizada sobre la consideración de E_i como sucesos independientes. Dos sucesos son independientes si el resultado de uno

no influye en el resultado del otro. Esta afirmación claramente no se cumple considerando que la clasificación E_i realizada por la heurística i, está relacionada con la clasificación E_j realizada por la heurística j por corresponder al mismo flujo. Este problema es analizado en [Ris05], donde se concluye que los clasificadores *Naïve Bayes* obtienen buenos resultados cuando los atributos son completamente independientes o cuando éstos son funcionalmente dependientes, que es nuestro caso, lo que habilita su aplicación.

El denominador de la expresión 3.5 es igual para todas las hipótesis H consideradas y depende únicamente de la evidencia $\{E_1 \cdots E_m\}$, ésto permite evitar su cálculo requiriendo después de finalizado normalizar el resultado llevando las sumas de todas las posibles hipótesis H a 1. La ecuación 3.6 presenta el cálculo requerido.

$$Pr[H|E_1 \cdots E_m] \propto Pr[H] * \prod_{i=1}^{m} (Pr[E_i|H]) \tag{3.6}$$

Ésta metodología propuesta recibe los resultados de los m mecanismos de clasificación de entrada en formato *determinístico*, y devuelve sus resultados en formato *probabilístico*.

3.5.1. Cálculo para *Clasificación Bayesiana*

Supongamos que deseamos conocer la probabilidad de la hipótesis H que el flujo estudiado sea de clase c_b, y contamos con la evidencia E del resultado obtenido $\{cme_1, \cdots, cme_m\}$, donde $-1 \leq cme_i \leq (n-1)$ y $1 \leq i \leq m$. El cálculo a realizar es el siguiente:

$$Pr[c_b|cme_1, \cdots, cme_m] = Pr[c_b] * \prod_{i=1}^{m} (Pr[cme_i|c_b])$$

A la fórmula de cálculo bayesiano propuesta agregaremos dos elementos que permiten mejorar los resultados obtenidos. Inicialmente para evitar que la falta de información en la base estadística descarte un posible valor devolviendo probabilidad 0 por tener un factor con éste valor, es que se plantea agregar un parámetro δ de probabilidad mínima.

En segundo término se desea dar un peso relativo a cada heurística para el cálculo bayesiano. Para esto planteamos aplicar un peso relativo γ_i, similar al utilizado en *Combinación con Peso por Heurística*. Debe cumplirse que $\sum_{i=1}^{m} \gamma_i = 1$. La fórmula de cálculo aplicada entonces es 3.7.

$$Pr[c_b|cme_1, \cdots, cme_m] = (Pr[c_b] + \delta) * \prod_{i=1}^{m} (Pr[cme_i|c_b] * \gamma_i + \delta) \tag{3.7}$$

Para el cálculo de los componentes que integran la fórmula 3.7, se considera $\mathcal{I}$ un conjunto de instancias ya clasificadas y conocidas.

El conjunto $\mathcal{I}$ contiene elementos $I = \{cm_1, \cdots, cm_m, c\}$, donde cm_i con $1 \leq i \leq m$ es la clase asignada por la heurística i, o -1 en caso de no obtener resultado, y c la clasificación final obtenida por el sistema.

El cálculo de $Pr[cme_i|c_b]$ y $Pr[c_b]$ se realiza utilizando fórmulas básicas de cálculo probabilístico

$$\frac{\#casos\_favorables}{\#casos\_posibles}$$

sobre el conjunto $\mathcal{I}$ de flujos clasificados.

Para obtener el valor de la expresión 3.7, debemos resolver $Pr[cme_i|c_b]$, lo que estima la probabilidad de obtener una clase resultado c_b, dado que la heurística i presentó un resultado igual cme_i. Se considera entonces:

$$c_{pos}^{c_b} = |\{j \in \mathcal{I} / j.c = c_b\}|$$

$$c_{fav}^{c_b,i} = |\{j \in \mathcal{I} / j.c = c_b \wedge j.cm_i = cme_i\}|$$

Para el tratamiento de valores faltantes (*missing values*), correspondientes a metodologías que no clasifican con éxito y no devuelven resultado, se presentan dos opciones:

- Agregar una clase *sin resultado*, y tratarla como un tipo de tráfico más.
- Omitir el elemento faltante en el cálculo. Las probabilidades resultantes cuando se realiza la omisión son mucho mayores, pero esto ocurre para todas las clases de tráfico analizadas, por lo que se resuelve al normalizar las probabilidades.

En función de pruebas realizadas se encontró que la segunda opción devuelve mejores resultados, en el sentido que en casos de existencia de pocas heurísticas con resultados, se logra igualmente identificar el tráfico.

La ecuación 3.8 presenta el cálculo de $Pr[cme_i|c_b]$ en función de los valores $c_{pos}^{c_b}$ y $c_{fav}^{c_b,i}$ presentados anteriormente.

$$Pr[cme_i|c_b] = \begin{cases} 1 & \text{si } cme_i = \text{-}1 \\ \frac{c_{fav}^{c_b,i}+1}{c_{pos}^{c_b}+1} & \text{si } cme_i = c_b \\ \frac{c_{fav}^{c_b,i}}{c_{pos}^{c_b}+1} & \text{en otro caso} \end{cases} \tag{3.8}$$

El sumar 1 al numerador cuando $cme_i = c_b$ se plantea para considerar el caso de no tener registrados flujos en $\mathcal{I}$ donde genero como condición inicial que la metodología es certera, dando mayor probabilidad al acierto. El sumar 1 en el denominador busca evitar errores (división por 0) cuando no existen instancias referentes a una clase.

El caso particular que debe considerarse es cuando ninguna de las metodologías obtiene resultados, donde la probabilidad de cada clase $Pr[c_b]$ es el único valor considerado. En éste caso debe forzarse un resultado de no clasificado.

El cálculo de $Pr[c_b]$ se realiza en forma similar. La ecuación 3.9 muestra el procedimiento.

$$Pr[c_b] = \frac{ct_{fav}^{c_b}}{ct_{pos}+1} \tag{3.9}$$

donde

$$ct_{pos} = |\mathcal{I}|$$

$$ct_{fav}^{c_b} = |\{j \in \mathcal{I}/j.c = c_b\}|$$

Para obtener el valor $Pr^{Bys}[c_b|cme_1, \cdots, cme_m]$ es necesario normalizar el resultado obtenido, llevando a que la suma de probabilidades para todas las clases c_b sea 1. La ecuación 3.10 presenta la normalización realizada:

$$Pr^{Bys}[c_b|cme_1, \cdots, cme_m] = \frac{Pr[c_b|cme_1, \cdots, cme_m]}{\sum_{k=0}^{n} Pr[c_k|cme_1, \cdots, cme_m]} \tag{3.10}$$

La selección de flujos a incluir en el conjunto $\mathcal{I}$ es un elemento fundamental. La inclusión de elementos que no sean certeros pueden generar desviaciones en

las cálculos bayesianos realizados. Una buena selección de $\mathcal{I}$ asegura la calidad de la clasificación. Además debe buscarse incluir la mayor cantidad de registros para lograr que la clasificación sea representativa del tráfico analizado.

3.5.2. Aplicación de *Clasificación Bayesiana*

La utilización de *Clasificación Bayesiana* se entiende que resuelve problemas de errores sistemáticos en las metodologías aplicadas. Éstos errores representan casos donde las metodologías aisladas confunden tipos de tráfico.

Esta metodología de combinación en función de los registros tomados de la red (conjunto $\mathcal{I}$) permitirá generar parámetros de certeza por mecanismo de clasificación y por clases de tráfico, elementos que no permite la aplicación de la metodología propuesta anteriormente. Para generar correctamente los niveles de certeza de metodologías y clases de tráfico, se debe tener cuidado con la información incluida en $\mathcal{I}$ dado que de la misma depende los resultados.

La aplicación de esta metodología se sugiere en casos donde no se tiene confianza en algunos de los resultados que obtiene algunas de las metodologías aplicadas.

Capítulo 4

Herramienta *trafChar* para la Clasificación de Tráfico

4.1. Introducción

En éste capítulo se presenta la arquitectura y funcionamiento del software desarrollado *trafChar*. Éste implementa mecanismos para combinar los resultados obtenidos a través de diferentes metodologías de clasificación.

Los objetivos del software son computar las diferentes metodologías de clasificación de tráfico y contar con herramientas para operar con los resultados obtenidos en las mismas.

Los requerimientos específicos exigidos son:

- Permitir la inspección de los paquetes para obtener las *características* referidas a los flujos en estudio, así como el análisis de los datos de los diferentes paquetes.
- Permitir la interacción con Bases de Datos que permitan almacenar información intermedia requerida para el procesamiento.
- Contar con herramientas de cálculo estadístico y procedimientos *Support Vector Machines* (SVM) de *Machine Learning* (ML).

Las metodologías de clasificación de tráfico sobre las que se realizó el trabajo de combinación son *Well Known Ports*, Análisis de patrones de *bit strings*, y una

Técnica *Support Vector Machines* (SVM) de ML. Igualmente se propone no perder las generalidades para facilitar la inclusión de otras metodologías diferentes.

En el presente capítulo presentaremos la herramienta *trafChar* de clasificación de tráfico desarrollada.

4.2. Clases de Tráfico

La tabla 4.1 presenta las clases de tráfico utilizadas en las pruebas.

Cuadro 4.1: *Tipos de tráfico de Internet y aplicaciones representativas*

Id.	Clase	Aplicación ejemplo
0	Bulk	ftp
1	Interactive	ssh, telnet
2	Mail	smtp, pop2 y 3, imap
3	Services	dns, ldap, ntp, auth
4	Web	http, https
5	Multimedia	rtsp, Real_media, SIP, H323, Skype
6	P2P	KaaZa, BitTorrent, GnuTella, eDonkey, Napster, DirectConnect
7	Chat	Yahoo, AOL, MSN, IRC

Las clases de tráfico utilizadas por *trafChar* son configurables. La única exigencia es que la respuesta de las metodologías aisladas utilizadas deben devolver el mismo conjunto de clases de tráfico utilizando los mismos identificadores.

4.3. Esquema General de la Solución Propuesta

Para probar los mecanismos de combinación de metodologías de clasificación de tráfico, *Combinación con Peso por Heurística* y *Combinación por Clasificación Bayesiana*, se desarrolló software que permite analizar los resultados de los mismos.

La propuesta es extender el software *open source* Snort [SS08], permitiendo la clasificación del tráfico *onLine*, aplicando algunas de las metodologías presentadas en el capítulo 2.

Snort es un software de *Intrusion Detection System* (IDS), el cual es *open source* y se encuentra ampliamente distribuido en Internet.

La elección del software de IDS Snort, se fundamenta en los siguientes elementos:

- Los objetivos de un software IDS son similares a los buscados para clasificar tráfico.
- Por ser *open source* se cuenta con el código fuente.
- Provee mecanismos para extender sus funcionalidades.
- Tiene implementadas herramientas para el análisis del tráfico.
- Está desarrollado en C, lo que facilita agregar librerías para manejo estadístico y de técnicas SVM.

El conocimiento del Snort en función de desarrollos previamente realizados, se entendió como un elemento adicional facilitando el desarrollo realizado.

La arquitectura del software Snort se presenta en la figura 4.1 (ver [AB05]). Los módulos que componen el software son los siguientes:

- *Decodificador*: carga los paquetes capturados en estructuras de datos identificando los protocolos de red involucrados. Una vez decodificados, toma la información útil (direcciones de red o puertos TCP o UDP).
- *Preprocesador*: Pueden ser vistos como una clase de filtros, los que identifican elementos a ser chequeados en los siguientes módulos. La función de los preprocesadores es encontrar patrones de comportamiento que serán aplicados posteriormente por la *máquina de detección*.
- *Archivo de Reglas*: Es el archivo de texto plano que contiene una lista de reglas las que se encuentran escritas con un metalenguaje definido por Snort. La sintaxis permite especificar protocolos, direcciones y *plug-ins de salida* asociados con los mismos.

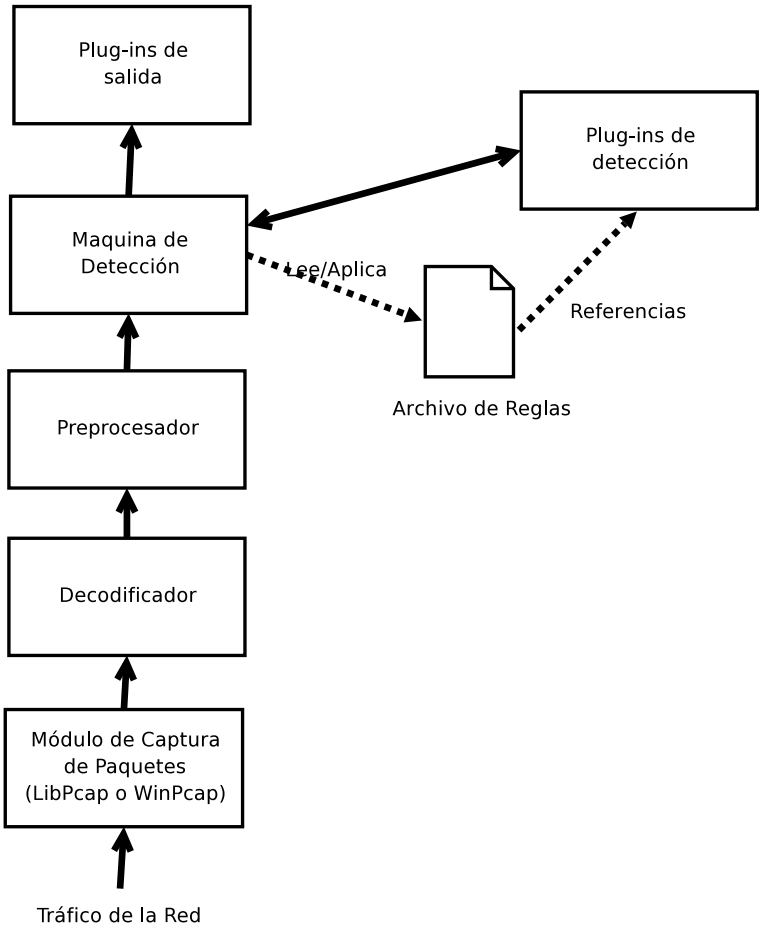


Figura 4.1: Arquitectura del software Snort

- *Plug-ins de Detección*: estos módulos son referenciados en el *archivo de reglas*, y buscan identificar patrones ante la evaluación de una regla.
- *Máquina de Detección*: Utilizando los *plug-ins de detección*, realiza la busqueda de coincidencias en paquetes contra la reglas cargadas en la inicialización del Snort.
- *Plug-ins de salida*: Estos módulos permiten dar formato a las notificaciones (logs, alertas), permitiendo al usuario acceder a las mismas en muchas maneras (consola, archivos externos, base de datos etc).

El software *trafChar* se implementó a través de un preprocesador Snort [How09], dado que en ésta instancia ya se encuentra decodificado el paquete de red, y se cuenta con la información necesaria para realizar los estudios requeridos para la caracterización del tráfico.

Snort provee un preprocesador genérico para extender las funcionalidades del software. El mismo se encuentra como parte del código fuente, en los programas *spp_ template.c* y *spp_ template.h* en el subdirectorio *src/preprocessors*. En base a éstos se desarrolló el preprocesador *spp_ trafchar.c* que se compila conjuntamente con el Snort e implementa *trafChar*.

El lenguaje de programación utilizado para el desarrollo es C y se utilizaron librerías para el acceso a la base de datos MySQL [MyS09] que almacena la información intermedia, GSL GNU Scientific Library para procesamiento estadístico [GSL09], y LIBSVM para implementar SVM [CL01a].

4.3.1. Arquitectura del Preprocesador *trafChar*

La arquitectura del preprocesador presentada indica los módulos de software aplicados por *trafChar* ante la llegada de cada paquete al sistema.

Para la implementación de ambas metodologías de combinación presentadas en el capítulo anterior se utiliza un módulo que realiza los cálculos necesarios para la obtención del resultado. Éste módulo posee dos implementaciones, una para cada una de las metodologías, *Combinación con Peso por Heurística* y *Clasificación Bayesiana*. Resumiendo para ambas metodologías de combinación se ejecuta el mismo código, invocando según el caso el módulo que corresponde.

La figura 4.2 contiene un diagrama de interacción de los distintos módulos del sistema *trafChar*.

Las tareas realizadas por los módulos se presentan a continuación:

- *Calculo Parámetros*, en función de los registros del sistema, genera la información requerida por el análisis Bayesiano, calculando cantidad de casos registrados para los diferentes eventos. Además regenera el modelo SVM para mejorar la clasificación en base a nueva información recabada.

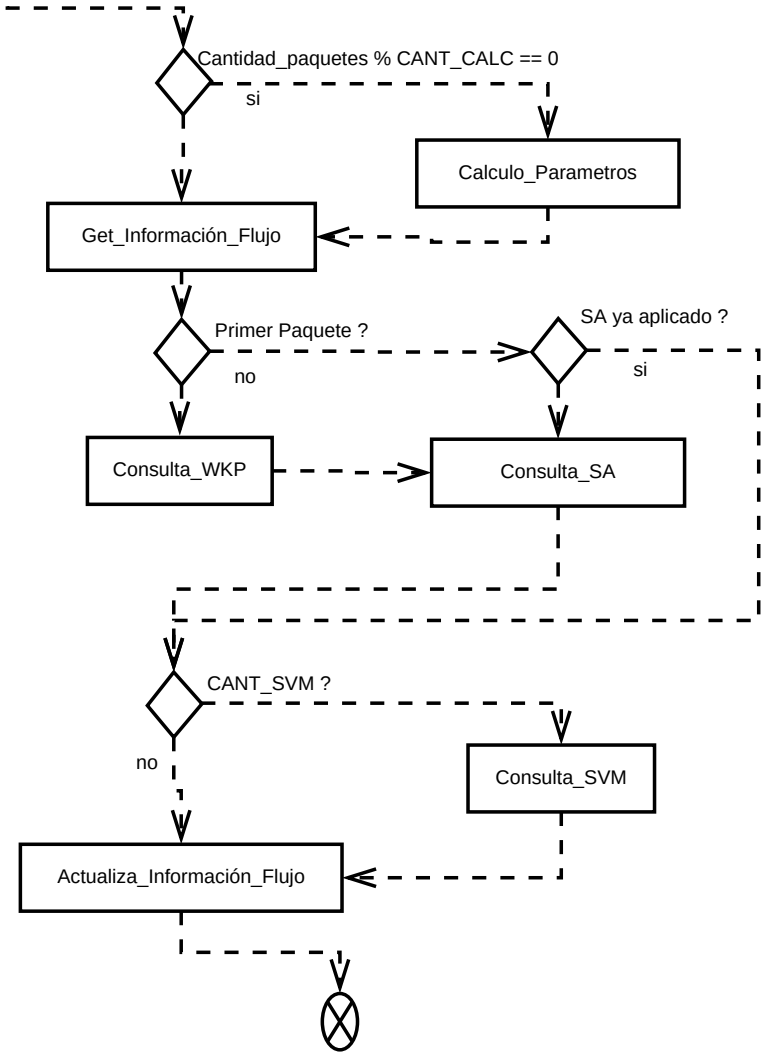


Figura 4.2: Módulos que componen el *trafChar*

- *Get Información Flujo*, dada la 5 tupla que identifica el flujo, *protocolo*, *IP origen*, *IP destino*, *puerto origen* y *puerto destino*, se consulta la base de información para determinar si el flujo ya se encuentra registrado (si se había recibido paquetes con la misma identificación en un período reciente). En caso afirmativo se devuelve la siguiente información cantidad de paquetes ida/vuelta, metodologías de clasificación aplicadas y sus resultados.
- *Consulta WKP*, dado la identificación del flujo, y en presencia del primer

paquete recibido para el mismo, se aplica la heurística de *Well Known Ports*, devolviendo la clase de tráfico asignada y probabilidad para cada clase de tráfico (ver sección 4.5.1).

- *Consulta SA* , dado el payload del paquete recibido, se analiza si existen patrones que permitan su identificación. Devuelve la clase asignada y probabilidad para cada clase de tráfico (ver sección 4.5.2).
- *Consulta SVM*, dada la información de tamaño y tiempo de arribo de cada paquete se aplica la heurística de SVM. Devuelve la clase asignada y probabilidad para cada clase de tráfico (ver sección 4.5.3).
- *Actualiza Información Flujo*, en caso de modificaciones en la información referente al flujo, realiza la invocación del módulo de clasificación por metodologías combinatorias (ver sección 4.4) y actualiza/almacena en la Base de Datos la información obtenida en base al presente paquete.

La información intermedia requerida para el proceso de los diferentes flujos, se encuentra disponible en una Base de Datos MySQL [MyS09]. La selección de MySQL se fundamenta en que presenta las prestaciones requeridas, puede invocarse fácilmente desde Snort y su operación es personalmente conocida. En la misma se almacenan tanto información parcial de los flujo que se presentan activos, así como la información requerida por las metodologías aisladas de clasificación.

Para una mejor comprensión del software, el Apéndice A presenta el *Modelo de Casos de Uso* incluyendo *Diagrama de Secuencia* del sistema y *Contratos de Software* respectivos.

El Apéndice C contiene la estructuras de las tablas MySQL utilizadas por *trafChar*.

4.4. Implementación de las Metodologías de Combinación

En ésta sección se presenta la implementación realizada de las metodologías de combinación de resultados. La aplicación de la metodología de combinación

deseada se realiza a través de un módulo de clasificación el que cuenta con una implementación para *Combinación con Peso por Heurística* y otro para *Combinación Bayesiana*.

4.4.1. Implementación de Combinación con Peso por Heurística

La implementación de *Combinación con Peso por Heurística* se limita a la aplicación de la ecuación 3.3. Las variables involucradas en ésta se encuentran disponibles en la tabla *flujos* (ver C.2) en los campos *probTrfWKP*, *probTrfCNT* y *porbTrfIA* que contiene el resultado de la clasificación por *Well Known Ports*, *Análisis de Patrones* y *Técnica SVM* respectivamente. Los pesos por heurística γ_i se toman de la configuración inicial del preprocesador *trafChar*.

En caso de no contar con resultados de alguna de las metodologías de clasificación, se considera el vector de probabilidades resultado de aplicar la ecuación 3.1 con $k = -1$ y certeza μ_{unclf}.

En nuestro caso la asignación de los valores γ_i se realizó en forma empírica, elemento que puede mejorarse considerando de la información obtenida la certeza de las diferentes heurísticas, elemento propuesto como trabajo futuro en la sección 6.3.

4.4.2. Implementación de Clasificación Bayesiana

La implementación de *Clasificación Bayesiana* presenta mayores complejidades. El primer elemento a considerar es el requerimiento de flujos clasificados para obtener los elementos requeridos por la ecuación 3.7. El conjunto $\mathfrak{I}$ de elementos $I = \{cm_1, \cdots, cm_m, c\}$ requerido para generar la estadística bayesiana se consideró entre los flujos clasificados que cumplen alguna de éstas condiciones:

- Si $c = cm_i$ con $i = SignaturesAnalysis$. Dado que *Análisis de patrones* es considerada una heurística certera.
- Si $|\{cme_i = c_b\}| >= BYS\_ACCEPT$. Si es correctamente clasificado por más de BYS_ACCEPT heurísticas.

Los flujos considerados en las estadísticas bayesianas se almacenan en la tabla MySQL *trafCharStatistic* (ver C.4). A partir de éstos valores cada $CANT\_CALC$ paquetes (ver figura 4.2) se calculan los valores requeridos para el cálculo de probabilidades. Estos cálculos se realizan generando los valores $c_{pos}^{c_b}$, $c_{fav}^{c_b,i}$, ct_{pos} y $ct_{fav}^{c_b}$ para las i metodologías de clasificación y las c_b clases de tráfico permitiendo generar los valores de las ecuaciones 3.8 y 3.9, considerando como conjunto $\mathfrak{I}$ el registrado en la tabla *trafCharStatistic*.

Los datos requeridos para la clasificación se obtienen de la tabla *flujos* (ver C.2) en los campos *probTrfWKP*, *probTrfCNT* y *porbTrfIA*, convirtiendo los mismos a clasificación determinística aplicando la ecuación 3.2.

Los pesos por heurística γ_i, y probabilidad mínima δ se toman de la configuración inicial del preprocesador *trafChar*.

4.5. Implementación de las Metodologías de Clasificación

A continuación se presentan los mecanismos utilizados para la implementación de las diferentes metodologías de clasificación. Éstas se desarrollaron sobre el software *open source* Snort [SS08].

4.5.1. Implementación de Well Known Ports

La heurística *Well Known Ports*, utiliza una base de información, conteniendo para cada par $\{protocolo, puerto\_TCP/UDP\}$ un par $\{k, \mu\}$ indicando clase a la que pertenece el flujo k y nivel de certeza μ del mismo. A partir de ésta información se genera el vector de probabilidades resultado correspondiente según la fórmula 3.1.

El algoritmo aplicado es el presentado en 2.1.

En C.5 se presenta la estructura de la tabla *wellKnownPorts* que almacena la información requerida por la heurística.

Como se indica en la figura 4.2 la heurística es aplicada una vez, cuando se detecta el primer paquete del flujo.

4.5.2. Implementación de Análisis de Patrones

Snort cuenta con una herramienta para el análisis de patrones, en base a la cuál se implementó la presente metodología. Ésta se encuentra en el modulo *Máquina de Detección* presentado en la figura 4.1.

El software Snort carga el archivo de reglas en una estructura de dos dimensiones. La misma se presenta en la figura 4.3. A cada elemento *Chain Header* se le asocia una lista conteniendo *Chain Options*, que asignan nuevas condiciones a las presentadas en el *Header*. Esto permite agilizar el funcionamiento del la búsqueda ya que solamente se realizará en las entradas que cumplan las condiciones exigidas en *Chain Header*. Para cada regla existente en el archivo de reglas se crea un *Chain Options* conectado al *Chain Header* correspondiente.

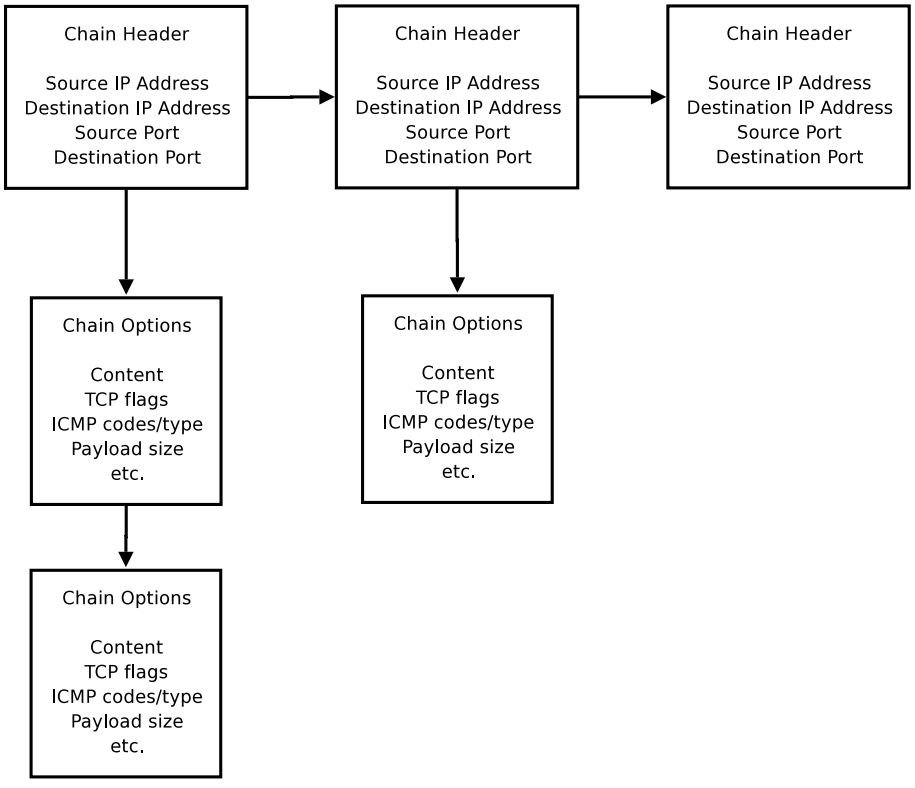


Figura 4.3: Lista para almacenar reglas en Snort

Recursivamente se busca en la lista de reglas para cada paquete recibido,

realizando la búsqueda en ambas direcciones. Para la primer regla que encuentre una coincidencia se activa la acción especificada en la definición de la misma.

Adicionalmente Snort provee para la definición de reglas la opción *tag*, que activa la regla no solamente para el paquete inspeccionado sino para otros siguientes [Sno09]. La activación se puede realizar por cantidad de paquetes, segundos o cantidad de bytes. En la implementación de Snort se puede condicionar para todos los paquetes que incluyan dirección IP origen, dirección IP destino o ambas. Ésta opción es muy útil en protocolos que inician otro flujo. Un ejemplo es el protocolo *File Transfer Protocol* (FTP) que para la transmisión de un archivo establece otro flujo.

En el desarrollo realizado se extendió la opción *tag*, permitiendo definir el protocolo de capa transporte a buscar en nuestro caso protocolo TCP o UDP, y no exclusivamente las direcciones IP involucradas en el flujo a analizar.

Como puede observarse en la figura 4.1 la invocación de los módulos preprocesadores se realiza antes del de la máquina de detección, donde se investiga la búsqueda de conincidencias de reglas del Snort. Por ésto debió resolverse el pasaje de información entre los módulos, realizándose a través de una estructura a la cual el preprocesador y la máquina de detección tienen acceso y registran el hallazgo de coincidencias de las mismas. Se entendió que la presente solución es más eficiente que invocar dos veces la máquina de detección, y el único elemento nocivo que presenta es que el registro de coincidencias será procesado en el siguiente paquete recibido para el flujo.

Por lo tanto en la solución propuesta las *signatures* se ingresan en el archivo de reglas de Snort y con el metalenguaje definido por Snort con tal fin.

Para registrar el par $\{k, \mu\}$ con la clase perteneciente k y el valor de certeza μ en caso de encontrar coincidencia (ver sección 3.1) se utiliza el campo *msg* donde se plantean los valores separados por ",".

Ejemplo:

```
alert tcp any 22 -> any any
(msg: "1,0.001"; content: "SSH-2.0-OpenSSH";
nocase; offset:0; sid:1100001;)
```

Ésta regla detecta los paquetes transmitidos por TCP, cuyo puerto destino es 22 y contienen la cadena *SSH-2.0-OpenSSH*, en el offset 0 (inicio de los datos), devolviendo el valor clase de tráfico 1 (*Interactive*) y $\mu = 0{,}001$, donde aplicando la fórmula 3.1 se genera el vector de probabilidades.

Los patrones utilizados provienen de artículos donde fundamentalmente se encuentran *signatures* para identificar tráfico P2P y además de páginas referentes a productos para identificar tráfico [HiP09].

4.5.3. Implementación de la Técnica Support Vector Machine de ML

En el trabajo preliminar [1] realizado, se investigó sobre la aplicación de *Classification and Regression Trees* (CART) y *Support Vector Machines* (SVM) para la clasificación de tráfico, obteniendo resultados similares pero levemente superiores para SVM. Finalmente se implementó la técnicas de ML supervisada SVM presentadas en la sección 2.4.2.3.

Un elemento que facilitó el desarrollo es la existencia de LIBSVM [CL01b], una implementación *open source* de SVM. La implementación se realizó invocando funciones que calculan SVM contenidas en la librería.

Los atributos considerados para clasificar los flujos son 13. Estos son:

- *Relación entre paquetes de upstream y downstream* (1).
- *Media del tamaño de paquete (total, upstream y downstream)* (3).
- *Varianza del tamaño de paquete (total, upstream y downstream)* (3).
- *Media del tiempo entre paquetes (total, upstream y downstream)* (3).
- *Varianza del tiempo entre paquetes (total, upstream y downstream)* (3).

tomándose como referencia $CANT\_SVM$ paquetes (ver tabla 5.1) para el cálculo.

[1] *Network Traffic Characterization using, DTW, CART, SVM and Neural Network* (F. Rodriguez Teja, G. Tejera y F. Benavides) Instituto de Computación - Facultad de Ingeniería (Universidad de la República)

Los parámetros utilizados para la configuración del software SVM se presentan en la tabla 4.2.

Cuadro 4.2: *Parámetros utilizados por Metodología SVM*

Parámetro	Valor	Observaciones
SVM Type	C-SVM	C-Support Vector Classification (C-SVC)
Kernel	Radial Basis Function (rbf)	$K(x_i, x_j) = exp(-\gamma \parallel x_i - x_j \parallel^2)$
Parámetro γ Kernel	0.0769231	se obtiene a partir de $\frac{1}{cantidad\_atributos} = \frac{1}{13}$
metodología multi-class	one-against-one	

El algoritmo utilizado para obtener la información requerida y el cálculo de SVM se presenta en 4.1.

La funciones utilizadas se explican a continuación:

- *registro_info_flujo(pkt)*, registra los datos de paquete recibido, indicando tiempo de llegada del paquete, tamaño y dirección del mismo (*upstream* o *downstream*)

- *calculo_atributos(DATO_FLOW)* procesa los datos registrados para el flujo y devuelve $ATRIB$ el valor de los 13 atributos considerados para la clasificación. Para el procesamiento de los datos registrados se utilizó el paquete Scientific Library (GSL) [GSL09] que resuelve procesamiento estadístico (medias y varianzas).

- *clasifico(ATRIB)* invoca procedimientos de LIBSVM y devuelve un vector con la clasificación resultado.

- *registro_info_SVM(RESULT,ATRIB)*, registra los datos del flujo procesado (valores obtenidos de atributos del mismo $ATRIB$ y clase de tráfico

Algoritmo 4.1 Algoritmo *Implementación SVM*

Require: *pkt*, $DATO\_FLOW$ ▷ $DATO\_FLOW$ contiene datos del flujo
Ensure: $clase\_trafico$ o $noClasificado$

```
if DATO_FLOW.cant <= CANT_SVM then
    ▷ Si cantidad paquetes del flujo <CANT_SVM
    ▷ Registro información del flujo
    registro_info_flujo(pkt)
end if
if DATO_FLOW.cant = CANT_SVM then
    ▷ Si cantidad paquetes del flujo igual CANT_SVM
    ▷ Clasifico flujo y registro información
    ATRIB ← calculo_atributos(DATO_FLOW)
    RESULT ← clasifico(ATRIB)
    if registro válido para entrenamiento then
        registro_info_SVM(RESULT, ATRIB)
    end if
    return RESULT
end if
return noClasificado
```

$RESULT$), para ser utilizado en próxima instancia de entrenamiento de SVM.

Ésta metodología devuelve siempre un resultado probabilístico, por la forma de invocación de las librerías LIBSVM. El mismo no contiene la componente probabilidad de no clasificado p_{i0}, por lo que para normalizar se agrega la misma con valor 0.

Otro elemento importante de la metodología SVM de ML es la obtención de un conjunto de flujos clasificados de distintas clases que permitan entrenar al módulo SVM (dado que es una metodología supervisada). Con éste fin, si la clasificación realizada tiene un nivel aceptable de certeza, entonces los valores de sus 13 atributos son almacenados junto con la clase de tráfico asignada.

Cada $CANT\_CALC$ paquetes analizados se generan nuevamente la configuración del SVM, compuesto por el modelo (archivo *model*) y la escala de los

atributos (archivo *scale*).

En [CL01b] se puede obtener más información sobre la implementación de LIBSVM utilizada.

Capítulo 5

Resultados Obtenidos

5.1. Introducción

En éste capítulo se presentan los resultados obtenidos.
Las pruebas se realizaron sobre las siguiente trazas de tráfico:

1. Para la depuración del software (*debug*) se utilizaron pruebas sobre una red hogareña conectada a un servicio *Asymmetric Digital Subscriber Line* (ADSL) de Internet. El objetivo fue corregir el funcionamiento de *trafChar*, sobre un tráfico conocido.

2. Traza de servicios ADSL de Internet, conteniendo 236293 flujos con 36058 TCP y 200235 UDP, de 1044 segundos de duración.

3. Traza de servicios ADSL de Internet, conteniendo 386885 flujos con 66258 TCP y 320627 UDP, de 2135 segundos de duración.

Los resultados obtenidos para las distintas trazas son similares. Los valores presentados en éste capítulo corresponden a la tercera traza, dado que contiene más flujos y mayor duración.

5.2. Información General

En ésta sección se presentan los parámetros utilizados para la configuración del *trafChar*, y algunos resultados generales obtenidos.

5.2.1. Parámetros utilizados en las pruebas realizadas

La tabla 5.1 presenta los parámetros utilizados en la configuración del software *trafChar* para las pruebas realizadas:

Cuadro 5.1: *Parámetros utilizados por trafChar.*

Parámetro	Valor	Observaciones
m	3	cantidad de metodologías aplicadas
n	8	cantidad de clases de tráfico
μ	0,001	nivel de certeza (ecuación 3.1)
μ_{unclf}	0,75	nivel certeza en flujos no clasificados
δ	0,01	probabilidad mínima (ecuación 3.7)
γ_i para WKP γ_i para SA γ_i para SVM	0,4 0,5 0,1	peso de heurísticas (ecuaciones 3.3 y3.7)
BYS_ACCEPT	2	cantidad clasificados correctamente para selección Clasificación Bayesiana
$CANT\_SVM$	10	paquete en el que se realiza cálculo para SVM

Los parámetros utilizados se basan en experiencias empíricas y el *tuning* realizado. Se entiende que la generación de procedimientos que permitan obtener mejores configuraciones del sistema permitirán lograr soluciones con mayor nivel de certeza. Entre estos se propone como trabajo futuro la determinación de los valores γ_i por medio de cálculos de valores estadísticos tomados en la certeza de los resultados obtenidos por heurística.

5.2.2. Información General de los Resultados Obtenidos

En ésta sección se presenta información general en relación con el tamaño de flujos procesados medido en KBytes y paquetes, y el tiempo requerido por las diferentes metodologías para la clasificación.

5.2.2.1. Tamaño de Flujos Procesados

El tamaño de los flujos procesados es información adicional importante para el sistema de clasificación de tráfico. Dependiendo de la metodología de clasificación, se obtendrán resultados con la visualización de un número de paquetes o bytes del flujo. Conociendo el comportamiento de los flujos es posible estimar la probabilidad de éxito de las diferentes metodologías.

Un elemento adicional es la función de distribución presentada por cantidad de bytes y paquetes de los flujos. Éste fenómeno es conocido en la academia como la presencia de flujos "elefantes" y "ratones" (*the elephant and mice phenomenon*) [EMA+07].

Él fenómeno de flujos "elefantes" y "ratones" plantea que la mayoría de los flujos de Internet son flujos "ratones" de tamaño pequeño y sólo una pequeña fracción son flujos "elefante" grandes. La clasificación de flujos "elefantes" genera que un pequeño porcentaje del total de flujos permita clasificar un porcentaje importante de paquetes y bytes transmitidos en el Internet.

La figura 5.1 compara la cantidad de paquetes por flujo y cantidad de KBytes por flujo y la *Cumulative Distribution Function* (CDF) para flujos de protocolo UDP, TCP y del total de los flujos, donde puede apreciarse claramente fenómeno descripto anteriormente.

Del gráfico se desprende que el comportamiento de TCP y UDP no es igual. Los flujos UDP presentan más claramente la presencia de flujos "ratones", dado que el 96,67 % de los flujos tienen un tamaño menor a 1 KByte y el 94,12 % contiene menos de 4 paquetes. En TCP ya no se cumple tan directamente, donde un 44,26 % de los flujos son de tamaño menor a 1 KBytes. Como la cantidad de flujos UDP es cinco veces mayor, ésto genera que el comportamiento del total de los flujos sea similar al comportamiento de UDP.

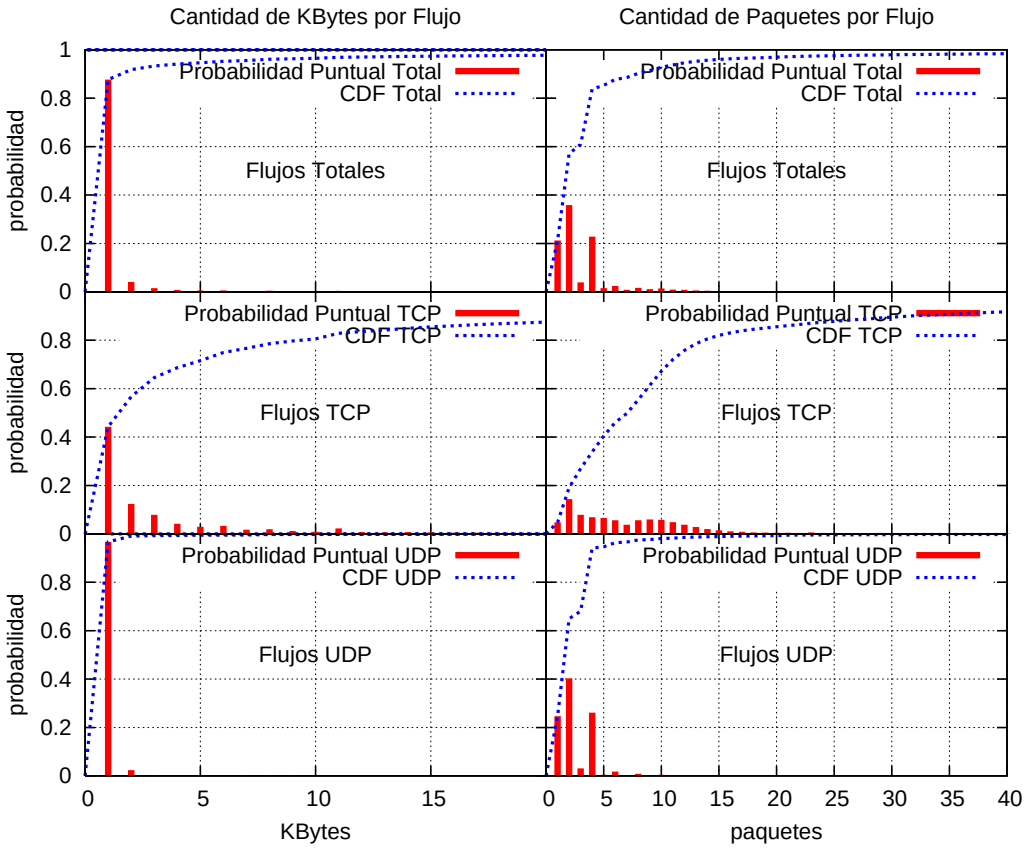


Figura 5.1: Cantidad de Paquetes y KBytes por Flujo

De las metodologías de clasificación aplicadas en las pruebas, la *Técnica SVM* requiere la visualización de $CANT\_SVM$ paquetes para obtener resultados. Lo expresado anteriormente indica que es una limitación importante para la metodología, dado que de la información obtenida se desprende que en condiciones para ser clasificados se encuentra el 2,39 % de los flujos UDP, el 38,65 % de los flujos TCP que corresponden al 8,6 % del total de los flujos.

5.2.2.2. Tiempo de Clasificación de Flujos Procesados

Para la aplicación *OnLine* de las diferentes heurísticas, se debe estudiar el tiempo requerido por éstas para clasificar. Es importante destacar que el método cooperativo presentado no mejora el tiempo de clasificación de las heurísticas ais-

ladas. La mejora se produce cuando dos heurísticas devuelvan el mismo resultado, donde el tiempo de clasificación será el menor de ambos.

La figura 5.2 presenta los tiempo requeridos para las metodologías de clasificación utilizadas en las pruebas, presentando su distribución empírica CDF. El tiempo de clasificación se computa calculando la diferencia de tiempos entre la llegada del paquete que permitió la clasificación y el primer paquete recibido del flujo.

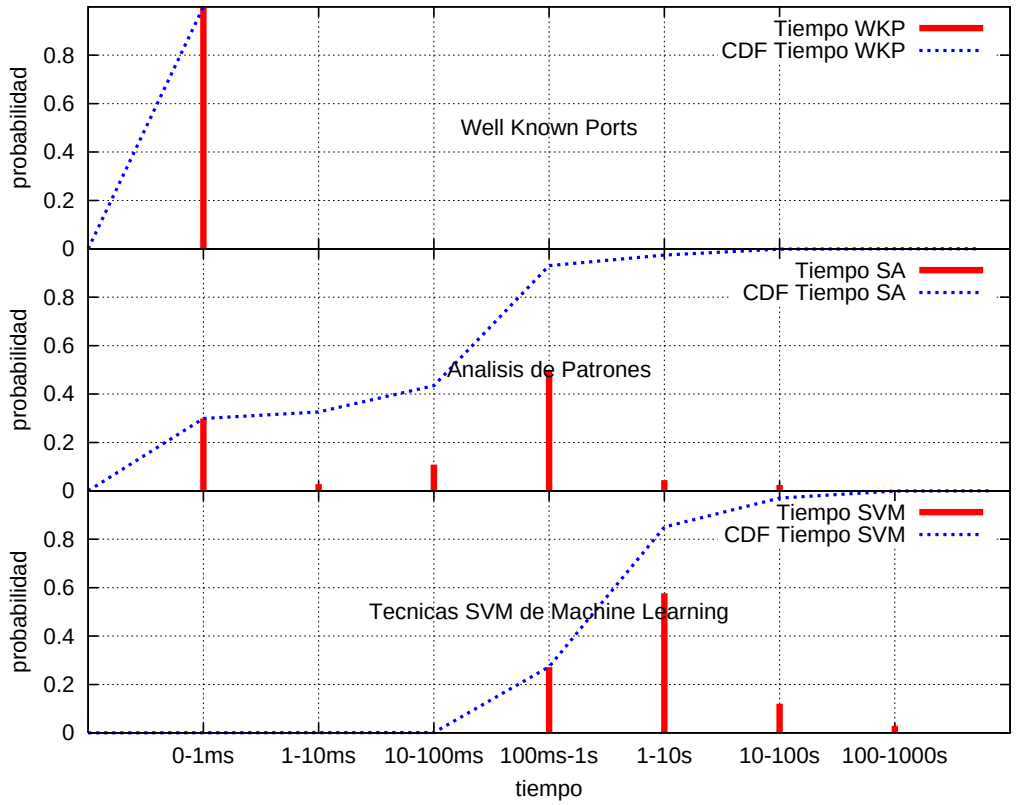


Figura 5.2: Tiempo de Clasificación por Heurística

Para el estudio se consideraron el total de flujos clasificados por cada heurística, 87924 flujos clasificados por *Well Known Ports*, 73683 clasificados por *Análisis de Patrones* y 13216 por *Técnicas SVM*. Para el estudio se consideran intervalos

de tiempo $0-1ms$, $1-10ms$, $10-100ms$, $100ms-1s$, $1-10s$, $10-100s$, $100-1000s$ y se contaron los casos registrados por intervalo.

El análisis de los resultados obtenidos en referencia al tiempo de clasificación indica:

- *Well Known Ports* clasifica en el primer paquete visualizado del flujo. En este caso el resultado refleja un tiempo igual y constante para todos los flujos menor a $1ms$.
- *Análisis de Patrones* clasifica siempre en un tiempo menor a $10seg.$. El 29,86 % se clasifica en menos de $1ms$, el 43,45 % de los flujos en menos de $100ms$ y el 93,01 % de los flujos en menos de $1seg$. La clasificación se realiza cuando se encuentra un patrón que identifique alguna clase de tráfico. En general se busca que los patrones buscados aparezcan en los primeros paquetes.
- *Técnicas SVM de Machine Learning* requiere más tiempo para la clasificación que las restantes heurísticas, lo que se justifica en el requerimiento de visualizar $CANT\_SVM$ paquetes para realizar la clasificación. Su cota superior es $10seg$, no clasifica en menos de $100ms$, y en el intervalo $100ms$ - $10seg.$ clasifica un 85 % del tráfico.

5.3. Análisis de los Resultados Obtenidos

Analizaremos los mismos en función de los siguientes elementos:

- Análisis del volumen de tráfico clasificado.
- Comparación de las metodologías de combinación respecto a las metodologías aisladas.
- Comparación de las metodologías de combinación.

Para los gráficos presentados en las futuras secciones, los datos identificados con WKP corresponden a resultados de la metodología *Well Known Ports*, los identificados con SA pertenecen a *Análisis de Patrones* y los presentados como SVM corresponden a *Técnicas SVM*.

5.3.1. Análisis del Volumen de Tráfico Clasificado

La figura 5.3 muestra el comparativo entre cantidad de flujos y bytes clasificados. Como puede observarse los flujos clasificados son para *Combinación con Peso por Heurística* un 27,24 % y para *Clasificación Bayesiana* el 25,24 %, mientras que la cantidad de bytes clasificados son 62,52 % y 62,32 % respectivamente del tráfico total. Ambas metodologías de combinación de resultados clasificaron cantidad similar de flujos y bytes, obteniendo un resultado levemente superior en *Combinación con Peso por Heurística*.

Los resultados presentados en el párrafo anterior reafirman el fenómeno de flujos "elefantes" y "ratones", dado que la clasificación únicamente de un 27 % de flujos logra clasificar el 62 % del total de bytes transimitidos.

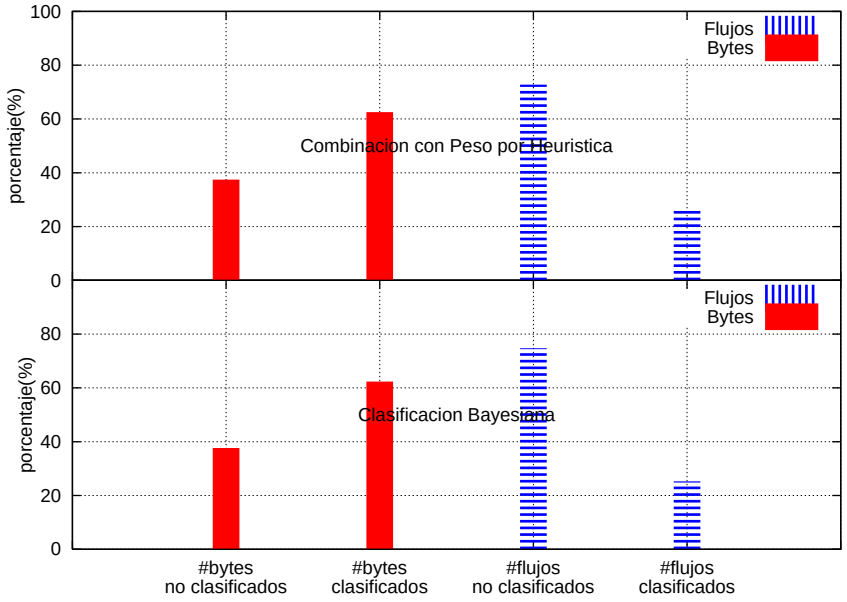


Figura 5.3: Comparativo de porcentaje de Flujos y Bytes clasificados

El aumento del volumen de tráfico clasificado (62 %) puede realizarse incluyendo metodologías de clasificación con resultados mayores del 62 % del tráfico, entonces los resultados obtenidos igualarán o superarán éste valor.

5.3.2. Comparación de las Metodologías de Combinación Respecto a las Metodologías Aisladas

La figura 5.4 muestra para las dos metodologías de combinación *Combinación con Peso por Heurística* y *Clasificación Bayesiana*, la comparación de sus resultados con los de las heurísticas aplicadas en forma aislada.

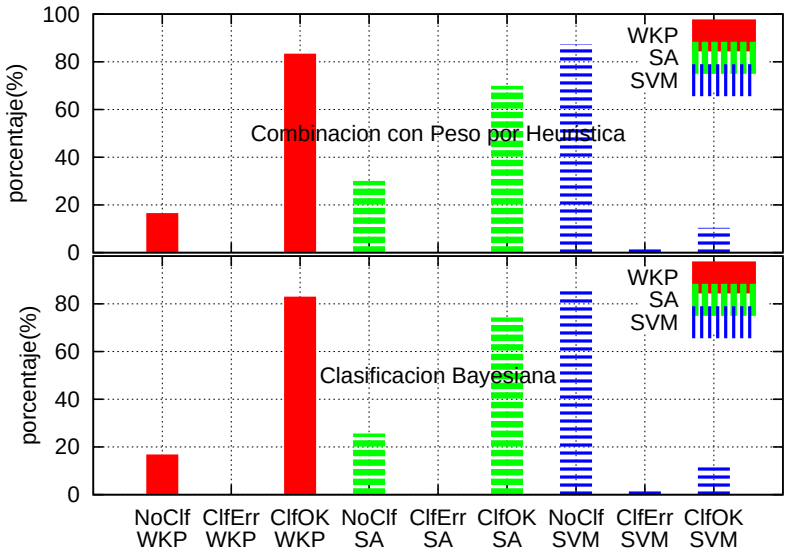


Figura 5.4: Resultados Obtenidos por Heurística

El gráfico presenta para los flujos clasificados por las metodologías de combinación, considerados como el 100 % de flujos, los resultados obtenidos por las metodologías *Well Known Ports*, *Análisis de Patrones* y *Técnicas SVM*, desglosados en: *no clasificados*, *clasificados correctamente* y *clasificados con error*.

El análisis de los resultados es el siguiente:

- Los elementos *no clasificados* (NoClf) por la metodología aislada, corresponden a la mejora obtenida por la aplicación de metodologías de combinación, dado que se obtuvo resultado en éstas mientras que si se hubiese aplicado únicamente la metodología aislada no se hubiese obtenido resultados.

- Los elementos *clasificados con error* (ClfErr) por la metodología aislada, corresponden a rectificaciones obtenidas por la aplicación de metodologías de combinación. Estas corresponden en casos de obtener clasificaciones discordantes por la aplicación de las metodologías aisladas. Por lo tanto corresponden a flujos donde se ha corregido la clasificación realizada por la metodología aislada.

- Los elementos *clasificados correctamente* (ClfOK) por la metodología aislada, corresponden a los resultados correctos que se obtendría aplicando únicamente la metodología asilada.

Los resultados obtenidos por ambas metodologías de combinación son similares. Como puede observarse, el aumento en la clasificación obtenida es para *Well Known Ports* un 16 % en ambas metodologías de combinación. *Análisis de patrones* mejora en un 30,08 % para *Combinación con Peso por Heurística* y 25,61 % en *Clasificación Bayesiana* y las *Técnicas de SVM* obtienen una mejora del 86 % en las dos metodologías.

Los elementos *clasificados con error* no son representativos para las heurísticas *Well Known Ports* y *Análisis de patrones*, mientras que para la metodología de *Técnicas de SVM* se encuentran en 2,12 % para *Combinación con Peso por Heurística* y 1,57 % en *Clasificación Bayesiana*.

De los elementos presentados se desprende que la heurística más utilizada es *Well Known Ports*, las más confiables son *Análisis de patrones* y *Well Known Ports*, y la que presenta peores resultados es la de *Técnicas de SVM*.

Un análisis importante es considerar el comportamiento de las diferente metodologías aisladas discriminado por tipo de tráfico. El estudio es equivalente al realizado anteriormente pero considerando resultados según el tipo de tráfico.

En la figura 5.5 se presentan los resultados separados por clase de tráfico. El elemento importante del gráfico es la confirmación de la disparidad en el comportamiento de las diferentes metodologías aisladas para las diferentes clases de tráfico, elemento que justifica la utilización de metodologías de combinación de resultados. Los resultados obtenidos para ambas metodologías de combinación son similares. Del mismo se desprenden los siguientes elementos:

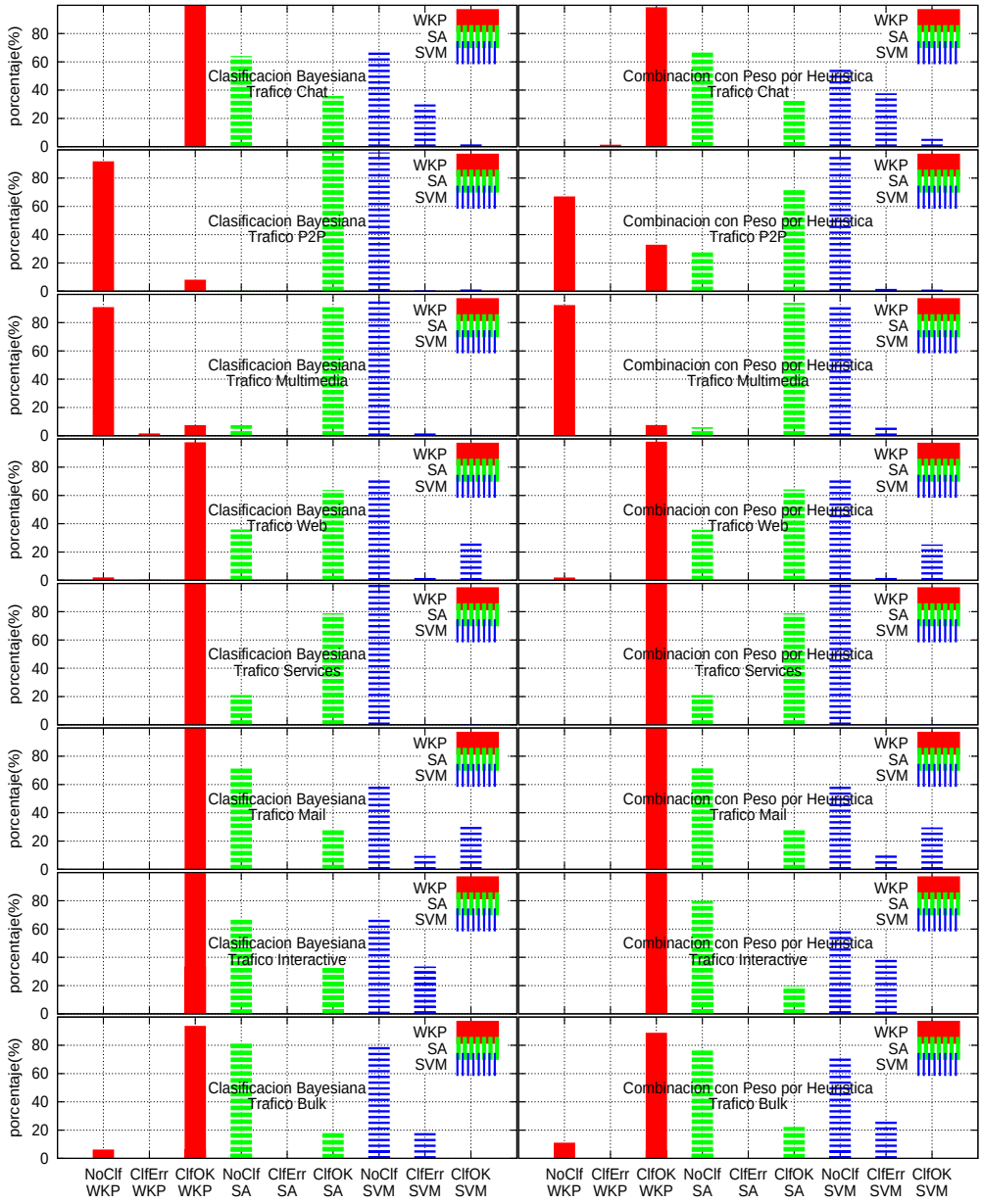


Figura 5.5: Resultado Obtenido por Tipo de Tráfico

- Como puede observarse, la metodología *Well Known Ports* responde en forma aceptable en todos las clases de tráfico excepto en *P2P* y *Multimedia*.

- La *Técnica SVM* para la clase de tráfico *Servicios* no presenta ningún resultado. La explicación a éste hecho se basa en que utiliza flujos de muy corta duración impidiendo llegar al paquete donde se aplica la metodología.

- Los únicos casos en los que la metodología aislada obtuvo el mismo resultado que una metodologías de combinación es para *Well Known Ports* y los tipos de tráfico *Interactive*, *Mail* y *Services*. Para el resto de los casos siempre las metodologías de combinación mejoran los resultados y para ciertos casos en porcentajes importantes.

- En relación a los *clasificados con error* en las metodologías aisladas respecto a las combinadas, debe considerarse primero que los parámetros γ_i (peso de cada metodología) es determinante en el mismo. El peso mínimo asignado a la *Técnica SVM* justifica la gran tasa de error encontrada. En segundo lugar como éstos errores se generan cuando se logra clasificar por más de una metodología aislada y los resultados son diferentes, estos errores pueden considerarse como correcciones realizadas a las metodologías de clasificación aisladas. Igualmente es importante aclarar que como se presenta en el gráfico 5.4, en términos generales para ningún caso presenta errores significativos.

5.3.3. Comparación de las Metodologías de Combinación

Inicialmente estudiaremos la cantidad de flujos clasificados por cada una de las metodologías de combinación de heurísticas. Los resultados son, *Combinación con Peso por Heurística* clasificó 105391 flujos, mientras que *Clasificación Bayesiana* 97657. Los resultados se presentan en la tabla 5.2.

Cuadro 5.2: *Cantidad y Porcentaje de flujos clasificados por metodologías de combinación*

Metodología de Combinación	Cant. clasificados	Cant. No clasificados	% clasificado	% No clasificado
Combinación con Peso por Heurìstica	105391	281494	27,25 %	72,75 %
Clasificación Bayesiana	97657	289228	25,25 %	74,75 %

Un elemento destacable es que el conjunto de los flujos clasificados por la metodología de *Clasificación Bayesiana* está contenido en el conjunto de flujos clasificados por *Combinación con Peso por Heurística*. La diferencia entre las dos metodologías son 7734 flujos, 1,99 % no clasificados por la metodología de *Clasificación Bayesiana*. Puede concluirse entonces que la metodología *Combinación con Peso por Heurística* es menos exigente en los requerimientos para resolver la clasificación, elemento que puede además deducirse del análisis de la metodología en si misma.

Las coincidencias entre los resultados obtenidos por ambas metodologías de combinación, presentadas en la tabla 5.3 contiene la cantidad y porcentaje de coincidencias en los resultados. De las 7966 diferencias deben considerarse los 7734 flujos para los cuales *Clasificación Bayesiana* no obtuvo resultado. Esto indica que de 386885 flujos procesados, solamente 232 un 0,06 % presentan resultados diferentes para ambas metodologías de combinación. Estos valores se calcularon sobre la totalidad de los flujos procesados, clasificados o no, por lo que los clasificados por *Combinación con Peso por Heurística* y no clasificados por *Clasificación Bayesiana* se consideran como diferencias.

Cuadro 5.3: *Comparación de Resultados de Metodologías de Combinación*

	Cantidad	Porcentaje (%)
coincidencias	378919	97,95
diferencias	7966	2,05

Analizaremos a continuación los resultados obtenidos para ciertos flujos en particular. Presentamos tres flujos, *flujo 1* puede considerarse como un flujo *P2P* (clase 6) dado que fue identificado por la metodología de *Analisis de Patrones*, *flujo 2* del cual se tiene solamente la clasificación por *Well Known Ports* con resultado *P2P* (clase 6) y *flujo 3* puede considerarse como un flujo *Mail* (clase 2) dado que fue identificado por la metodología de *Well Known Ports*. Los resultados presentados por las metodologías aisladas, *Well Known Ports*, *Analisis de Patrones* y *Técnicas SVM*, así como los resultados de las metodologías de combinación *Combinación con Peso por Heurística* y *Clasificación Bayesiana* se presentan en la tabla 5.4.

Las ecuaciones utilizada para la realización de los cálculos de *Combinación con Peso por Heurística* y *Clasificación Bayesiana* son 3.3 y 3.7 respectivamente. Los valores considerados para la aplicación de las fórmulas, son los presentados en la sección 5.2.1 ($\gamma_{WKP} = 0{,}4, \gamma_{SA} = 0{,}5, \gamma_{SVM} = 0{,}1$ y $\delta = 0{,}01$).

En el caso del *flujo 1* el error en *Clasificación Bayesiana* se debe a la baja probabilidad existente en $Pr[cme_{SVM} = 4|c_b = 6]$ y además por la existencia de muchos flujos *Web* clase 4 que genera una probabilidad $Pr[c_b = 4]$ alta y pocos *P2P* dando $Pr[c_b = 6]$ baja. De haber clasificado la *Técnica SVM* como *Chat* (clase 7) se habría obtenido el resultado deseado, dado que $Pr[c_b = 7]$ y $Pr[cme_{SVM} = 7|c_b = 6]$ son bajos, destacando el resultado obtenido por *Análisis de Patrones*.

En el *flujo 2* la no clasificación de *Clasificación Bayesiana* se debe a la existencia en el espacio de cálculo de los valores bayesianos (conjunto $\mathcal{I}$ en sección 3.5) de varios flujos clasificados por *Well Known Port* como *P2P* (clase 6), pero donde se les asignó como clase de tráfico -1 lo que genera un valor alto de $Pr[cme_{WKP} = 6|c_b = -1]$. Esto se complementa además con el hecho de

Cuadro 5.4: *Ejemplos de Flujos Clasificados*

Flujo 1										
Resultado obtenido por cada Metodología										
	Result.	-1	0	1	2	3	4	5	6	7
WKP	-1	0.250	0.094	0.094	0.094	0.094	0.094	0.094	0.094	0.094
SA	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.999	0.000
SVM	4	0.000	0.000	0.000	0.000	0.004	0.680	0.052	0.118	0.050
Resultado $\mathbf{Pr^{Heur}}$										
$\mathbf{Pr^{Heur}}$	6	0.1	0.04	0.04	0.04	0.04	0.11	0.04	**0.55**	0.04
Resultado $\mathbf{Pr^{Bys}}$										
$\mathbf{Pr^{Bys}}$	4	0.113	0.002	0.002	0.015	0.010	**0.687**	0.008	0.161	0.002
Flujo 2										
Resultado obtenido por cada Metodología										
	Result.	-1	0	1	2	3	4	5	6	7
WKP	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.999	0.000
SA	-1	0.250	0.094	0.094	0.094	0.094	0.094	0.094	0.094	0.094
SVM	-1	0.250	0.094	0.094	0.094	0.094	0.094	0.094	0.094	0.094
Resultado $\mathbf{Pr^{Heur}}$										
$\mathbf{Pr^{Heur}}$	6	0.15	0.06	0.06	0.06	0.06	0.06	0.06	**0.46**	0.06
Resultado $\mathbf{Pr^{Bys}}$										
$\mathbf{Pr^{Bys}}$	-1	**0.593**	0.006	0.006	0.025	0.020	0.272	0.007	0.063	0.008
Flujo 3										
Resultado obtenido por cada Metodología										
	Result.	-1	0	1	2	3	4	5	6	7
WKP	2	0.000	0.000	0.000	0.999	0.000	0.000	0.000	0.000	0.000
SA	-1	0.250	0.094	0.094	0.094	0.094	0.094	0.094	0.094	0.094
SVM	3	0.000	0.000	0.000	0.000	0.347	0.033	0.125	0.159	0.336
Resultado $\mathbf{Pr^{Heur}}$										
$\mathbf{Pr^{Heur}}$	2	0.125	0.047	0.047	**0.447**	0.082	0.050	0.060	0.063	0.081
Resultado $\mathbf{Pr^{Bys}}$										
$\mathbf{Pr^{Bys}}$	2	0.041	0.001	0.001	**0.882**	0.024	0.038	0.002	0.004	0.007

que $Pr[c_b = -1]$ presenta un valor alto. Éste resultado no presenta un error en la clasificación, sino que incide sobre el volumen de tráfico clasificado para la metodología *Clasificación Bayesiana*.

El *flujo 3* permite verificar propiedades importantes de los clasificadores bayesianos. En el mismo puede verse que como $Pr[cme_{SVM} = 3|c_b = b]$ es muy alto para c_b *Mail*, *Services*, *Multimedia*, *P2P* y *Chat* (clases 2, 3, 5, 6 y 7), lo que demuestra que en muchos flujos clasificados como $cme_{SVM} = 3$ tiene resultados diferentes en c_b. Esto genera que *Clasificación Bayesiana* retorne una probabilidad $\mathbf{Pr^{Bys}}$ de 0,882 en *Mail* (clase 2), mientras que *Combinación con Peso por Heurística* $\mathbf{Pr^{Heur}}$ devuelve 0,447. La interpretación del presente ejemplo indica que las *Técnicas SVM* cometen errores sistemáticos que confunden éstas clases de tráfico con el tráfico tipo *Services* (clase 3).

Extendiendo el concepto presentado, supóngase que exista una metodología que comete errores sistemáticos para la clasificación donde contesta muchas veces que cierto tipo de flujos pertenece a la clase de tráfico a mientras que es b, pero se obtienen resultados correctos $c_b = b$, por lo que $Pr[cme_i = a|c_b = b]$ será alto. Entonces a pesar de la incorrecta clasificación realizada por la metodología aislada, la *Clasificación Bayesiana* devolverá un valor correcto.

Resumiendo, la cantidad de resultados obtenidos por ambas metodologías son similares, levemente superior *Combinación con Peso por Heurística*, mientras que *Clasificación Bayesiana* presenta la ventaja de adaptarse a errores sistemáticos de las metodologías aisladas aplicadas.

En el apéndice B se dispone de un estudio sobre los resultados de la metodología *Clasificación Bayesiana*, con ejemplos de su aplicación, basado en resultados tomados de las pruebas realizadas.

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Introducción

En el presente capítulo se desarrollan las conclusiones del estudio realizado y se proponen trabajos futuros.

6.2. Conclusiones

Las metodologías propuestas para la combinación, *Combinación con Peso por Heurística* y *Clasificación Bayesiana* mejoran el volumen de tráfico clasificado. Puede verse que la cota inferior del tráfico clasificado es la metodología aislada que obtuvo mayor clasificación, dado que para la clasificación por los métodos combinados es suficiente haber obtenido resultados en una metodología.

En el caso concreto implementado, con metodologías *Well Known Ports*, *Análisis de Patrones* y *Técnicas SVM*, la mejora obtenida medida sobre la cantidad de flujos, superó en un 20 % comparado con la aplicación de cualquiera de las tres en forma aislada.

Otro elemento importante en referencia con las metodologías de combinación, es la capacidad de complementarse para mejorar los resultados. Como ejemplo, las gráficas 5.5 muestran que la clasificación a través de *Well Known Ports* para el caso de tráfico *Multimedia* y *P2P* es pobre, mientras que *Análisis de Patrones* devuelve porcentajes altos de clasificación. La cooperación entre las diferentes

metodologías aisladas permite una respuesta mejor, donde cada metodología aislada aportará resultados para los diferentes tipos de tráfico.

La comparación del nivel de certeza de los resultados obtenidos por las metodologías de combinación comparado con la aplicación de heurísticas en forma aislada, indica que la combinación no desvirtúa los resultados obtenidos, dado que existe un porcentaje muy bajo de flujos con diferencia entre la clasificación de las metodologías combinadas con las aisladas.

El sistema *trafChar* propuesto permite además el manejo de pesos en las diferentes heurísticas (γ_i), los que permiten reformular el modelo, asignando la participación de las metodologías aisladas utilizadas dependiendo de su certeza.

En referencia a la comparación de las metodologías *Combinación con Peso por Heurística* y *Clasificación Bayesiana* puede afirmarse los siguiente:

- *Combinación con Peso por Heurística* es muy simple en su algoritmia, y es muy fácil su implementación y aplicación. Los resultados obtenidos por ésta son comparables a la otra metodología. Se puede afirmar que su simpleza no genera errores en la clasificación obtenida como resultado.

- *Clasificación Bayesiana* presenta la característica de permitir la corrección de errores sistemáticos realizados por las heurísticas aplicadas en forma aislada. Éstas correcciones solamente pueden realizarse en los casos en que más de una metodología aislada logra clasificar un flujo, y el resultado de la metodología combinatoria es correcto.

La obtención de resultados pobres en la metodología *Técnicas SVM* se entiende es causada por la falta de registros para su entrenamiento en algunas clases de tráfico. Los resultados obtenidos en ejecuciones de mayor tiempo, realizadas en la red hogareña indican que el mecanismo clasifica y lo hace correctamente.

Lo destacable de la implementación propuesta de *Técnica SVM*, es que permite generar el mecanismo sin entrenamiento previo y con el pasaje del tiempo éste genera el conocimiento que permite su aplicación. El tiempo requerido de aprendizaje no es grande en las clases de tráfico en las que se está porcesando tráfico.

Finalmente la prueba de concepto realizada permite afirmar que la combinación de metodologías mejora los resultados obtenidos por las heurísticas de

clasificación aisladas. Los resultados presentados por las metodologías aisladas muestran diferentes tasa de éxito para los diferentes tipos de tráfico, mejorando así el éxito global de la clasificación en las metodologías de combinación. En términos generales la combinación de resultados permite obtener resultados más completos y certeros.

Las propuestas de la academia estudiadas, plantean la combinación sobre metodologías de clasificación de tráfico concretos, mientras que nuestras propuestas plantean mecanismos de combinación genéricos. En el caso de las metodologías de clasificación por comportamiento presentadas en la sección 2.3, las mismas obtienen resultados para tráfico *P2P* únicamente, mientras que metodologías del tipo *Well Known Ports* los obtienen en aplicaciones tipo *Mail* o *Web*. Combinando los resultados se obtienen los beneficios de las dos propuestas. Se entiende entonces que las metodologías de combinación son un paso importante para la clasificación de tráfico en Internet.

6.3. Trabajo Futuro

Los trabajos futuros propuestos mejoran la clasificación final, y resuelven carencias encontradas en el sistema actual.

Un objetivo buscado es lograr que él sistema *trafChar* mejore su funcionamiento en base a la información tomada de la red. Un mecanismo que permita calibrar automáticamente los parámetros del sistema $\gamma_i, \mu, \mu_{unclf}$, permitirá una autoconfiguración del mismo mejorando la clasificación de tráfico final obtenida. Para cada uno de los parámetros se requiere un estudio particular.

- Parámetro μ. Éste parámetro indica el nivel de certeza del resultado *probabilísticos* devueltos por las diferentes metodologías de clasificación. El valor empírico considerado en las pruebas es 0,001, sin considerar valores diferentes para las diferentes metodologías de clasificación. Pueden considerarse valores μ para las diferentes metodologías e inclusive se pueden considerar valores diferenciados dependiendo del resultado devuelto (por ejemplo para *Well Known Ports* dependiendo del puerto considerado devolver un valor μ diferente).

- Parámetro μ_{unclf}. Éste parámetro indica el nivel de certeza para el caso de no clasificado en resultados *probabilísticos*. Debe tenerse en cuenta que un valor pequeño genera que se descarten metodologías de clasificación, en el mecanismo de *Combinación con Peso por Heurística*. Estudios acerca del valor de éste parámetro pueden mejorar el resultado de la clasificación final obtenida.

- Parámetro γ_i. Éste parámetro en ambos métodos de combinación representa el peso dado a cada heurística. En el presente trabajo se realizó en forma empírica y en base a la certeza considerada para los diferentes mecanismos de clasificación utilizados. Se propone generar éstos parámetros en función de la información obtenida por el sistema para la aplicación de *Clasificación Bayesiana*, donde puede determinarse el acierto obtenido por cada una de las diferentes metodologías aisladas. Esto permite independizar al sistema del estudio previo de certeza de las metodologías aisladas utilizadas.

Las pruebas realizadas no incluyeron mecanismos basados en *Elementos de Comportamiento* (ver sección 2.3). Se propone como trabajo futuro estudiar el comportamiento de los mecanismo de combinación con la incorporación de éstas metodologías de clasificación. Se plantea la implementación dentro del preprocesador *trafChar* desarrollado de metodologías basadas en el comportamiento como puede ser *IP pairs*, estudiando los resultados que se obtienen.

Por último el desarrollo de una herramienta que permita el testeo de mecanismos de combinación de metodologías de clasificación se entiende facilitaría posteriores estudios. Ésta herramienta debe incluir la posibilidad de invocar módulos de clasificación desarrollados en forma separada y permitir aplicar una metodología de combinación. La herramienta permitirá el testeo de mecanismos de clasificación y combinación de resultados obtenidos. La diferencia con el preprocesador de Snort desarrollado es que el objetivo del software propuesto es la clasificación del tráfico, mientras que Snort se propone la detección de intrusos IDS.

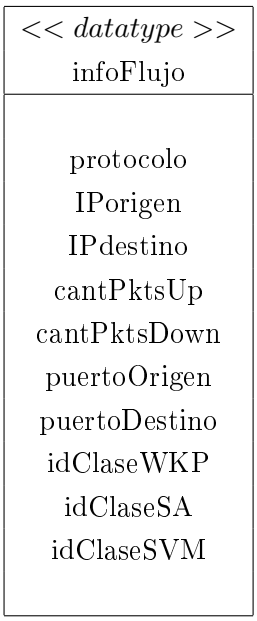
Apéndice A

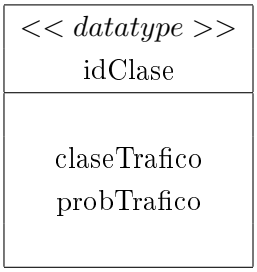
Modelo de Casos de Uso

El presente Apéndice presenta el diagrama de secuencia del preprocesador *trafChar* desarrollado y los contratos de software de las operaciones incluidas en el mismo.

A.1. Diagrama de Secuencia de *trafChar*

Los tipos de datos (*datatypes*) utilizados se presentan a continuación:





El tipo *infoFlujo* contiene la información almacenada por el sistema para cada

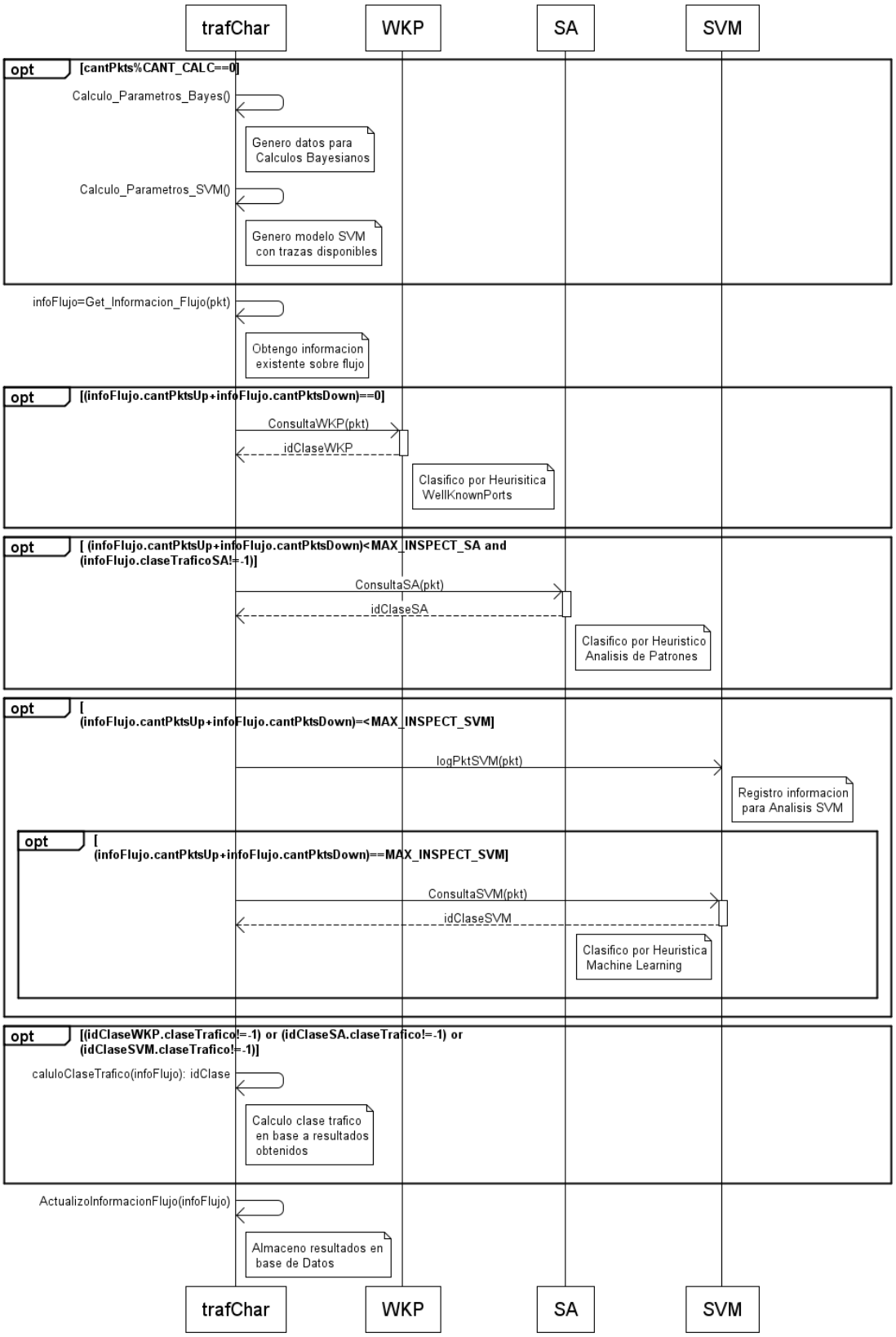
flujo procesado. El tipo *idClase* contiene el resultado genérico de la aplicación de un mecanismo de clasificación, se compone de la clase de tráfico y el vector de probabilidades de pertenencia a cada clase.

La figura A.1 muestra el diagrama de secuencia de *trafChar*.

El sistema utiliza tres bases de información:

- *Base de Flujos*. Contiene la información recabada por el sistema para cada flujo analizado. Su estructura se presenta en C.2.
- *Base Bayesiana*. Contiene información histórica para la aplicación de metodologías bayesianas. Su estructura se presenta en C.4.
- *Base SVM*. Contiene información histórica para la aplicación de las técnicas de SVM, con el fin de entrenar la ML. La información almacenada contiene valores de las diferentes características para flujos ya clasificados, incluyendo la clase asignada.

Figura A.1 Diagrama de Secuencia de *trafChar*



A.2. Contratos de Software

Operación	Calculo_Parametros_Bayes()
Entrada	
Salida	
Descripción	Genera la información requerida para el Cálculo Bayesiano que se realiza para clasificar cada flujo. A partir de la información existente en *Base Bayesiana*, contabiliza para cada clase de tráfico los resultados obtenidos por cada metodología

Operación	Calculo_Parametros_SVM()
Entrada	
Salida	
Descripción	Genera a partir de información disponible en *Base SVM*, el modelo y escala para la aplicación de la metodología SVM

Operación	Get_Informacion_Flujo(pkt): datosFlujo
Entrada	paquete
Salida	infoFlujo
Descripción	Dado el paquete pasado como parámetro, a partir de los datos que identifican el flujo *protocolo*, *IP origen*, *IP destino*, *puerto origen* y *puerto destino*, se devuelve la información disponible en la *Base Flujos* del mismo. En caso de existir registro se carga la información disponible en *infoFlujo*, en caso contrario se cargan valores por defecto.

Operación	ConsultaWKP(pkt) : idClase
Entrada	paquete
Salida	idClase
Descripción	Para el paquete pasado como parámetros se toman de éste el *protocolo*, *puertoOrigen* y *puertoDestino*. Se aplica la metodología *Well Known Ports*. En caso de clasificar devuelve *idClase* conteniendo *{claseTrafico,probTrafico}*, en caso contrario devuelve *claseTrafico* -1..

Operación	ConsultaSA(pkt) : idClase
Entrada	paquete
Salida	idClase
Descripción	Para el paquete pasado como parámetro se aplica la metodología de *Análisis de Patrones*. La información de los patrones se encuentra almacenada en el sistema. En caso de clasificar devuelve *idClase* conteniendo *{claseTrafico,probTrafico}*, en caso contrario devuelve *claseTrafico* -1.

Operación	logPktSVM(pkt)
Entrada	paquete
Salida	
Descripción	Dado el paquete pasado por parámetro toma de éste el tiempo de captura, tamaño y determina su dirección (*up* o *down*). Almacena esta información para los cálculos de la metodología SVM.

Operación	ConsultaSVM(pkt): idClase
Entrada	paquete
Salida	idClase
Descripción	Para el paquete pasado como parámetros se identifica el flujo y se aplica la metodología SVM en función de la información registrada para el mismo a través de *logPktSVM*. Devuelve *idClase* conteniendo *{claseTrafico,probTrafico}*.

Operación	caluloClaseTrafico(infoFlujo) : idClase
Entrada	infoFlujo
Salida	idClase
Descripción	Para la información contenida en le parámetro *infoFlujo*, realiza la metodología combinada de clasificación. En caso de clasificar devuelve *idClase* conteniendo *{claseTrafico,probTrafico}*, en caso contrario devuelve *claseTrafico* -1.

Operación	ActualizoInformacionFlujo(infoFlujo)
Entrada	infoFlujo
Salida	
Descripción	Actualiza/almacena en la información de *Base Flujos*. En caso de contener información válida para la metodología SVM la almacena en la *Base SVM*. Si su información se considera certera para la metodología bayesiana la almacena en la Base Bayesiana.

Apéndice B

Resultados Obtenidos por *Clasificación Bayesiana*

Éste apartado presenta un estudio detallado de los resultados obtenidos por la metodología de combinación *Clasificación Bayesiana*. Para el presente estudio se toma como base de información para el cálculo de la metodología, datos obtenidos en las pruebas realizadas.

B.1. Base de Información para el Cálculo de *Clasificación Bayesiana*

En la presente sección se presentan los valores utilizados para el cálculo de la *Clasificación Bayesiana*. Éstos valores son utilizados en la fórmula B.1 Los parametros utilizados son, $\gamma_{WKP} = 0{,}4$, $\gamma_{SA} = 0{,}5$, $\gamma_{SVM} = 0{,}1$ y $\delta = 0{,}01$.

$$Pr[j|cm_{WKP}, cm_{SA}, cm_{SVM}] = (Pr[j] + \delta) \prod_i (Pr[cm_i|j]\gamma_i + \delta) \qquad \text{(B.1)}$$

donde

$$i \in \{WKP, SA, SVM\}$$

La tabla B.1 contiene los valores $Pr[j]$ y $Pr[cm_i|j]$. Como se ha presentado, dado el resultado determinístico de las metodologías de clasificación *Well Known*

Ports (WKP), *Análisis de Patrones* (SA) y *Técnica SVM* (SVM) se aplica para cada clase de tráfico j la fórmula B.1 y posteriormente se normaliza aplicando ecuación B.2.

$$Pr^{Bys}[j|cm_{WKP}, cm_{SA}, cm_{SVM}] = \frac{Pr[j|cm_{WKP}, cm_{SA}, cm_{SVM}]}{\sum_{j=0}^{n} Pr[j|cm_{WKP}, cm_{SA}, cm_{SVM}]} \quad \text{(B.2)}$$

B.2. Resultados Obtenidos

Inicialmente presentamos los resultados para el caso en que las tres metodologías de clasificación devuelven la misma clase de tráfico se presentan en la tabla B.2.

Los valores obtenidos se encuentran en el rango $0{,}932 - 1$, donde el resultado presenta un nivel alto de certeza. El valor más bajo corresponde a la clase de tráfico 6 (P2P) la que como puede verse en la tabla B.1 presenta gran cantidad de resultados no certeros. El resultado 1 corresponde al tráfico *Web*, que se encuentra en gran cantidad y es clasificado en forma certera.

En segundo término presentamos el caso de clasificación por dos y una metodología. La tabla B.3 contiene los resultados obtenidos en el caso de tráfico Web.

De los resultados presentados analizando los casos donde clasifica una sola metodología, se desprende que la metodología más certera es *Análisis de Patrones*, seguida por *Well Known Ports* y en tercer término la *Técnicas SVM*.

Por último la tabla B.4 presenta los resultados para el caso de tráfico P2P.

Los resultados obtenidos muestran que la metodología que logra clasificar es *Análisis de Patrones* (SA), dado que solamente se logra clasificar cuando ésta metodología obtiene resultados.

Por último en la tabla B.5 presentamos el caso de resultados discordantes de las diferentes metodologías de clasificación aplicadas. Se considera todas las posibilidades de resultados discordantes de clasificación con clases *Mail*, *P2P* y *Chat* (clases 2, 6 y 7 respectivamente). Buscamos verificar que la metodología considera la respuesta en función de las diferentes clases de tráfico. Del gráfico 5.5 se desprende que la clases de tráfico *Mail* y *Chat* responden en forma correcta para las metodologías *Well Known Port* (WKP) y *Análisis de Patrones* (SA) y la metodología *P2P* solamente presenta resultados para *Análisis de Patrones* (SA).

Cuadro B.1: *Base de Información Bayesiana*

$Pr[j]$									
$j \Rightarrow$	-1	0	1	2	3	4	5	6	7
$Pr[j]$	0.50	0.00	0.00	0.03	0.02	0.43	0.00	0.01	0.00
$Pr[cme_i \mid j]$									
$cme_{WKP} \Rightarrow$ $j \Downarrow$	-1	0	1	2	3	4	5	6	7
-1	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00
0	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
5	0.17	0.00	0.00	0.00	0.00	0.00	0.83	0.00	0.00
6	0.86	0.00	0.00	0.00	0.00	0.00	0.03	0.11	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
$Pr[cme_{SA} \mid j]$									
$cme_{SA} \Rightarrow$ $j \Downarrow$	-1	0	1	2	3	4	5	6	7
-1	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
0	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
3	0.09	0.00	0.00	0.00	0.91	0.00	0.00	0.00	0.00
4	0.10	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
7	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.95
$Pr[cme_{SVM} \mid j]$									
$cme_{SVM} \Rightarrow$ $j \Downarrow$	-1	0	1	2	3	4	5	6	7
-1	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.05	0.00	0.00	0.00	0.59	0.13	0.18	0.04	0.00
3	0.02	0.00	0.00	0.00	0.85	0.09	0.00	0.03	0.01
4	0.12	0.00	0.00	0.00	0.01	0.85	0.01	0.00	0.00
5	0.00	0.00	0.00	0.00	0.17	0.33	0.50	0.00	0.00
6	0.25	0.00	0.00	0.00	0.15	0.00	0.15	0.40	0.05
7	0.27	0.00	0.00	0.00	0.59	0.00	0.00	0.05	0.09

Cuadro B.2: *Resultados* $\mathbf{Pr^{Bys}}$ *con clasificación igual*

			$Pr^{Bys}[j\|cme_{WKP}, cme_{SA}, cme_{SVM}]$									
clf WKP	clf SA	clf SVM	-1	0	1	2	3	4	5	6	7	result final
0	0	0	0.002	**0.995**	0.000	0.000	0.000	0.002	0.000	0.000	0.000	0
1	1	1	0.002	0.000	**0.995**	0.000	0.000	0.002	0.000	0.000	0.000	1
2	2	2	0.006	0.000	0.000	**0.988**	0.000	0.005	0.000	0.000	0.000	2
3	3	3	0.001	0.000	0.000	0.000	**0.998**	0.001	0.000	0.000	0.000	3
4	4	4	0.000	0.000	0.000	0.000	0.000	**1.000**	0.000	0.000	0.000	4
5	5	5	0.004	0.000	0.000	0.001	0.000	0.004	**0.989**	0.001	0.000	5
6	6	6	0.046	0.000	0.000	0.002	0.002	0.016	0.000	**0.932**	0.001	6
7	7	7	0.010	0.000	0.000	0.001	0.001	0.009	0.000	0.001	**0.978**	7

Cuadro B.3: *Resultados* $\mathbf{Pr^{Bys}}$ *para tráfico Web*

			$Pr^{Bys}[j\|cme_{WKP}, cme_{SA}, cme_{SVM}]$									
clf WKP	clf SA	clf SVM	-1	0	1	2	3	4	5	6	7	result final
4	-1	-1	0.027	0.001	0.001	0.002	0.002	**0.966**	0.001	0.001	0.001	4
-1	4	-1	0.024	0.000	0.000	0.002	0.002	**0.969**	0.001	0.001	0.001	4
-1	-1	4	0.102	0.002	0.002	0.019	0.013	**0.846**	0.009	0.004	0.003	4
4	4	-1	0.001	0.000	0.000	0.000	0.000	**0.999**	0.000	0.000	0.000	4
4	-1	4	0.003	0.000	0.000	0.001	0.000	**0.996**	0.000	0.000	0.000	4
-1	4	4	0.003	0.000	0.000	0.000	0.000	**0.996**	0.000	0.000	0.000	4

Cuadro B.4: *Resultados* $\mathbf{Pr^{Bys}}$ *para tráfico P2P*

			$Pr^{Bys}[j\|cme_{WKP}, cme_{SA}, cme_{SVM}]$									
clf WKP	clf SA	clf SVM	-1	0	1	2	3	4	5	6	7	result final
6	-1	-1	**0.591**	0.006	0.006	0.025	0.020	0.274	0.007	0.063	0.008	-1
-1	6	-1	0.308	0.005	0.005	0.018	0.015	0.198	0.005	**0.442**	0.006	6
-1	-1	6	**0.418**	0.008	0.008	0.047	0.035	0.377	0.009	0.081	0.016	-1
6	6	-1	0.185	0.001	0.001	0.006	0.005	0.063	0.002	**0.735**	0.002	6
6	-1	6	**0.460**	0.005	0.005	0.028	0.021	0.219	0.005	0.248	0.009	-1
-1	6	6	0.109	0.002	0.002	0.009	0.007	0.072	0.002	**0.795**	0.003	6

Los resultados probabilísticos muestran el comportamiento para las diferentes clases de tráfico, donde por ejemplo en el caso del resultado $WKP = 7$, $SA = 2$ y $SVM = 6$ donde las *Técnicas SVM* devuelven un resultado de una clase donde

Cuadro B.5: *Resultados* $\mathbf{Pr^{Bys}}$ *para resultados no coincidentes*

			$Pr^{Bys}[j\|cme_{WKP}, cme_{SA}, cme_{SVM}]$									
clf WKP	clf SA	clf SVM	-1	0	1	2	3	4	5	6	7	result final
2	6	7	0.165	0.002	0.002	**0.380**	0.008	0.103	0.002	0.330	0.006	2
2	7	6	0.119	0.002	0.002	**0.529**	0.010	0.104	0.002	0.022	0.209	2
6	2	7	0.259	0.003	0.003	**0.555**	0.009	0.121	0.003	0.040	0.007	2
6	7	2	**0.425**	0.004	0.004	0.019	0.015	0.200	0.005	0.045	0.283	-1
7	2	6	0.107	0.002	0.002	**0.603**	0.009	0.094	0.002	0.020	0.161	2
7	6	2	0.253	0.004	0.004	0.015	0.012	0.160	0.004	**0.356**	0.193	6

la metodología no responde correctamente. La probabilidad obtenida por la clase *P2P* es muy baja 0,02, mientras el resultado para la clase *Mail* es 0,603 la mas alta y la obtenida por *Análisis de Patrones* (SA).

Otro resultado interesante es $WKP = 2$, $SA = 6$ y $SVM = 7$ donde se analiza el caso de dos clases de tráfico que responden en forma correcta para los tipos de tráfico *Mail* y *P2P*. El resultado probabilístico presenta la situación mencionada dado que devuelve 0,380 para clase *Mail* y 0,330 para *P2P*, valores muy cercanos.

Apéndice C

Estructura de Tablas Utilizadas

Éste apartado presenta las tablas utilizadas por el preprocesador *trafChar* desarrollado y su estructura.

Las mismas son soportadas por MySQL [MyS09].

C.1. Tablas MySQL utilizadas

```
mysql> show tables;
+--------------------+
| Tables_in_trafChar |
+--------------------+
| flujos             |
| tipoTrafico        |
| trafCharStatistic  |
| wellKnownPorts     |
+--------------------+
4 rows in set (0.00 sec)
```

C.2. Estructura de la tabla *flujos*

```
mysql> describe flujos;
+---------------+------------------+------+-----+---------------------+-------+
| Field         | Type             | Null | Key | Default             | Extra |
+---------------+------------------+------+-----+---------------------+-------+
| t_ultimo      | timestamp        | NO   | MUL | CURRENT_TIMESTAMP   |       |
| t_inicial     | timestamp        | NO   |     | 0000-00-00 00:00:00 |       |
```

```
| srcaddr        | int(10) unsigned | NO   | PRI | 0                    |        |
| dstaddr        | int(10) unsigned | NO   | PRI | 0                    |        |
| srcport        | int(10) unsigned | NO   | PRI | 0                    |        |
| dstport        | int(10) unsigned | NO   | PRI | 0                    |        |
| prot           | int(10) unsigned | NO   | PRI | 0                    |        |
| up_pkts        | int(10) unsigned | YES  |     | NULL                 |        |
| down_pkts      | int(10) unsigned | YES  |     | NULL                 |        |
| metodosAplic   | int(1) unsigned  | YES  |     | NULL                 |        |
| codigoTrafico  | int(11)          | YES  |     | NULL                 |        |
| probTrfWKP     | varchar(100)     | YES  |     | NULL                 |        |
| probTrfCNT     | varchar(100)     | YES  |     | NULL                 |        |
| probTrfIA      | varchar(100)     | YES  |     | NULL                 |        |
+----------------+------------------+------+-----+----------------------+-------+
14 rows in set (0.09 sec)
```

C.3. Estructura de la tabla *tipoTrafico*

```
mysql> describe tipoTrafico;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| codigoTrafico | int(11)     | NO   | PRI | 0       |       |
| tipoTrafico   | varchar(25) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

C.4. Estructura de la tabla *trafCharStatistic*

```
mysql> describe trafCharStatistic;
+---------------+------------------+------+-----+-------------------+-------+
| Field         | Type             | Null | Key | Default           | Extra |
+---------------+------------------+------+-----+-------------------+-------+
| t_registro    | timestamp        | NO   | MUL | CURRENT_TIMESTAMP |       |
| srcaddr       | int(10) unsigned | YES  |     | NULL              |       |
| dstaddr       | int(10) unsigned | YES  |     | NULL              |       |
| srcport       | int(10) unsigned | YES  |     | NULL              |       |
| dstport       | int(10) unsigned | YES  |     | NULL              |       |
| prot          | int(10) unsigned | YES  |     | NULL              |       |
| codigoTrafico | int(11)          | YES  | MUL | NULL              |       |
| codigoWKP     | int(11)          | YES  |     | NULL              |       |
```

```
| codigoCNT     | int(11)          | YES  |     | NULL              |       |
| codigoIA      | int(11)          | YES  |     | NULL              |       |
+---------------+------------------+------+-----+-------------------+-------+
10 rows in set (0.00 sec)
```

C.5. Estructura de la tabla *wellKnownPorts*

```
mysql> describe wellKnownPorts;
+-----------------+------------------+------+-----+---------+-------+
| Field           | Type             | Null | Key | Default | Extra |
+-----------------+------------------+------+-----+---------+-------+
| prot            | int(10) unsigned | NO   | PRI | 0       |       |
| port            | int(10) unsigned | NO   | PRI | 0       |       |
| portDescription | varchar(100)     | YES  |     | NULL    |       |
| clase           | int(11)          | YES  |     | NULL    |       |
| valorMu         | decimal(6,4)     | YES  |     | NULL    |       |
+-----------------+------------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Referencias Bibliográficas

[AB05] Andrés Felipe Arboleda and Charles Edward Bedón, *Snort(tm) diagrams for developers*, April 2005, http://afrodita.unicauca.edu.co/ cbedon/snort/snortdevdiagrams.html.

[AC75] Alfred V. Aho and Margaret J. Corasick, *Efficient string matching: an aid to bibliographic search*, Commun. ACM **18** (1975), no. 6, 333–340.

[CAI09] CAIDA : research : traffic-analysis : classification-overview, http://www.caida.org/research/traffic-analysis/classification-overview/, 2009.

[Cis09] Cisco WAN and Application Optimization Solution Guide, http://www.cisco.com/en/US/docs/nsite/enterprise/wan/wan_optimization /wan_opt_sg.pdf, 2009.

[CL01a] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM: a library for support vector machines*, 2001, Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[CL01b] Chih-Chung Chang and Chih-Jen Lin, *Libsvm: a library for support vector machines*, 2001.

[CST00] Nello Cristianini and John Shawe-Taylor, *An introduction to support vector machines: and other kernel-based learning methods*, Cambridge University Press, New York, NY, USA, 2000.

[EAM06] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti, *Traffic classification using clustering algorithms*, MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data (New York, NY, USA), ACM, 2006, pp. 281–286.

[EMA06] Jeffrey Erman, Anirban Mahanti, and Martin F. Arlitt, *Internet traffic identification using machine learning.*, GLOBECOM, IEEE, 2006.

[EMA+07] Jeffrey Erman, Anirban Mahanti, Martin Arlitt, Ira Cohen, and Carey Williamson, *Offline/realtime traffic classification using semi-supervised learning*, Performance Evaluation **64** (2007), no. 9-12, 1194–1213.

[GS99] Anup K. Ghosh and Aaron Schwartzbard, *A study in using neural networks for anomaly and misuse detection*, SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium (Berkeley, CA, USA), USENIX Association, 1999, pp. 12–12.

[GSL09] GSL - GNU Scientific Library, http://www.gnu.org/software/gsl/, 2009.

[HiP09] HiPPIE: Hi-Performance Protocol Identification Engine, http://hippie.oofle.com/about, 2009.

[How09] *Snort preprocessors kickstart*, 2009, http://snort-ai.wiki.sourceforge.net/Snort+Preprocessors+Kickstart.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The elements of statistical learning: Data mining, inference, and prediction*, Springer, August 2001.

[IAN09] IANA(Internet), *Iana, protocol registries*, April 2009, http://www.iana.org/protocols/.

[IKF+09] M. Iliofotou, Hyun-Chul Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, and G. Varghese, *Graph-based p2p traffic classification at the internet backbone*, INFOCOM Workshops 2009, IEEE, 2009, pp. 1–6.

[IPF+07] Marios Iliofotou, Prashanth Pappu, Michalis Faloutsos, Michael Mitzenmacher, Sumeet Singh, and George Varghese, *Network monitoring using traffic dispersion graphs (tdgs)*, Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference (New York, NY, USA), ACM, 2007, pp. 315–320.

[KBFC04] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kc Claffy, *Transport layer identification of p2p traffic*, 2004.

[KCF+] Hyun-Chul Kim, Kc Claffy, Marina Fomenkov, Dhiman Barman, Michalis Faloutsos, and Kiyoung Lee, *Internet traffic classification demystified: Myths, caveats, and the best practices*, ACM CoNEXT 2008.

[LD00] Roger J. Lewis and Ph. D, *An introduction to classification and regression tree (cart) analysis*, Annual Meeting of the Society of Academic Emergency Medicine in, 2000.

[Lew00] Roger J. Lewis, *An introduction to classification and regression tree (cart) analysis; presented at annual meeting of the society for academic emergency medicine*, Annual Meeting of the Society of Academic Emergency Medicine in, 2000.

[MCM+05] Andrew Moore, Michael Crogan, Andrew W. Moore, Queen Mary, Denis Zuev, Denis Zuev, and Michael L. Crogan, *Discriminators for use in flow-based classification*, Tech. report, 2005.

[MyS09] MySQL: The world's most popular open source database., http://www.mysql.com/, 2009.

[MZ05] Andrew W. Moore and Denis Zuev, *Internet traffic classification using bayesian analysis techniques*, SIGMETRICS Perform. Eval. Rev. **33** (2005), no. 1, 50–60.

[Pos81] Jon Postel, *RFC793: Transmission control protocol*, September 1981.

[Ris05] Irina Rish, *An empirical study of the naive bayes classifier*, IJCAI-01 workshop on Empirical Methods in AI, 2005.

[SAM09] *State of the art in traffic classification: A research review.*, Seoul, Korea, 2009.

[Sno09] Snort Users Manual, http://www.snort.org/docs/snort_manual/2.8.3/snort_manual.pdf, 2009.

[SS08] Inc. Snort Sourcefire, *Snort - the de facto standard for intrusion detection/prevention*, October 2008, http://www.snort.org/.

[SSO07] Geza Szabo, Istvan Szabo, and Daniel Orincsay, *Accurate traffic classification*, A World of Wireless, Mobile and Multimedia Networks, International Symposium on **0** (2007), 1–8.

[WF05] Ian H. Witten and Eibe Frank, *Data mining: Practical machine learning tools and techniques, second edition (morgan kaufmann series in data management systems)*, Morgan Kaufmann, June 2005.

[YKL04] Fang Yu, Randy H. Katz, and T. V. Lakshman, *Gigabit rate packet pattern-matching using tcam*, Tech. Report UCB/CSD-04-1341, EECS Department, University of California, Berkeley, Jul 2004.