MÁSTER EN COMUNICACIONES

TRABAJO FIN DE MÁSTER

Metodología fiable de detección de duplicados para el análisis de tráfico de red IP

CURSO: 2012/2013

Iñaki Úcar Marqués

El tribunal constituido para la evaluación del proyecto TFM titulado:

Metodología fiable de detección de duplicados para el análisis de tráfico de red IP

Realizado por el alumno: **Iñaki Úcar Marqués**
Y dirigido por el tutor: **Daniel Morató Osés**

Ha resuelto asignarle la calificación de:

- □ SOBRESALIENTE (9 - 10 puntos)
- □ NOTABLE (7 - 8.9 puntos)
- □ APROBADO (5 - 6.9 puntos)
- □ SUSPENSO

Con la nota: puntos.

El Presidente:

El Secretario:

El Vocal:

Pamplona, a de de 2013

MÁSTER EN COMUNICACIONES

Metodología fiable de detección de duplicados para el análisis de tráfico de red IP

REALIZADO POR:
Iñaki Úcar Marqués

DIRIGIDO POR:
Daniel Morató Osés

ESCUELA:
ETS de Ingenieros Industriales y de Telecomunicación

Pamplona, junio de 2013

Metodología fiable de detección de duplicados para el análisis de tráfico de red IP

Iñaki Úcar Marqués

PALABRAS CLAVE:
duplicación de datos, monitorización de redes, análisis de tráfico

RESUMEN:
Todo sistema de monitorización de tráfico de red se enfrenta a un problema fundamental: el proceso de captura de tráfico, casi invariablemente, produce una duplicación de datos que falsea cualquier análisis posterior. En contraste con otros campos relacionados con la ingeniería de datos, no existe literatura científica que aborde este problema.

En este contexto, el presente trabajo establece los fundamentos teóricos de la duplicación de datos en el ámbito de la monitorización de red. Para ello, se describen y analizan los mecanismos generadores, los tipos de duplicados existentes y sus características.

Sobre esta base, se aporta una metodología fiable para la detección de duplicados que ha sido implementada y profusamente utilizada en proyectos empresariales. Se recoge, asimismo, un estudio analítico validado experimentalmente del cual se deriva una regla de dimensionamiento para maximizar la eficiencia de la metodología sin perder capacidad de detección.

D. Daniel Morató Osés, profesor del Área de Ingeniería Telemática de la ETS de Ingenieros Industriales y de Telecomunicación de la Universidad Pública de Navarra, como director del Trabajo Fin de Máster de D. Iñaki Úcar Marqués

Informa:

que el presente trabajo, titulado:

Metodología fiable de detección de duplicados para el análisis de tráfico de red IP

Ha sido realizado y redactado por el mencionado alumno bajo su dirección, y con esta fecha autoriza a su presentación.

Pamplona, a de de 2013

Fdo. Daniel Morató Osés

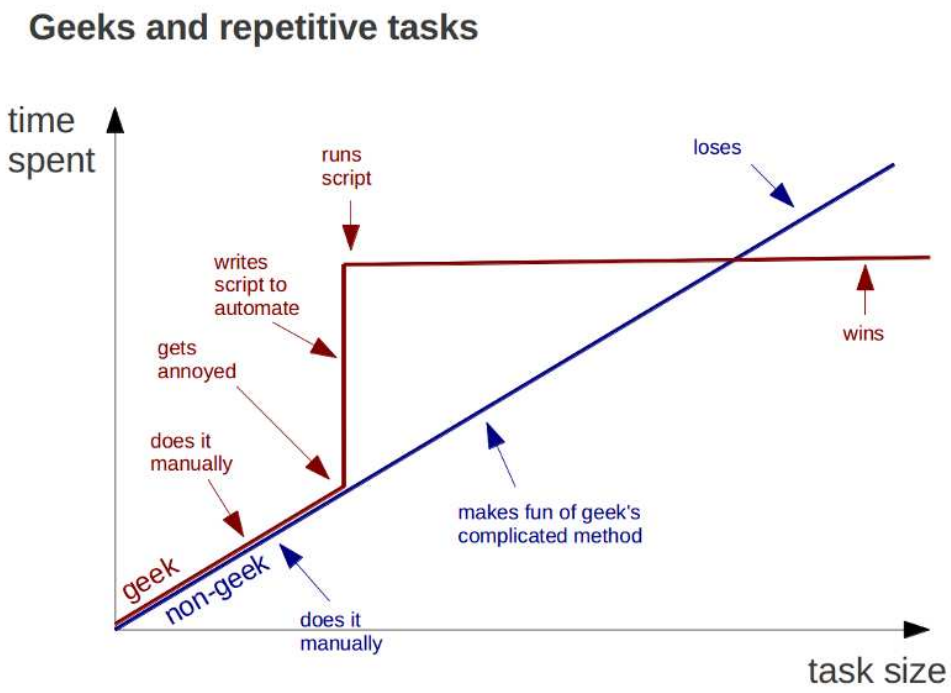
Los abajo firmantes autorizan a que la presente copia de Trabajo Fin de Máster se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Pamplona, a de de 2013

Fdo. Iñaki Úcar Marqués

Fdo. Daniel Morató Osés

Estoy desarrollando un generador automático de agradecimientos.



Mientras tanto...

Al GRSST, por todo lo que he aprendido aquí.


INGREDIENTS:
HYDROGEN, TIME

Índice general

Índice de figuras

Índice de tablas

Glosario

Símbolos

802.1Q Estándar de red definido por el IEEE que define un etiquetado de tramas Ethernet para la coexistencia de múltiples LAN virtuales (VLAN) en un mismo segmento físico.

A

ACK *Acknowledgement*, campo de la cabecera TCP que lleva cuenta de la cantidad de bytes recibidos dentro de una conexión. Asimismo, el *flag* de la misma cabecera que activa el campo descrito también se nombra con estas siglas.

ALG *Application Layer Gateway*, extensión implementada sobre cortafuegos o dispositivos NAT para la traducción de mensajes en protocolos que portan información de la conexión a nivel de aplicación y que, de otra manera, serían incapaces de atravesar estos dispositivos.

D

DSCP *Differentiated Services Code Point*, hace referencia al campo de la cabecera IPv4 utilizado para marcar tráfico en función de distintos tipos de servicios en arquitecturas de QoS basadas en Servicios Diferenciados (DiffServ).

F

FTP *File Transfer Protocol*, protocolo de transferencia de ficheros basado en TCP y una arquitectura de cliente-servidor. Utiliza dos conexiones: una de control, basada en texto plano, y otra para la transferencia de ficheros.

I

IGMP *Internet Group Management Protocol*, protocolo de nivel de red utilizado en redes IP para la gestión de grupos *multicast*.

IP *Internet Protocol*, protocolo de nivel de red de la pila de protocolos TCP/IP. También se llama así a la dirección de nivel de red que identifica a una máquina.

M

MAC *Media Access Control address*, identificador único asignado a las interfaces de red y usado como dirección de red en la comunicación a nivel de enlace.

MTU *Maximum Transmission Unit*, tamaño máximo de datagrama que un nivel determinado de la pila de protocolos puede manejar.

GLOSARIO

N

NAT *Network Address Translation*, mecanismo de traducción de direcciones IP entre redes que asignan direcciones mutuamente incompatibles.

Q

QoS *Quality of Service*, la totalidad de características de un servicio de telecomunicaciones que determinan su capacidad para satisfacer las necesidades explícitas e implícitas del usuario del servicio.

R

RST *Reset*, *flag* de la cabecera TCP usado como mensaje de control con el propósito de forzar el cierre de una conexión.

S

SPAN *Switched Port Analyzer*, denominación de Cisco Systems para el sistema de monitorización de tráfico de red mediante *port mirroring* que implementan sus equipos.

T

TCP *Transmission Control Protocol*, protocolo de nivel de transporte de la pila de protocolos TCP/IP orientado a conexión.

TTL *Time To Live*, campo de la cabecera IPv4 que alberga el tiempo de vida de un paquete, es decir, el número de veces que puede ser encaminado antes de que sea descartado.

U

UDP *User Datagram Protocol*, protocolo de nivel de transporte de la pila de protocolos TCP/IP orientado a datagrama.

V

VLAN *Virtual Local Area Network*, red de área local virtual. Un *switch* o un *router* es capaz de particionar los puertos de una LAN a nivel lógico para formar múltiples dominios de *broadcast* o VLANs.

Capítulo 1

Introducción

La duplicación de datos es un problema genérico de las tecnologías y sistemas de información concerniente al campo de la ingeniería de datos. Como su propio nombre indica, consiste en la presencia de repeticiones de un mismo elemento dentro de un conjunto de datos.

Las consecuencias de este hecho pueden ser muy variadas dependiendo del ámbito que consideremos. La más evidente es que el conjunto de datos es más grande, y por tanto ocupa más espacio, de lo que sería estrictamente necesario. En términos de teoría de la información, la entropía de la fuente está lejos de ser óptima. Así pues, los algoritmos de compresión sin pérdidas pueden entenderse como simples desduplicadores. En esta misma línea, muchas soluciones de *backup* y almacenamiento de archivos contienen mecanismos de desduplicación: incluso Mega, el flamante sucesor de Megaupload que vio la luz en enero de este mismo año, anunció que su servicio de almacenamiento cifrado en la nube poseía esta característica.

El ejemplo anterior es consecuencia de la redundancia intrínseca que posee la representación de la información que manejamos. Fuera de esta estructura interna, esta codificación, otro caso frecuente hoy en día es la integración de información desde múltiples fuentes o bases de datos. Sirva como ejemplo sencillo la información que puedan contener diferentes bases de datos sobre las publicaciones científicas de determinado autor: habitualmente se producen solapamientos y los buscadores que operan con varias de estas fuentes muestran entradas duplicadas.

En definitiva, una somera reflexión hace patente que esta problemática es fundamental en un gran número de disciplinas y, como se verá, un análisis más profundo arroja como conclusión que las características de cada una de estas disciplinas hacen de la desduplicación un reto único en su campo.

En el ámbito que nos concierne, el del análisis de tráfico de red IP, la unidad básica de información es el paquete. A partir de ahí, se construyen jerarquías más altas de información: conjuntos de paquetes constituyen flujos, conjuntos de flujos forman conexiones, etc. En este contexto, la duplicación de datos se traduce en duplicación

de paquetes. Más concretamente, se traduce en duplicación de la carga útil de los paquetes. Es decir, el segmento de datos que se genera en lo más alto de la pila de protocolos, a nivel de usuario, es la verdadera unidad de información, puesto que ese mismo datagrama puede encontrarse con diferentes cabeceras conforme va viajando por los niveles más bajos (véase la Figura 1.1).

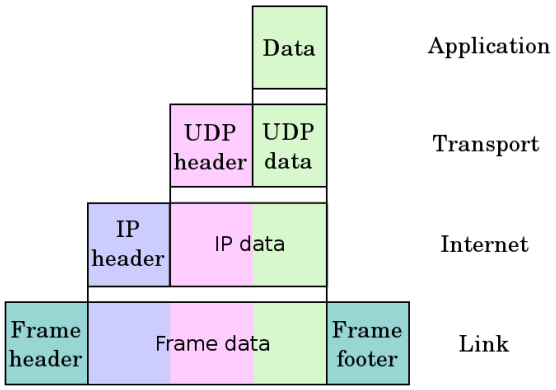


Figura 1.1: Ejemplo de encapsulación sobre la pila TCP/IP

Sin embargo, el análisis de red se realiza generalmente sobre el nivel más bajo de esa pila. Todo proceso de análisis descansa sobre una primera etapa de importancia crítica: la captura de los paquetes de red, que, en última instancia, como se discutirá más adelante, es el proceso causante de los paquetes duplicados.

1.1. Motivación

La motivación del presente trabajo es eminentemente práctica. Como se ha comentado y se analizará en la próxima sección, el análisis de tráfico de red descansa invariablemente sobre un proceso de monitorización que falsea lo que realmente está sucediendo en la red de datos. Por ejemplo, si en cierta fracción de tiempo y en un determinado punto de una red pasan dos paquetes pero medimos cuatro, automáticamente hasta las medidas más simples quedarán pervertidas: *throughput*, tiempo entre paquetes, etc. Por tanto, queda patente que la desduplicación constituye un paso previo fundamental.

No obstante, y en contra de lo que se pueda pensar *a priori*, lo cierto es que no existe literatura científica que describa rigurosamente y aborde este fenómeno. Tan solo hemos sido capaces de encontrar al respecto referencias informales, aisladas e incompletas [1, 2].

Por consiguiente, este trabajo está dedicado a cubrir esta carencia teórica y metodológica en cuanto a desduplicación de datos en el campo de la monitorización y análisis de redes IP.

1.2. Objetivos

A continuación, se enumeran los principales objetivos y contribuciones del presente trabajo:

- Contextualización del caso particular de la duplicación de tráfico de red en el campo más amplio de la ingeniería de datos.
- Realización de un análisis teórico y descriptivo de la problemática de los duplicados.
- Diseño de una metodología fiable para la detección y, en su caso, eliminación del tráfico duplicado.
- Implementación práctica de dicha metodología para su evaluación y puesta a punto.

Capítulo 2

Estado del arte

Tal y como se ha avanzado anteriormente, no existe —o al menos no hemos sido capaces de encontrar— literatura científica que aborde la duplicación de datos en el campo específico de las redes de datos. Por ello, este capítulo está dedicado a la realización de una revisión generalista del trabajo previo en otros campos con el objetivo de contextualizar el caso particular que nos atañe.

2.1. Introducción

Dentro de la comunidad estadística, se han venido empleando desde hace más de cincuenta años los términos *record linkage* y *record matching* para designar el problema de la identificación de diferentes representaciones de una misma entidad. Más recientemente, se han añadido otras denominaciones en el ámbito de las bases de datos o la inteligencia artificial, como *merge-purge*, *data deduplication*, *instance identification*, *database hardening*, *name matching*, *coreference resolution*, *identity uncertainty* y *duplicate detection* [3, 4]. En el presente trabajo, se utilizarán indistintamente los términos *duplicación/desduplicación de datos*, *detección de duplicados* o *identificación de duplicados*.

Nótese la importante matización del párrafo anterior cuando se dice *diferentes representaciones de una misma entidad*. Esta descripción explicita quizás el concepto más importante de la duplicación de datos: cada, digamos, instancia de información duplicada es, en general, *distinta* de las demás, ya sea por diferencia de criterio, por error o por necesidad. Esto, que puede parecer confuso en un primer momento, se aclara recuperando un ejemplo del capítulo anterior: una base de datos puede recoger publicaciones de un tal «Iñaki Úcar» y de «Iñaki Úcar Marqués», pero realmente son la misma persona y dichas entradas deberían ser fusionadas. Obviamente, la desduplicación conlleva otros retos, pero este hecho es uno de los principales contribuyentes a que se aleje mucho de ser trivial.

A continuación, se hará un breve repaso de las principales técnicas existentes de

detección de duplicados y se discutirán cuestiones de eficiencia en base a [3].

2.2. Técnicas de detección

Existe una gran variedad de técnicas para comparar datos consistentes en un solo campo. Desde la más sencilla y obvia, que pasa por comparar los elementos *byte-by-byte*, hacia las más sofisticadas, que tienen en cuenta la naturaleza de los datos y que sí contemplan variaciones sobre los mismos. El grueso maś importante de este tipo de técnicas se corresponde con los métodos de comparación de cadenas de caracteres.

No obstante, en la mayoría de situaciones de la vida real, los datos a comparar poseen múltiples campos, lo que eleva notoriamente la complejidad del problema. Las técnicas utilizadas en este caso pueden subdividirse en probabilísticas, basadas en clasificadores determinísticos y de aprendizaje automático, supervisado y no supervisado.

2.2.1. Técnicas probabilísticas

El proceso de desduplicación se reduce a un simple proceso de clasificación. Cada pareja de elementos $\langle \alpha, \beta \rangle$ puede pertenecer a dos clases: la clase M (*match*) contendrá todas aquellas parejas de duplicados, mientras que la clase U contendrá el resto. Si se conocen las funciones de densidad de ambas clases, la desduplicación se convierte en un problema de inferencia bayesiana.

A partir de ahí, pueden utilizarse diferentes criterios a la hora de definir el umbral entre ambas clases, como por ejemplo, la minimización de la probabilidad de error o del coste de decisión.

2.2.2. Técnicas de aprendizaje supervisado

Así como las técnicas probabilísticas parten del conocimiento de las funciones de densidad de las clases M y U, las de aprendizaje supervisado parten de la existencia de un conjunto de datos de entrenamiento en el que cada pareja $\langle \alpha, \beta \rangle$ está etiquetada como perteneciente a una de las dos clases.

Muchas de estas técnicas tratan cada par de manera independiente. Otra aproximación, que también es aplicable a técnicas probabilísticas, consiste en postprocesar las parejas clasificadas como duplicados para formar grafos. Así, aplicando la asunción de transitividad, si $\langle \alpha, \beta \rangle$ y $\langle \beta, \gamma \rangle$ son duplicados, se asume que $\langle \alpha, \gamma \rangle$ también lo son.

Las técnicas de aprendizaje supervisado pasivo requieren grandes conjuntos de datos de entrenamiento que, si son creados artificialmente, difícilmente reflejan los casos ambiguos. El aprendizaje activo semisupervisado es una evolución en ese sentido que utiliza directamente datos reales sin etiquetar. Parte de la base de que la gran mayoría de casos pueden clasificarse como duplicados o no duplicados con un nivel de certeza alta de manera automática. En cambio, para las parejas que presentan cierto grado de incertidumbre, se delega la decisión en un humano durante la fase de entrenamiento.

2.2.3. Técnicas basadas en clasificadores determinísticos

Cuando tampoco se dispone de un conjunto de datos de aprendizaje o de la interacción de un humano, se requieren técnicas basadas únicamente en los datos sin ningún tipo de información previa. La clasificación se produce en función de una medida sobre los elementos y un umbral de decisión.

Esta métrica puede ser la distancia entre los elementos de la pareja en campos de tipo carácter o reglas lógicas más complejas basadas en comparaciones multicampo.

2.2.4. Técnicas de aprendizaje no supervisado

Tienen su origen en las técnicas probabilísticas. En ausencia de funciones de densidad, las parejas pueden agruparse en *clusters* de forma automática y sin aprendizaje previo en base únicamente a su similitud. Una vez creados los *clusters*, que se corresponden con clases desconocidas, parte de la premisa de que el hecho de identificar unas pocas parejas como pertenecientes, por ejemplo, a la etiqueta «duplicados», hace que esta identificación pueda extenderse a todo el *cluster*.

2.3. Eficiencia

Otro de los grandes retos de la desduplicación es la eficiencia. Una técnica elemental para descubrir duplicados en un conjunto de N elementos tiene un coste $O(N^2)$, que a menudo resulta prohibitivo. Es por ello que a menudo se concentra un gran porcentaje del esfuerzo en mejorar al máximo el rendimiento del proceso. En esta dirección, existen dos vías principalmente: reducir el número de comparaciones y reducir el coste por comparación.

2.3.1. Eficiencia en el número de comparaciones

Las técnicas más elementales tratan de comparar todos los elementos con todos. Sin embargo, a menudo no todas las comparaciones son necesarias. Existen estrategias para realizar un proceso de descarte *a priori* y así reducir notablemente el número de comparaciones totales. De entre ellas, podemos destacar las siguientes:

Blocking
: Consiste en dividir los datos, mediante algún procedimiento, en subconjuntos mutuamente exclusivos, de forma que pueda asumirse que no se producen duplicados entre elementos pertenecientes a diferente conjunto.

Vecinos ordenados
: Para cada elemento, se extrae una clave formada por un conjunto de campos relevantes. La lista de claves se ordena y se comparan solo aquellos elementos cuyas claves son vecinas en dicha lista ordenada.

2.3.2. Eficiencia en el coste por comparación

Otra vía para mejorar la eficiencia pasa por reducir el propio coste de cada comparación. Para ello, existen dos máximas a seguir. Por un lado, dos elementos no duplicados deben ser diferenciados cuanto antes. Por otro, dos elementos duplicados deben ser identificados con el menor número de campos posibles. Dicho en forma de reglas:

- Se deben seleccionar los campos mínimos e imprescindibles —es decir, suficientes y necesarios— para identificar un duplicado.
- Es importante el orden de comparación: debe establecerse del campo más diferenciador al menos, en la medida de lo posible.

Capítulo 3

Duplicados en tráfico de red

Las técnicas revisadas en el Capítulo 2, tanto de detección como de mejora de la eficiencia, son muy genéricas en el sentido de que su aplicación queda supeditada en gran medida a la naturaleza de los datos. De hecho, una de las enseñanzas más importantes que proporciona el estudio de dichas técnicas y de los ámbitos de aplicación es que no resulta factible una implementación directa sin una adaptación severa al problema específico [5].

En el ámbito que nos ocupa, los datos son de una naturaleza muy particular. Se generan tras un proceso de captura de paquetes en un determinado punto de la red en el que confluyen múltiples enlaces, típicamente un *switch* o un *router*.

A continuación, se realizará una revisión de los diferentes métodos de monitorización de tráfico existentes. Esto nos permitirá sentar las bases para un posterior análisis del origen de los duplicados de red y los distintos tipos implicados.

3.1. Métodos de monitorización de tráfico

La monitorización de tráfico de red consiste en la extracción de los paquetes que fluyen por un determinado punto de la red para su posterior análisis. Se trata de un procedimiento pasivo, esto es, se realizan copias de los paquetes sin interferir en el normal funcionamiento de la red en cuestión. El resultado es un flujo o *stream* de datos ordenados temporalmente según su momento de ocurrencia en el punto de monitorización.

En un entorno *switched Ethernet* cableado, disponemos principalmente de cuatro métodos para realizar esta tarea: mediante un *hub*, un computador en línea, un *tap* o *port mirroring* [6].

3.1.1. *Network hub*

Un concentrador o *hub* es un dispositivo eléctrico multipuerto que opera a nivel físico como simple repetidor de la señal: cada paquete entrante se replica por todos los

puertos salvo por el que lo recibió. Une múltiples enlaces en modo *half-duplex* como si constituyeran un mismo segmento de red a nivel físico. Esto significa que los dispositivos conectados a él deben estar sincronizados a la misma velocidad y deben coordinarse para transmitir uno cada vez o, de lo contrario, se producen colisiones.

Se trata de un dispositivo barato y cómodo, puesto que basta con monitorizar uno de los puertos para recibir todo el tráfico que lo atraviesa. No obstante, esta característica de medio compartido limita su capacidad. Un *hub* nunca supera los 100 Mbps, por lo que han quedado obsoletos en favor de los conmutadores o *switches*, y ya no se fabrican.

3.1.2. En línea

Aunque el resto de métodos de monitorización pueden considerarse elementos en línea, aquí nos referimos con este apelativo al hecho de realizar un *Machine-In-The-Middle* pasivo. Esto es, colocar una computadora en mitad de un enlace (Figura 3.1). Técnicamente es posible monitorizar de esta manera varios enlaces, pero resulta poco práctico: requiere un recableado complejo y dos interfaces por cada par de enlaces a monitorizar que actuarán como puente e introducirán un retardo adicional en los paquetes.

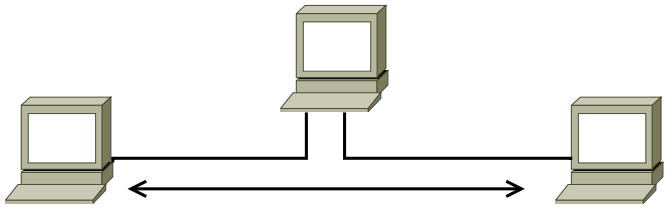


Figura 3.1: Machine-In-The-Middle

3.1.3. *Network tap*

Un *tap* clásico hereda características de los dos métodos anteriores. Por un lado, consiste en un dispositivo que se coloca en línea y parte un enlace a través de dos interfaces. Por otro lado, se trata de un dispositivo pasivo en el que la señal fluye entre ambas interfaces sin sufrir retardo, pero en el proceso replica tan solo uno de los sentidos del enlace (*half-duplex*) por una tercera interfaz.

Los hay que funcionan tanto en el dominio eléctrico como en el óptico, por lo que es una tecnología que no está limitada por la velocidad. Adicionalmente, existen fabricantes, como NetOptics, que disponen de *taps* que realizan *link aggregation* para la monitorización de múltiples enlaces en modo *full-duplex* a través de una sola interfaz.

Aunque es el mejor método de los revisados hasta ahora, sigue teniendo la desventaja del recableado —que provoca un corte temporal en el servicio— y la introducción de un equipamiento adicional voluminoso, lo que en ocasiones no es deseable o factible.

3.1.4. *Port mirroring*

Los *hubs* evolucionaron en conmutadores o *switches*, dispositivos que convierten el medio compartido en enlaces punto a punto conmutables en función de la MAC de destino. El dominio compartido sigue existiendo a nivel de enlace, pero no a nivel físico. Esto aporta múltiples ventajas, como la comunicación *full-duplex* en cada puerto con sincronización independiente, ya sin posibilidad de colisión; pero también produce una desventaja de cara a la monitorización: una máquina conectada a un puerto pasa a ver únicamente su propio tráfico (más el de *broadcast* y *multicast*).

Así pues, *port mirroring* es una funcionalidad integrada en el equipamiento de red de capa 2 y 3 —*switches*, *routers*, aunque muchas veces son indistinguibles puesto que operan en ambas capas— de alta gama para paliar la desventaja antes mencionada. *Port mirroring* es el término —podríamos llamar *genérico*— que utilizan fabricantes como HP o Juniper, aunque habitualmente se intercambia con la denominación utilizada por Cisco: *Switched Port Analyzer* o SPAN. Consiste en la capacidad de copiar el tráfico entrante —saliente, o entrante y saliente— por uno o múltiples puertos del dispositivo (fuentes) hacia un puerto de monitorización llamado puerto de captura, espejo o *mirror* (destino).

Algunos fabricantes, como HP y Cisco, ofrecen además otra característica muy útil denominada SPAN de VLAN, que consiste en escoger VLANs como fuentes del SPAN. En última instancia, esto no supone más que un atajo a la hora de monitorizar puertos, puesto que, a bajo nivel, el SPAN de VLAN es un SPAN de puertos. Este hecho queda patente al observar el resultado de monitorizar el tráfico entrante a una determinada VLAN: cuando el tráfico entra por otra VLAN no sujeta a monitorización e internamente es enrutada hacia la primera, el puerto de captura no transmite ningún paquete de dicho flujo (véase el ejemplo de la Figura 3.2).

La puesta en marcha de este método de monitorización, a diferencia de los anteriores, es inmediata y poco invasiva. Se realiza mediante una sencilla configuración en un dispositivo preexistente en la red a analizar, sin necesidad de recablear ni interferir en el servicio.

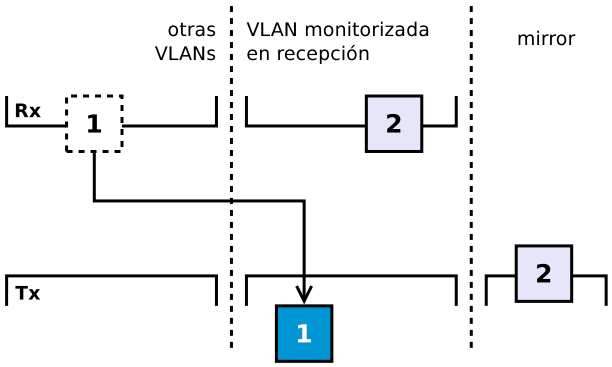


Figura 3.2: Monitorización del tráfico de ingreso a cierta VLAN.

3.2. Origen de los duplicados

Dadas sus inherentes ventajas, el *port mirroring* es el método de monitorización más utilizado en la práctica, quizás junto con el basado en *taps*, puesto que también permite el seguimiento de múltiples enlaces a las altas velocidades (1-Gigabit y 10-Gigabit) habituales en las redes de hoy en día. Los ejemplos que se utilizarán de aquí en adelante estarán basados únicamente en casos de *port mirroring* por comodidad, pero ha de notarse que serían totalmente equivalentes con el uso de *taps*.

Una vez sentadas las bases en cuanto a técnicas de monitorización, el siguiente paso es comprender el mecanismo de generación del tráfico duplicado. Existen dos fuentes: los duplicados generados por la red y los generados por el propio esquema de monitorización.

3.2.1. Generados por la red

La propia red de datos puede generar paquetes duplicados:

Debido a un defecto en la configuración
: En determinadas circunstancias, un mal funcionamiento de mecanismos de balanceo o de actualización de tablas de rutas puede producir que temporalmente un conjunto de flujos pasen varias veces por un mismo punto de la red. Los casos de malas configuraciones permanentes son más extrañas, pero también pueden producirse. En otras ocasiones, se encuentran paquetes producto de ataques y escaneos de red que entran en bucles de enrutamiento debido a que las direcciones de destino no existen.

Debido al funcionamiento inherente de la red
: Las configuraciones en las que un mismo paquete retorna a un equipo de monitorización no son comunes, pero de hecho existen. La Figura 3.3 recoge dos ejemplos, en capa 2 y en capa 3.

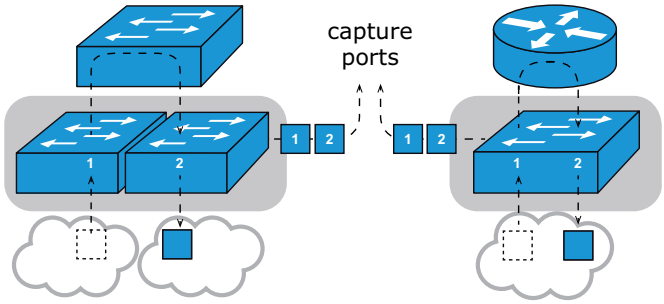


Figura 3.3: La figura de la izquierda muestra como un paquete puede retornar a un equipo de monitorización en capa 2, mientras que en la de la derecha se produce en capa 3.

Hay que tener en cuenta que estos duplicados tienen lugar durante el normal funcionamiento de una red, por lo que, aunque son susceptibles de ser analizados en busca de defectos en la misma, no interesa identificarlos en el proceso de desduplicación previo al análisis.

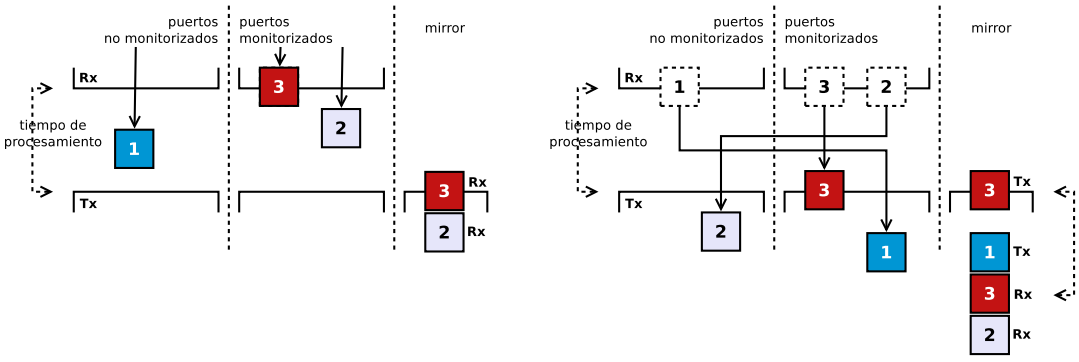
3.2.2. Generados en el equipo de captura

Como se ha visto, tenemos la capacidad de medir lo que entra o lo que sale (o ambas opciones) para cada uno de los enlaces que confluyen en un determinado punto de la red. Por lo tanto, es evidente que capturar todo el tráfico entrante implicaría un conjunto de datos libre de duplicados. No obstante, en general no resulta viable o deseable capturar todo el tráfico, ya sea por el volumen implicado, que podría saturar el puerto de captura, o por el objetivo que se persiga en análisis posteriores.

Así pues, resultan habituales configuraciones como la mostrada en la Figura 3.2, en la que se ha realizado un SPAN de VLAN y, como resultado, solo un subconjunto de los puertos son monitorizados. Este hecho obliga a monitorizar ambos sentidos —entrante y saliente— con el objetivo de no perder paquetes.

En esta situación, considérese el ejemplo de la Figura 3.4a en el que se representa un esquema de *port mirroring* en el que existen puertos monitorizados en recepción y en transmisión —que pueden pertenecer a un SPAN de VLAN— y puertos no monitorizados:

- Los paquetes del flujo (1) entran por un puerto no monitorizado y, tal y como muestra la Figura 3.4b, salen por un puerto monitorizado. Por tanto, en el puerto de captura, se genera lo que llamaremos una *copia en transmisión* (1 Tx). Nótese que esta copia nunca se habría obtenido si la monitorización hubiera sido configurada solo para el tráfico entrante.



(a) Instante de llegada de tres paquetes **(b)** Instante de salida de los paquetes.

Figura 3.4: Esquema de *port mirroring* con un subconjunto de puertos monitorizados en ambos sentidos.

- Los paquetes del flujo (2) entran por un puerto monitorizado y salen por un puerto no monitorizado. Así pues, en el puerto de captura, se general lo que llamaremos una *copia en recepción* (2 Rx). Nótese que esta copia nunca se habría obtenido si la monitorización hubiera sido configurada solo para el tráfico saliente.
- Los paquetes del flujo (3) entran y salen por enlaces monitorizados. Como resultado, en el puerto de captura aparece tanto una copia en recepción (3 Rx) como una copia en transmisión (3 Tx) del mismo paquete, lo que constituye una pareja de duplicados.

Nótese además que estos duplicados se producen de manera determinística: dada una configuración de monitorización, todos los paquetes que atraviesen el dispositivo utilizando cualquier combinación de pares de puertos monitorizados se verán duplicados, mientras que el resto no.

En este caso, estamos ante duplicados que no estaban circulando por la red: están generados artificialmente por el esquema de monitorización empleado, por lo que conviene realizar una desduplicación de estos paquetes antes de realizar cualquier análisis.

¿Cómo distinguirlos entonces de los generados por la red? Mediante el tiempo de separación entre pares. En la Figura 3.4b está marcada la separación que sufren los duplicados por *port mirroring* con lo que se ha llamado tiempo de procesamiento, es decir, el tiempo en el que el paquete es encaminado desde un puerto de entrada a uno de salida. Los duplicados generados por la red, por su parte, tendrán una separación necesariamente mayor, probablemente un orden superior.

3.3. Tipos de duplicados

Considérese de nuevo la Figura 3.4a,3.4b. El conmutador recibe un paquete del flujo (3) y lo transmite en capa 2, con lo que se generan las correspondientes copias en el puerto de captura. De acuerdo con esto, la correspondiente captura de tráfico mostrará dos paquetes exactamente iguales en un lapso corto de tiempo en el que realmente debería haber uno.

El caso descrito es el más simple y más fácil de detectar. Tanto es así que se ha comprobado que algunos dispositivos, como los HP ProCurve —no así el equipamiento de Cisco—, automáticamente detectan estos duplicados y los suprimen. La mayor parte de las veces, sin embargo, algunos campos de las cabeceras podrían cambiar en varios niveles de la pila de protocolos. En el ejemplo anterior, el citado conmutador podría cambiar, de estar presente, el identificador de VLAN 802.1Q a nivel Ethernet; o podría estar aplicando reglas de QoS y, por tanto, marcando los paquetes salientes con distintos valores DSCP a nivel IP (con lo que el checksum de la cabecera IP también cambiaría).

En resumidas cuentas, no es suficiente con buscar paquetes exactamente iguales en la identificación de duplicados como hacen algunas aproximaciones [2]. En general, algunos campos cambiarán, otros podrían o no cambiar, y otros permanecerán invariables, como la carga útil del paquete o *payload*.

Tabla 3.1: Comportamiento de los diferentes tipos de duplicados.

Tipo	Capa	Cambia	Puede cambiar
Switching	2		etiqueta VLAN
	3		etiqueta DSCP checksum
Routing	2	dirección de origen dirección de destino	etiqueta VLAN
	3	TTL checksum	etiqueta DSCP
NAT routing	2	dirección de origen dirección de destino	etiqueta VLAN
	3	TTL checksum	etiqueta DSCP dirección de origen dirección de destino
	4	checksum	puerto de origen puerto de destino
Proxying	2	dirección de origen dirección de destino	etiqueta VLAN
	3	checksum	etiqueta DSCP dirección de origen dirección de destino
	4	checksum	número de secuencia número de ACK

A continuación, definiremos distintos tipos de duplicados en función de los cambios que se produzcan al atravesar el dispositivo de captura. Las denominaciones están basadas en el mecanismo generador (más bien, en el procesado que sufre el paquete antes de volver a aparecer como copia). La Tabla 3.1 resume los cambios esperados en los diferentes niveles de la pila de protocolos para cada tipo de duplicado.

3.3.1. Duplicados de *switching*

Tal y como se ha descrito anteriormente, el conmutador de la Figura 3.4a,3.4b podría cambiar las etiquetas VLAN o DSCP (nótese que esto es cierto para cualquier paquete que atraviese un *switch* o un *router* y, por tanto, para cualquier tipo de duplicado). La característica distintiva de los duplicados de *switching* es que están constituidos, habitualmente, por parejas de paquetes idénticos, pero en general no podemos asumir que sean idénticos (véase la Tabla 3.2).

Tabla 3.2: Cambios en los duplicados de *switching*.

Ethernet	Destination MAC				
	Source MAC				
	802.1Q (optional)				Ethertype
IPv4	Ver.	HL	DSCP	E	Total length
	IP ID			F	Offset
	TTL		Protocol		Checksum
	Source IP				
	Destination IP				
	Options				
TCP	Source port				Destination port
	Sequence number				
	ACK number				
	Off	0	Flags		Window size
	Checksum				URG pointer
	Options				

Leyenda: Cambia | Puede cambiar

Tabla 3.3: Cambios en los duplicados de *routing*.

Ethernet	Destination MAC				
	Source MAC				
	802.1Q (optional)				Ethertype
IPv4	Ver.	HL	DSCP	E	Total length
	IP ID			F	Offset
	TTL		Protocol		Checksum
	Source IP				
	Destination IP				
	Options				
TCP	Source port				Destination port
	Sequence number				
	ACK number				
	Off	0	Flags		Window size
	Checksum				URG pointer
	Options				

Leyenda: Cambia | Puede cambiar

3.3.2. Duplicados de *routing*

Tómese de nuevo el ejemplo anterior y considérese que los paquetes están siendo enrutados a nivel de red en lugar de conmutados. Durante el proceso, se modifican los siguientes campos (véase la Tabla 3.3):

- La MAC de origen es reemplazada por la dirección del puerto 2.
- La MAC de destino es reemplazada por la dirección del siguiente salto.
- El TTL es decrementado en una unidad.
- El *checksum* a nivel IP es recalculado.

Tabla 3.4: Cambios en los duplicados de *routing* NAT.

Ethernet	Destination MAC					
	Source MAC					
	802.1Q (optional)				Ethertype	
IPv4	Ver.	HL	DSCP	E	Total length	
	IP ID			F	Offset	
	TTL		Protocol		Checksum	
	Source IP					
	Destination IP					
	Options					
TCP	Source port				Destination port	
	Sequence number					
	ACK number					
	Off	0	Flags		Window size	
	Checksum				URG pointer	
	Options					

Leyenda: Cambia | Puede cambiar

3.3.3. Duplicados de *routing* NAT

Si el *router* del caso precedente actúa también como dispositivo NAT [7], es esperable que sucedan cambios adicionales (véase la Tabla 3.4):

- La dirección IP de origen es reemplazada con la dirección externa del NAT si el paquete está abandonando la red privada. En la dirección contraria, es la dirección IP de destino la reemplazada por la dirección interna de la conexión a la que pertenece.
- El *checksum* a nivel de transporte también es recalculado.

El comportamiento anterior se corresponde con un NAT básico, pero puede extenderse con traducción de puertos:

- El puerto TCP/UDP de origen es reemplazado por un puerto mapeado si el paquete está abandonando la red privada. En la dirección contraria, es el puerto de destino el reemplazado por el puerto original que utilizó la máquina de la red interna.

3.3.4. Duplicados de *proxy* transparente

Los *proxies* transparentes y los *proxies* inversos (balanceadores de carga) son comunes en granjas de servidores. Estos dispositivos actúan como un NAT básico, pero con la capacidad de reescribir ciertos parámetros de la cabecera de transporte (véase la Tabla 3.5):

- El número de secuencia de TCP es modificado si el paquete viaja del cliente al servidor. En la dirección contraria, es el número de ACK el modificado.

Tabla 3.5: Cambios en los duplicados de *proxy* transparente.

Ethernet	Destination MAC					
	Source MAC					
	802.1Q (optional)				Ethertype	
IPv4	Ver.	HL	DSCP	E	Total length	
	IP ID			F	Offset	
	TTL		Protocol		Checksum	
	Source IP					
	Destination IP					
	Options					
TCP	Source port				Destination port	
	Sequence number					
	ACK number					
	Off	0	Flags		Window size	
	Checksum				URG pointer	
	Options					

Leyenda: Cambia | Puede cambiar

3.3.5. Otros casos raros

Las infraestructuras actuales están salpicadas con una gran variedad de dispositivos —cortafuegos, detectores de intrusiones, limitadores de ancho de banda, y un largo etcétera— que a veces dan lugar a comportamientos inesperados. Muchos de estos comportamientos responden a razones de seguridad. Por ejemplo, un *router* puede no decrementar el TTL para ocultar su presencia.

En otros casos, el comportamiento es consecuencia de estas medidas. Por ejemplo, un dispositivo NAT aisla de manera efectiva una red privada ante conexiones entrantes, pero también da problemas con protocolos especiales que llevan información de conexión (IP, puerto) a nivel de aplicación (la carga útil), tal y como hace el FTP activo. Por ello, algunos de estos NATs trabajan también como pasarelas a nivel de aplicación (ALG), y son capaces incluso de modificar la carga útil del paquete [8].

Muchos de estos casos son muy raros y requieren un tratamiento especial. Es por ello que la metodología desarrollada por el presente trabajo los evita.

3.4. Conclusiones

- La monitorización de redes se realiza principalmente mediante *port mirroring*. Se trata de un método que permite la extracción de tráfico a altas velocidades desde múltiples enlaces a un solo puerto de monitorización sin introducir equipamiento adicional y sin necesidad de parar el servicio de las redes en producción.

- La red, ya sea por defecto de configuración o no, puede dar lugar a duplicados que serán susceptibles de ser analizados, pero, dado que son parte del funcionamiento de la red, no interesará identificarlos en el proceso de desduplicación.

- El propio esquema de monitorización produce artificialmente duplicados que será ne-

cesario evitar en subsiguientes análisis.

- La generación de los duplicados es un proceso determinístico, es decir, dada una configuración de monitorización, todos los paquetes que sigan determinados caminos físicos se verán duplicados.
- Una captura de tŕafico de red es un *stream* de datos ordenado en el tiempo.
- Debido a los mecanismos de generación, los duplicados podrán ser de distinto tipo —lo que se traduce en que las copias no serán idénticas— y estarán muy próximos en el tiempo.

Capítulo 4

Metodología de detección de duplicados

Otros trabajos anteriores ya han abordado el tratamiento de *streams* de datos genéricos mediante el uso de ventanas deslizantes [9, 10, 11, 12, 13, 14]. Todos ellos asumen duplicados idénticos y *streams* probabilísticos. Desafortunadamente, en el Capítulo 3 se ha llegado a la conclusión de que los duplicados de red no son en general idénticos y que se generan de forma determinística, por lo que, una vez más, estos estudios no son aplicables directamente al campo que nos ocupa.

El presente capítulo está destinado a la definición de una metodología basada en ventana deslizante para la desduplicación de tráfico de red. Adicionalmente, se proporcionarán algunas pautas y consideraciones que se han seguido a la hora de llevar a cabo la implementación.

4.1. Introducción

Del Capítulo 3, se extrae que el *payload* o carga útil de los paquetes es el único fragmento que en general puede considerarse invariable tras su paso por un dispositivo de red. Por tanto, comparar tan solo el *payload* parece una buena aproximación. Además, esto simplificaría enormemente el problema, ya que los duplicados pasarían a ser idénticos. Sin embargo, conocer el tipo de los duplicados es muy útil desde el punto de vista del análisis de red de cara a entender los patrones de tráfico. Y esta información, como se ha visto, solo puede obtenerse a partir de las distintas cabeceras.

El conocimiento de los tipos de duplicados presentes en una captura permitirá realizar desduplicaciones parciales en función del objetivo. Por ejemplo, para el cálculo del factor de utilización por VLAN, los duplicados de *switching* deben ser eliminados, pero no así los de *routing*, puesto que cada pareja pertenecerá a distinta VLAN.

Nótese, además, que habrá muchos paquetes exentos de *payload* (como los paquetes

de control de TCP: SYN, ACK, FIN, RST), hecho que descarta definitivamente una aproximación que obvie las cabeceras.

La metodología descrita en las subsiguientes secciones está diseñada para ofrecer identificación exacta (no probabilística) de los duplicados sobre un modelo de ventana deslizante, al igual que otros estudios sobre *streaming data*. Dado el elevado coste computacional que esto supone, tal y como ha constatado el Capítulo 2, su aplicación estará restringida al procesado *off-line* de capturas de tráfico.

4.2. Diseño de la ventana deslizante

La Figura 4.1 describe el comportamiento de la ventana deslizante. A continuación se detallan los diferentes pasos:

1. A partir de una ventana de tamaño k, se procede a extraer el paquete n-ésimo del fichero de traza.
2. El paquete n se compara con cada uno de los paquetes de la ventana, desde el $n-1$ hasta el $n-k$.
3. Si se identifica como duplicado en la comparación i-ésima, el paquete n se marca como duplicado del $n-i$, y este último abandona la ventana. La búsqueda a través de la misma cesa.
4. El paquete n se añade a la ventana.
5. Se reajusta el tamaño de la ventana, eliminando los paquetes más viejos que procedan.

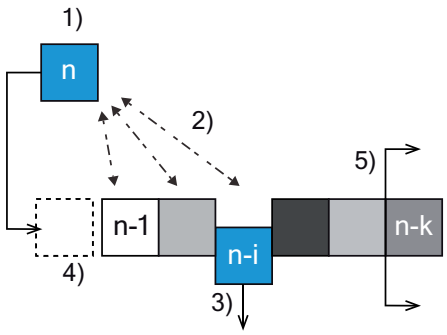


Figura 4.1: Ventana deslizante para la detección de duplicados.

Dado que la complejidad del algoritmo crece linealmente con el tamaño de la ventana, será deseable utilizar un tamaño lo más ajustado posible a la separación entre copias obtenida en el puerto de monitorización. Dicha separación puede medirse en tiempo o en número de paquetes, al igual que la propia ventana. Hallar una cota superior

para esta distancia en los duplicados debidos al proceso de monitorización, que es el propósito del próximo capítulo, permitirá reducir al máximo el número de comparaciones manteniendo la seguridad de no perder duplicados.

4.3. Procedimiento de comparación de paquetes

Recordemos, tal y como establece el Capítulo 2, que otra vía de mejora de la eficiencia consiste en reducir en lo posible el coste por comparación a través de dos medidas: escoger los campos necesarios y suficientes, y establecer un orden de forma que los datos no duplicados se descarten lo antes posible.

En el caso que nos ocupa, es lógico suponer que el *payload* será el rasgo más diferenciador entre dos paquetes cualesquiera. Dado que no está sujeto a un tamaño fijo, muchas comparaciones de *payloads* ni siquiera se producirán al no coincidir sus tamaños o el protocolo de nivel inferior. Con el objetivo de contrastar esta suposición, se ha trazado la curva de supervivencia para el proceso de comparación *byte-by-byte* de *payloads*.

A tal efecto, se obtuvo una captura en el enlace de salida de la Universidad Pública de Navarra en un día laboral. A partir de la traza desduplicada (de unos $2{\cdot}10^9$ paquetes), se hicieron comparaciones de parejas de *payloads* para cada paquete contra ventanas deslizantes de cuatro tamaños diferentes:

- 0.1 ms (total: $\sim 9 \cdot 10^9$ comparaciones).
- 1 ms (total: $\sim 6 \cdot 10^{10}$ comparaciones).
- 10 ms (total: $\sim 5 \cdot 10^{11}$ comparaciones).
- 100 ms (total: $\sim 5 \cdot 10^{12}$ comparaciones).

La Figura 4.2 muestra la curva de supervivencia para cada uno de estos tamaños de ventana. Queda patente que, en el caso peor, aproximadamente el 50 % de los pares no duplicados han quedado descartados al no coincidir el protocolo de nivel inferior o los tamaños de *payload* (en la gráfica se corresponde con la caída en $x = 0$), y el siguiente 49 % ha sido descartado mediante la comparación de tan solo dos bytes.

Este hallazgo constata que

1. comparar el *payload* de los paquetes en primer lugar es el enfoque más eficiente.
2. a la vista de la gráfica, 60 bytes deberían ser suficientes para asegurar que dos *payloads* son iguales.

Una vez se ha determinado que dos paquetes contienen el mismo *payload*, el siguiente paso consiste en comparar las cabeceras de los paquetes, tal y como establece el análisis realizado en el Capítulo 3, con el objetivo de verificar si se trata de un duplicado y de qué tipo.

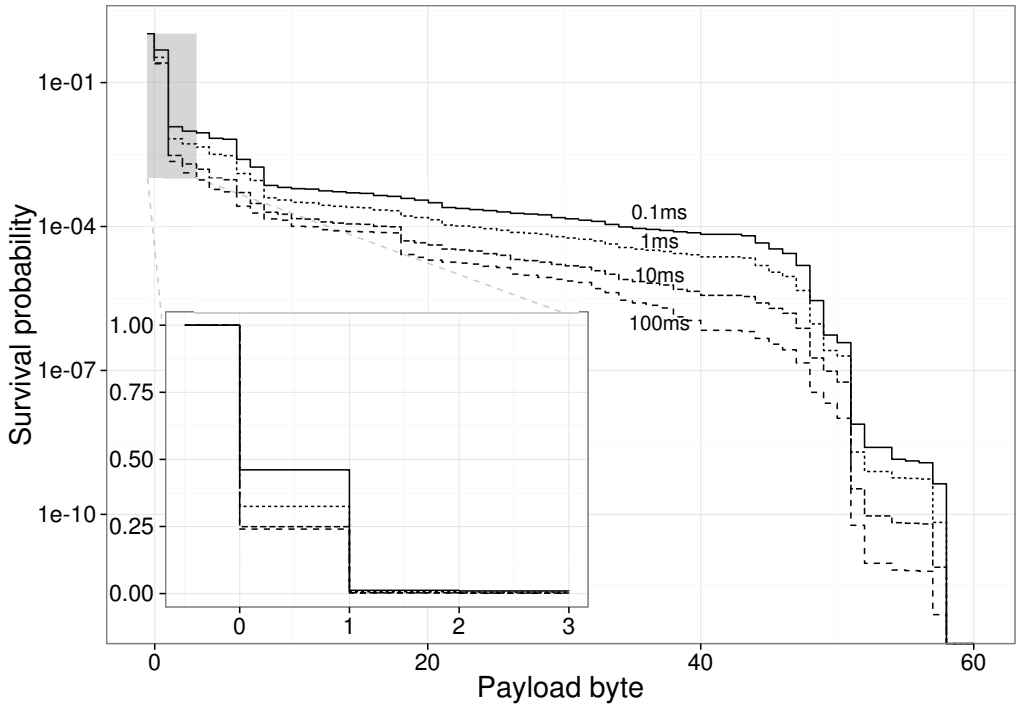


Figura 4.2: Curvas de supervivencia para el proceso de comparación *byte-by-byte* de *payloads* de paquetes en una traza desduplicada. Se han considerado cuatro tamaños diferentes para la ventana deslizante.

4.4. Consideraciones sobre la implementación

La metodología aportada describe unas reglas generales de diseño. A partir de ahí, las implementaciones concretas pueden variar en los detalles. Para cerrar el capítulo, se recogen unas pinceladas de las consideraciones que se han tomado a la hora de implementar un prototipo.

4.4.1. Gestión de la memoria

La técnica de ventana deslizante conlleva introducir y sacar paquetes de la misma constantemente. Si por cada paquete nuevo se produce una reserva de memoria y por cada salida se procede a liberar la memoria previamente ocupada, significa que, para N paquetes, se llevan a cabo N reservas y N liberaciones. Para evitar este derroche de recursos, se ha implementado un *buffer* circular doblemente enlazado para almacenar la ventana de paquetes (véase la Figura 4.3).

Dado que el tamaño de dicha ventana estará limitado, la memoria utilizada no crecerá indefinidamente, sino que alcanzará una cota máxima en algún momento. Por tanto, la estrategia más eficiente consiste en reservar memoria en grandes porciones

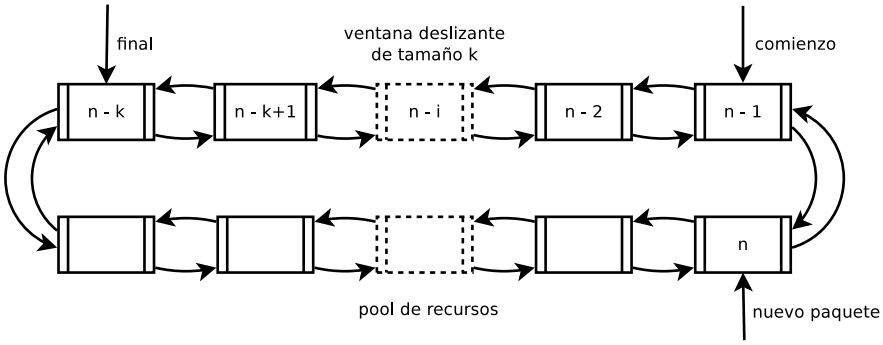


Figura 4.3: *Buffer* circular doblemente enlazado.

y no liberarla hasta finalizar la ejecución. Así, para sacar un paquete de la ventana, simplemente se mueve el puntero de final de ventana, y la memoria que ocupa queda en el *pool* de recursos disponibles. Al introducir un paquete, se utiliza una estructura de dicho *pool*, si existe alguna disponible. En caso contrario, se vuelve a reservar otra porción grande de memoria.

4.4.2. Lógica de detección

Para cada paquete extraído de la traza, se reserva memoria dentro del *buffer* circular y se pasa por referencia a una función que realiza la búsqueda sobre la ventana deslizante. Dicha función realiza las comparaciones generales, comenzando, de acuerdo con la sección anterior, por el *payload*. Si se cumplen estas condiciones generales, se llama a los comparadores específicos de cada tipo de duplicado, los cuales se encargan de verificar los campos de las cabeceras implicadas. La Figura 4.4 representa esta lógica de detección.

La implementación realizada soporta únicamente IPv4 a nivel de red. Este protocolo permite que un *router* produzca fragmentación al encaminar los paquetes por un enlace de salida cuya unidad máxima de transferencia (MTU) sea inferior a la de entrada. Nuestro prototipo contempla dicha posibilidad y es capaz de detectar fragmentos IP como duplicados de un paquete anterior completo.

4.4.3. Salida

El hecho de que esta metodología trabaje *off-line* sobre capturas almacenadas en disco, unido al interés en desduplicar diferentes tipos en función del análisis que se desee realizar, desaconseja la eliminación directa de los duplicados. En su lugar, el prototipo ofrece una salida de texto plano indicando la posición, el tipo e información adicional de cada duplicado. Posteriormente, los programas de análisis podrán utilizar esta salida para evitar o no dichos paquetes en función del tipo u otros parámetros. La Tabla 4.1 recoge una posible línea de salida.

Tabla 4.1: Ejemplo de línea de salida para un duplicado.

Índice	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Código	35	2	1	0	0	0.01	1

1. Posición del paquete duplicado dentro de la traza.
2. Diferencia en número de paquetes con el que ha dado positivo. En este caso, el original se encontraba dos paquetes antes, en la posición 33.
3. Tipo de duplicado, según un código numérico. En este caso, se trata de un duplicado de *routing*.
4. Indicador de cambio en identificador de VLAN. La etiqueta 802.1Q, así como el campo DSCP de IP, puede variar en todos los tipos de duplicados. Por ello, ambos se comparan *a posteriori*, una vez detectado un duplicado, y se incluyen como información adicional en la salida.
5. Indicador de cambio en etiqueta DSCP.
6. Diferencia de tiempo, en milisegundos, con el paquete original.
7. Diferencia de TTL con el paquete original. Este campo tampoco se compara nunca, y se incluye como información adicional. Esto permite detectar duplicados tras varios saltos, duplicados desordenados o aquellos en los que no se modifica el TTL para ocultar el dispositvo.

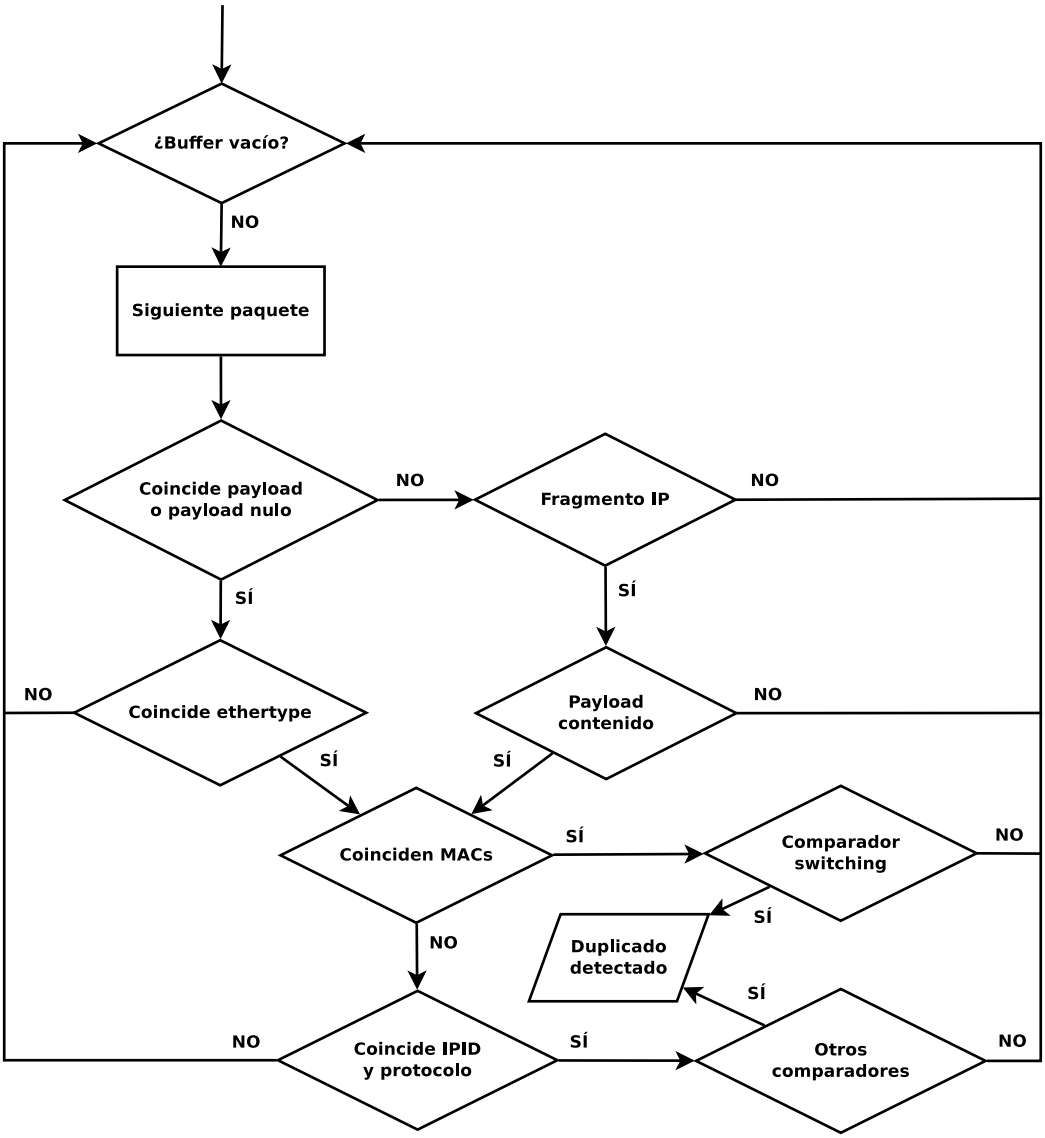


Figura 4.4: Lógica de detección de la implementación realizada.

Capítulo 5

Estudio analítico y experimental

En el Capítulo 3, se han analizado las causas de los duplicados de red y, en el Capítulo 4, se aporta una metodología basada en ventana deslizante para su identificación exacta. Se ha llegado a la conclusión de que el proceso de desduplicación debe ceñirse a eliminar los duplicados generados artificialmente por el propio esquema de monitorización, puesto que los generados por la red existen durante el normal funcionamiento de la misma y son susceptibles de ser analizados posteriormente en busca de fallos o vectores de ataque.

Estos duplicados generados artificialmente se traducen en parejas de paquetes compuestas por lo que hemos llamado *copia en recepción* y *copia en transmisión* —aludiendo a su momento de generación—. Dichas copias se encuentran separadas en el puerto de captura por un intervalo de tiempo y un número de paquetes presumiblemente muy pequeño, pero desconocido *a priori*. Asimismo, se han descrito las fuentes de esta separación temporal, que probablemente es la causante de que, en general, las parejas de duplicados no aparezcan de forma consecutiva en el puerto de captura.

En aras de la eficiencia de la metodología propuesta, se hace necesario tratar de buscar una cota superior para la separación entre la copia en recepción y la copia en transmisión, en términos de tiempo o de número de paquetes, si alguno de los parámetros resulta predecible. Así pues, este capítulo recoge un estudio analítico y experimental que analiza el comportamiento de ambos parámetros.

5.1. Modelo analítico

Los fabricantes no aportan demasiados detalles de la arquitectura interna de sus dispositivos. Sin embargo, una sencilla prueba como es sobrecargar —digamos, al doble de su capacidad— un puerto de salida monitorizado de un conmutador da como resultado pérdidas en el puerto de captura —cuando la capacidad de este último es superior y, por tanto, sería capaz de acoger todo ese tráfico—. De aquí se puede inferir que las copias hacia el puerto de captura se realizan una vez pasadas las colas y no antes de entrar en ellas.

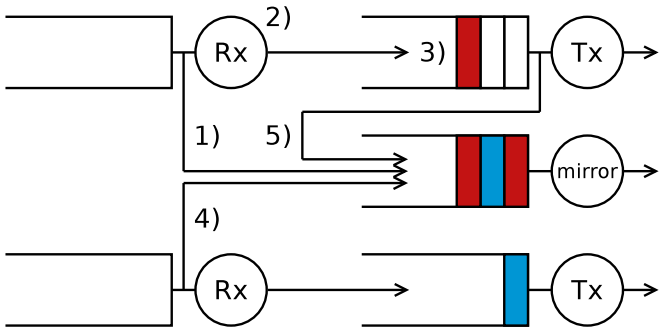


Figura 5.1: Modelo de colas para un esquema de *port mirroring*.

Basándonos en este dato, podemos modelizar un esquema de *port mirroring* tal y como muestra la Figura 5.1.

5.1.1. Separación temporal

A continuación y a partir del modelo de la Figura 5.1, se detallan los retardos que sufren los paquetes durante la generación de una pareja de duplicados:

1. El paquete n-ésimo llega a una interfaz monitorizada y se copia al puerto de captura. Esta copia en recepción sufre un retardo en cola w'_n.

2. Dicho paquete tiene que ser encaminado (conmutado o enrutado) hacia una interfaz de salida. Así pues, sufre un retardo en recepción que llamaremos tiempo de servicio en recepción o x_n.

3. El paquete ya ha sido encaminado y espera en la cola de salida de una de las interfaces monitorizadas un tiempo w_n.

4. Durante el tiempo de procesamiento $s_n = x_n + w_n$ del paquete n, otros paquetes pueden ser servidos por cualquier interfaz monitorizada (la Figura 5.1 recoge un ejemplo en recepción, pero puede suceder en transmisión) y, por tanto, aparecerán en el puerto de captura.

5. El paquete n termina su tiempo de espera en cola y se transmite, no sin antes enviar una copia al puerto de captura. Esta copia en transmisión sufre un retardo en cola w''_n.

En resumidas cuentas, la diferencia de tiempo Δt_n entre duplicados, es decir, entre copia en recepción y copia en transmisión, para el paquete n-ésimo vendrá dada por la siguiente expresión:

$$\Delta t_n = -w'_n + x_n + w_n + w''_n = s_n + \left(w''_n - w'_n\right) \tag{5.1}$$

En un estado estacionario, el retardo que sufrido por la copia en recepción será —en promedio— igual al de la copia en transmisión, por lo que $\bar{w}'' - \bar{w}' = 0$. Por tanto, aplicando el operador esperanza a la Ecuación (5.1):

$$\bar{\Delta t} = \bar{x} + \bar{w} = \bar{s} \tag{5.2}$$

5.1.2. Separación en paquetes

Durante el tiempo de procesamiento s_n de un paquete que generará un duplicado (que ha generado una copia en recepción y próximamente generará otra copia en transmisión), todo aquel paquete que sea procesado por un puerto monitorizado pondrá una copia en el puerto de monitorización. Por tanto, la separación en número de paquetes entre la copia en recepción y la copia en transmisión, siempre según nuestro modelo de la Figura 5.1, será proporcional al tiempo de procesamiento s_n del paquete que genera los duplicados, al número de puertos monitorizados y a la tasa de tráfico en cada uno de ellos.

En definitiva, sea μ_i la tasa de servicio instantánea del i-ésimo puerto monitorizado, podemos definir la separación media en paquetes, $\bar{\Delta m}$, de la siguiente manera:

$$\bar{\Delta m} = \sum \mu_i \bar{s} \tag{5.3}$$

5.2. Estudio experimental

Para el estudio experimental, contamos con el siguiente equipamiento:

- Un conmutador capa 2/3 Cisco Catalyst 3560.
  - 8 puertos FastEthernet, a 100 Mbps.
  - 1 puerto Gigabit (utilizado como puerto de captura).
- Dos servidores Dell PowerEdge T110.
  - CPU Intel Xeon X3460 a 2.8 GHz.
  - RAM 4 GB DDR3.
  - Tarjeta PCIe Intel E1G44ET con 4 puertos Gigabit.

El modelo analítico propuesto predice que la separación media en tiempo entre copias de un mismo paquete será igual al tiempo de procesamiento de dicho paquete, lo cual depende del tiempo de servicio en recepción y el tiempo de espera en cola de salida, pero no del estado de otros puertos, incluido el puerto de captura.

Por otro lado, también predice que la separación media en número de paquetes dependerá de la carga en otros puertos monitorizados, por lo que su variabilidad, en principio, será mayor.

El objetivo, por tanto, será el estudio de la validez de las Ecuaciones (5.2, 5.3) como herramientas de análisis a la hora de establecer una cota para el tamaño de ventana necesario.

5.2.1. Escenario sin encolado

En primer lugar, se plantea un escenario sin encolado en el puerto de salida ($\bar{w} = 0$) para aislar el tiempo de servicio en recepción, $\bar{x}$, en la Ecuación (5.2).

5.2.1.1. Diseño experimental

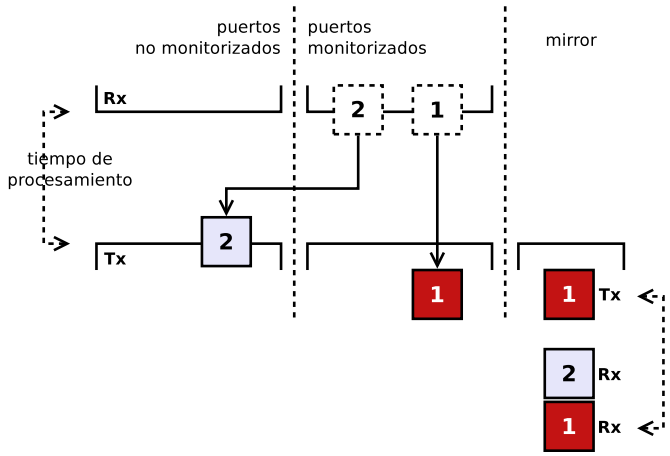


Figura 5.2: Escenario sin encolado

La Figura 5.2 esquematiza el montaje de este escenario. Se monitorizan un conjunto de puertos FastEthernet y se escoge el puerto Gigabit para la sonda. Se establecen dos flujos de paquetes. Un servidor actúa como emisor a través de dos de sus interfaces y otro como receptor y sonda, a través de tres interfaces. Los flujos aparecen numerados en la figura como 1 y 2:

1. **Flujo principal**. Se fuerza a que sea monitorizado tanto en recepción como en transmisión, por lo que generará duplicados en el puerto de captura.

2. **Flujo interferente**. Se monitoriza únicamente en recepción.

Para simplificar el análisis, los flujos se configuran a tasa constante y tamaño de paquete constante mediante *pktgen* [15], el generador de paquetes del Kernel de Linux. El flujo principal se fija con paquetes de tamaño máximo (*payload* + cabeceras + preámbulo MAC + delimitador de trama + CRC + *inter-frame gap* = 1538 octetos) para maximizar el tiempo de servicio, con un *throughput* de 95 Mbps (ligeramente por debajo de la capacidad máxima para asegurar que no se forman colas).

Mediante un barrido del flujo interferente, seremos capaces de verificar la relación de la Ecuación (5.3), que para este escenario se simplifica en la siguiente expresión:

$$\bar{\Delta m} = \mu\bar{x} \tag{5.4}$$

donde el tiempo de procesamiento se reduce al tiempo de servicio en recepción y μ es la tasa del flujo interferente en paquetes por segundo (pps). S escogen diversas configuraciones del flujo interferente mediante combinaciones de 3 tamaños de paquete —84, 354 y 624 bytes, siempre contando hasta el *inter-frame gap* a no ser que se diga lo contrario— y de 7 tasas de bit —desde 10 hasta 70 Mbps por segundo en pasos de 10—, tal y como muestra la Tabla 5.1 junto con su equivalencia en paquetes por segundo.

Tabla 5.1: Configuraciones del flujo interferente.

Tamaño (bytes)	**Tasa (Mbps)**	**Tasa (pps)**
84	10	20832
84	20	41664
84	30	62496
84	40	83329
84	50	104161
84	60	124994
84	70	145826
354	10	3788
354	20	7576
354	30	11363
354	40	15151
354	50	18938
354	60	22725
354	70	26513
624	10	2084
624	20	4166
624	30	6250
624	40	8333
624	50	10416
624	60	12500
624	70	14583

Para cada una de las configuraciones del tráfico interferente, se tomó una captura con aproximadamente 20000 duplicados que posteriormente se analizó para extraer diferencias de tiempos y de paquetes entre parejas de duplicados.

5.2.1.2. Resultados y discusión

El tiempo de servicio en recepción x_n, es decir, el tiempo de decisión para encaminar un paquete por una interfaz de salida, debe ser del orden de la velocidad de línea del puerto con más capacidad. Dicho de otra forma, para poder utilizar el puerto de 1 Gigabit, el conmutador C3560 tiene que *pensar* al menos a 1 Gigabit. En este escenario, dado que los paquetes son de tamaño máximo, podemos estimar este tiempo de servicio en recepción de la siguiente manera:

$$\bar{x} = \frac{1538 \cdot 8 \text{ [bits]}}{1 \text{ [Gbps]}} = 0{,}0123 \text{ [ms]} \tag{5.5}$$

La Figura 5.3 muestra cómo la media experimental —que es independiente del tráfico interferente, como se esperaba— se ajusta al valor calculado. Asimismo, al introducir dicho tiempo de servicio en la Ecuación (5.4), se obtiene el número medio de paquetes entre parejas de duplicados en función de la tasa del flujo interferente. La Figura 5.4 muestra la predicción teórica y el ajuste experimental.

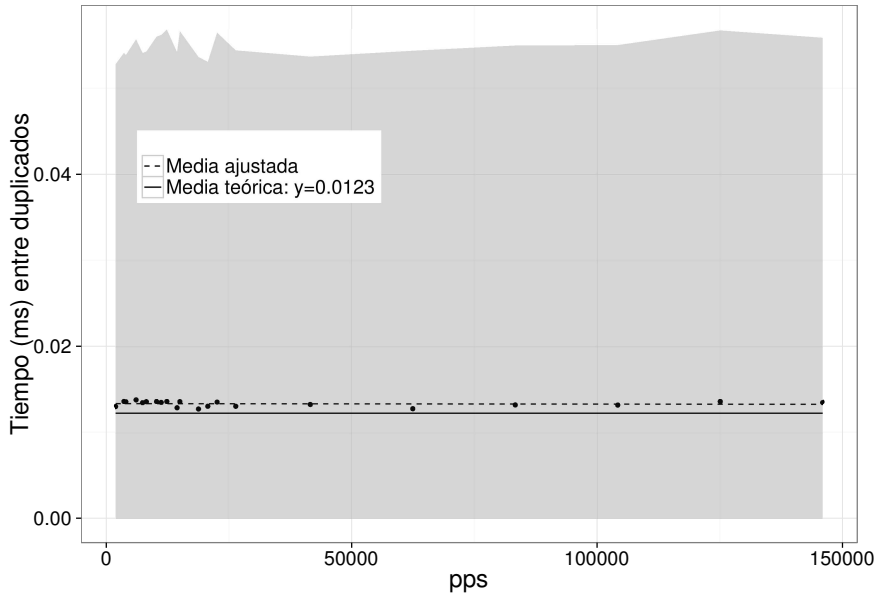


Figura 5.3: Diferencia de tiempo media entre parejas de duplicados (el área coloreada muestra la desviación estándar).

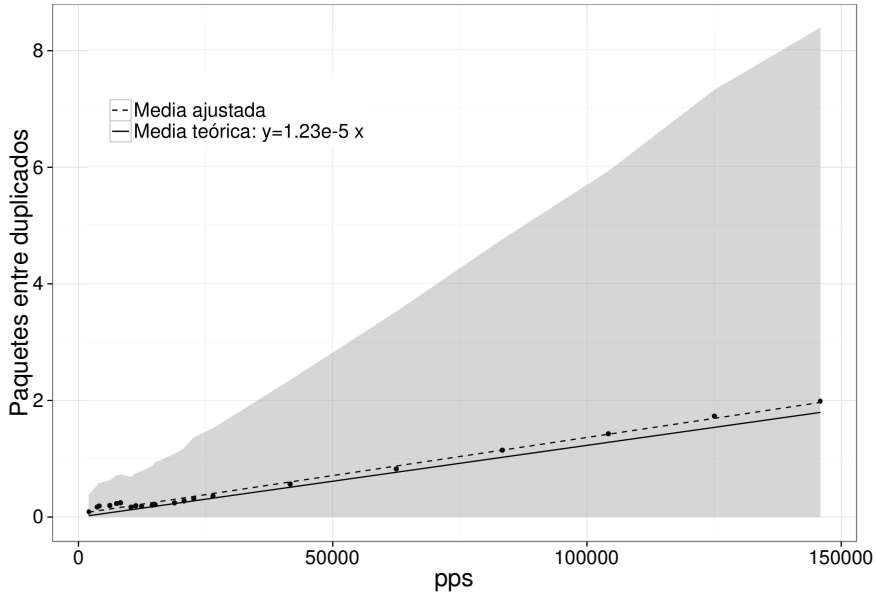


Figura 5.4: Diferencia de paquetes media entre parejas de duplicados (el área coloreada muestra la desviación estándar).

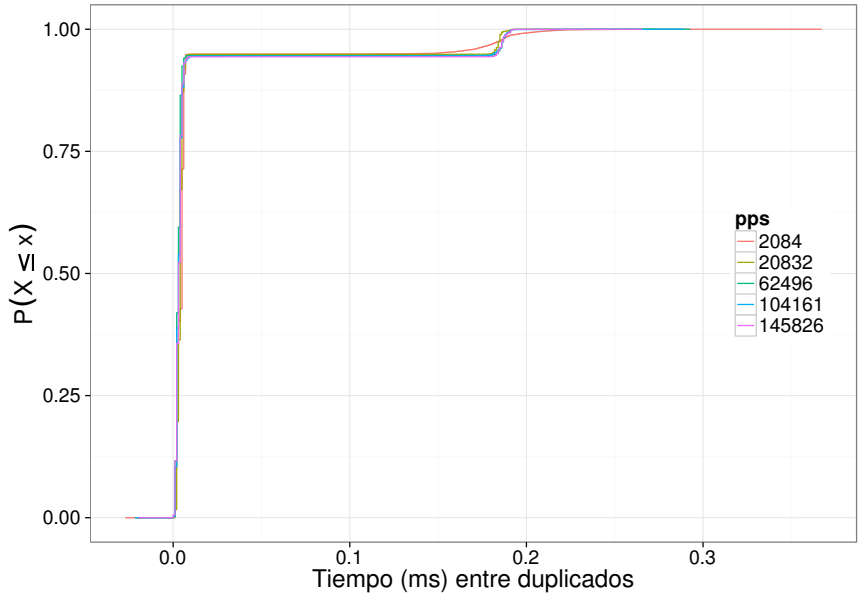


Figura 5.5: Función de distribución acumulativa (CDF) de la diferencia de tiempo entre parejas de duplicados.

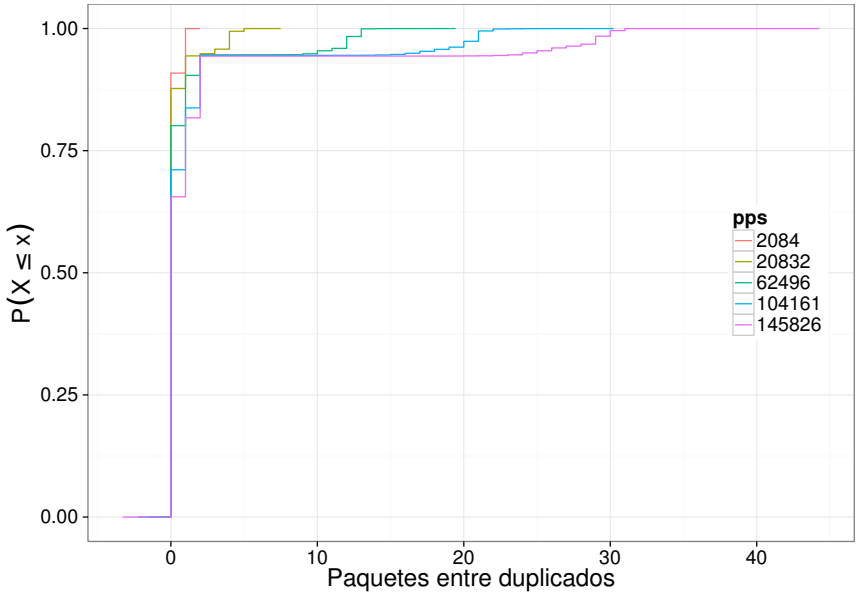


Figura 5.6: Función de distribución acumulativa (CDF) de la diferencia de paquetes entre parejas de duplicados.

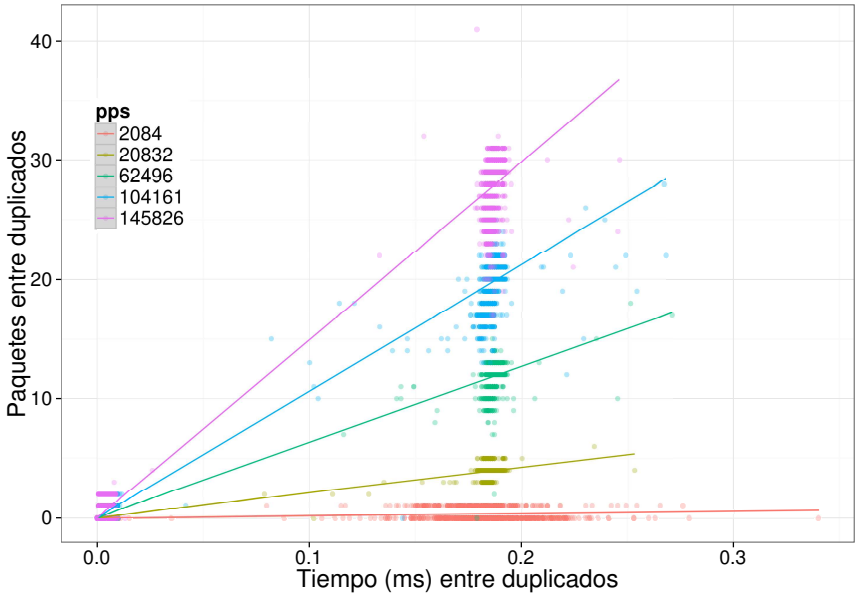


Figura 5.7: Correlación entre la tasa de tráfico interferente, la diferencia de tiempo y la diferencia de paquetes.

La distribución acumulativa de ambos parámetros aporta algo más de información. La Figura 5.5 recoge la CDF de las diferencias de tiempos, que en algún caso llegan a superar 0,35 ms. Pero lo más notable es la presencia de un escalón: aproximadamente un 5 % de los duplicados sufren un retardo de alrededor de 0,185 ms, un valor muy superior al tiempo de servicio medio.

Esta anomalía, que quizás se aprecia mejor en la Figura 5.7, podría ser achacable a posibles efectos de espera en cola en el puerto de captura si no fuera porque también se encuentra presente en la CDF de diferencias de paquetes (Figura 5.6), puesto que la cola del puerto de captura —recordemos, Ecuación (5.3)— no afecta al número de paquetes entre duplicados (siempre asumiendo ausencia de pérdidas en colas). Por consiguiente, la única explicación plausible es que se deba a algún proceso interno de actualización de los esquemas de conmutación o similar.

5.2.2. Escenario con encolado

Una vez caracterizado el tiempo de servicio en recepción, se plantea un escenario que explore el efecto de la cola del puerto de salida causante de los duplicados.

5.2.2.1. Diseño experimental

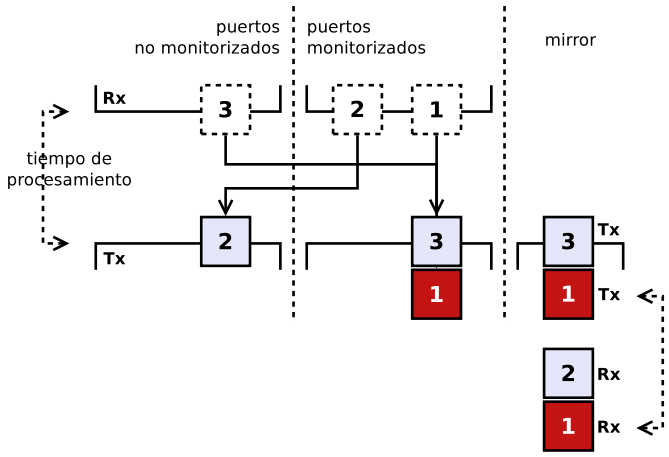


Figura 5.8: Escenario con encolado

Se propone el escenario de la Figura 5.8. El montaje, en esencia, es el mismo que el de la Figura 5.2: un flujo principal generador de duplicados (1) y un flujo interferente (2) con el que se realizarán diferentes experimentos. A esto se añade un **flujo auxiliar** (3) entrante por un puerto no monitorizado y saliente por el mismo puerto que el flujo principal. El objetivo de la adición de este tercer flujo será forzar la formación de colas en el puerto de salida del flujo principal para estudiar el impacto de este retardo adicional ($\bar{w} \neq 0$).

Tanto el flujo principal como el auxiliar quedan fijados con paquetes de tamaño máximo (1538 octetos). Dado que en el escenario anterior se ha determinado que el tamaño de paquete del tráfico interferente no influye en los parámetros a medir, se fijan paquetes de tamaño mínimo (84 octetos) para este flujo, lo que permite barrer un mayor rango de paquetes por segundo. Los tres flujos son generados con tiempos entre llegadas exponenciales haciendo uso del *software* D-ITG [16].

Si no hay pérdidas en la cola de salida, es equivalente a una cola infinita. Además, si suponemos llegadas de Poisson y un tiempo de servicio en transmisión determinístico e igual al tiempo de tranmisión de un paquete de tamaño máximo, la cola de salida puede ser descrita mediante un modelo M/D/1. Basándonos en esta premisa, podemos calcular el tamaño medio de la cola [17]:

$$\bar{N}_q = \frac{\rho^2}{2(1-\rho)} \tag{5.6}$$

y el tiempo medio en el sistema:

$$\bar{s} = \frac{\delta}{2}\left(\frac{2-\rho}{1-\rho}\right) \tag{5.7}$$

donde ρ, δ son el factor de utilización y el tiempo de servicio en transmisión respectivamente. La longitud de la cola de salida por defecto en el C3560 es de 40 paquetes, aunque puede configurarse hasta un máximo de 4096. Si establecemos un tamaño medio $\bar{N}_q = 5$, la Ecuación (5.6) da un factor de utilización $\rho \approx 0{,}916$, o 91,6 de 100 Mbps de capacidad del puerto, repartidos a partes iguales entre los flujos principal y auxiliar (45,8 Mbps, o 3722 pps, por flujo).

Por último, la Tabla 5.2 recoge las tasas generadas en el flujo interferente para cada uno de los experimentos realizados.

Tabla 5.2: Configuraciones del flujo interferente.

Rate (pps)	Rate (Mbps)
5000	3.36
25000	16.8
45000	30.24
65000	43.68
85000	57.12
105000	70.56
125000	84.0

5.2.2.2. Resultados y discusión

El tiempo de servicio en transmisión se obtiene como sigue:

$$\delta = \frac{1538 \cdot 8 \text{ [bits]}}{100 \text{ [Mbps]}} = 0{,}123 \text{ [ms]} \tag{5.8}$$

Con este valor y el factor de utilización empleado en los experimentos ($\rho = 0{,}916$), la Ecuación (5.7) predice un tiempo medio en el sistema $\bar{s} \approx 0{,}79$ ms. La Figura 5.9 compara este valor con el obtenido experimentalmente. Al igual que en el caso del escenario sin encolado, es constante e independiente del flujo interferente.

Como era de esperar, el modelo M/D/1 sobreestima ligeramente dicho tiempo medio, dado que supone llegadas de Poisson y el escenario propuesto inyecta únicamente dos flujos en la cola. No obstante, predice con bastante precisión el número de paquetes entre duplicados, tal y como se aprecia en la Figura 5.10.

Las distribuciones acumulativas muestran que la diferencia de tiempos ocasionalmente llega hasta los 8 ms (Figura 5.11), mientras que la diferencia de paquetes supera los 800 (Figura 5.12). Por último, la Figura 5.13 muestra la correlación entre la tasa de tráfico interferente, la diferencia de tiempo y la diferencia de paquetes.

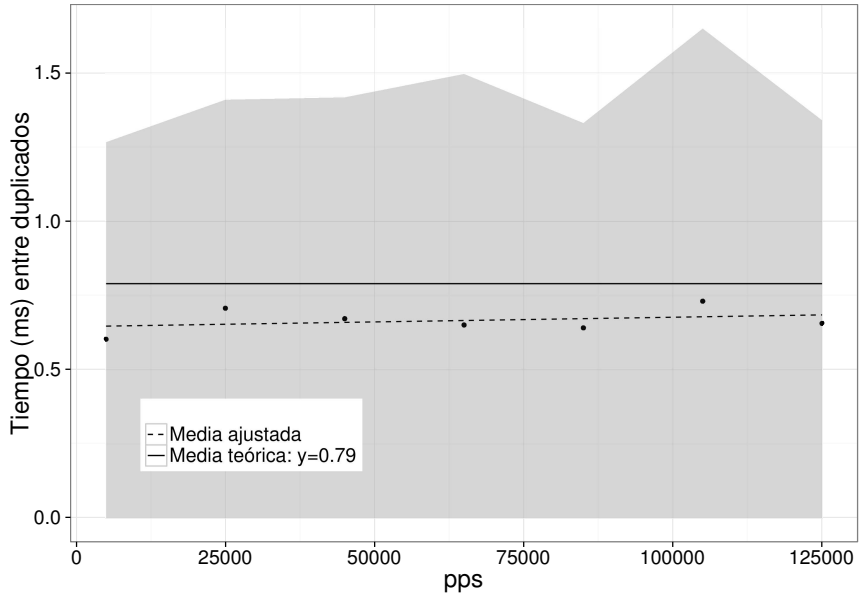


Figura 5.9: Diferencia de tiempo media entre parejas de duplicados (el área coloreada muestra la desviación estándar).

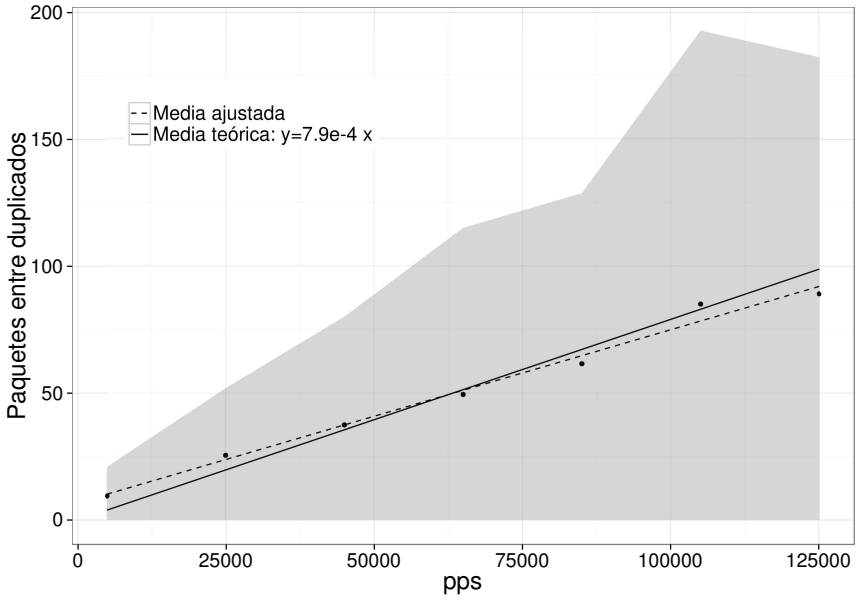


Figura 5.10: Diferencia de paquetes media entre parejas de duplicados (el área coloreada muestra la desviación estándar).

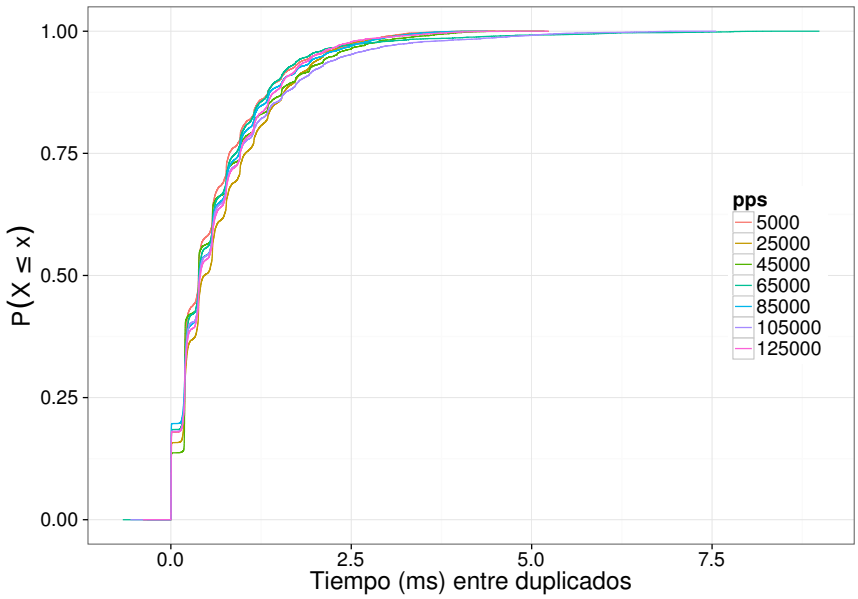


Figura 5.11: Función de distribución acumulativa (CDF) de la diferencia de tiempo entre parejas de duplicados.

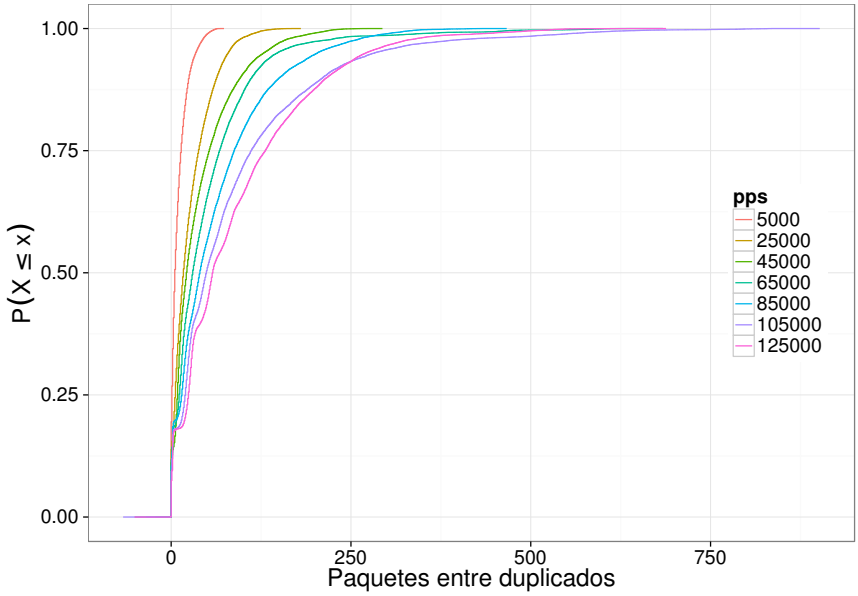


Figura 5.12: Función de distribución acumulativa (CDF) de la diferencia de paquetes entre parejas de duplicados.

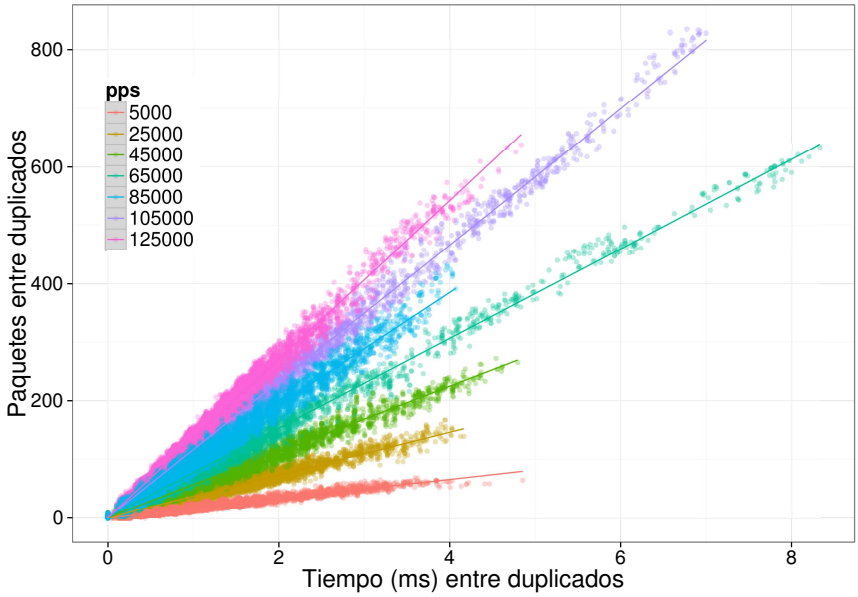


Figura 5.13: Correlación entre la tasa de tráfico interferente, la diferencia de tiempo y la diferencia de paquetes.

5.3. Dimensionamiento de la ventana deslizante

Los resultados obtenidos reflejan que la distancia entre duplicados muestra mayor variabilidad en términos de paquetes. De hecho, el número de paquetes que se introducen entre la copia en recepción y la copia en transmisión depende tanto del tiempo de procesamiento como de la tasa servicio de otros puertos monitorizados, como se aprecia claramente en la Figura 5.13. Mientras tanto, la diferencia de tiempo depende únicamente del tiempo de procesamiento; de hecho, es igual al tiempo de procesamiento, entendido como el tiempo de servicio en recepción, más el tiempo de espera en cola de salida (véase la Ecuación (5.2)).

Se ha visto que el tiempo de servicio en recepción es despreciable comparado con el tiempo de espera en cola. Matemáticamente, $x_n << w_n$. Este hecho sugiere que puede establecerse un límite superior para el tiempo de procesamiento como sigue:

$$max(s_n) \approx max(w_n) = \frac{N_q M}{C} \tag{5.9}$$

donde N_q, M, C son la longitud máxima de la cola de salida, la longitud máxima de paquete y la capacidad del enlace respectivamente. Con los valores de nuestro escenario, $N_q = 40$, $M = 1538 \cdot 8$ bits y $C = 100$ Mbps, obtenemos $max(s_n) \approx 5$ ms.

No obstante, y aunque la Figura 5.13 muestra que en general se cumple este límite, puede apreciarse que algunos puntos lo exceden hasta en 3 ms. Además, este retardo adicional viene acompañado por un aumento proporcional de la diferencia de paquetes, por lo que no puede ser un efecto de la cola del puerto de captura. En definitiva, los procesos internos del conmutador pueden aumentar considerablemente el tiempo de servicio en recepción en algunos casos.

Por otra parte, este estudio no tiene en consideración los efectos potenciales de un puerto de captura congestionado. Los escenarios propuestos manejan factores de utilización de hasta 0,2 en dicho puerto, por lo que el retardo adicional que introduce es despreciable.

En resumen, la diferencia de tiempo entre copias de un paquete en el puerto de captura tiene tres contribuciones:

- El tiempo de servicio en recepción, x_n.
- El retardo de la cola de salida, w_n.
- El retardo de la cola del puerto de captura, $w''_n - w'_n$.

En vista de los resultados y siendo muy cautelosos, asumiremos que se puede llegar al extremo de que $x_n, (w''_n - w'_n) \sim w_n$. De esta forma, puede modificarse la Ecuación (5.9) para inferir una regla de dimensionamiento la para ventana deslizante en términos de tiempo:

$$WS = \frac{3 N_{qmax} M}{C_{min}} \tag{5.10}$$

donde N_{qmax}, M, C_{min} son el tamaño máximo de la cola más larga, el tamaño máximo de paquete y el enlace con menor capacidad respectivamente. En nuestro escenario, sería recomendable una ventana de $WS = 15$ ms.

5.4. Conclusiones

- Se ha propuesto un modelo analítico para caracterizar la separación media de duplicados generados en la monitorización basada en port mirroring.
- Las predicciones de dicho modelo han sido validadas experimentalmente.
- En base a estos resultados, se ha propuesto una cota superior en términos de tiempo para la ventana deslizante de la metodología de detección desarrollada.
- Esta regla de dimensionamiento supone una cota muy conservadora. Por tanto, para reducir al máximo el número de comparaciones y así aumentar la eficiencia en la medida de lo posible, sería interesante la exploración de algoritmos dinámicos de adaptación de la ventana.

Capítulo 6

Discusión

Para finalizar, se ofrece una breve discusión sobre los usos de la herramienta desarrollada y las posibles líneas futuras.

6.1. Usos de la herramienta

El presente trabajo surge principalmente de la necesidad, como se ha tratado de demostrar en los Capítulos 1 y 2, y su aplicación práctica es directa en los ámbitos de la monitorización y análisis de tráfico de red. No en vano, el prototipo implementado en base a la metodología propuesta en el Capítulo 4 ha sido profusamente testeado y empleado con éxito por la empresa Naudit [18] en diversos proyectos. Esta experiencia de uso ha ido moldeando las capacidades de la herramienta, pero sobre todo ha confirmado su practicidad en dos vertientes: como paso previo al análisis y como herramienta de análisis en sí misma.

6.1.1. Como paso previo al análisis

Este es su cometido original y el objeto de estudio del Capítulo 5: la detección de duplicados generados artificialmente por el esquema de monitorización. En este aspecto, destaca como herramienta fundamental posterior al proceso de captura y anterior a cualquier tipo de análisis que se plantee sobre los datos recogidos.

6.1.2. Como herramienta de análisis

Durante el Capítulo 3, se ahondan en las causas de los duplicados de red. Así, se ha visto que en la propia red, en ausencia de monitorización, se producen duplicados de diversa índole. Por tanto, como efecto colateral positivo, el prototipo desarrollado ha demostrado su utilidad como herramienta de análisis de estos fenómenos. Más concretamente, es capaz de detectar los siguientes efectos indeseables:

- Bucles de *switching*, que pueden producirse por errores de configuración o fallos en el protocolo de *spanning tree*.
- Bucles de enrutamiento, de nuevo producidos por errores de configuración o por ataques y escaneos de red hacia direcciones inexistentes dentro de organizaciones, lo que produce a menudo paquetes erráticos en sus redes.

El uso como herramienta de análisis requiere la ampliación del tamaño de la ventana deslizante, puesto que este tipo de duplicados se dan en intervalos de tiempo más grandes, del orden de decenas o centenares de milisegundos. Este hecho, en principio, aumenta la probabilidad de falsos positivos porque la ventana abarca un tiempo suficiente como para acoger parejas de paquetes que portan datos iguales pero no constituyen un duplicado, a saber:

- Retransmisiones inherentes al funcionamiento de TCP.
- Paquetes de *broadcast* o *multicast* repetidos. Algunos protocolos que utilizan estos métodos de comunicación, como IGMP, repiten los mensajes múltiples veces para minimizar problemas derivados de pérdidas puntuales.
- Paquetes sin *payload* como el mensaje de control RST de TCP, utilizado para cerrar abruptamente una conexión y que habitualmente envían por parejas o tríos.

Todos estos posibles falsos positivos se discriminan fácilmente por el campo de identificación de IPv4, diferente en todos los paquetes IPv4, por lo que en la práctica no suponen un problema. No obstante, con la inminente implantación de IPv6, este campo desaparece de la cabecera IP, lo que supondrá un reto de cara a evitar tales falsos positivos.

Un falso positivo frecuente que sucede incluso con IPv4 es el provocado por paquetes RST consecutivos con IP ID nulo. Esto se debe a que muchos equipos, al rechazar conexiones mediante estos paquetes, no se molestan siquiera en rellenar este campo para evitar el consumo de recursos que esto supone. A pesar de ello, no resulta de importancia puesto que son paquetes relativamente infrecuentes —o deberían serlo— y, como se ha dicho, de control, sin datos.

6.2. Hacia una metodología de detección on-line

Este trabajo propone una metodología *off-line* que identifica y marca duplicados con el objetivo de evitarlos selectivamente durante los procesos de análisis. Sin embargo, existen muchas herramientas que realizan sus análisis *on-line*, sin la necesidad de guardar capturas en disco. Además, uno de los efectos más perniciosos de los duplicados es que consumen gran parte del ancho de banda de la sonda (hasta el 50 % del *throughput*, en el peor de los escenarios, puede deberse a duplicados). Por consiguiente, queda patente

que una metodología que trabaje *on-line* en el propio dispositivo de captura puede ser muy beneficiosa. Desafortunadamente, las búsquedas sobre ventanas deslizantes suponen un gran reto en cuanto a coste temporal. A pesar de todas las mejoras introducidas e incluso con el uso de paralelización, la metodología propuesta está lejos de poder implementarse para la detección *on-line*.

No obstante, como ya se ha comentado previamente en el Capítulo 3, los duplicados en la monitorización emergen de una configuración fija y, por tanto, se producen de forma determinística: los paquetes de un cierto conjunto de flujos aparecerán siempre duplicados, mientras que los del resto nunca lo harán. Esto abre la posibilidad de encontrar un algoritmo capaz de aprender a discernir entre ambos conjuntos de flujos. Después de la fase de aprendizaje, este hipotético algoritmo sería capaz de eliminar automáticamente los duplicados únicamente en función de la procedencia y el destino de los paquetes. En este sentido, una futura metodología de detección *on-line* resultaría viable y requeriría una investigación más profunda.

Capítulo 7

Conclusiones

7.1. Resultados

- Se ha realizado una revisión del estado del arte en métodos de desduplicación de datos genéricos, un campo ampliamente investigado de la ingeniería de datos, con el objetivo de contextualizar la problemática particular que atañe a este trabajo.

- Se aporta un completo análisis teórico y descriptivo del problema de los duplicados en el tráfico de red, del que se pueden destacar los siguientes aspectos:

  - Las técnicas de desduplicación genéricas no son aplicables directamente en general, ni en este caso en particular, sin una severa adaptación a las especificidades de cada campo.
  - En redes de datos, se encuentran duplicados de muy diversos tipos que, en general, no son idénticos debido a las cabeceras.
  - En las propias redes en producción se generan duplicados, por diversas razones, que no interesa eliminar.
  - Los esquemas de monitorización introducen duplicados artificiales, no derivados del funcionamiento de la red, que sí interesa evitar para no falsear los análisis.

- Se aporta una metodología fiable para la detección y marcado del tráfico duplicado que ha sido implementada y utilizada con éxito por una empresa dedicada al análisis de tráfico.

- Se aporta un estudio analítico con su correspondiente validación experimental para el análisis del proceso de generación de duplicados en esquemas de *port mirroring*. Asimismo, se deriva una regla de dimensionamiento para maximizar la eficiencia de la metodología sin perder capacidad de detección.

7.2. Líneas futuras

- La inminente implantación de IPv6 requerirá una revisión de la implementación, basada hasta ahora en IPv4.
- Queda todavía margen para la mejora de la eficiencia mediante esquemas de ventana deslizante adaptativa y paralelización del proceso de búsqueda y comparación que deberán ser analizados.
- Dado el carácter determinístico de la generación de los duplicados por efecto de la monitorización, puede ser viable la búsqueda de un algoritmo que aprenda a desduplicar en base a una clasificación de los flujos. Esta posibilidad requiere mayor investigación con el objetivo de lograr una futura metodología de detección *on-line*.

Referencias

[1] **Are duplicate packets interfering with network monitoring?**, Abril 2013. Disponible en: `http://www.ixiacom.com/pdfs/network_visibility/library/white_papers/Anue_Duplicate_Packets_wp.pdf`. 2

[2] **Editcap man page**, Abril 2013. Disponible en: `http://www.wireshark.org/docs/man-pages/editcap.html`. 2, 14

[3] A.K. Elmagarmid, P.G. Ipeirotis, and V.S. Verykios. **Duplicate Record Detection: A Survey**. *Knowledge and Data Engineering, IEEE Transactions on*, **19**(1):1–16, 2007. 5, 6

[4] Gianni Costa, Alfredo Cuzzocrea, Giuseppe Manco, and Riccardo Ortale. **Data De-duplication: A Review**. In Marenglen Biba and Fatos Xhafa, editors, *Learning Structure and Schemas from Documents*, **375 of *Studies in Computational Intelligence*, pages 385–412. Springer Berlin Heidelberg, 2011. 5**

[5] Peter Christen. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection (Data-Centric Systems and Applications).* Springer, 2012. Disponible en: `http://www.amazon.com/Data-Matching-Techniques-Data-Centric-Applications/dp/3642311636`**. 9**

[6] Richard Bejtlich. *The Tao of Network Security Monitoring: Beyond Intrusion Detection.* Addison-Wesley Professional, 2004. Disponible en: `http://www.amazon.com/The-Tao-Network-Security-Monitoring/dp/0321246772`**. 9**

[7] IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663, Enero 2001. Disponible en: `http://www.ietf.org/rfc/rfc2663.txt`**. 17**

[8] **Traditional IP Network Address Translator (Traditional NAT). RFC 3022, Agosto 1999. Disponible en:** http://www.ietf.org/rfc/rfc3022.txt. **18**

[9] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. **Duplicate detection in click streams. In *Proceedings of the 14th international conference on World Wide Web - WWW '05*, page 12, New York, New York, USA, Mayo 2005. ACM Press. Disponible en:** http://dl.acm.org/citation.cfm?id=1060745.1060753. **21**

[10] Fan Deng and Davood Rafiei. **Approximately detecting duplicates for streaming data using stable bloom filters. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data - SIGMOD '06*, page 25, New York, New York, USA, Junio 2006. ACM Press. Disponible en:** http://dl.acm.org/citation.cfm?id=1142473.1142477. **21**

[11] Hong Shen and Yu Zhang. **Improved Approximate Detection of Duplicates for Data Streams Over Sliding Windows. *Journal of Computer Science and Technology*, 23(6):973–987, 2008. Disponible en:** http://dx.doi.org/10.1007/s11390-008-9192-1. **21**

[12] Parikshit Gopalan and Jaikumar Radhakrishnan. **Finding duplicates in a data stream. pages 402–411, Enero 2009. Disponible en:** http://dl.acm.org/citation.cfm?id=1496770.1496815. **21**

[13] Xiujun Wang and Hong Shen. **Approximately Detecting Duplicates for Probabilistic Data Streams over Sliding Windows. In *2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming*, pages 263–268. IEEE, Diciembre 2010. Disponible en:** http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5715092&contentType=Conference+Publications&searchField=Search_All&queryText=duplicate+sliding+window. **21**

[14] Jiansheng Wei, Hong Jiang, Ke Zhou, Dan Feng, and Hua Wang. **Detecting Duplicates over Sliding Windows with RAM-Efficient Detached Counting Bloom Filter Arrays. In *2011 IEEE Sixth International Conference on Networking, Architecture, and Storage*, pages 382–391. IEEE, Julio 2011. Disponible en:** http://

ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6005483&contentType=Conference+Publications&searchField=Search_All&queryText=duplicate+sliding+window. **21**

[15] Robert Olsson. **pktgen: the linux packet generator. In *Proceedings of the Linux Symposium*, Julio 2005. Disponible en:** https://www.kernel.org/doc/ols/2005/ols2005v2-pages-19-32.pdf. **32**

[16] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. **A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531–3547, Octubre 2012. Disponible en:** http://dx.doi.org/10.1016/j.comnet.2012.02.019. **38**

[17] Leonard Kleinrock. ***Queueing Systems: Theory, Volume 1.*** **Wiley, 1976. 38**

[18] **Naudit - High Performance Computing and Networking. Disponible en:** http://www.naudit.es/. **45**