Monitorización con FPGAs de flujos y sesiones TCP en enlaces de 40 Gbit/s

Tobías Alonso[1,2], Mario Ruiz[1], Gustavo Sutter[1], Cristian Sisterna[2], Sergio López-Buedo[1,3], Jorge E. López de Vergara[1,3]

Resumen— **En este trabajo se presenta una arquitectura basada en FPGA, diseñada para la agregación y posterior exportación de registros de sesiones TCP en enlaces de hasta 40 Gbit/s sin realizar muestreo de paquetes, incluso a la máxima tasa de paquetes. De esta manera, se descarga a exportadores de flujos basados en hardware de propósito específico de tareas para las cuales las FPGA ofrecen una flexibilidad y un desempeño adecuados, reduciendo los requerimientos del sistema completo. Un prototipo funcional del sistema ha sido implementado en la plataforma NetFPGA-SUME, donde fue sometido a tráfico real. En el mismo, se incorporó una estimación de las retransmisiones por flujo, además de otras estadísticas estándar, tales como número de bytes y paquetes de cada conexión de red.**

Palabras clave— **NetFPGA, exportador de flujos, tasa de línea, retransmisiones TCP.**

I. Introducción

El análisis de tráfico de red permite obtener información útil para encaminar dicho tráfico en *routers* y *switches*, así como para la administración de alto nivel en antivirus y cortafuegos o *firewalls*. Dentro de las herramientas de análisis de red, existen sistemas que agregan el tráfico en flujos, a partir de los cuales se puede conocer el estado de conexiones, utilización del ancho de banda, así como analizar ataques a un servidor. Esta técnica, a costa de perder información por paquete, tiene la ventaja de requerir considerablemente menos recursos de memoria.

La agregación de tráfico se suele llevar a cabo en *routers* o *switches*, aprovechando los recursos que estos disponen, como es el caso de NetFlow de Cisco. Sin embargo, estos suelen utilizar técnicas de muestreo de paquetes, las cuales son función del nivel de congestión al que son sometidos. A esto se le suma que las redes de alta velocidad (10 Gbit/s y superiores) comienzan a ser más frecuentes, dificultando aún más la tarea, por lo que, si se desea una mayor precisión en los análisis, se requiere un dispositivo dedicado a esta tarea de monitorización.

Consideramos que tanto una solución 100 % hardware como 100 % software no podrán tener simultáneamente suficiente capacidad de procesamiento y memoria con la latencia y capacidad necesaria. Del mismo modo, deberán ser suficientemente flexibles, lo cual es un factor importante en estos días, debido a la velocidad a la que se desarrollan nuevos sistemas y protocolos. Por otro lado, las FPGAs nos permiten resolver, con hardware diseñado a medida, problemas que serían inviables con tecnologías de circuitos de aplicaciones específicas (ASICs) debido a sus elevados costes, otorgando adicionalmente una mayor flexibilidad dada la posibilidad de actualización del hardware implementado.

Teniendo esto en cuenta, proponemos un sistema híbrido FPGA-CPU, en el que la FPGA no solo disecciona los paquetes y distribuye la carga de los procesadores, sino que también realiza un preprocesado de los paquetes, agregándolos en flujos. Esto es posible gracias a la localidad temporal entre paquetes de un mismo flujo, la cual es esperada debido a los protocolos utilizados en las redes de computadoras, que típicamente envían el tráfico a ráfagas. El proceso de agregación en flujos deberá ser completado por posteriores etapas de procesamiento.

El resto de este artículo se estructura de la siguiente manera: En la sección II, se comenta el estado del arte de sistemas para el análisis de flujos. Luego, en la sección III se presenta la monitorización de flujos y el algoritmo utilizado para detectar retransmisiones, mientras que en la sección IV, se desarrolla la metodología y el análisis previo al diseño del sistema. La arquitectura se detalla en la sección V, exponiendo los procedimientos seguidos para la verificación del sistema y los resultados de los mismos en la sección VI. Por último, en la sección VII, se resumen las contribuciones realizadas y se plantean trabajos futuros.

II. Estado del Arte

En el estado del arte se observan diversas implementaciones de exportadores de flujos utilizando CPUs, GPUs, FPGAs y versiones híbridas que reparten la carga del sistema entre una combinación de estos dispositivos.

Marco Forconesi *et al.* en [1] propusieron una arquitectura para exportar flujos en redes de 10 Gbit/s implementada en la plataforma NetFPGA-10G, la cual permite el procesamiento de la máxima cantidad de paquetes por segundos sin muestreo, soportando el manejo de hasta 786.432 flujos concurrentes. El problema de este sistema es su falta de escalabilidad, dado el uso de memoria SRAM y el *throughput* del procesador de flujos.

Paula Roquero *et al.* en [2] propusieron un sistema CPU-GPU generador de flujos capaz de obtener información compleja de computar sobre el protoco-

[1]Grupo HPCN, Escuela Politécnica Superior, Universidad Autónoma de Madrid, España. {tobias.alonso, mario.ruiz, gustavo.sutter, sergio.lopez-buedo, jorge.lopez_vergara}@uam.es

[2]Facultad de Ingeniería, Universidad Nacional San Juan, Argentina, cristian@unsj.edu.ar

[3]Naudit HPCN, S.L., España, {sergio,jorge}@naudit.es

lo TCP, como es la detección de retransmisiones con memoria a nivel de paquete. Sin embargo, el enlace soportado se encuentra por debajo de 10 Gbps dado que la tasa de paquetes capaz de ser procesada es de 4,4 Mpps, siendo 14,88 Mpps la máxima tasa.

Otras soluciones, basadas únicamente en software, hacen uso de *drivers* de captura de alto rendimiento [3], alcanzando los 10 Gbit/s. Así, es posible monitorizar flujos con métricas complejas, como estimaciones del número de retransmisiones [4]. Sin embargo, para soportar enlaces de 40 Gbit/s se requerían procesadores de muy alto rendimiento, lo cual aumentaría considerablemente el coste del sistema.

Viktor Puš *et al.* [5] implementaron un sistema FPGA-CPU capaz de soportar exportación de flujos en enlaces de 100 Gbit/s. Este resultado se logra utilizando una FPGA como NIC inyectando los flujos en 16 colas implementadas en la memoria principal (64 GB DDR4) de la CPU de 20 núcleos, según el resultado de una función *hash* aplicada a campos de la cabecera de los paquetes. En este caso, se observa que, delegando a una FPGA, el servidor utilizado se vuelve asequible, pero consideramos que la CPU sigue realizando tareas para las cuales la FPGA es más adecuada, con lo que se podría reducir en mayor medida el coste del sistema.

III. Monitorización de flujos y detección de retransmisiones

A. Monitorización de flujos

De acuerdo a la RFC 7011, un flujo se define como una secuencia unidireccional de paquetes que comparten una serie de parámetros. En el presente trabajo, estos son las direcciones IP de origen y destino, y los puertos de origen y destino, quedando definida de esta manera la 4-tupla que identifica a cada flujo. Una sesión (o flujo bidireccional) está conformada por un par de flujos recíprocos, es decir, aquellos que conectan los mismos puntos, pero tienen sentido contrario.

Como ejemplo de un sistema para el tratamiento de flujos podemos observar la figura 1, donde se ilustra un esquema de ejemplo. El mismo está compuesto por tres componentes. El exportador de flujos obtiene estadísticas de los flujos que atraviesan el *router* y luego exporta estos registros a uno o más colectores de flujos, donde se almacenan y procesan. Por último, una aplicación de análisis realiza peticiones de información a los colectores y la analiza.

La tabla de sesiones es aquella memoria que contiene los registros temporales de las sesiones no expiradas. Cada vez que se recibe un paquete de una nueva sesión se crea una entrada en la tabla, la cual será liberada al transmitir los registros de la misma para su posterior análisis. Esta transmisión se denomina exportación de la sesión.

Las sesiones pueden ser exportadas por cuatro razones:

- *Por tiempo de inactividad*: Una sesión es exportada si en los últimos T_i segundos no se recibe al menos un paquete perteneciente a la misma.
- *Por tiempo de actividad*: Si una sesión se mantiene activa por más de T_a segundos, es exportada.
- *Por banderas TCP*: Si se recibe una bandera de FIN de cada flujo o una bandera de RST de cualquiera de ellos, la sesión es exportada.
- *Por colisión*: Si llega un paquete de una nueva sesión al sistema y no hay lugar en la memoria para almacenarla, se exporta una de las sesiones que generan el conflicto.

En el prototipo implementado T_i se fijó a 15 segundos, y T_a a 15 minutos, dado que son valores frecuentemente usados en despliegues de red.

Para profundizar en el tema de la monitorización de flujos, en [6] se dispone de un tutorial que describe todas las partes que componen estos sistemas.

B. Detección de retransmisiones

Un indicador de que existe un problema en la red es que haya excesivas retransmisiones en una conexión TCP, por lo que se propuso la implementación de un algoritmo que las detecte. Para esto se empleó un método sencillo, utilizado en [4], el cual se basa en la hipótesis de que los segmentos TCP llegan de forma ordenada (según orden de envío). Esta hipótesis no es necesariamente cierta, dado que no todos los paquetes transitan el mismo camino. Sin embargo, según [7], la cantidad de paquetes que llegan desordenados habitualmente es muy baja (inferior al 0,2 %).

Para detectar una retransmisión, se compara el número de secuencia del último byte de datos del flujo TCP recibido con el primer byte del nuevo segmento, como se observa en la figura 2.

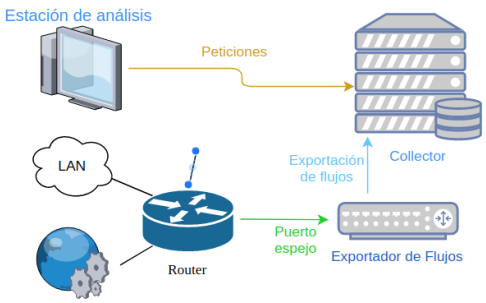

Fig. 1: sistema de exportación de flujos

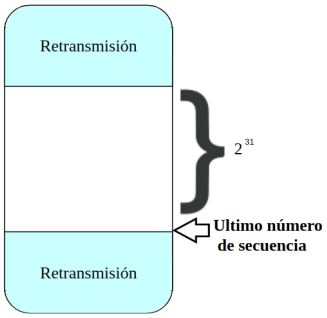

Fig. 2: Método de determinación de retransmisiones

IV. Diseño del sistema

A. Metodología

Para el diseño FPGA, se usó una metodología basada en IP Cores usando la herramienta Vivado de Xilinx, donde se desarrollaron módulos en VHDL, integrándolos con diferentes bloques predefinidos. Para intercomunicar los módulos se utilizó un protocolo estándar, AXI4-Stream, lo cual aumenta las posibilidades de reutilización de los mismos.

En busca de una mayor flexibilidad, se buscó un gran nivel de desacople entre el control y el camino de los datos, reduciendo los módulos a ser modificados para cambiar información almacenada, allanando el camino para futuros proyectos donde se buscará definir esto a través de lenguajes de alto nivel. A modo de prueba, se implementaron estadísticas típicas en sistemas basados en flujos, añadiendo estimaciones de las retransmisiones de paquetes TCP.

B. Dimensionamiento del sistema

Para tomar decisiones de diseño, primero fue necesario realizar una serie de análisis y estudios respecto de las restricciones de tiempos y especificaciones de memoria requerida.

B.1 Restricciones de tiempos

Por razones de compatibilidad con la plataforma utilizada (NetFPGA-SUME [8]), en la implementación se utilizó una interfaz de entrada de 32 bytes de ancho. Teniendo en cuenta esto y las especificaciones del protocolo Ethernet, llegamos a la siguiente formula, que define la máxima velocidad de enlace soportado en la interfaz interna de la FPGA, considerando el peor de los casos:

$$\textit{Enlace soportado } [bits/s] = CF * 712/3 \quad (1)$$

donde CF es la frecuencia de reloj.

Si se desea soportar un enlace de 40 Gbit/s, la fórmula arroja que debemos temporizar el bus interno a 168,54 MHz o más.

Por otro lado, para que en el sistema de procesamiento no se forme un cuello de botella, se debe poder procesar los paquetes a la máxima tasa a la cual estos pueden aparecer. Si el ancho del bus interno y la frecuencia del reloj (CF) permiten soportar una determinada velocidad de enlace, entonces:

$$\frac{\textit{Mín ciclos}}{\textit{paquete}} = \left\lfloor CF * \frac{672\ [bits/paquete]}{Velocidad\ Enlace} \right\rfloor \quad (2)$$

donde $\lfloor x \rfloor$ es la función suelo.

Reemplazando CF por 168,54 MHz y una velocidad de enlace de 40 Gbit/s, obtenemos que la mínima cantidad de ciclos entre dos paquetes es 2.

B.2 Memoria para la tabla de sesiones

El tamaño de la tabla de sesiones dependerá de la velocidad y tipo de enlace, el tiempo de expiración, el porcentaje de paquetes filtrados, así como su tamaño medio y la localidad temporal existente entre ellos (esta última fue observada en análisis realizados, pero se requirieren de estudios más profundos). Teniendo esto en cuenta y basándonos en [9–11], obtenemos una estimación gruesa de 6 GB para un enlace de 40 Gbit/s filtrando TCP y usando entradas de 64 bytes. Esto es considerando una memoria totalmente asociativa, por lo que esto establecería un límite inferior para memorias de menor grado de asociatividad.

En cuanto al requisito de velocidad de la memoria, se observa que esta es función de la velocidad del enlace, el tipo de paquetes y el tamaño de los mismos. Basándonos en los citados artículos llegamos a un límite inferior de 160 $ns/(n^\circ\ de\ accesos\ por\ paquete)$.

Analizando las especificaciones de memorias disponibles en la actualidad, las que más se aproximan a las especificaciones son las DDR SDRAM, considerando que se tienen 4 accesos a memoria por paquete (a partir de un primer diseño del sistema) y un enlace 40 Gbit/s típico. Sin embargo, al introducir las latencias del controlador de memoria disponible, los requisitos de tiempos se dejan de cumplir.

Es por esto que, bajo la hipótesis de localidad temporal, se propuso una arquitectura con una pequeña tabla de sesiones en la FPGA que agregue fracciones de sesiones (fraccionadas debido a las colisiones, dado el tamaño reducido de la tabla), proporcionando la información necesaria para reconstruir los flujos completos a posteriori por etapas de procesamiento posteriores.

V. Arquitectura propuesta

A. Interfaces

La interfaz de entrada del sistema es un bus de 32 bytes que acepta tramas Ethernet sin el preámbulo, el delimitador de trama y el CRC, por razones de compatibilidad con las interfaces en los diseños de referencia para la NetFPGA-SUME. En esta primera versión, solo se aceptan paquetes IPv4 con protocolo de transporte TCP.

Como resultado del procesamiento, se obtiene una palabra de 536 bits para cada sesión. Esta contiene una serie de registros elegidos para poder identificar las sesiones, conocer el número de retransmisiones de los paquetes de cada flujo en relación a la cantidad de paquetes enviados en ese sentido, así como información del tamaño promedio de los paquetes y del control de flujo de los mismos. También permite obtener información del tiempo de inicio y duración de las sesiones. Además, dado que existe la posibilidad de que existan colisiones, se introdujo la información necesaria para reconstruir los registros en estos casos.

El sistema posee una interfaz para obtener una referencia de tiempo externa. Las marcas de tiempo utilizadas internamente por el sistema son de 32 bits: 20 bits para los μs y 12 bits para los s. Aunque sería interesante alcanzar una precisión mayor a los μs, se ha optado por dicha magnitud dadas las limitaciones de bits disponibles, y que el retardo de red está habitualmente en el orden de los ms [7].

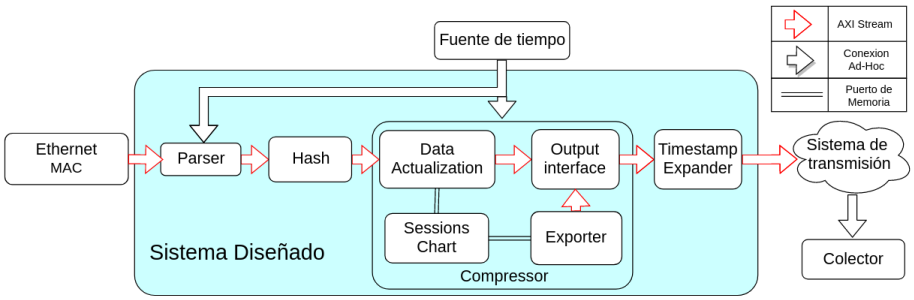

Fig. 3: Diagrama de alto nivel del sistema

B. Descripción de alto nivel

El sistema tiene una arquitectura en *pipeline* para maximizar la tasa de procesamiento y aprovechar las capacidades de las FPGA. Adicionalmente, nos permite lograr un desacople entre los diferentes módulos del sistema que facilita el desarrollo independiente de los mismos, siempre que se respeten los protocolos e interfaces. En la figura 3 se observan los módulos que componen el sistema.

El flujo comienza en el *Parser*, que filtra paquetes y extrae información de ellos. Luego, la función *hash* calcula la dirección de la tabla de sesiones (*Sessions Chart*) donde se almacenará la sesión de este paquete. El módulo *Data Actualization* es el encargado de agregar toda la información de una sesión en los registros. El *Exporter* examina continuamente la tabla en busca de flujos expirados, los cuales se pasarán a la interfaz de salida y luego, al *Timestamp Expander*, donde se amplía la base de las marcas de tiempo para, finalmente, transferir los registros de la sesión. A continuación, se procede a detallar cada uno de estos módulos.

C. Disector (Parser)

Este módulo analiza la trama entrante y decide si la acepta en función de los protocolos que utiliza. Si es aceptada, se extrae la 4-tupla, el número de secuencia y las banderas del encabezado TCP. Adicionalmente, realiza el marcado de tiempo, cuenta la cantidad de bytes de datos y se calcula el número de secuencia del último byte. Esta información es luego transferida al siguiente módulo.

Respecto al desempeño de este módulo, el mismo es capaz de recibir un nuevo paquete cada dos ciclos de reloj y soporta cualquier cantidad de etiquetas VLAN. En el presente, los campos extraídos y calculados se definieron en código HDL. Sin embargo, en futuros trabajos se buscará que los mismos puedan ser definidos con un lenguaje de más alto nivel, dando más flexibilidad al sistema, como Zazo *et al* propusieron en [12].

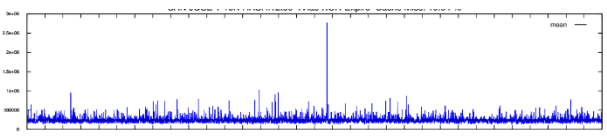

Fig. 4: Distribución de flujos por dirección de la tabla

D. Hash

El módulo *Hash* es el encargado de definir la correspondencia entre las sesiones y una dirección de la tabla de sesiones. La elección de la función a implementar persigue principalmente dos objetivos:

- Obtener la misma dirección para los dos flujos de la sesión.
- Obtener una distribución aproximadamente uniforme de direcciones, reduciendo las colisiones.

Adicionalmente, si se posee un sistema de memoria con memoria cache, es deseable que sea una función rápida de calcular, ya que evitaría la necesidad de almacenar el resultado de la misma.

Atendiendo a lo anterior, se decidió realizar una XOR de palabras formadas con la 4-tupla, con una simetría tal que el resultado es independiente de la dirección del flujo, obteniendo una distribución como la observada en la figura 4. Calculada la dirección, la misma es adjuntada a los datos entregados por el *Parser*, como se observa a la izquierda en la figura 5.

E. Tabla de sesiones (Sessions Chart)

El sistema emplea memorias *true dual port* para implementar la tabla de sesiones, utilizando un puerto para el módulo *Data Actualization* y otro para el *Exporter*. En cada dirección de memoria se podrá almacenar más de una sesión, distinguiendo cada entrada por vías, al estilo de una memoria caché, lo cual disminuye la probabilidad de colisiones.

F. Actualizador de sesiones (Data Actualization)

Este módulo condensa en un conjunto de registros (entrada de la tabla) toda la información proveniente de paquetes de una sesión logrando reducir de esta manera la tasa de información a la salida del mismo. El mismo es capaz de procesar un nuevo paquete cada dos ciclos de reloj siempre que la tabla de sesiones posea un tiempo de acceso de un ciclo de reloj o menos. Para obtener el desempeño adecuado se dispuso de dos ciclos de reloj para las operaciones sobre los datos (*multicycle path*).

En la figura 5 se observa la relación del *Exporter* y la Interfaz de Salida (*Output Interface*) con este módulo, así como también se aprecian las dos etapas del mismo: Pre-búsqueda (*Pre-fetch*) y Actualización (*Actualization*). A pesar de ser un *pipeline* no lineal, estas poseen un gran grado de desacople gracias a los protocolos de comunicación definidos.

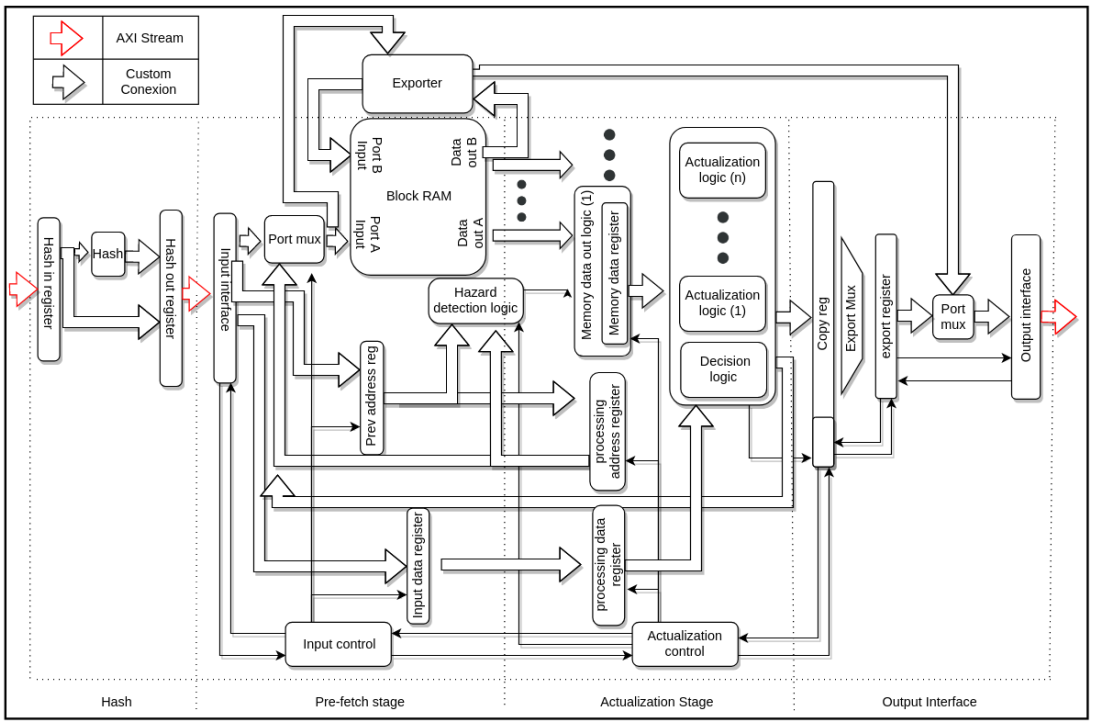

Fig. 5: Arquitectura del *Compressor*

Otro punto notable de este sistema es el desacople existente entre el camino de control y el de datos, permitiéndole a este módulo manejar cualquier base de datos cambiando únicamente dos módulos que son los que analizan y operan sobre los datos. Por esta razón, se buscará en futuros proyectos definir estos módulos con lenguajes de alto nivel para permitir mayor flexibilidad en los campos almacenados y operaciones realizadas, así como aumentar la posible reutilización de este diseño.

G. *Exportador (Exporter)*

Este módulo es el encargado de transmitir y eliminar las sesiones en la memoria que han expirado. El mismo tendrá control del segundo puerto de la memoria, debiendo monitorizar las operaciones que se realizan en el puerto usado por el sistema de actualización de datos de manera que no se corrompa la información.

Este recorre la memoria comparando la marca de tiempo del primer y el último paquete de la sesión con el tiempo actual. Si el tiempo transcurrido desde el primer paquete es mayor al tiempo de actividad (T_a) y/o el tiempo transcurrido desde último paquete es mayor al tiempo de inactividad (T_i), la sesión debe ser exportada.

H. *Expansor de marca de tiempo*

El *Timestamp Expander* extiende la base de tiempo de las marcas de tiempo, obteniendo 32 bits para los segundos, de manera que posteriores análisis de las exportaciones puedan determinar unívocamente la ocurrencia de los diferentes eventos.

VI. Verificación y Resultados de Implementación

A. *Depuración y verificación del sistema*

Para verificar el correcto funcionamiento del sistema se realizaron tanto simulaciones como pruebas de campo. Debido a la dificultad para obtener patrones contra los que comparar los resultados, en un principio se realizaron comprobaciones con tráfico generado de forma sintética[1], cuyos paquetes fueron creados con el fin de probar partes específicas del sistema, y luego se sometió a un régimen de trabajo más realista utilizando trazas capturadas y comparando los resultados con los provistos por una conocida herramienta para el análisis de tráfico, `Wireshark`[2]. El prototipo implementado en la fase de pruebas fue configurado para almacenar hasta 16384 sesiones de forma concurrente, distribuidas en cuatro vías.

B. *Pruebas de campo*

Se implementó un sistema de prueba en la plataforma NetFPGA-SUME (FPGA: Xilinx Virtex-7 690T). Dado que no se disponía de una interfaz de 40 Gbit/s en el momento de realizar las pruebas, el sistema se verificó a 10 Gbit/s, quedando este trabajo para cuando se disponga del equipamiento adecuado. Se utilizó un PC con dos interfaces de 10 Gbit/s. Una de ellas fue usada como generador de tráfico, corriendo `tcpreplay` y la segunda capturando la salida con `tcpdump`.

Para las pruebas se utilizaron trazas de distintas características, primero simples, para verificar comportamientos específicos y luego, trazas de mayor volumen, como se observa en la tabla I.

[1]Por ejemplo, con `http://packeth.sourceforge.net/`
[2]`https://www.wireshark.org/`

TABLA I: Trazas utilizadas

Traza	Verificación
Un paquete con RST	Determinar si responde y si exporta ante un RST
Sesión completa	Almacenamiento y actualización correcta de datos, detección de retransmisiones, marcado de tiempos y exportación por banderas FIN
Flujos de más de 15 min	Exportación por tiempo de actividad(T_a)
Tráfico TCP de una PC	Tráfico real a tasa de envío original y a máxima tasa
Tráfico TCP sin datos	Comportamiento ante gran tasa de paquetes
Tráfico de una PC	Comportamiento ante tráfico real simple a máxima tasa
Tráfico de una institución	Comportamiento ante tráfico real de gran volumen a tasa de envío original y a máxima tasa

Analizando el comportamiento del sistema en simulación y en las pruebas de campo podemos decir que el sistema funciona de acuerdo a las especificaciones. Se observaron pequeños errores en el número de retransmisiones, debido a cierto desorden de los paquetes, y algunas diferencias en las exportaciones, debido a colisiones (subsanables agregando las exportaciones parciales). No obstante, este comportamiento era el esperado. Sin embargo, se considera que se deben realizar pruebas más rigurosas respecto al marcado de tiempos, dado que el patrón de que se disponía no es suficientemente preciso.

C. Especificaciones del hardware implementado

El sistema diseñado fue implementado con un reloj de 180 MHz, el cual (en base a las ecuaciones 1 y 2) permite sostener la velocidad de enlaces de 40 Gbps, dada su capacidad de procesar un nuevo paquete cada dos ciclos de reloj.

Respecto de los recursos utilizados en la implementación del sistema diseñado y del sistema de prueba completo (incluyendo interfaces, un soft-μP, ...), los mismos se exponen en la tabla II. Se observa que se disponen de recursos suficientes para aumentar el tamaño de la tabla de sesiones si así se requiriese.

TABLA II: Utilización de recursos

Módulo		LUT	FF	BRAM
Parser		782	459	0
Hash		156	222	0
Compressor		8443	6745	228
	Sessions Chart	16	0	228
	Exporter	683	1168	0
Timestamp Expander		30	84	0
Total Diseño		9454	8769	228
Total Diseño(%)		2.2	1	15.5
Sistema de prueba (%)		5.8	3.1	18

VII. Conclusiones y trabajos futuros

Este trabajo aporta una arquitectura eficiente en términos de desempeño para la obtención de una serie de estadísticas sobre las sesiones TCP establecidas a través de un enlace de Ethernet a 40 Gbit/s, descargando a sistemas basados en hardware de propósito general de las tareas de disección y agregación fina de paquetes en flujos. Gracias al desacople logrado entre el camino de datos y la lógica de control, esta arquitectura puede ser reutilizada para el cálculo de diferentes estadísticas a las implementadas y bajo un flujo de información distinto sin realizar cambios estructurales.

Se considera que todavía hay un largo camino por recorrer, quedando para futuros proyectos la verificación en campo a 40 Gbit/s, el estudio de funciones *hash* óptimas y el análisis riguroso de la existencia de localidad temporal en enlaces de alta velocidad, la cual puede verse debilitada debido a la agregación de usuarios en una sola línea. Así mismo, dado que se comienzan a observar enlaces a 100 Gbit/s, se buscará adaptar este sistema a esta tasa de enlace.

Agradecimientos

Este trabajo ha sido parcialmente financiado por una beca de movilidad de la Universidad de San Juan, por el Ministerio de Economía y Competitividad y el Fondo Europeo de Desarrollo Regional con el proyecto TRÁFICA (MINECO/FEDER TEC2015-69417-C2-1-R), y por la Comisión Europea con el proyecto H2020 METRO-HAUL (id. 761727).

Referencias

[1] M. Forconesi *et al.*, "Accurate and flexible flow-based monitoring for high-speed networks," in *Proc. 23rd Int. Conf. Field Programmable Logic and Applications*, 2013.

[2] P. Roquero *et al.*, "High-speed TCP flow record extraction using GPUs," *The Journal of Supercomputing*, vol. 71, no. 10, pp. 3851–3876, Oct 2015.

[3] V. Moreno *et al.*, "Commodity packet capture engines: Tutorial, cookbook and applicability," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1364–1390, 2015.

[4] E. Miravalls-Sierra *et al.*, "Online detection of pathological TCP flows with retransmissions in high-speed networks," *Computer Communications*, vol. 127, pp. 95–104, Sept. 2018.

[5] V. Puš *et al.*, "Hardware accelerated flow measurement of 100 gb ethernet," in *Proc. 2015 IFIP/IEEE Int. Symp. Integrated Network Management*, May 2015.

[6] R. Hofstede *et al.*, "Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX," *IEEE Communications Surveys & Tutorials*, vol. 16, pp. 2037–2064, 2014.

[7] D. Murray and T. Koziniec, "The state of enterprise network traffic in 2012," in *Proc. 18th Asia-Pacific Conference on Communications (APCC)*, 2012.

[8] (2018, Mayo) NetFPGA SUME. [Online]. Available: https://netfpga.org/site/#/systems/1netfpga-sume/details/

[9] K. Pentikousis and H. Badr, "Quantifying the deployment of TCP options - a comparative study," *IEEE Communication Letters*, vol. 8, pp. 647–649, 2004.

[10] W. John and S. Tafvelin, "Analysis of internet backbone traffic and header anomalies observed," in *Proc. 7th ACM SIGCOMM Conf. Internet Measurement*, 2007.

[11] M. Zhang *et al.*, "Analysis of UDP traffic usage on internet backbone links," in *Proc. 9th Annual Int. Symposium on Applications and the Internet*, 2009.

[12] J. F. Zazo *et al.*, "Automated synthesis of FPGA-based packet filters for 100 Gbps network monitoring applications," in *2016 Int. Conf. ReConFigurable Computing and FPGAs (ReConFig)*, Nov. 2016.