REVISTA COLOMBIANA DE COMPUTACIÓN
Volumen 14, número 1
Págs. 78 - 95

Esquema de Timestamp en la Recepción de Paquetes de Prueba a Través de la Netfpga para la Estimación de Ancho de Banda Disponible en Traceband

Nydia S. Sandoval[1], César D. Guerrero[1]

Fecha de recibido:30/01/2013 Fecha de aprobación: 03/03/2013

Resumen

Los servicios y aplicaciones que se ofrecen actualmente sobre internet como video bajo demanda, requieren estimación de parámetros que determinen la calidad de una conexión como el ancho de banda disponible. Para la estimación del ancho de banda disponible (ABW) se han desarrollado herramientas de software como Pathload, Spruce y Traceband. Estas herramientas presentan limitaciones en la precisión de la estimación debido a los retardos que sufren los paquetes en los procesos inherentes del sistema operativo. La herramienta requiere precisión en la captura del tiempo de llegada de los paquetes y esto se logra realizando el timestamp a nivel físico. Para ello se utiliza la NetFPGA dado que permite modificar el comportamiento como tarjeta de red adicionando módulos de acuerdo a las necesidades del proyecto. En el diseño del esquema de marcación de los tiempos de llegada de los paquetes de prueba se utiliza la herramienta Traceband, esta se encarga del envío y cálculo del ABW y la NetFPGA se encarga de realizar la marca de tiempo de llegada de los paquetes en la recepción en hardware y basados en este valor se realiza la estimación del ABW. Se diseñaron los módulos Timestamp e Identificación en la NetFPGA, y los cambios requeridos en Traceband para comunicarse con la NetFPGA.

Palabras Clave: *Ancho de banda disponible, NetFPGA, Redes, Traceband.*

Abstract

Applications and services currently being offered on the Internet as video on demand, require estimation parameters that determine the quality of a connection as the available bandwidth. To estimate the available bandwidth (ABW) have developed software tools to level as Pathload, Spruce and Traceband. These tools have limitations on the accuracy of the estimate due to delays suffered by packets in the inherent operating system processes. The tool

[1] Universidad Autónoma de Bucaramanga. E-mail: {nsandoval697, cguerrer} @unab.edu.co

requires precision in capturing the arrival time of packages and this is achieved by performing the timestamp on the physical level. To do the NetFPGA is used as it allows modifying the behavior as adding network card modules according to the needs of the project. In designing the scheme marking the arrival times of the test packets the Traceband tool is used, it is in charge of sending and calculation of ABW and NetFPGA is responsible for conducting the timestamp of packet arrival at the receiving hardware and based on this estimation value is performed ABW. The timestamp and Identification modules in NetFPGA, and the required changes in Traceband to communicate with NetFPGA were designed.

Keywords: *Bandwidth Estimate, NetFPGA, Network, Traceband.*

1. Introducción

Las aplicaciones y servicios utilizados hoy día sobre internet son dinámicos y complejos. Estimar la calidad de la conexión de extremo a extremo de la red es difícil de predecir por el comportamiento del tráfico variable. El ancho de banda es una medida de velocidad que se define como la cantidad de bits transmitidos en un canal de comunicaciones por unidad de tiempo. Hay dos métricas asociadas con el ancho de banda: capacidad y ancho de banda disponible. La capacidad de un enlace es el ancho de banda máximo o la máxima cantidad de bits que se pueden transmitir al enlace por unidad de tiempo. Por el contrario, el ancho de banda disponible (ABW) de extremo a extremo (end-to-end) es la mínima capacidad no utilizada de todos los enlaces en una ruta de comunicación entre dos nodos extremos, este concepto ha recibido gran atención en los últimos años debido a que puede ser usado para monitoreo de la red, verificación de la calidad del servicio, selección de la mejor ruta. Para estimar el ancho de banda disponible de extremo a extremo sobre una ruta de red, se han desarrollado diferentes aplicaciones de software, las cuales se realizan a través de mediciones activas y requieren operaciones estadísticas sobre los valores de dispersión de los paquetes. Las herramientas estudiadas basan sus estimaciones en los modelos Probe Rate Model (PRM) [1] y Probe Gap Model (PGM) [2]. El primero induce paquetes de prueba incrementando la velocidad de transferencia hasta encontrar la velocidad a la que el canal se congestiona, el segundo envían pares de paquetes de prueba de igual tamaño separados de acuerdo con el tiempo de transmisión de las pruebas sobre el cuello de botella del enlace.

El contenido de este artículo se encuentra estructurado de la siguiente manera: en la sección 2 se presentan las herramientas de estimación de ancho de banda disponible de extremo a extremo y los estudios comparativos entre estas herramientas, evaluando parámetros como

precisión en la estimación, *overhead* sobre el canal de comunicaciones y tiempo estimación. En la tercera sección se describen las aplicaciones del *timestamp* en la NetFPGA, las ventajas y desventajas que presentan la forma de realizar el *timestamp*. En la cuarta sección se presenta el esquema propuesto para la marcación de tiempo en los paquetes de llegada en la NetFPGA que permite la interacción con la herramienta *TRACEBAND* para la estimación ABW. En la sección 5 de artículo se presentan las conclusiones de la investigación.

2. Análisis de las Herramientas de Estimación de Ancho de Banda Disponible

Para estimar el ancho de banda disponible de extremo a extremo sobre una ruta de red, se han encontrado en la literatura 14 herramientas; estas aplicaciones basan sus cálculos en la dispersión de los paquetes de prueba al pasar por una ruta. Los tiempos de llegada de los paquetes son tomados de software.

Las herramientas encontradas en la literatura son de sondeo activo, es decir, envían paquetes al canal de comunicaciones, se pueden organizar de acuerdo al modelo aplicado la estimación de esta manera: las que desarrollan el modelo PRM son Pathload [1], PathChirp [3], YAZ [4], Nestet [5], TOPP [6], ASSOLO [7], y FEAT [8] , aquellas que aplican el modelo PGM son IGI/PTR [9], Spruce [10], ProbeGap [11], Delphi [12], IGMPS [13], Abing [14] y Traceband [15]. Observándose que el 50% de las herramientas utilizan el modelo PRM y el 50% restante el modelo PGM.

Se han encontrado además estudios comparativos de estas herramientas, como los realizados por Guerrero et il [15] [16], donde se comparan las herramientas Pathload, IGI, Spruce y Traceband. Los autores evalúan las aplicaciones bajo los parámetros de evaluación: Error de estimación, tiempo de estimación, sobrecarga y fiabilidad. Con la realización de 11760 experimentos, se concluye que Pathload presenta menor error de estimación, IGI tiene un tiempo de estimación más pequeño y Spruce presenta la menor sobrecarga en el cuello de botella del enlace de extremo a extremo. Sin embargo Traceband presenta mejor rendimiento en cuanto a la baja sobrecarga similar con el comportamiento de Spruce y presenta una precisión en la medición similar con la que presenta Pathload.

En cuanto a las comparaciones realizadas por Goldoni [17] y se comparan varias herramientas en un banco de pruebas real con enlaces a 100 Mbps, y tanto con la velocidad de bits constante (CBR) y un tráfico cruzado tipo Poisson. Los autores evalúan la exactitud, la intrusión, y el

tiempo de convergencia de las herramientas. Reportando en los resultados que la más alta precisión se proporciona por Pathload y YAZ, independientemente de la clase de tráfico cruzado. La intrusión y el tiempo de convergencia de Pathload y YAZ son más significativos, especialmente durante las pruebas en una red de área amplia emulada. Entre herramientas restantes, Pathchirp, IGI/PTR, y Diettopp logran un rendimiento promedio en todas las situaciones.

Una de las limitaciones de las herramientas de estimación de ancho de banda disponible tiene relación con los procesos como la generación y recepción de paquetes de prueba, estos son complejos de controlar y en algunos casos están fuera del alcance de cualquier herramienta de estimación [18] [19] [20] debido a que son procesos inherentes al sistema operativo como se muestra en la Fig. 1. La generación de paquetes y el registro del tiempo de llegada (Timestamp) de los mismos, es determinada por la aplicación (Userspace), los paquetes pasan a la cola de procesos del sistema operativo (Kernelspace) donde se colocan los encabezados del paquete, los retardos que los paquetes sufren en esta conmutación dependen de la velocidad de procesamiento y las restricciones del mismo, finalmente el paquete pasa a la tarjeta de red, donde el paquete se acondiciona para la transmisión, en la recepción ocurre el proceso inverso y en el momento de realizar el marcado de tiempo en los paquetes existe un retardo adicional generados por los múltiples procesos antes de llegar a la aplicación donde se toma el retardo total con el cual se realiza la estimación. Dando lugar a registros incorrectos en los tiempos de salida o llegada de paquetes.

Además pueden presentarse errores como la recepción fuera de orden, replicación y corrupción de paquetes, la sobrecarga generada en la red por la transmisión de los paquetes de prueba, el comportamiento del tráfico cruzado y la especificación de un conjunto de parámetros antes de ejecutar la aplicación de estimación, afectan la precisión de las herramientas de estimación de ABW.

A pesar de los esfuerzos por mejorar el performance de las mediciones, aun las herramientas requieren a mayor precisión y fiabilidad en la estimación, utilizando un menor tiempo de estimación, y una baja sobrecarga en el canal, para poder ser utilizadas en aplicaciones como la selección de la ruta o medición de la calidad de servicio de un enlace de comunicaciones real.

Dado que la herramienta Traceband presenta un comportamiento similar a otras herramientas en relación a la precisión, baja intrusión o sobrecarga en la red y un tiempo de estimación moderado, será la herramienta utilizada para el desarrollo de este trabajo de investigación.

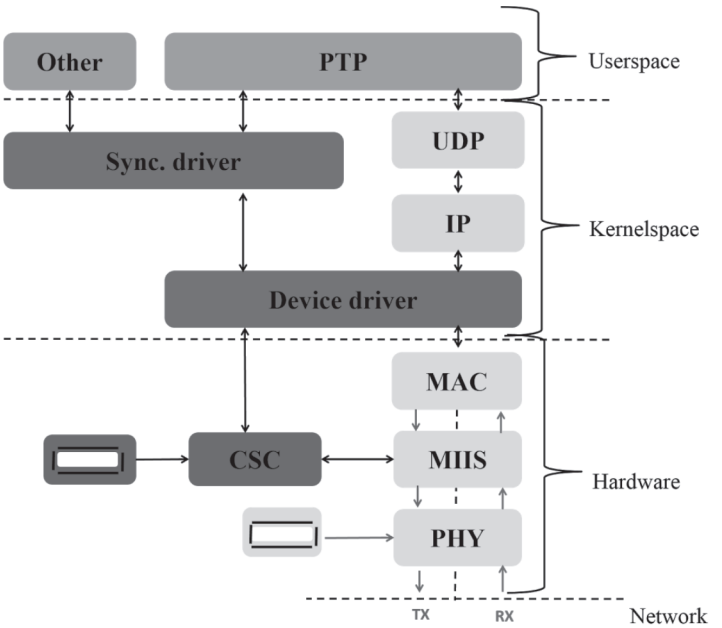

Fig. 1. Ruta de los paquetes de datos.
Fuente: [21].

2.1 Descripción del Algoritmo de *TRACEBAND*

TRACEBAND es una herramienta de estimación de ancho de banda disponible desarrollada por Guerrero y Labarador [15]. Es una herramienta cliente-servidor escrita en *ANSI C* que utiliza Modelos Ocultos de Markov (*HMM*) para proporcionar estimaciones de *ABW* de forma rápida, continua y precisa, bajo el modelo de estimación *PGM*.

El cliente en *TRACEBAND* ejecuta ciclos de diez estimaciones. En la primera estimación de la herramienta envía 50 pares de paquetes *UDP* de 1498 bytes de longitud por el puerto 3504.

Las nueve estimaciones restantes se llevan a cabo con 30 pares de paquetes de cada una. Esta reducción es posible gracias a *HMM*, el cual es capaz de aprender la dinámica del *ABW* con una muestra inicial y mantener el modelo actualizado con muestras de tamaño reducido. Se encontró por experimentación que diez estimaciones fueron suficientes para mantener una buena precisión con bajo costo operativo.

Al realizar un estudio del código de la aplicación Traceband_rcv la cual se encarga de recibir los paquetes de prueba y el procesamiento de estos paquetes para la estimación de ancho de banda disponible, se puede describir su funcionamiento de la siguiente manera:

Al llegar un paquete, se realiza el marcado de tiempo y se guarda la información del paquete, luego se registra en una tabla, los datos como número de paquete(pck_num), tamaño del paquete (pck_size), marca de tiempo de trasmisión en segundos y microsegundos (snd_time); y la marca de tiempo de recepción (rcv_time), se revisa si se finalizó la recepción de los paquetes, si es así, se procede a realizar la estimación de ancho de banda disponible con los datos guardados en la tabla y el resultado del Modelo Oculto de Markov, sino se procede a recibir el siguiente paquete hasta que llegue el último paquete de prueba cuyo tamaño es 0bytes.

Una vez se tenga el resultado del ABW se organiza un paquete con la información y se envía a la aplicación cliente donde se muestra en pantalla el resultado.

3. NetFPGA y Aplicaciones del *TIMESTAMP*

La plataforma NetFPGA fue desarrollada por la Universidad de Stanford, es un dispositivo de bajo costo. Permite a los investigadores e instructores construir prototipos de sistemas de redes de alta velocidad, acelerados por hardware. La plataforma puede ser utilizada en el aula para enseñar a los estudiantes como construir conmutadores Ethernet y routers IP (del inglés *Protocol Internet)* además puede ser utilizada por los investigadores para crear prototipos de servicios avanzados para las redes. [22]

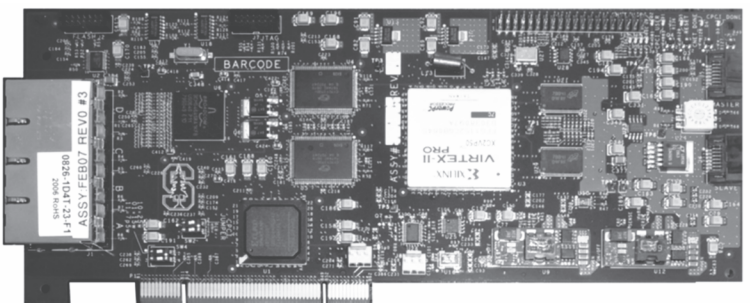

Fig. 2. Tarjeta de NetFPGA.
Fuente: [23]

Uno de los diseños de referencia de la NetFPGA es tarjeta de red, la cual tiene una estructura lógica compuesta por los módulos *Input Arbiter, Output Port Lookup, Output Queues* los cuales conforman el corazón de la plataforma o el *User Data Path* debido a que son los encargados de direccionar los paquetes desde el Host hacia la red y viceversa. Además cuenta con los módulos *SRAM interface, DRAM Interface y Register IO,* este último permite el intercambio de información de la NetFPGA con una aplicación en el Host. Este diseño se muestra en laFig. 3.

Dada la versatilidad de la NetFPGA para el desarrollo de nuevos prototipos de equipos de *Networking* a través de la adición de módulos programados en lenguaje VHDL de la FPGA de la cual dispone, se puede desarrollar un módulo que permita la marcación de tiempo en la recepción de trafico de prueba a nivel físico antes de que el paquete pase a los procesos del sistema operativo del host. De esta manera mejorar la precisión de la estimación de ABW debido a las características físicas de la tarjeta como son una alta velocidad en el procesamiento de los paquetes, un reloj de 125MHz y cuatro puertos para entrada y salida de datos de un Gbps de velocidad.

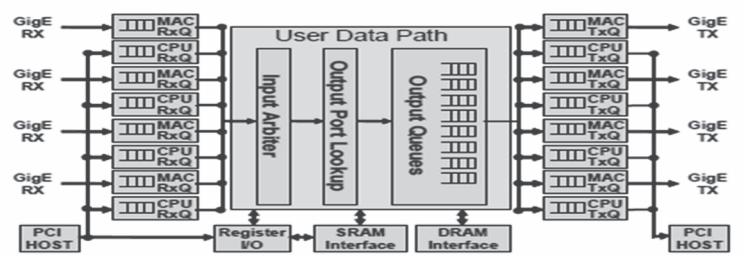

Fig. 3. Estructura Lógica de la NetFPGA como Tarjeta de Red.
Fuente: [24]

3.1 Estudio de Módulo de *timestamp* en Aplicaciones con la NetFPGA

En la literatura han sido encontrados y seleccionados tres proyectos que permiten identificar dos esquemas de Timestamp en sus aplicaciones a través de la NetFPGA. Estos son el proyecto ***Packet Generator*** desarrollado por Covington, et il [24], y el trabajo de investigación "***HATS: High Accuracy Timestamping System Based on NetFPGA***" desarrollado por Zhou, et il [25].

En el proyecto *Packet Generator*, está basado en el diseño de referencia en la NetFPGA como tarjeta de red estándar o NIC. La estructura lógica

de la *NetFPGA* se muestra en la Fig. 4 en esta se observan los módulos *timestamp, packet capture, Rate Limiter* y *Delay,* adicionales a la estructura lógica del diseño de referencia. Este proyecto utiliza el timestamp para registrar el tiempo de llegada de los paquetes. Cuando los paquetes de prueba ingresan a la NetFPGA se toma el tiempo de llegada en el módulo *timestamp*, creando una tabla con los datos contador de paquetes y el valor del timestamp este último tiene un tamaño de 64 bits. El valor del tiemsptamp es guardado en la carga útil del paquete en el módulo *packet capture* en, se calcula nuevamente el tamaño del paquete para modificar el encabezado con el valor adecuado. Entre las características principales del módulo *timestamp* se encuentra que trabaja con una frecuencia de reloj de 125MHz lo que genera un periodo de 8ns. Se realiza un conteo de los ciclos de reloj a partir de la llegada del primer paquete.

En el trabajo de investigación "***HATS: High Accuracy Timestamping System Based on NetFPGA***" desarrollado por Zhou, et il [25] se realiza el marcado de tiempo en hardware desde la *NetFPGA*. Los módulos utilizados en el *timestamping* se muestran en la Fig. 5. En este caso, el módulo *timestamp* se encuentra por fuera del *User Data Path,* este tiene una resolución de 8ns debido a que trabaja con el reloj de 125MHz de la *NetFPGA*. Adicionalmente puede realizar el marcado tanto en la transmisión como en la recepción, al utilizar un contador general, el cual está conectado a las interfaces Ethernet de transmisión y recepción de la tarjeta.

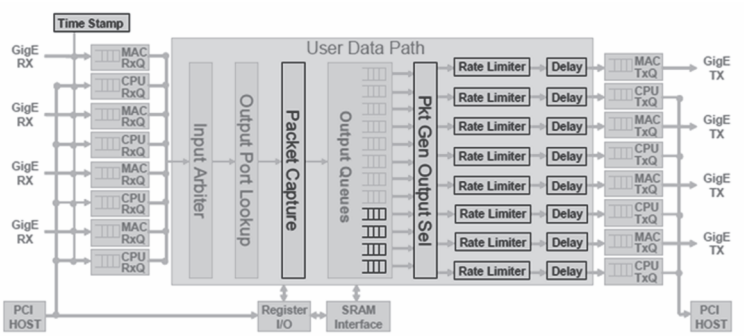

Fig. 4. Estructura lógica del proyecto "*Packet Generator".*
Fuente: [24]

Cuando ingresa una paquete a la tarjeta, se activa el modulo Time Stamp Rx. Este maneja registros de 64bits, de los cuales 16 sirven para llevar el conteo de los paquetes y 48bits para registrar el valor timestamp generado por el módulo Times Stamp Counter, esta información es guardada en la RAM Rx. Cuando termina la recepción de los paquetes

una aplicación se comunica a través de los registros IO para leer los datos guardados en la RAM Rx. Como la lectura de los registros no se realiza directamente sobre la RAM se utiliza el módulo Time Stamp Register para el control de la lectura de los registros. De forma similar funciona para la Transmisión.

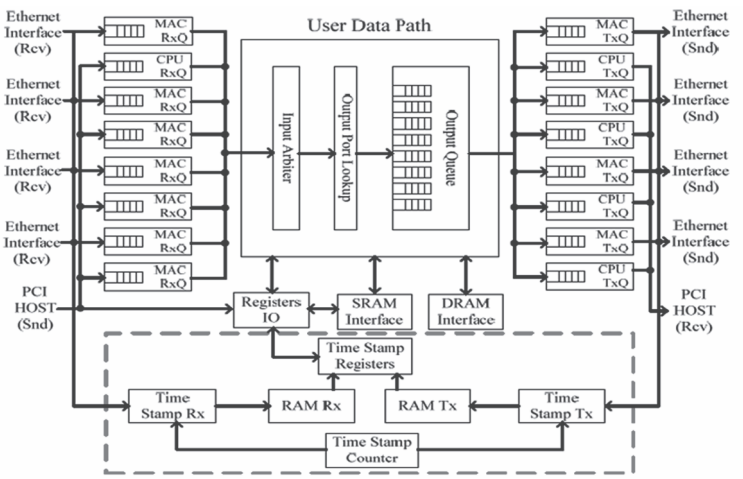

Fig. 5. Estructura lógica del proyecto "*HATS*"**.**
Fuente: [25]

3.2 Comparación entre los Esquemas de Timestamp

En la Tabla 1 se exponen las ventajas y desventajas más relevantes relacionadas con el módulo *Timestamp* de acuerdo con los proyectos anteriores mencionados. El módulo *timestamp*, se puede desarrollar de dos formas similares en cuanto al momento en que toma el *timestamp* se realiza antes del ingreso del paquete a los *FIFOS MAC* y distintos en cuanto a la forma de hacer llegar la información a la aplicación. En el proyecto *Packet Generator* adiciona la información al paquete mientras en *HATS* la información del timestamp se guarda en la RAM y luego se pasa a la aplicación.

Para este diseño, de acuerdo con lo anterior la opción de trabajo es el esquema utilizado en HATS pero solo para la Recepción de los paquetes se realiza el *timestamp.* Para ello se va a usar dos registros de 32bits formando un registro de 64bits, de los cuales los 16 bits más significativos corresponden a un identificador de paquete y los 48 restantes a la marca de tiempo, al utilizar el reloj de núcleo de 125Mhz, se tiene una resolución de 8ns por ciclo de reloj, y utilizando dos

registros de 32bits en el contaron y solo utilizando 48 bits, se tiene 2^48 o 281474976710656 valores por 8ns, de cada ciclo, dando un tiempo de marcado de 625 horas aproximadamente, lo suficiente para realizar una prueba de estimación de ancho de banda disponible.

Módulo timestamp	Ventajas	Desventajas
HATS	No genera retardos adicionales dados por el procesamiento de los paquetes	Requiere sincronización para la lectura de los registros guardados en la *RAM* y el acceso a esta desde la aplicación se hace de manera indirecta a través de los registros IO de la *NetFPGA*
PACKET GENERATOR	Información del timestamp es agregado en el paquete. Selección del tipo de paquetes al que se le realiza la marca de tiempo	Mayor Retardo en el procesamiento por los módulos *packet capture* en la ruta de datos de la NetFPGA

Tabla 1. Comparación entre los módulos de Timestamp.

4. Diseño del Esquema de Marcación

En esta sección se describe el diseño plantado para realizar el marcado de tiempo en la recepción de paquetes de prueba a través de la NetFPGA y la transferencia de la información entre la NetFPGA y la herramienta de estimación de ancho de banda disponible *TRACEBAND* como se muestra en la Fig. 6.

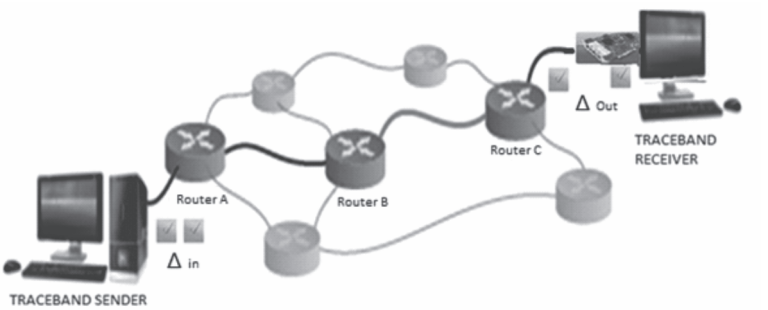

Fig. 6. Esquema general de herramienta propuesta.

4.1 Diseño del módulo *Timestamp*

En la Fig. 7 se muestra el modelo a utilizar en este diseño basado en la estructura HATS descrita en la sección 3.1, este módulo funciona de la siguiente manera, al detectar en las entradas de las interfaces Ethernet (Recepción) la llegada de un paquete el módulo *Time Stamp Rx* se habilita y guarda el *timestamp* junto con el identificador del paquete, este último es el valor de un contador de paquetes; para obtener la marca de tiempo, el módulo se comunica con el módulo *Time Stamp Counter* y este le envía el valor del contador en ese momento. El módulo *Time Stamp*

counter comienza su conteo una vez se descargan los *bitfiles* a la NetFPGA y aumenta con cada ciclo de reloj. Al tener los dos valores *Id_pack y timestamp* se almacena en la RAM de la FPGA como un registro de 64bits. Para la lectura de la RAM es necesario un módulo que lea la RAM cada vez que la aplicación lo solicite, esto es, la aplicación se comunica con REGISTER IO le solicita el envío de los timestamp después del tiempo de muestreo, REGISTER IO se enlaza con el módulo TIME STAMP REGISTER, este hace un barrido en las direcciones de la memoria y guarda el valor para pasarlo a REGISTER IO quien se encarga de pasarlo a la aplicación. Si se hace el barrido de todos los paquetes pueden tenerse en cuenta paquetes descartados en el módulo de IDENTIFICACIÓN; para ello se realiza un conteo de los paquetes descartados y del número de paquete entrantes hasta ese momento almacenados en un registro de 64bits, donde estará formado por el registro de 32bit cnt_pck_desc es el número de paquetes descartados y el número de paquetes total. Al realizar la lectura de los *timestamp* se compara el identificador del paquete con el valor guardado, si estos no coinciden, el timestamp pasa al registro de REGISTER IO y de allí seguirá a la aplicación donde se realiza la estimación de ancho de banda disponible.

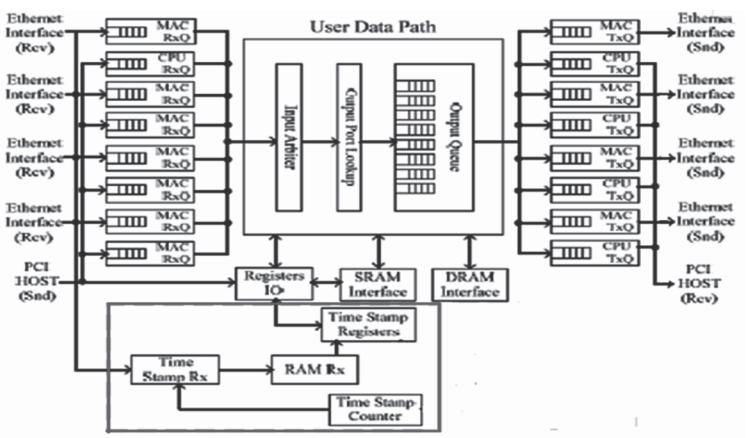

Fig. 7. Módulo Timestamp.

4.2 Diseño del Módulo de Identificación

De acuerdo con el diseño planteado en el módulo Timestamp, se hace necesario reconocer que paquete entrante, pertenece a los paquetes de prueba y que permita transferir a la aplicación los datos de *timestamp* correspondientes. El objetivo del módulo de identificación es reconocer que el paquete que ingresa a la ruta de datos de la *NetFPGA*, es un paquete UDP correspondiente a los paquetes de prueba enviados por

TRACEBAND. Para ello, se inserta en el User Data Path de la NetFPGA, un módulo que permita la identificación a través de los parámetros del encabezado TCP/IP como son protocolo y puerto; en este caso y en conformidad con Traceband corresponden a los valores 17 y 3504 respectivamente.

Debido al funcionamiento de la NetFPGA como tarjeta de Red, y teniendo en cuenta que la transferencia de información entre los diferentes módulos de esta se realiza a través de paquetes de datos de 64bits, paquetes de control de 8bits, y dos bit para la habilitación de lectura y escritura. Se plantea en la Fig. 8 la máquina de estados para el módulo de identificación del paquete.

Este módulo se basa en cinco estados, SIGUIENTE PAQUETE, EXTRACCION DE CAMPOS, IDENTIFICACIÓN Y ESCRITURA DE PAQUETE. La función de SIGUIENTE PAQUETE es anunciar la llegada de un nuevo paquete. Una vez la FIFO contiene los datos y encabezados del paquete, para ello requiere verificar el estado de la FIFO y decirle al módulo anterior que está en proceso de lectura. En el estado EXTRACCIÓN DE CAMPOS se encarga de guardar los valores de los campos como direcciones IP destino y IP origen, Puertos origen y destino, tipo de protocolo que corresponden al encabezado del paquete.

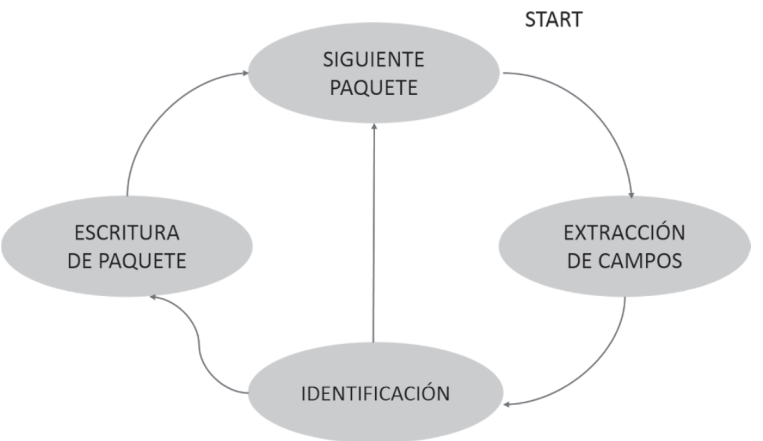

Fig. 8. Máquina de estado para el módulo Identificación.

En la Tabla 2 se pueden ver las variables involucradas en el proceso. La variable fsm_state es la encargada del cambio de estado en la máquina de estado, cnt_reset y cnt son variables que controlan el ciclo de extracción de campos; en este estado, se tienen en cuenta las variables in_fifo_data_rdy la cual indica que se está leyendo los datos de la fifo, e in_fifo_ctrl es la variable que se verifica para conocer el estado de la

lectura de la fifo, cuando esta llega a 'hFF significa que es el última palabra, protocol_field es la encargada de guardar la información del protocolo y port_dtn del puerto destino permiten verificar sí el paquete corresponde a un paquete de prueba, se lleva la cuenta de los paquetes entrantes de prueba, a los cuales se les coloca un identificador que será usado en el momento de pasar el *timestamp* a la aplicación, y se cuentan también los paquetes que son descartados y se lleva la cuenta del total de los paquetes con las variables cnt_pck_prueba, cnt_pck_des y cnt_pck_total.

ESTADO ACTUAL	ESTADO SIGUIENTE	VARIABLES DE ENTRADA	VARIABLES DE SALIDA
Siguiente Paquete	Extracción De Campos	Fsm_state=0	Cnt_reset=1 Nfsm_state=1
Extracción de Campo	Identificación	In_fifo_data_rdy=1 In_fifo_ctrl='hFF Cnt=1	Nfsm_state=2
Identificación	Escritura de paquete	Protocol_field=17 Port_dtn=3504	Nfsm_state=4
Escritura de paquete	Siguiente paquete	Out_rdy	Nfsm_state=0 Out_data
Identificación	Siguiente paquete	Protocol_field=?	Nfsm_state=0

Tabla 2. Relación de estados y variables de entrada y salida.

4.3 Comunicación entre Traceband_rcv y la NetFPGA

Al revisar del código de la aplicación Traceband_rcv la cual se encarga de recibir los paquetes de prueba y el procesamiento de estos paquetes para la estimación de ancho de banda disponible. Su funcionamiento se inicia llevando a las condiciones iniciales las variables vinculadas a cada estado. Al llegar un paquete, se realiza el marcado de tiempo y se guarda la información del paquete, luego se registran en una tabla los datos como número de paquete(pck_num), tamaño del paquete (pck_size), marca de tiempo de trasmisión en segundos y microsegundos (snd_time); y la marca de tiempo de recepción (rcv_time), se revisa si se finalizó la recepción de los paquetes; si es así, se procede a realizar la estimación de ancho de banda disponible con los datos guardados en la tabla y el resultado del Modelo Oculto de Markov, si no se procede a recibir el siguiente paquete hasta que llegue el último paquete de prueba cuyo tamaño es 0bytes.

En el apartado anterior se menciona que en la aplicación Traceband_rcv al ingresar un paquete de prueba se realiza el *timestamp* esto lo realiza con la función SCM_TIMESTAMP, y el resultado es guardado en la

tabla Traceband_table[pck_cnt]. En este diseño se pretende realizar esta función en hardware desde la NetFPGA para dar mayor precisión a la estimación de ancho de banda disponible. El proceso que se lleva a cabo en la NetFPGA se describe a continuación.

El *Timestamp* generado en el módulo Time_Stamp_Rx se guarda en la RAM de la misma. La RAM tiene una capacidad de 24576*64bits, lo que permite guardar 24575 datos, cuando se llega a la posición 24576, se reescribe la posición 0. Sin embargo, el máximo número de paquetes utilizado por Traceband_snd es en promedio 320; lo que es mucho menos que la capacidad de la RAM. Una vez se haya terminado de recibir los paquetes y en la RAM esté registrado el tiempo en que llegaron dichos paquetes, la aplicación debe solicitar el envío de la información y realizando un barrido de la RAM, esta información se guarda en la tabla traceband_table[pck_con]. rcv_time de *Traceaband*. Antes de guardar el *timestamp* en la tabla se debe realizar la conversión de nanosegundos a segundos y micro segundos puesto que en estas unidades de tiempo se halla snd_time.

El código en traceband_rcv para la comunicación con la NetFPGA, utiliza llamadas ioctl para hacer lecturas y escrituras de los registros del módulo REGISTER I/O. Las llamadas ioctl se desarrollan bajo dos funciones simples READREG y WRITEREG. Una vez se haya terminado de recibir los paquetes de prueba, se inicia la lectura de los registros de la RAM. En ese momento, el número total de paquetes estará guardado en el registro pkt_cnt y será las veces que se leerá la RAM.

Visto desde la NetFPGA, la comunicación puede darse de acuerdo con el trabajo desarrollado por Zhou Z [25]. En este, se incorpora un mecanismo de sincronización para realizar la lectura de los *timestamp* de los paquetes recibidos. Este mecanismo es utilizado en este proyecto, y su diagrama de flujo se muestra en la Fig. 9. La transferencia de la información es iniciada por la aplicación, esta escribe en el registro bandera (REG_BAN) el valor del estado leer, este registro es evaluado por el hardware y mientras esté en este estado, él procede a verificar si se leyó completamente la RAM, si no guarda en los registros reg_time_lo y reg_time_hi, de la marca de tiempo, el registro vuelve a cambiar de estado, cuando se está leyendo la RAM, este estado es leído por la aplicación guardando los valores y coloca el REG_BAN en Estado leer, cuando se ha terminado la lectura de la RAM se cambia el REG_BAN y tanto la aplicación como el hardware cierran el proceso hasta un nuevo llamado.

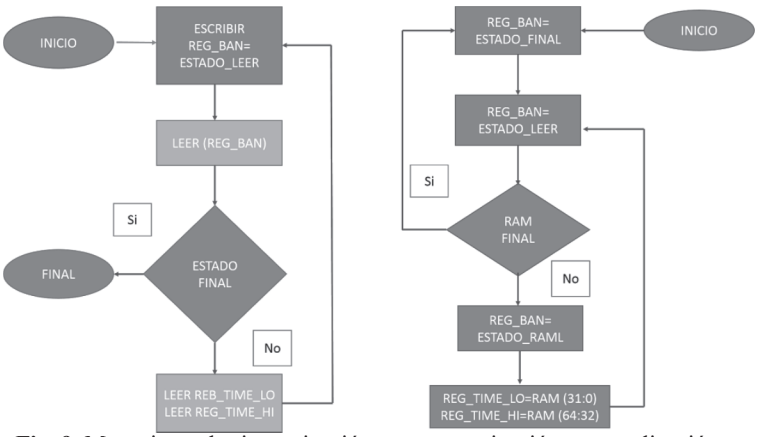

Fig. 9. Mecanismo de sincronización para comunicación entre aplicación y NetFPGA
Fuente: [25]

Es necesario declarar registros software cuando la información viaja de la aplicación al hardware y para comunicar el hardware con la aplicación deberá usarse registros tipo hardware, en este caso se utilizarán tres tipo hardware (reg_ban1, timestamp_lo y timestamp_hi) y uno tipo software, (reg_ban2) se exportan a través de la interfaz PCI del host.

Además de realizar la lectura de la RAM debe identificarse en hardware si el *timestamp* guardado corresponde a un paquete rechazado en el módulo identificador de la NetFPGA, de ser así, el registro no se pasa a la aplicación y se continúa con la lectura de la siguiente posición de memoria.

5. Conclusiones

En este trabajo se diseñaron los mecanismos para realizar la estimación de ancho de banda disponible utilizando la herramienta TRACEBAND y NetFPGA. Esta medición requiere tiempo en que llegan los paquetes de prueba al receptor. Para dar mayor precisión a la estimación, se realizó el timestamp en hardware y en software el cálculo del ancho de banda disponible.

En la NetFPGA se utilizaron los módulos Time_Stamp_Counter, Time_Stamp_Rx y Time_Stamp_Register, y el módulo Register IO, y la memoria RAM para el Timestamp. Estos módulos se encuentran fuera del *user_data_path* para no tener en cuenta el tiempo de procesamiento del paquete. El primer módulo mencionado es

encargado de conteo de pulsos del reloj, a una frecuencia de 125Mhz, para una resolución de 8ns por ciclo en la NetFPGA de esta desde que se descargan los bitfile. En el módulo Time_Stamp_Rx se diseñó de tal manera que al ingresar un paquete por las interfaces Ethernet se registre el tiempo en que llegó tomando el tiempo del Time_Stamp_Counter y es almacenado en la memoria RAM de la NetFPGA. En el módulo Time_Stamp_Register se encarga de almacenar el número de paquetes descartados y su posición dentro de los paquetes totales entrantes, mientras se realiza el muestreo. Cuando tranceband_rcv lo requiere, este se encarga de leer la RAM para pasar el timestamp solo de cada paquete de prueba.

Adicionalmente se diseñó un módulo llamado IDENTIFICACIÓN, ubicado luego del módulo Output_Port_Lookup, del user data path el cual permite comprobar si el paquete entrante es un paquete de prueba, utilizando los parámetros puerto y protocolo del encabezado TCP/IP de los paquetes. Se lleva la cuenta de los paquetes de prueba, paquetes descartados y el total de los paquetes, esto se envía al módulo Time_Stamp_Register.

Para comunicar Traceband con la NetFPGA, se hizo necesario declarar tres registros de tipo hardware y uno de tipo software, los cuales permiten la transferencia de la información del timestamp y del estado del registro bandera, que es el encargado de dar inicio a la lectura y estado del proceso de lectura de la RAM. En traceband_rcv se incluyen las librerías que permiten la identificación del dispositivo, la interface Ethernet de la NetFPGA. La utilización de las llamadas ioctl con el uso de las funciones writereg y readreg se realiza la escritura y lectura de los registros del módulo REGISTER IO respectivamente

Referencias

[1] M. &. D. C. Jain, «Pathload: A measurement tool for end-to-end available bandwidth.,» *In proceedings of Passive and Active Measurement,* 2002.

[2] J. Strauss, F. Kaashoek y D. Katibí, «A measurement study of available bandwidth estimation tolls,» *Proceedings of the 3rd ACM SIGCOMM conference on Internet Measurement,* pp. pp 39-44, 2003.

[3] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil y L. Cottrell, «Pathchirp efficient available bandwith estimation for network path,» *In Passive and Active Measurement Workshop,* 2003.

[4] J. Sommers, P. Barford y W. Willinger, «A Proposed Framework

for Calibration of Available Bandwidth Estimation Tools,» *Computer and Caomunications IEEE,* pp. 709-718, 2006.

[5] G. Jin y B. Tierney, «Nestet: A tool to measure the maximum burts size, available bandwidth and achievable throughput,» *Proceedings In information Technology: Research and Education ,* pp. 578-582, 2003.

[6] B. Melander, M. Bjorkman y P. Gunniengberg, «A new ent to end probing and analysis method for estimating bandwidth bottlenecks,» *Global Telecommunications Conference IEEE,* pp. 415- 420, 2000.

[7] T. Goldoni y E. Rossi, «Assolo, a new method for available bandwidth estimation,» *Internet and protection IEEE,* pp. 130-136, 2009.

[8] Q. Wang y L. Cheng, «FEAT: Improving accuracy in end to end available bandwidth measurement,» *Global Telecommunications Conference IEEE,* pp. 1-4, 2006.

[9] N. Hu y P. Steenkiste, «Estimating Available Bandwidth Using Packet Pair Probing,» *IEEE,* p. 27, 2002.

[10] J. Strauss, D. Katabi, F. Kaashoek y B. Prabhakar, «Spruce: A lightweight end to end toll for measuring available bandwidth,» *Internet measurement Conference IEEE,* 2003.

[11] P. Sharma, «Study and analysis of bandwidth flow estimation techniques for wired/wireless networks,» *Computer techology and applications,* vol. 3, nº ISSN:2229-6093, pp. 130-137, 2012.

[12] L. Lao, C. Dovrolis y M. Sanadidi, «The probe gab model can underestimate the available bandwidth of multihop path,» *ACM SIGCOMM Computer comunication review,* vol. 36, pp. 29-34, 2006.

[13] A. Alí y F. Lepage, «IGMPS, a new tool for estimating end to end available bandwidth in IP network paths,» *Networking and service IEEE,* pp. 115-120, 2007.

[14] J. Navratil y R. Cottrell, «ABING: A practical approach to available bandwidth estimation,» *Passive and active measurement workshop IEEE,* 2003.

[15] C. D. Guerrero y M. A. Labrador, «Traceband: A fast, low overhead

and accurate tool for available bandwidth estimation and monitoring,» *Computer Network Elsevier,* pp. 977-990, 2009.

[16] C. D. Guerrero y M. A. Labrador, «Experimental and Analytical Evaluation of Available Bandwidth Estimation Tools,» *IEEE,* pp. 710-717, 2006.

[17] E. Goldoni y M. Schivi, «End to end available bandwidth estimation tools, an experimental comparison,» *Proceedings of TMA, IEEE,* pp. 171-182, 2010.

[18] V. Paxson , «End-to-end internet packet dynamics,» *IEEE/ACM transactions on networking,* vol. 7, nº 3, pp. 277-292, 1999.

[19] F. Michaut y F. Lepage, «Application oriented network metrology: metrics and active measurement tool,» *IEEE communications survey y tutorials,* vol. 7, nº 2, pp. 2-24, 2005.

[20] H. Zhou, Y. Wang, X. Wang y X. Huai, «Difficulties in estimating available bandwidth,» *IEEE internacional conference on comunications,* 2006.

[21] P. Loschmidt, R. Exel y G. Gadere, «Highly Accurate Timestamping for Ethernet-Based,» *Journal of Computer Networks and Communications,* p. 11, Diciembre 2011.

[22] Digital Design Engenier's Source, «DIGILENT Inc,» 2000. [En línea]. Available: http://www.digilentinc.com/NavTop/AboutUs.cfm#info. [Último acceso: 3 JULIO 2013].

[23] NetFPGA, «NetFPGA,» 2013. [En línea]. Available: www.netfpga.org. [Último acceso: 15 07 2013].

[24] G. A. Covington, G. Gibb, N. McKeown y J. W. Lockwood, «A packet generator on the NetFPGA plataform,» *Field programmable custom computing maquines IEEE,* pp. 235-239, 2009.

[25] Z. Zhou, L. Cong, G. Lu, B. Deng y X. Li, «HATS: High Accuracy Timestamping System Based on NetFPGA,» *International journal of future generation communication and Networking,* p. 12, Septiembre 2010.

[26] G. Salmon, M. Ghobadi, M. Labrecque, Y. Ganjali y J. G. Steffan, «NetFPGA based precise traffic generation,» *Proc of NetFPGA Developers Workshop,* 2009.