ESQUEMA DE TIMESTAMP EN LA RECEPCIÓN DE PAQUETES DE PRUEBA A TRAVÉS DE LA NETFPGA PARA LA ESTIMACION DE ANCHO DE BANDA DISPONIBLE EN TRACEBAND

NYDIA S. SANDOVAL*§, CESAR D. GUERRERO*

Resumen

Los servicios y aplicaciones que se ofrecen actualmente sobre internet como video bajo demanda, requieren de 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 a nivel de software como Pathload, Spruce y Traceband. Estas herramientas presentan limitaciones en la estimación debido a los errores por retardos que sufren los paquetes en los procesos del sistema operativo, la intrusión que se genera sobre el canal, y el tiempo de estimación. En el diseño del esquema de marcación de los tiempos de llegada de los paquetes de prueba se utiliza la herramienta Traceband, está se encarga del envió 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 obtener los datos de 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 in the estimation due to errors delays experienced by packets in the operating system processes. 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 reception hardware and based on this estimation value is performed ABW. The timestamp and Identification modules in NetFPGA, and the required changes in Traceband for communication and data collection from the NetFPGA were designed.

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

* Maestría en Telemática, Facultad de Sistemas, Universidad Autónoma de Bucaramanga. Avenida 42 No 48 – 11 Bucaramanga Colombia. {nsandoval697, cguerrer}@unab.edu.co

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 su variabilidad en el comportamiento del tráfico. La medición del ancho de banda es importante, dado que éste, es finito, no es gratuito, y su demanda sigue creciendo. Para estimar el ancho de banda disponible de extremo a extremo sobre una ruta de red, se han desarrollado diferentes aplicaciones a nivel de software, las cuales se realizan a través de mediciones activas y requiere de operaciones estadísticas sobre los valores de dispersión de los paquetes, basados en los modelos *Probe Rate Model* (PRM) y *Probe Gap Model* (PGM). 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 acorde al tiempo de transmisión de las pruebas sobre el cuello de botella del enlace.

1.1 TRABAJOS RELACIONADOS

Las herramientas encontradas en la literatura que aplican el modelo PRM son Pathload [1], PathChirp [2], YAZ [4] [3], Nestet [4], TOPP [5], ASSOLO [6], y FEAT [7] , aquellas que aplican el modelo PGM son IGI/PTR [8], Spruce [9], ProbeGap [11], Delphi [12], IGMPS [10], Abing [11] y Traceband [12].

De acuerdo a los trabajos de Guerrero [13], en el que se comparan las herramientas Pathload, IGI y Spruce a través de los parámetros de evaluación: Error de estimación, tiempo de estimación, sobrecarga y fiabilidad, realizando 11760 experimentos, donde 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.

En cuanto a las comparaciones realizadas por Goldoni y Shivi entre diferentes herramientas [14], los autores comparan varias herramientas bien conocidas 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 muestran 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, mientras que Assolo exhibe una alta precisión, baja intrusión, y el tiempo de convergencia es corto.

Traceband desarrollada por Guerrero y Labrador [12] es una herramienta de estimación de ancho de banda disponible de extremo a extremo que basa la estimación en el modelo PGM, la cual en estudios comparativos con Pathload y Spruce, presenta mejor rendimiento en cuanto a la baja sobrecarga similar con Spruce y presenta una precisión en la medición similar con la que presenta Pathload.

A pesar de los esfuerzos por mejorar el *performance* de las mediciones, las herramientas requieren a mayor precisión y fiabilidad en la estimación, utilizando el menor tiempo de medición, y una baja sobrecarga en el canal.

1.2 LIMITACIONES DE LAS ABET

Una de las limitaciones de las herramientas de estimación de ancho de banda disponible tiene relación con los procesos como la generación de paquetes y registro del tiempo de llegada (*Timestamp*) de los mismos, ya que la separación inicial de los paquetes de prueba es determinada por la aplicación (*Userspace*) de la **¡Error! No se encuentra el origen de la referencia.** y 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.

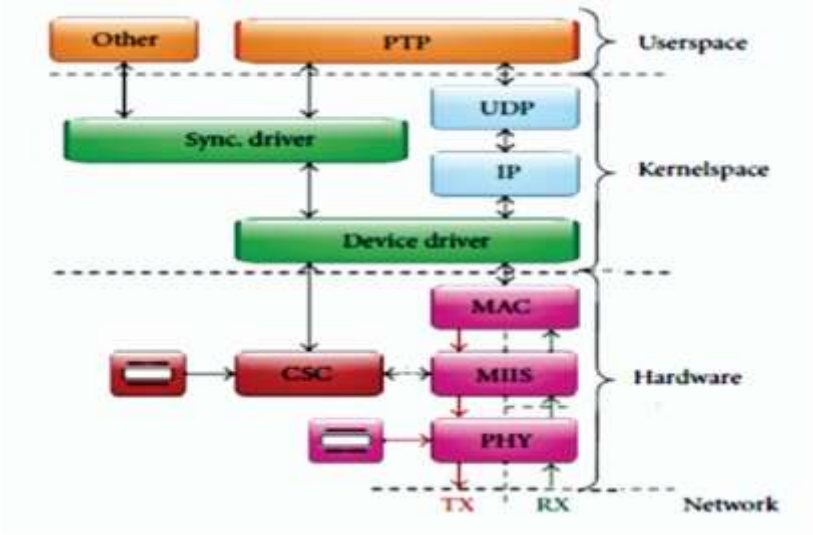

Fig. 1 Ruta de los paquetes de datos Fuente [15]

2. METODOLOGÍA

En esta investigación se utiliza como base la herramienta de estimación de ancho de banda disponible *TRACEBAND* y la plataforma *NetFPGA* como se observa en la Fig. 2. *TRACEBAND* funciona mediante dos aplicaciones *sender* y *receiver*. La aplicación *TRACEBAND* sender se encarga del envió de los paquetes de prueba a un tiempo de separación Δin determinado. En el receptor, se colocará la *NetFPGA* como tarjeta de red, esta realizará las marcas de tiempo en cada paquete de prueba. Este resultado, es necesario para realizar la estimación de ABW por la aplicación *TRACEBAND* receiver.

En este proyecto, para cumplir con el diseño planteado anteriormente se establecen 3 fases de desarrollo, estas son:

- Análisis de *TRACEBAND*, en ella se estudian los algoritmos de *TRACEBAND* sender y *TRACEBAND* receiver, esto permite determinar en el momento en qué se hace el *timestamp* en el paquete de prueba y cuando se utiliza el resultado de la marcación para la estimación del ancho de banda disponible.
- Diseño del módulo de *Timestamp* y filtrado en la *NetFPGA*, en esta fase, se determina donde se inserta el m~~o~~ódulo y sus características, es decir la frecuencia de reloj, la precisión en la marca de tiempo, registro del *timestamp* entre otras, para incluirlas en el módulo encargado de la marcación en hardware del paquete.
- Diseño del módulo de comunicación *TRACEBAND* y *NetFPGA*, una vez se cuente con el módulo de marcación el siguiente paso es determinar el protocolo que se usará para hacerle llegar el paquete de prueba con la marca de tiempo realizada en la *NetFPGA* a la herramienta de estimación *TRACEBAND*.

3. RESULTADOS

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*.

3.1 Análisis de *TRACEBAND*

TRACEBAND es una herramienta de estimación de ancho de banda disponible desarrollada por Guerrero y Labarador (2009). 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. Con 50 pares de paquetes el modelo mostró los mejores resultados de la estimación.

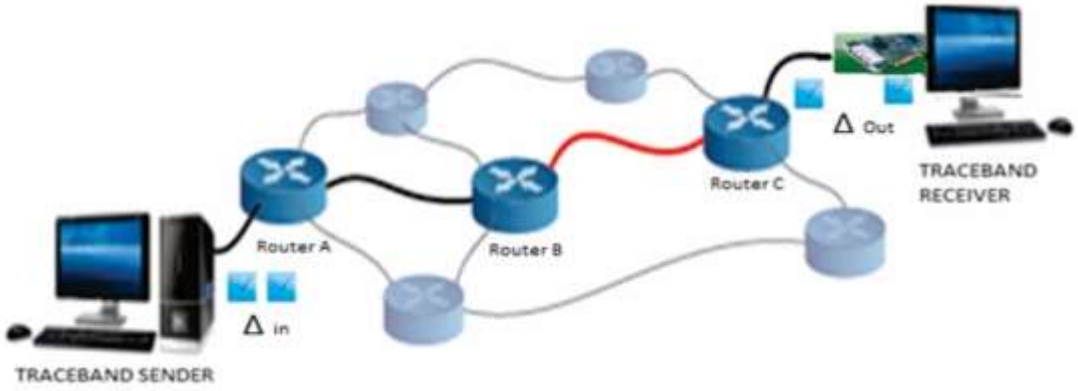

Fig. 2 Esquema general de herramienta propuesta

Al realizar una revisión del código de la aplicación de 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.
Al llegar un paquete, se realiza el marcado de tiempo y se guarda la información del paquete, luego se registra en un 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.

3.2 Utilización del módulo -TIMESTAMP en aplicaciones con la NetFPGA

En el proyecto ***NetFPGA-Packet Generator*** desarrollado por -Covington, Gibb, McKeown, & Lockwood [16] y ***NetFPGA –based Precise Traffic Generation*** (PTG) desarrollado por Salmon, Ghobadi, Labrecque, Ganjali, & Steffan —[17] en los dos proyectos se implementan procesos de marcación de paquetes, este proceso se realiza operando la NetFPGA como una tarjeta de red estándar o NIC de referencia, donde su uso está orientado a garantizar la precisión a la hora de la captura de los paquetes recibidos. La estructura lógica de la *NetFPGA* para el proyecto *Packet Generator* se muestra en la Fig. 3, en la cual se observa que el módulo *Timestamp* se encuentra fuera del *USER DATA PATH* y el valor de la marca de tiempo la guarda dentro del paquete en el módulo "*Packet Capture"*.
En el trabajo de investigación "***HATS: High Accuracy Timestamping System Based on NetFPGA***" desarrollado por Zhou, Cong, Lu, Deng y Li [18] se realiza marcado de tiempo en hardware desde la *NetFPGA*. Los módulos utilizados en el *timestamping* se muestran en la xx
En este caso, el módulo -*timestamp* se encuentra por fuera del *User Data Path,*

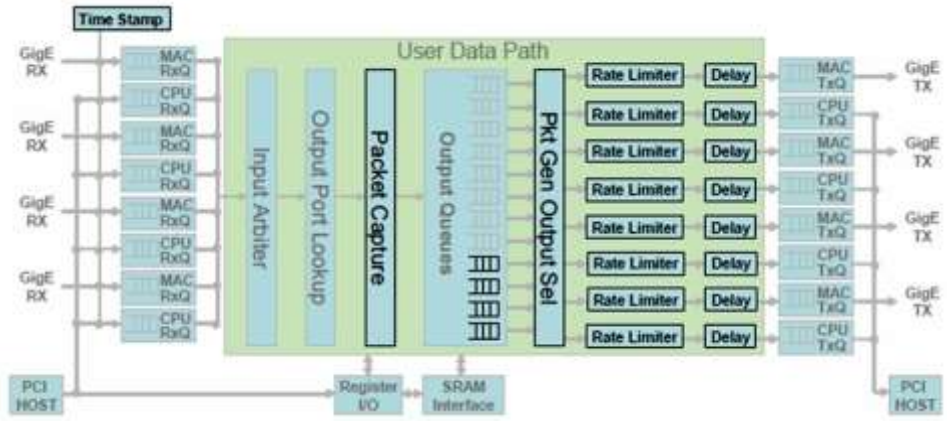

Fig. 3. Estructura lógica del proyecto "*Packet Generator*"

Fuente: [16]

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.

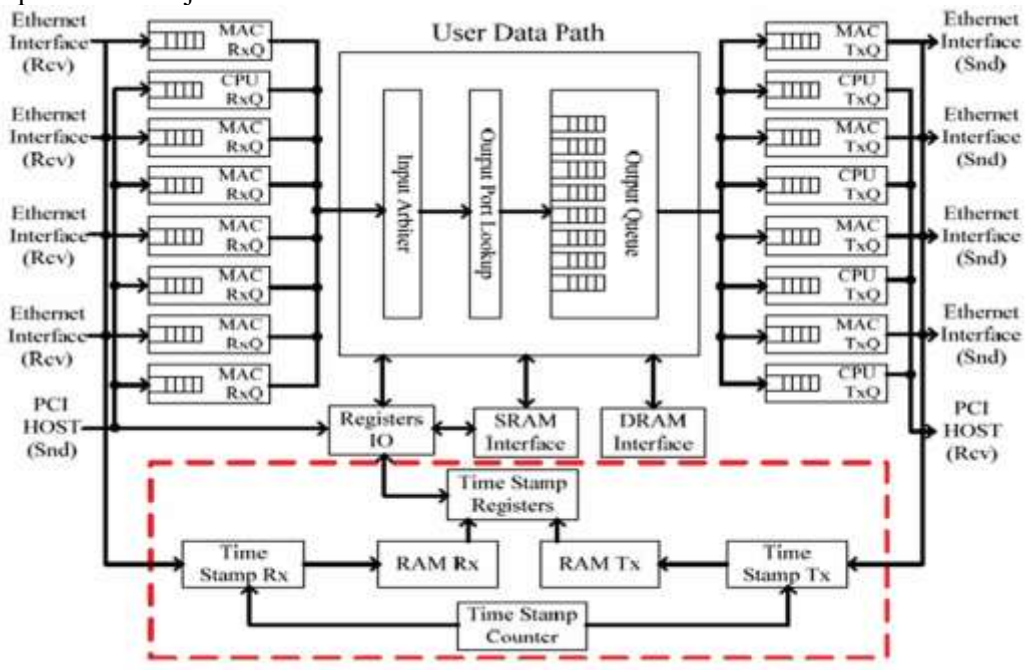

Fig. 4. Estructura lógica del proyecto "*HATS*"

3.2.1 Ventajas y desventajas de los módulos de Timestamp

En la Tabla 1 se exponen las ventajas y desventajas más relevantes relacionadas con el módulo *Timestamp* de acuerdo a los proyectos anteriores mencionados. El módulo *timestamp*, se puede desarrollar de dos formas de acuerdo a lo discutido en la sección 3.2 y a los resultados

expuestos en la Tabla 1, la primera donde el módulo se halla antes de los *FIFOS MAC*, el mayor inconveniente de esta es que se requiere de 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*. La segunda opción requiere de mayor tiempo de procesamiento, puesto que se adiciona *el timestamp* como un paquete *NetFPGA* de 64 bits, la cual no permitiría mejorar la precisión en el estimador de ancho de banda disponible.
Para este diseño, de acuerdo a lo anterior la mejor opción es la primera forma de realizar 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, dando un tiempo de marcado de 78 horas aproximadamente, lo suficiente para realizar las pruebas de estimación de ancho de banda disponible

Tabla 1. Comparación entre los módulos de Timestamp

Módulo timestamp	Ventajas	Desventajas
Fuera del USER DATA PATH	No genera retardos adicionales dados por el procesamiento de los paquetes	Mayor uso de los recursos de la NetFGA como lo es la RAM en HAST
Dentro del USER DATA PATH	Información del timestamp agregado en la carga útil del paquete UDP Selección del tipo de paquetes al que se le realiza la marca de tiempo	Mayor Retardo por la creación de un módulo en la ruta de datos de la NetFPGA

3.3 Diseño del módulo *Timestamp*

En la Fig. 5 Módulo TimestampFig. 5 se muestra el modelo a utilizar en este diseño basado en la estructura HATS descrita en la sección 3.2-, 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 modulo *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 IDENTIFICACION, 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 para al registro de REGISTER IO y de allí pasará a la aplicación.

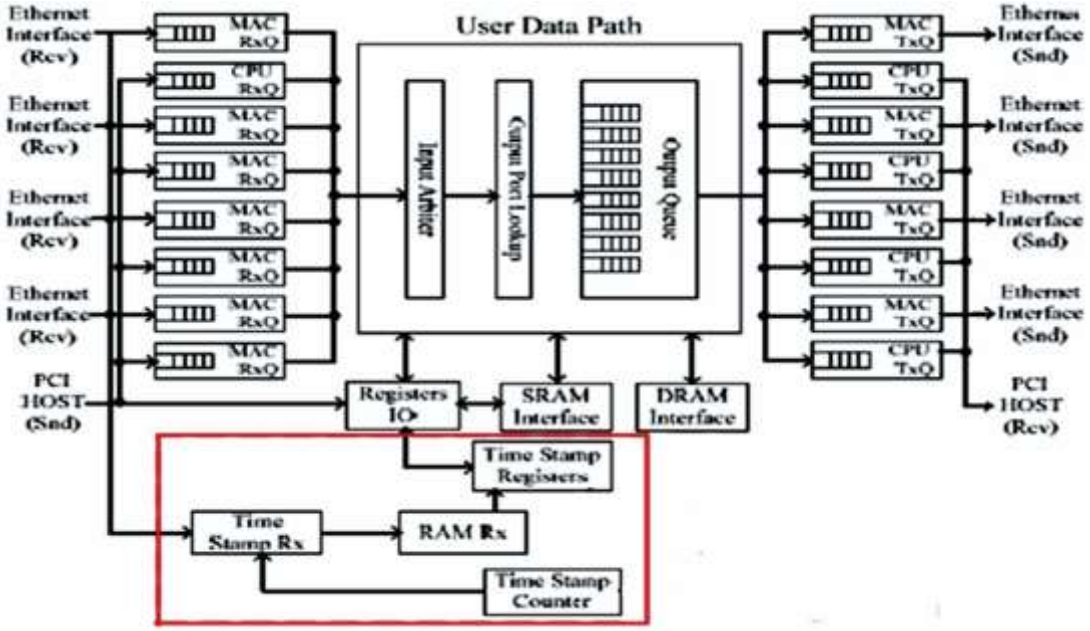

Fig. 5 Módulo Timestamp

3.4 Diseño del módulo de identificación

De acuerdo al diseño planteado en el módulo Timestamp, se hace necesario comprobar que los paquetes entrantes son paquetes de prueba. Para ello, se inserta en el User Data Path de la NetFPGA, un módulo que permita la identificación de estos paquetes 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 la siguiente máquina de estados:

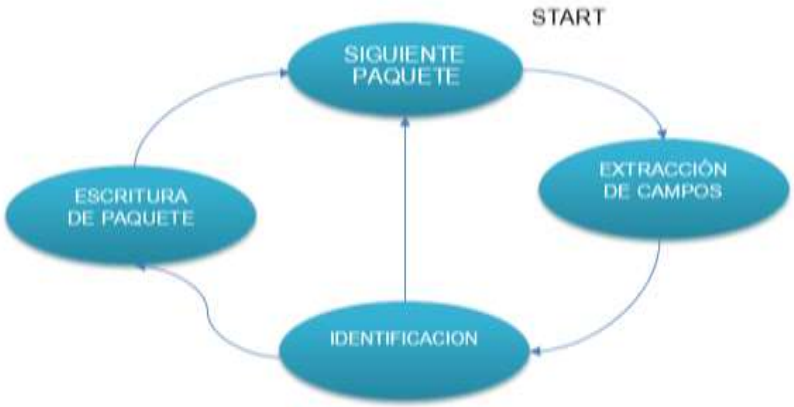

Fig. 6 Máquina de estado para el módulo Identificación

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.

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

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

3.5 Comunicación entre Traceband_rcv y la NetFPGA

Al realizar una revisión del código de la aplicación de 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 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.

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 este 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.

En el trabajo desarrollado por Zhou Z -[18] 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 . La transferencia de la información es iniciada por la aplicación, ésta escribe en el registro bandera (REG_BAN) el valor del estado leer, este registro es evaluado por el hardware y mientras este en este estado, el procede a verificar si se leyó completamente la RAM, sino 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 ésta 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.

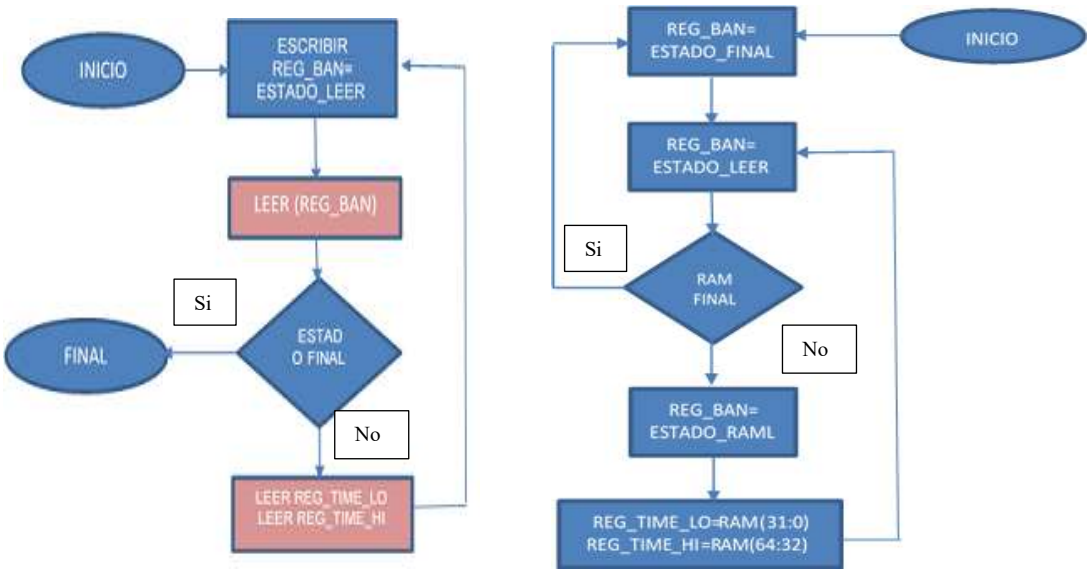

Fig. 7 Mecanismo de sincronización para comunicación entre aplicación y NetFPGA

Fuente [18]

En este diseño, basado en el mecanismo de sincronización descrito anteriormente, para que se pueda comunicar la aplicación con el hardware es necesario declarar el registro como registro software y para comunicar el hardware con la aplicación deberá usarse registros tipo hardware, en este caso se utilizaran tres tipo hardware (reg_ban1, timestamp_lo y timestamp_hi) y uno tipo software, (reg_ban2) se exportan a través de la interfaz PCI.

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.
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 envuelven en dos funciones simples READREG y WRITEREG. Una vez se haya terminado de recibir los paquetes de prueba, para ese momento, el número total de paquetes estará guardado en pkt_cnt y será las veces que se leerá la RAM. En el encabezado será necesario incluir las librerías para el manejo de los registros de la NetFPGA

4. 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 del tiempo en que llegan los paquetes de prueba al receptor. Para dar mayor precisión a la estimación, se realizó el timestamp a nivel de hardware y a nivel de 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
dicionalmente, se diseñó un módulo llamado IDENTIFICACION, 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

5. REFERENCIAS

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

[2] R. R. R. B. J. N. a. L. C. Vinay Ribeiro, «Pathchirp Efficient available bandwith estimation for network path,» *In Passive and Active Measurement Workshop,* 2003.

[3] 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.

[4] 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.

[5] 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.

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

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

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

[9] 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.

[10] A. A. Ali 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.

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

[12] 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.

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

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

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

[16] 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.

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

[18] 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.