VNIVERSIDAD NACIONAL AVPNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

FACULTAD DE INGENIERÍA

"*Diseño e Implementación de un Sistema Regulador de Tráfico utilizando un Procesador de Red IXP*"

T E S I S

QUE PARA OPTAR POR EL GRADO DE

MAESTRO EN INGENIERÍA

INGENIERÍA ELÉCTRICA-TELECOMUNICACIONES

P R E S E N T A :

ING: ELBA KARÉN SÁENZ GARCÍA

TUTOR:
DR. JAVIER GÓMEZ CASTELLANOS

2006

JURADO ASIGNADO:

Presidente: Dr. Miguel Moctezuma Flores

Secretario: Dr. Landeros Ayala Salvador

Vocal: Dr. Javier Gómez Castellanos

1er. Suplente: Dr. Ramón Gutiérrez Castrejón

2do. Suplente: Dr. Víctor Rangel Licea

Ciudad Universitaria, México, D.F.

TUTOR DE TESIS:

DR. JAVIER GÓMEZ CASTELLANOS

FIRMA

D e d i c a t o r í a

Dedico este trabajo

A Dios

A mis seres más queridos; mis padres Cristina y Jorge Roberto, mis hermanos América y Jorge Raymundo, mis sobrinos Alexamé y Alejando; y a novio y amigo Oscar.

A todas aquellas personas que me han apoyado y brindado su amistad

A g r a de c i m i e n t o s

Agradezco muy en especial a la Facultad de Ingeniería que me ha permitido desarrollarme en mis estudios

A todos mis profesores que me impartieron clase y me transmitieron sus sabios conocimientos durante mis estudios d maestría

Al Dr. Javier Gómez Castellanos y al Dr. Víctor Rangel Licea, por permitirme desarrollar este trabajo.

Tabla de Contenido

Página

Lista de Tablas

Página

Lista de Figuras

Resumen

La congestión de una red se puede definir como la condición que existe cuando la red no es capaz de satisfacer algunos de los objetivos de funcionamiento que han sido especificados o bien cuando no existe el suficiente ancho de banda para soportar la carga de tráfico presente.

Los traffic shapers se utilizan para prevenir la congestión y suavizar el tráfico, juega un papel importante en la realización de esquemas de control de tráfico en redes de alta velocidad para asegurar requerimientos de calidad de servicio. Los traffic shapers realizan la función de red llamada traffic shaping para regular la tasa media de tráfico de entrada a una red o subred y así asegurar que se ajuste a ciertos límites estadísticos.

El objetivo de este trabajo es diseñar e implementar un traffic shaper, al cual en este trabajo se le da el nombre de regulador de tráfico. Se pretende que permita regular tres diferentes tipos de tráfico, en si que sea posible utilizar el ancho de banda no usado por clases de tráfico consideradas más "caras", para transmitir datos de clases más "baratas".

Los resultados obtenidos fueron favorables, debido a que se regulan los tres tipos de tráfico planteados, y si se atiende con prioridad a las clases más "caras". El inconveniente es la pérdida de paquetes por el tamaño pequeño de los buffers para cada clase de tráfico.

El diseño e implementación de este sistema de red se desarrolla en una tecnología llamada Procesador de Red (Network processor), en especifico el IXP1200. La cual es considerada como una nueva generación para el diseño de sistemas de red.

Un Procesador de Red (NP) es un dispositivo de hardware de propósito especial programable, que combina el bajo costo y flexibilidad de un procesador RISC con la velocidad y escalabilidad de hardware, diseñado para funciones especificas de red].

CAPÍTULO 1

Aquí empezaremos por introducir la idea del porque se decidió implementar un sistema Regulador de Tráfico en una de las tecnologías relativamente nuevas, un Procesador de Red. Además brevemente se presenta una estructura de este trabajo.

1. Introducción

1.1. Problemática

Dado que una de las principales causas de la congestión en una red es que el tráfico es a ráfagas y que la conmutación de paquetes tiende a producir tráfico no uniforme, lo que se busca es formar el tráfico así que se podría forzar a los *sistemas de red que procesan paquetes (como un host)* a transmitirlos a una tasa más predecible. Una de las formas para solucionar los problemas de control de congestión, es regular la tasa media y la variabilidad (*burstiness*) del tráfico de entrada a una red o subred. Este mecanismo es ampliamente usado en redes ATM, y se denomina regulación o modelado del tráfico de entrada (*traffic shaping*) Figura 1.1. Aunque esta regulación es más fácil de implementar con (Circuitos Virtuales) CVs, puede aplicarse igualmente a subredes de datagramas (Internet).

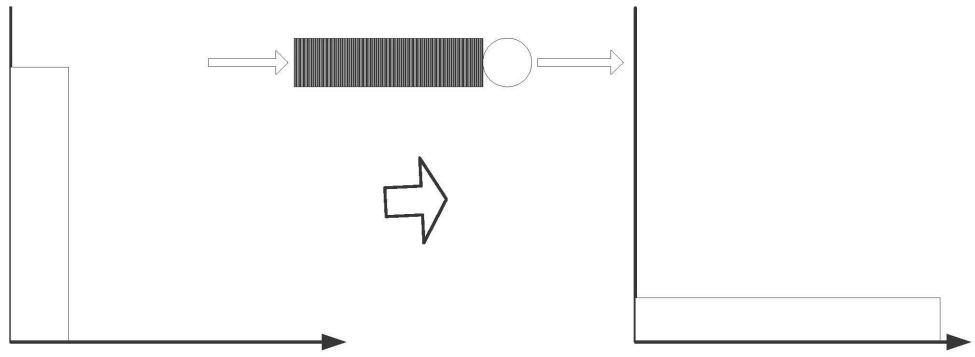

Figura 1.1 Conformado de tráfico (Shapping)

Desde el punto de vista de un proveedor de red, las ráfagas de tráfico hacen difícil fijar la capacidad necesaria o proveer de servicios garantizados. Por ejemplo, si un proveedor quiere ofrecer un servicio a una tasa de transmisión promedio de *N Mbps*, esto no sería posible. Él no puede asumir una máxima tasa de *N Mbps,* debido a que la transmisión puede incluir ráfagas. Esto es, un cliente puede enviar en un segundo ráfagas a la tasa de *4N Mbps* seguido por tres segundos de silencio.

Por lo anterior se considera que los Reguladores de Tráfico son necesarios en algunos sistemas de red.

Un sistema Regulador o de Tráfico (*Traffic Shaper*), es aquel que retarda algunos paquetes y permite a otros pasar más rápido, de acuerdo a un conjunto de especificaciones. Los ISPs lo utilizan, para prevenir congestiones y para dar prioridad a clientes que pagan más. Al obligar al tráfico a que se adhiera a ciertas especificaciones, el regulador puede asegurar que los paquetes puedan pasar sin que sean descartados [1].

El término regulación de tráfico (*Traffic Shaping*) se refiere al proceso de asegurar que el tráfico saliente se ajuste a ciertos límites estadísticos. La regulación puede ser aplicada a un flujo individual (por ejemplo al tráfico en una conexión especifica TCP) o al tráfico agregado (por ejemplo todo el tráfico de un sitio dado o a todo el tráfico destinado a un puerto TCP). Regular o moldear (*Shaping*) esta relacionado con aplicar políticas, pero los dos difieren en el objetivo global y los mecanismos usados. Las políticas implican límites duros, es decir, cuando el tráfico excede los límites las políticas de tráfico típicamente descartan los paquetes. En el caso de un regulador generalmente se implican límites suaves y no descarta paquetes.

Entonces se puede definir diferencias entre un sistema que aplica políticas de tráfico (*Traffic Policer*) y un sistema Regulador de Trafico; el primero funciona como un mecanismo para asegurar que el flujo de paquetes entrantes no exceda el ancho de banda convenido, la parte que excede este ancho de banda se marca como de baja prioridad o es descartado, y el Regulador de Tráfico como un mecanismo que reacomoda el patrón de llegada de los paquetes, manteniendo los paquetes por un cierto periodo de tiempo.

Cabe mencionar que la regulación de tráfico es una técnica utilizada para mejorar la calidad de servicio (QoS). La regulación de tráfico es una de las funciones del procesamiento de paquetes que puede ser incluida dentro de otro sistema de red (por ejemplo, algunos ruteadores lo tienen).

1.2. Objetivo

La idea de este trabajo es únicamente implementar un Regulador de Tráfico (*Traffic Shaper*) para modelar el tráfico saliente de una subred Ethernet a una tasa máxima limitada, específicamente para tres tipos de flujo:

- Paquetes
    - No Ip, e Ip no TCP ni UDP correspondientes a Ethernet
    - Paquetes 802.3
- Paquetes Ip UDP Ethernet
- Paquetes Ip TCP Ethernet

en una nueva tecnología, un Procesador de Red (IXP1200). De manera que el diseño pueda servir para utilizarlo en otros sistemas de red realizados en la misma tecnología, donde se requiera regular la tasa de salida de paquetes, dependiendo de una clasificación de los mismos.

Se usa un algoritmo que se denominó *Token Bucket* correlacionado y su objetivo es poder utilizar el ancho de banda no usado por clases de tráfico consideradas más "caras", para transmitir datos de clases más "baratas".

Algo importante es el manejo del término *ancho de banda*, que como se sabe es la capacidad de información transmitida en un canal de comunicación (el canal puede ser análogo o digital). En este trabajo no se referirá a las transmisiones analógicas, sino a transmisiones digitales que son medidas en bits por segundo. En si se refiere a la tasa de transferencia de datos.

1.3. ¿Porque el uso de un procesador de red?

Existen varias formas de llegar a implementar un sistema de red que procese paquetes, la más común, sencilla y barata es utilizando una arquitectura de una computadora convencional y software para la misma.

Existen otras que se mencionan en el capitulo 2 y no significa que la de utilizar la tecnología de un Procesador de Red sea la ideal.

El escoger esta tecnología es porque tiene la característica de ser flexible y programable, esto es, que se pueden agregar funciones extras a un sistema de red, programándolas en un software propio del fabricante. Además de ofrecer hardware especifico que realiza funciones de red que se pueden considerar comunes en la mayoría de los sistemas. Estas funciones al ser implementadas en hardware son más rápidas que las implementadas en software.

1.4. Estructura de la tesis

En esta tesis se constituye de la siguiente manera, en el capítulo 2 se explica la evolución de los sistemas de red hasta llegar a la tecnología de los Procesadores de Red, dando una breve descripción de su arquitectura de hardware y software. En el capítulo 3 se detalla a grandes rasgos la Arquitectura de hardware del Procesador de Red IXP1200 y su ambiente de desarrollo.

En el capítulo 4 se describe con un poco mas a detalle el sistema de red a implementar, así como su diseño en el Procesador de Red IXP1200.

El capítulo 5 se deja para la evaluación y análisis del sistema y el 6 para las conclusiones.

CAPÍTULO 2

En este capítulo se define lo que es un sistema de red y como han ido evolucionando hasta llegar a la cuarta generación, que es la aparición de los Procesadores de Red. Por último se da una breve descripción de que es un Procesador de Red, su arquitectura y funcionamiento.

2. Sistemas de red y tecnología de Procesadores de Red

Como se ha visto, la Internet ha ido creciendo a pasos agigantados, y con este crecimiento han surgido nuevas tecnologías de comunicaciones, así como el desarrollo de una amplia variedad de sistemas y tecnologías de red.

Muchas de las aplicaciones que actualmente se usan, fueron inventadas mucho tiempo después de que apareció Internet, como la www y la telefonía IP entre otras, y muchas de estas aplicaciones difieren bastante de las primeras. Lo mismo sucede con las tecnologías de comunicaciones, actualmente coexisten redes heterogéneas con nuevas y viejas tecnologías, tal que al incorporar las nuevas no se requiere cambiar los protocolos ni las aplicaciones. Con lo anterior surgen las siguientes preguntas. Si la Internet existente tiene suficientes funcionalidades para soportar todas las posibles aplicaciones.¿Por que se considera el diseño de nuevos sistemas de red?, ¿Porque el interés en crear nuevas arquitecturas e infraestructuras?.

Con respecto al crecimiento en general, una respuesta es, que debido al surgimiento de nuevas aplicaciones, muchas veces se requiere de un cambio en la infraestructura y arquitecturas disponibles, que permitan diferenciar los tipos de servicio, la seguridad y otras funciones de administración de red para que se pueda manejar todo el tráfico.

Por ejemplo, para que la telefonía IP trabaje bien se requiere de bajos retardos, y los bajos retardos solo se consiguen mejorando la infraestructura para reducir la congestión.

Internet es una red muy muy grande formada por varias redes de computadoras, y a su vez cada red contiene muchos componentes o dispositivos electrónicos, encargados de manejar paquetes de datos. A cada uno de estos dispositivos se le conoce también como un sistema de red y es así como se manejara este término durante el desarrollo de este trabajo.

Los sistemas de red han evolucionado, de tal forma que se puedan mejorar y se vayan ajustando a las nuevas tecnologías de comunicación. Por ello surge lo que es la cuarta generación de sistemas de red o bien los llamados **Procesadores de Red** (NP), sobre los cuales se pueden diseñar e implementar sistemas de red que procesen paquetes.

Hay varios sistemas de red utilizados en Internet, como ruteadores, switches, firewalls etc.. y otros muchos que son reservados para propósitos especiales que no son universalmente implementados, como el monitoreo, políticas, analizadores y reguladores de tráfico.

2.1. Sistemas de red

El término sistema de red se maneja en este trabajo como un dispositivo o componente electrónico capaz de manejar datos. Los de interés son aquellos que se encargan del procesamiento de paquetes, algunos ejemplos son: *bridge*, *switch, VLANSwitch*, VPN, NAT, modem DSL, ruteadores IP, *firewalls*, balanceadores de carga, *switches VoIP*, analizadores de flujo de paquetes, sistemas de monitoreo y control de tráfico etc..

2.2. Retos de los sistemas de red

El término Ingeniería de Red (*network engineering*) se refiere al diseño macroscópico de una red o intranet como el escoger el tipo de topología, y el término Ingeniería de sistemas de red se refiere al diseño de sistemas individuales como *switches* y ruteadores [1].

Un Ingeniero en sistemas de red debe considerar aspectos como nuevas maneras para emplear el hardware, ya que hasta ahora no se ha credo una nueva forma de transferir bits. En lo que se deben de concentrar, es, en como implementar el protocolo para asegurar que los sistemas operen correctamente con otros sistemas, un diseñador debe seguir los estándares publicados que especifican todos los detalles correspondientes a la comunicación digital.

Los Ingenieros y diseñadores de sistemas de red, tratan de perfeccionar los sistemas existentes, tal que los retos a vencer son:

a) Los sistemas operen a la velocidad del medio, sin que exista perdida de paquetes
b) El costo de los dispositivos sea bajo
c) Sean fáciles de operar y mantener

Las metas de los sistemas que procesan los paquetes, son generalidad, eficiencia, y escalabilidad. Esto es que pueda manejar cualquier tipo de aplicación, con velocidades del medio y sea capaz de ajustarse a las nuevas aplicaciones.

2.3. Procesamiento vs velocidad de una red

El término procesamiento de red (*Network Processing*) se utiliza cuando se refiere a procesamiento hecho a paquetes entrantes a una red de acuerdo a un conjunto de reglas y a la transmisión de los mismos a una línea de transmisión. Para implementar procesamiento de red existen varias alternativas de hardware entre las que se encuentran, los circuitos integrados para aplicaciones especificas ASIC, los FPGA (*Field Programmable Gate Array*), los procesadores de propósito general (PPG), etc.. [10]

Se requieren anchos de banda adecuados, para que una red garantice una alta calidad de servicio de transmisión, y de una u otra forma se puede tener el ancho de banda suficiente para la transmisión. Sin embargo los cuellos de botella de la velocidad de una red también tienen que ver con el procesamiento de red.

Por ejemplo, tradicionalmente el procesamiento de tareas TCP/IP fue primeramente desarrollado en software que se ejecutaba en un Procesador de Propósito General (PPG). Como las velocidades de algunas interfaces de red llegaron a alcanzar velocidades superiores a las del procesador, esta podría no mantener el ritmo con las velocidades del medio y entonces se tendría un cuello de botella.

Se sabe que las velocidades de los PPG también han tenido un crecimiento extraordinario, pero la rapidez con que las velocidades de transmisión de red han crecido es mayor al de los procesadores.

En la figura 2.1 se muestra como se ha desarrollado y como podría desarrollarse el crecimiento de ambas tecnologías.

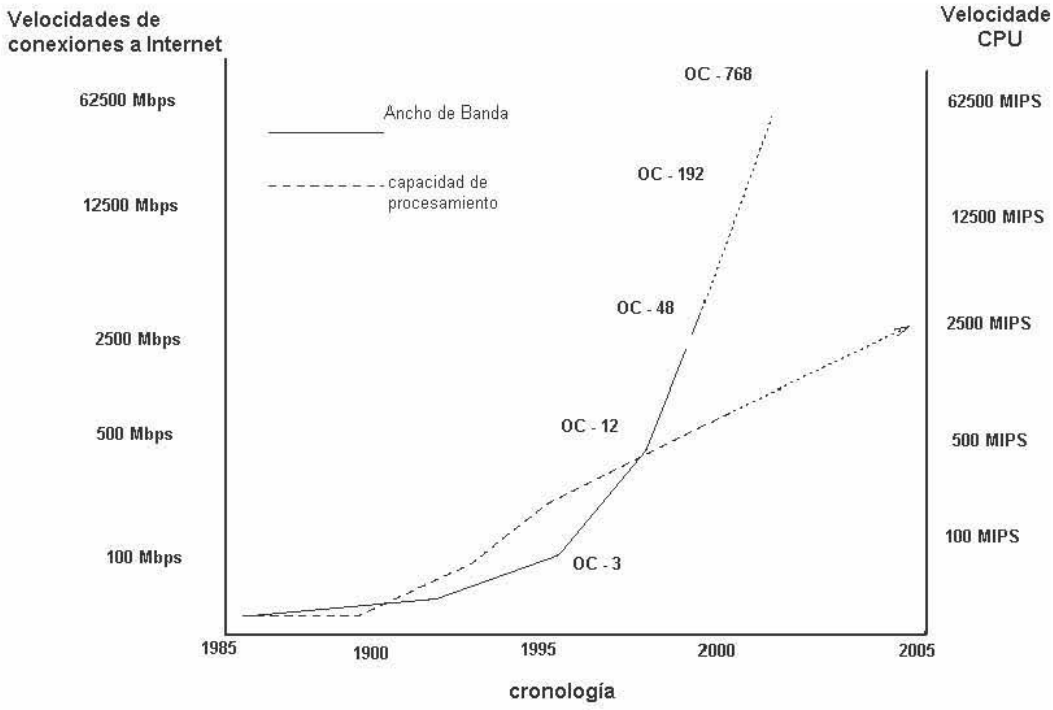

Figura 2.1 Crecimiento de BW y velocidades de procesamiento

Para este problema se ha propuesto el uso de hardware de propósito especial que forma parte de los sistemas de red, que hagan el procesamiento de red de una manera mas rápida.

2.4. Velocidades de una red y de un sistema de red

Cuando empezó Internet, la velocidad de una Red de Área Local (LAN) aumento de 3 Mbps a 10Mbps. Así un sistema de red de dos entradas (como un *bridge*) necesitaba manejar datos en una tasa agregada de 20 Mbps. Un sistema con 16 entradas necesitaba manejar una tasa agregada de 160 Mbps. A mediados de los 90s, una LAN operaba a 100 Mbps, lo que significa que un sistema de red con 16 entradas manejaba datos a 1.6 Gbps. Se observa que con el tiempo las velocidades han cambiado (Tabla 2.1) y por lo tanto los sistemas de red deben ser capaces de manejar mayor cantidad de datos.

Año	Velocidad
1960	10 Kbps
1970	1Mbps
1980	10Mbps
1990	100 Mbps
2000	1000 Mbps (1 Gbps)
2004	2400 Mbps (2.4 Gbps)
X	?

Tabla 2.1 Cronología del incremento de velocidades de operación de una Red

Los sistemas de red que conectan LANs, manejan circuitos digitales como líneas T1 Y OC3, las velocidades de estas líneas también han cambiado. En la tabla 2.2 se listan algunos circuitos digitales estándares en Norte América.

Nombre	Bit Rate Mbps
T1	1.555
T3	44.736
OC-1	51.840
OC-3	155.520
OC-12	622.080
OC-24	1,244,160
OC-48	2,488.320
OC-192	9,953.280
OC-768	39,813.120 Mbps

Tabla 2.2 Ejemplos de circuitos digitales en Norte América

2.5. Funcionalidades de un sistema de red

Los diseñadores buscan la mejor implementación de los protocolos en sus dispositivos, o bien que se realice un procesamiento optimo de los paquetes dependiendo del protocolo a utilizar. El procesamiento de protocolos puede ser expresado en algoritmos que especifican los pasos para tratar un paquete, cada algoritmo requiere de apoyo del sistema tanto en software como de hardware.

Los sistemas de red implementan una variedad de algoritmos correspondientes a las capas 2, 3 y 4, por ejemplo para *Brinding*, *hash table lookup*, fragmentación IP y reensamblaje, IP *forwarding*, reconocimiento de conexiones TCP, y TCP *splicing*.

Además de los algoritmos y estructuras utilizadas en los diseños de sistemas de red, se consideran otras funcionalidades, las cuales se definen en categorías, cada categoría corresponde a un aspecto fundamental del procesamiento del protocolo, que puede presentarse en muchas situaciones. Algunas de las categorías son [1]:

- Búsqueda de direcciones y reenvío de paquetes
- Detección y corrección de errores
- Fragmentación, segmentación y reensamblaje
- Clasificación de paquetes
- Encolar y descarta paquetes
- Planificador de paquetes
- Seguridad
- Mediciones de tráfico y políticas
- Regulación de tráfico

Hay categorías que tienen una gran relevancia dentro del sistema, por ejemplo, muchos sistema incluyen mecanismos de detección de errores ya que es fundamental en muchos protocolos.

Otras categorías relacionan más a la arquitectura de hardware del sistema que a los protocolos. En particular, la clasificación es una de las ideas importantes en el diseño del hardware de gran velocidad porque si se utiliza hardware que trabaje en paralelo se puede realizar la clasificación rápidamente para muchas capas de protocolos.

El hardware que forma parte del los sistemas de red es importante para el procesamiento de protocolos; hay una variedad de arquitecturas, mecanismos y técnicas de hardware que se usan con los sistemas de red.

2.6. Historia de los sistemas de red

Los sistemas de red han ido evolucionando debido a que los diseñadores buscan poder acercarse a los retos y metas de los sistemas de red. Por las características en común que han presentado los sistemas con el paso del tiempo, se puede dividir el diseño en tres generaciones.

Primera (1990) CPUs rápidas
Segunda (1995)
Tercera (1999)

La primera generación de sistemas de red se basa en que el procesamiento de paquetes se hace mediante software, y el hardware utilizado es el de una computadora convencional, la cual consta de una sola CPU, una memoria, uno o mas dispositivos de entrada y salida, un mecanismo conocido como bus, el cual interconecta los demás componentes, permitiendo así la comunicación entre ellos.

En estos primeros sistemas el hardware es usado para recibir y transmitir paquetes y el software para llevar a cabo el procesamiento de paquetes. El CPU llamado procesador de propósito general maneja casi todas las tareas, que tienen que ver con el procesamiento, mientras que las NICs sólo manejan tareas básicas de la capa 1 y 2 (figura 2.2). Estos sistemas funcionan bien en redes de baja velocidad.

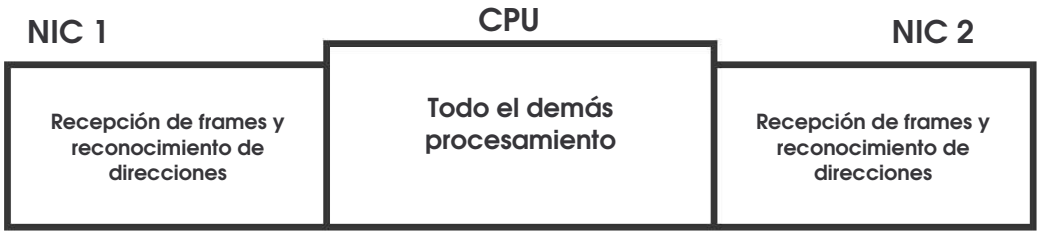

Figura 2.2 Arquitectura de hardware usada en sistemas de red basadas en software. La CPU maneja casi todas las tareas para el procesamiento de protocolos.

Cuando las proporciones de datos de red aumentaron, los ingenieros descubrieron que el uso del software en el procesamiento de paquetes de datos restringía el flujo de datos en los dispositivos de red de alta velocidad, así surgieron los sistemas de segunda generación, que tiene como propósito escalar a velocidades mas altas.

Aquí se extiende la arquitectura de la primera generación, agregando NICs inteligentes, tal que ayuden a las tareas de procesamiento de paquetes, y se reduzca la carga al CPU. Figura 2.3.

Se introduce el concepto de –*fast data path*- que consiste en mover paquetes de una interfaz a otra, sin pasar por el CPU.

Los diseños de segunda generación tienen ventajas y desventajas. La ventaja principal de estas arquitecturas es que ya se maneja más conexiones de red, a diferencia de la primera. Al descargar el procesamiento del CPU, los sistemas de segunda generación permiten una expansión: se pueden agregar mas interfaces de red y el *switching fabric*[1] se puede ajustar para las conexiones adicionales.

La desventaja de la segunda generación se debe a la dependencia del procesador de propósito general para realizar algunas tareas. Aunque el CPU sólo se usa para manejar un pequeño porcentaje de paquetes, el número de paquetes que pasa por el CPU incrementa linealmente la tasa de paquetes agregados (paquetes/segundo). Si el sistema tiene una alta tasa de paquetes agregados, el CPU puede llegar a ser un cuello de botella del sistema.

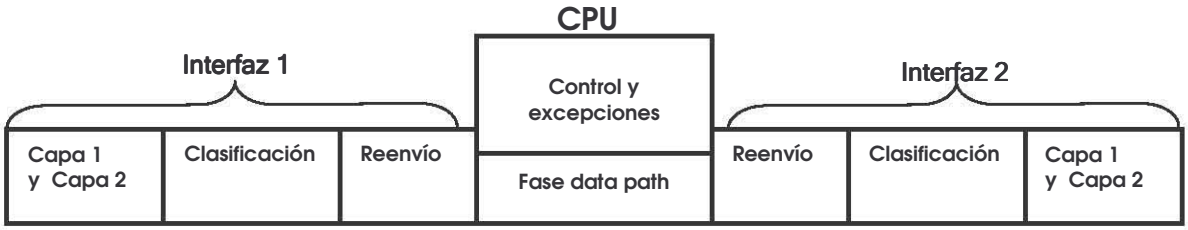

Figura 2.3 Organización conceptual de un sistema de red de segunda generación, con interfaces mas poderosas. La interfaz de entrada reenvía muchos paquetes por una ruta directa de datos (fase data path) a otra interfaz que sea la apropiada; la CPU se usa para manejar excepciones y errores.

Para reducir la carga del CPU, se diseñan otros sistemas de red de tercera generación. Estos sistemas usan hardware especializado para descentralizar el procesamiento de protocolos, el diseñador escoge el hardware apropiado que realice las tareas para cada protocolo, y repite este hardware en cada interfaz de red. Con esto se remueve completamente del CPU el procesamiento de protocolos y permite a las interfaces de red manejar excepciones y protocolos de las capas superiores. En la figura 2.4 se ilustra esta arquitectura.

[1] Mecanismo de hardware usado para interconectar múltiples unidades funcionales y permitir que intercambien datos. Puede ser usado como una interconexión interna entre los puertos de entrada y salida en un sistema de red.

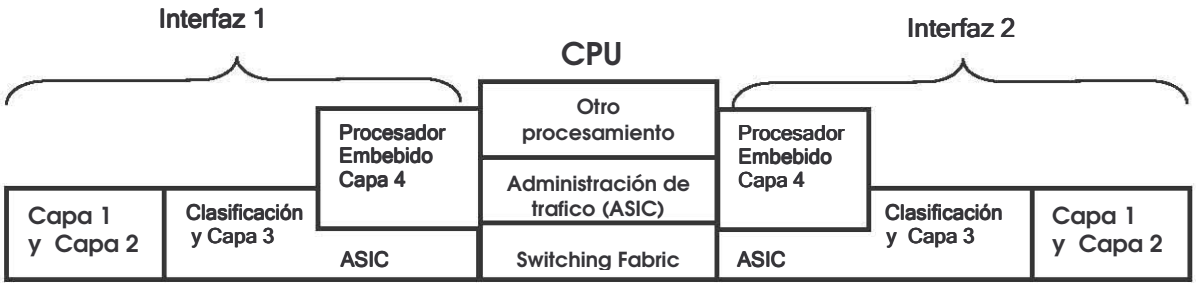

Figura 2.4 Arquitectura de tercera generación de los sistemas de red con un procesador embebido mas hardware ASIC en cada interfase. Un switching Fabric provee una ruta directa entre las interfaces . Una interfaz puede manejar tareas de capa 1 hasta la 4, dejando al CPU libre para ejecutar aplicaciones.

2.7. Procesadores de Red

Anteriormente se menciono que los sistemas de red se pueden dividir en tres generaciones, y como los diseñadores de sistemas de red buscan encontrar la manera de perfeccionar o mejorar el diseño y manufactura de los sistemas. Surge entonces, lo que se conoce como cuarta generación.

La cuarta generación busca la flexibilidad que proporciona la primera generación de programar ciertas funciones de red, y que se puedan ajustar a las nuevas tecnologías y la rapidez de la tercera generación, es decir la introducción de hardware de propósito especifico que normalmente no necesite modificación ya que las funciones que realice sean comunes que todos los sistemas de red realizan.

Entonces un Procesador de Red (NP), es un dispositivo de hardware de propósito especial programable, que combina el bajo costo y flexibilidad de un procesador RISC con la velocidad y escalabilidad de un dispositivo de hardware de propósito específico (*custom silicon)* como lo es un chip ASIC. Los NP están constituidos de distintos bloques y son usados para diseñar sistemas de red [1].

2.7.1. Arquitectura de los NPs

Existe una variedad de arquitecturas de NP, y se diferencian de acuerdo a las siguientes características.

- Jerarquía del procesador
- Jerarquía de la memoria
- Mecanismos de transferencia Internos
- Interfaz externa y mecanismo de comunicación
- Hardware de propósito especial
- Mecanismos de Poleo y notificación
- Soporte de ejecución concurrente
- Paradigma y Modelo de programación
- Mecanismos de Despacho de Software y Hardware
- Paralelismo Implícito o explicito

A continuación se describe brevemente a que se refiere cada una de las características

2.7.1.1. Jerarquía de procesador

El término jerarquía de procesador se refiere a la unidad de hardware que desarrolla varios aspectos del procesamiento de paquetes.

La jerarquía de un Procesador de Red se extiende de los niveles más bajos a los mas altos, abarca hardware de propósito especial dedicado a una sola tarea especifica, así como también CPUs de propósito general. En la tabla se muestra un ejemplo de 8 jerarquías de procesadores.

Nivel	Tipo de Procesador	¿Es Programable?	¿Dentro de chip NP?
8	CPU de propósito General	Si	posiblemente
7	Procesador Embebido	Si	típicamente
6	Procesador de E/S	Si	típicamente
5	Coprocesador	No	típicamente
4	*Fabric interface*	No	típicamente
3	Unidad de transferencia de datos	No	típicamente
2	*Framer*	No	posiblemente
1	Transmisor Físico	No	posiblemente

Tabla 2.3 Jerarquía de procesadores en un NP

No todos los procesadores pueden residir dentro del chip NP. El chip NP contiene solo el hardware suficiente para manejar algunos niveles, y se apoya en hardware externo para manejar otros niveles. La jerarquía de procesador en un NP es interesante ya que un solo NP incluye varios procesadores físicos que trabajan en conjunto.

Un NP puede contener: uno o varios procesadores embebidos RISC usados para manejar las capas superiores de protocolos y proveer un control global del sistema, uno o más coprocesadores especializados que realizan tareas especificas de procesamiento, múltiples procesadores de E/S que llevan a cabo el procesamiento de ingreso y egreso a la velocidad del medio (*wire speed*), una o más interfaces que manejan la interacción con el *switching fabric* y una o más unidades de transferencia de datos que tratan el movimiento de paquetes entre los dispositivos de E/S y la memoria.

El flujo de paquetes que pasa a través de una jerarquía de procesador en un NP, es más óptimo cuando los niveles bajos manejan paquetes localmente sin pasarlos a procesadores que se encuentran en un nivel más alto. Entonces es importante tomar en cuanta la jerarquía dentro de NP cuando se busca un desempeño más óptimo.

2.7.1.2. Jerarquía de memoria

La jerarquía de Memoria es utilizada por diseñadores para conseguir un alto desempeño con un bajo costo. Las diferentes tecnologías de memoria tienen diferentes características en su desempeño, tamaño y costo.

Una jerarquía de memoria se organiza de acuerdo a pequeñas cantidades de memoria que operan a altas velocidades (memoria con baja latencia y alto *throughput*), y una mayor cantidad de memoria que opera a velocidades más lentas y así sucesivamente, permitiendo a un NP alcanzar un alto desempeño a un costo razonable (*Tabla 2.4*). La mayoría de los NPs incluyen hardware que soporta una jerarquía de memoria multinivel, el chip tienen solo pequeñas cantidades de memoria y contiene interfaces de hardware que proveen acceso a memoria externa al chip y el programador del sistema de red decidirá que memoria usará y para que la empleara, puede colocar los datos que son utilizados con mas frecuencia en la memoria rápida.

Desde luego, la memoria asociada con un NP no es *lenta* comparada con la memoria de un sistema de una computadora convencional.

Tipo de memoria	Velocidad Real	Tamaño aproximado	¿Dentro del chip NP?
Almacenamiento de Control	100	10^3	Si
Registros de propósito General	90	10^2	Si
Cache	40	10^3	Si
RAM	10	10^3	Si
SRAM	5	10^7	No
DRAM	1	10^8	No

Tabla 2.4 Jerarquía de memoria en un NP

2.7.1.3. Mecanismos de transferencia internos

Este termino se refiere a cualquier mecanismo en un chip NP que provee una ruta de datos entre las diferentes unidades funcionales que lo conforman. Los NPs tienen varios componentes que necesitan "comunicarse" unos con otros por ejemplo procesadores con coprocesador, procesadores con los paquetes en memoria o procesadores con el procesador de control
La comunicación interna es importante, ya que si una ruta directa de datos es lenta puede haber una baja en el desempeño de todo el NP. Los mecanismos de transferencia que se usan se usan:

- Bus Interno
- Hardware FIFO
- Registros de transferencia
- Memoria compartida

2.7.1.4. Interfaces externas y mecanismo de comunicación

Un chip NP debe conectarse a otras unidades de hardware del sistema red externo, las conexiones principales incluyen.

Interfaces de Bus estándar y especializadas: Un NP se necesita conectar a uno o mas buses externos. El bus puede ser estándar en la industria de las computadoras como el PCI, o puede ser diseñado para el NP como un bus LA-2 que es especifico para el NP Forum.

En cualquier caso, muchos NP contienen un bus de interfaz de hardware que maneja los detalles de las conexiones eléctricas y provee un bus de acceso para los procesadores que están en el chip.

Interfaces de Memoria: Un NP necesita acceder a las memorias externas. En muchos casos, la memoria se conecta por medio de un bus. Sin embargo, un NP puede contener hardware adicional para optimizar la interacción con la memoria. (p.e. memoria cache). Las interfaces de memoria también permiten al programador controlar situaciones donde el orden de las operaciones de lectura y escritura es importante.

Interfaz directa de E/S: Los NPs se pueden conectar a dispositivos de E/S de alta velocidad por medio de un bus. Por ejemplo, la conexión entre un NP y un puerto Ethernet puede hacerse mediante un bus. Sin embargo también es posible para un NP tener una conexión directa a un dispositivo o tener una conexión a hardware intermedio el cual se conecta a un dispositivo específico. En cualquier caso, la interfaz de hardware en el chip NP permite a los procesadores que lo conforman acceder a dispositivos externos de E/S.

Interfaz Shitching Fabric: Para conectarse a un Switching Fabric especifico de cada NP, hay hardware especial que maneja estas interfaces.

2.7.1.5. Hardware de propósito especial

Además de coprocesadores, que son parte de la jerarquía de procesadores de un NP, un NP contienen al menos otros dos tipos de hardware de propósito especial.

Primero, un NP puede contener una o más unidades que proveen control, sincronización o coordinación entre los procesadores internos que compiten por el acceso a recursos compartidos. Segundo, un NP puede tener una o más unidades de hardware configurables que son usadas para manejar o controlar dispositivos de E/S u otras unidades de hardware.

2.7.1.6. Mecanismos de poleo y notificación

Un NP debe manejar eventos asíncronos tal como la llegada de un paquete en un puerto Ethernet, la expiración del *timer* de un protocolo, o la transferencia completa a través de un *Switching Fabric*. Existen dos mecanismos de hardware para manejar la asincronía. Poleo y notificación: Un NP puede emplear uno de los mecanismos o los dos.

El mecanismo de poleo requiere un elemento activo (como un procesador) para que repetidamente se verifique un hardware asociado con ciertos eventos. Por ejemplo, una interfase de E/S puede ser configurada activando un bit si el paquete llego. Un procesador limpia el bit, inicia el dispositivo de E/S y entonces repetidamente verifica este bit. Cuando encuentra el bit activo, recupera y procesa un paquete.

La notificación puede ser implementada con una interrupción basada en hardware o software. Un procesador especifica que notificación es necesaria, inicia una operación y entonces procede a otra tarea. Por ejemplo se puede generar una interrupción cuando llega un paquete.

2.7.1.7. Soporte de ejecución concurrente

Ejecución concurrente se refiere al bien conocido paradigma de programación usado para maximizar el desempeño. Para optimizar el *throughput*, un sistema concurrente permite a un conjunto de procesos o hilos independientes ejecutarse al mismo tiempo.

2.7.1.8. Paradigma y modelo de programación

Cada NP contiene soporte de hardware para un paradigma de programación. Las dos formas más comunes son:

Manejadores de eventos asíncronos: Un estilo de programación donde el programador crea un conjunto de manejadores y los asocia con un evento en especifico. Cuando ocurre el evento el sistema invoca al correspondiente manejador.

Hilos de comunicación: Es un modelo de programación convencional secuencial, donde uno o más hilos se ejecutan independientemente. Si existe más de un hilo, los hilos usan un mecanismo de comunicación entre procesos para pasarse datos.

2.7.1.9. Mecanismo de despacho en software y hardware

El término *despacho* se refiere al control global de las tareas concurrentes y paralelas cuando hay algún trabajo disponible, un despachador asigna el trabajo a un especifico proceso o hilo. Algunos NP proveen soporte de hardware para el despacho; otros permiten al software controlar el despachamiento. El despachamiento por software es usado en procesadores que tienen un sistema operativo, y el despachamiento por hardware es usado por procesadores de bajo nivel de E/S que no tienen sistema operativo.

2.7.1.10. Paralelismo implícito o explicito

Hay dos estilos de paralelismo usados por un NP, la principal diferencia consiste en como es estructurado el software. Una arquitectura de hardware que usa paralelismo explicito deja el paralelismo al programador, y requiere que el software sea construido para explotar el paralelismo de hardware, esto es, el programador escoge como y cuando usar las unidades funcionales repetidas, y escribir código que coordine las copias de hardware.

Una arquitectura de hardware que usa paralelismo implícito oculta el paralelismo de hardware formado al tener varias copias de unidades funcionales. El programador crea software para ejecutarse en una sola unidad funcional y el hardware maneja el paralelismo automáticamente repitiendo el programa como sea necesario en las unidades funcionales.

2.7.2. Categorías de arquitecturas

Una de las diferencias más importantes en la arquitectura de un NP se centra en la manera que fluye el tráfico a través del hardware. Se pueden considerar tres formas principalmente

Una arquitectura de hardware donde todo el tráfico entrante fluye por el procesador embebido que lleva a cabo el conjunto de operaciones en un paquete. Figura 2.5.

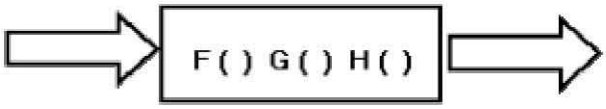

Figura 2.5 El procesador realiza las operaciones sobre los paquetes de una manera secuencial

La segunda forma a considerar es cuando los paquetes entrantes fluyen sobre una arquitectura paralela, en la cual el tráfico es dividido y repartido en un conjunto de procesadores que desarrollan las mismas funciones sobre los paquetes. Figura 2.6

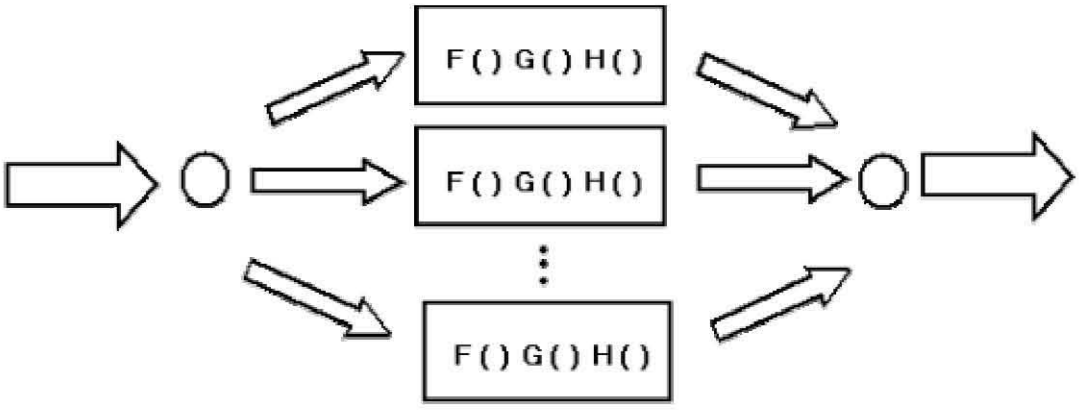

Figura 2.6 Varios procesadores pueden atender a distintos paquetes realizando las mismas funciones sobre los mismos

La última es una arquitectura llamada *pipeline* o entubamiento, donde cada procesador desarrolla una función diferente sobre un flujo de paquetes.

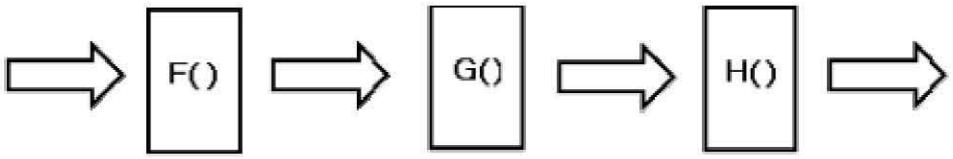

Figura 2.7 Varios procesadores realizan funciones diferentes sobre un flujo de paquetes

La principal diferencia entre estas arquitecturas es el *clock rate* necesario para procesar los datos

2.7.3. Arquitectura de software

La mayoría de las arquitecturas de software para un NP siguen los siguientes patrones

- Un programa central que invoca a coprocesadores como subrutinas
- Un programa central que interactúa con código inteligente de los procesadores de E/S
- Comunicación con Hilos
- Manejador de eventos
- Estilos RPC (Programa repartido entre procesadores)
- Pipelined (Si es que la arquitectura de hardware lo permite)
- Combinación de todos los anteriores

2.7.4. Composición general de un NP

Un procesador de red generalmente consta de un procesador o núcleo maestro (procesador embebido), varias maquinas (procesadores de E/S) que procesan los paquetes, interfaces de memoria y un bus con interfaz a un dispositivo de capa física. Figura 2.8.

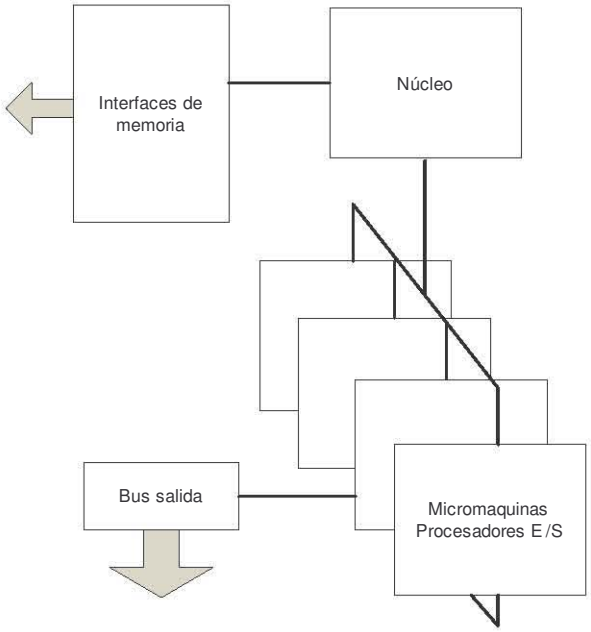

Figura 2.8 Composición general de un NP

El procesador o núcleo central coordina toda la unidad de procesador de red, administra el destino de los paquetes por medio de actualización de tablas o manejo de colas.

El procesador de red tiene la ventaja de ser manejado por herramientas y software externos para que el procesador central lleve a cabo sus objetivos.

Las micromáquinas se dedican al procesamiento de los datos, y sacrifican la facilidad de programación por un procesado rápido y eficiente de los paquetes. Dichas micromáquinas manejan un pequeño conjunto de instrucciones digamos un mini-RISC con una arquitectura que optimiza el procesamiento, como la copia de los datos, o conversión de protocolos.

Dependiendo del procesador de red, las maquinas pueden ser conectadas de varias maneras, ya sea organizadas para realizar operaciones independientemente (paralelo) o en forma de arreglos (*pipelined*).

Las interfaces de memoria se usan para conectar las memorias externas como la SDRAM con todo el procesador de red. Tanto el núcleo como las micromáquinas tienen acceso a esa memoria que puede usarse de manera compartida, y usan estas interfaces para compartirse información sobre el control y procesamiento de los paquetes.

2.7.5. Arquitecturas comerciales

Hay una variedad arquitecturas de procesados de red entre las que se encuentran

- Multi-Chip Pipeline (Agere)
- Augmented RISC Processor (Alchemy)
- Embedded Processor Plus Coprocessors (Applied Micro Circuit Corporation)
- Pipeline of Homogeneous Processors (Cisco)
- Pipeline of Heterogeneous Processors (EZchip)
- Configurable Instruction Set Processors (Cognigine)
- Extensive And Diverse Processors (IBM)
- Flexible RISC Plus Coprocessors (Motorola)
- Internet Exchange Processor (Intel) etc..

2.7.6. Funciones de los NPs

Los procesadores de red pueden llevar a cabo tareas, como por ejemplo procesamiento de paquetes, flujos de datos, o algunas otras tareas especificas para las cuales son programados.

Las funciones que realiza la unidad de procesador de red (NPU por sus siglas en ingles) se clasifican en funciones de [1]:

- Capa física
- De conmutación,
- De procesamiento de paquetes,
- Y de control del sistema.

2.7.6.1. Funciones de capa física

Las funciones de capa física se encargan de manejar la señalización en las conexiones de la red como en un puerto Ethernet 100 base T, conexión por fibra óptica o en una conexión de cable coaxial T3.

Las funciones de capa física se encargan de convertir los paquetes de datos en señales digitales que se transmiten en un medio físico. Trabajan en conjunto con la capa física y la MAC.

2.7.6.2. Funciones de conmutación

Las funciones de conmutación son responsables de dirigir el tráfico dentro del dispositivo. En general se encargan de conducir los datos desde el puerto de entrada hasta la red adecuada a través del puerto de salida apropiado. Otra de las funciones es ordenar los datos en los puertos de acuerdo a la prioridad que tengan (*queuing*).

2.7.6.3. Funciones de procesamiento de paquetes

Las funciones de procesamiento de paquetes se encargan de manejar el procesamiento de los protocolos de red.

En algunos casos las funciones de procesamiento de paquetes se pueden subdividir en procesamiento de paquetes de la capa de red y procesamiento de las capas superiores.

2.7.6.4. Funciones de control del sistema

Una unidad de procesador de red necesita de las funciones del control del sistema para manejar la administración de todos los componentes de hardware que lo conforman, tal como la administración de energía, dispositivos de control periféricos, puertos, etc..

Estas funciones también son necesarias para llevar a cabo la comunicación de los periféricos de los procesadores con la CPU, otras NPs u otros dispositivos periféricos.

CAPÍTULO 3

El capítulo esta enfocado al procesador de red IXP1200. Se dan las características más importantes en cuanto a su arquitectura de hardware y elementos de su programación.

3. Procesador de Red Intel IXP1200

El IXP1200 es uno de los primeros procesadores de red, su diseño fue inicialmente hecho por DEC, el cual adquirió INTEL. Es un Procesador de Red integrado por varios componentes de hardware, es parte de la familia *Internet Exchange Processor* (Intel) y forma parte de la arquitectura IXP.

El objetivo del IXP1200 como el de todos los NPs, es la creación de aplicaciones de red, esto es sistemas de red a nivel de capa 2,3 y 4, que requieren de un alto grado de flexibilidad, fácil programación, escalabilidad, desempeño y un bajo consumo de energía.

Su arquitectura de hardware permite llevar a cabo el procesamiento de paquetes de una manera paralela-concurrente, y proporciona características que simplifican el modelo de programación. Tiene una arquitectura de almacenamiento de datos distribuida que permite la comunicación entre procesadores y unidades funcionales además de compartir datos.

El NP IXP1200 permite a los diseñadores de sistemas de red programar funcionalidades de red que normalmente se implementan en bloques de hardware de propósito específico como los ASICs.

3.1. Arquitectura de hardware

El Procesador de Red IXP1200 internamente, tiene un procesador RISC embebido (*StrongARM*); seis procesadores programables llamados micromáquinas (también conocidos como procesadores de E/S); buses internos que interconectan los componentes; interfaz serial de baja velocidad para comunicación con el *StrongARM*; interfaz de acceso a la memoria externa SDRAM a la cual puede acceder el *StrongARM*, las micromáquinas y dispositivos conectados al bus PCI, interfaz de acceso a la memoria SRAM la cual es utilizada también por el *StrongARM* y las micromáquinas; interfaces de acceso a los buses externos de E/S PCI e IX, el primero puede ser usado para tener

acceso a otros dispositivos PCI u otro procesador como el de una PC, el segundo al que se hace referencia como unidad FBI (FIFO Bus Interface) es controlado por las micromáquinas y provee acceso al bus de datos externo de alta velocidad IX donde llegan y salen los paquetes del dispositivo, en el FBI se encuentra una unidad pequeña de memoria llamada *scratchpad*, un coprocesador para el cálculo de la función *hash*, registros de control y estado (CSRs), un pequeño procesador programable llamado *ready-bus secuence* y dos colas llamadas RFIFO – TFIFO donde se almacena temporalmente un paquete que entrará o saldrá del dispositivo. Más detalles sobre la arquitectura hardware se encuentran en *Intel "IXP1200 Network Processor Family Hardware Reference Manual"* .

En la figura 3.1 se muestra la arquitectura de hardware básica, con los componentes internos (dentro del chip NP) y externos.

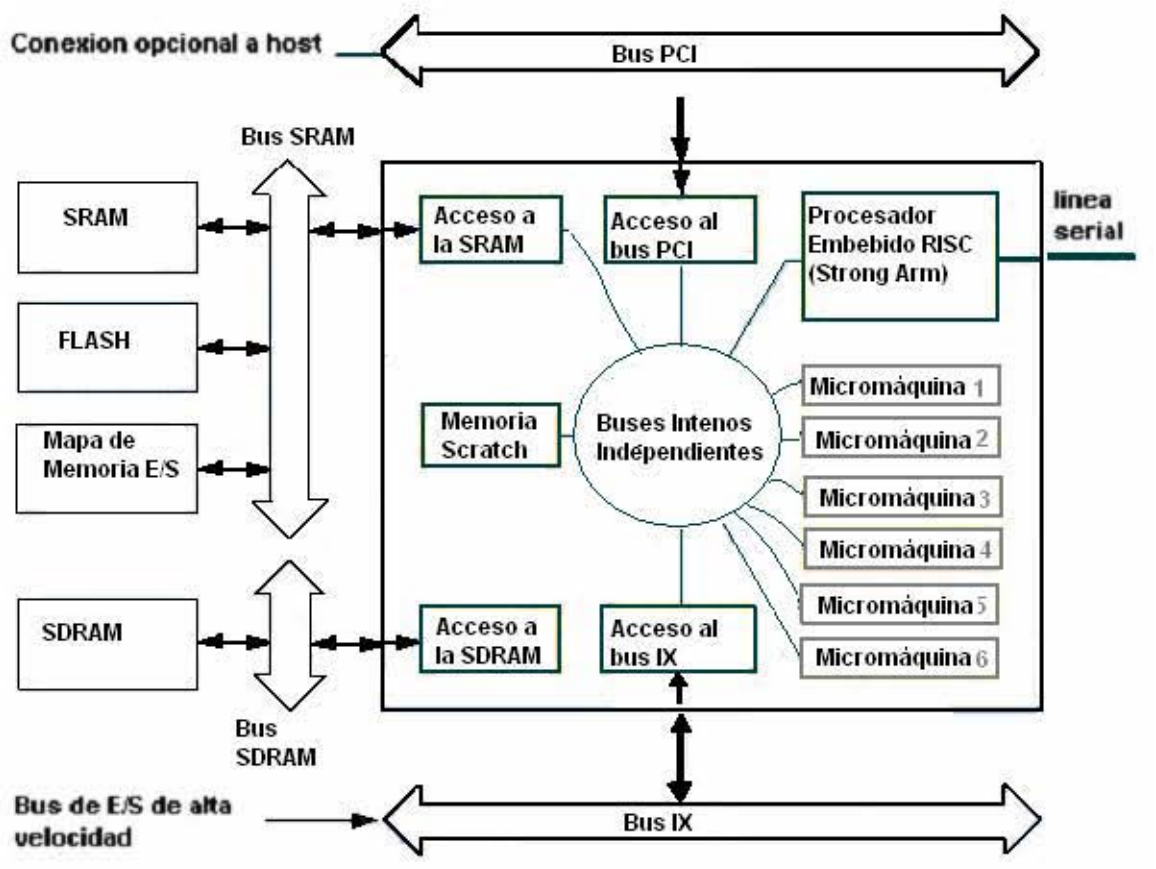

Figura 3.1 Arquitectura de Hardware del IXP1200

En el capítulo anterior se dio un vistazo general a los procesadores de red y se menciono que los NPs al manejar varios tipos de procesadores y memorias, éstos manejan cierta jerarquía. En éste NP encontramos su jerarquía de procesador y de memoria como se muestran en las tablas 3.1 y 3.2.

Tipo de Procesador	**¿En la tarjeta?**	**Programable**
PPG	No	Si
Procesador Embebido RISC	Si	Si
Procesadores de E/S	Si	Si
Coprocesadores	Si	No
Interfaces Físicas	No	No

Tabla 3.1 Jerarquía de procesadores en el IXP1200

Tipo de memoria	**Tamaño máximo**	**¿Dentro del Chip NP**	**Uso Típico**
Registros de Propósito General	128 registros	Si	Computo inmediato
Cache de Instrucciones	16 Kbytes	Si	Instrucciones usadas recientemente
Cache de Datos	8 Kbytes	Si	Datos recientemente usados
Mini - Cache	512 Bytes	Si	Datos que son re-usados una vez
Buffers de escritura	No especificada	Si	Escribir operaciones en un buffer
Memoria Scratchpad	4 Kbytes	Si	Sincronización
Almacenamiento de instrucciones	64 Kbytes	Si	Instrucciones de la micromáquina
Flash ROM	8 Mbytes	No	Bootstrap
SRAM	8 Mbytes	No	Tablas o cabecera de paquetes
DRAM	256 Mbytes	No	Almacenamiento de paquetes

Tabla 3.2 Jerarquía de memoria

Las memorias de interés para la programación son la *Scratch* para almacenar datos de acceso global y pasar mensajes entre procesadores e hilos, la memoria SRAM que normalmente se utiliza para almacenar cabeceras de paquetes, tablas de ruteo y la memoria SDRAM para el almacenamiento de los paquetes. Aunque cabe señalar que el desarrollador puede administrar la memoria de acuerdo a sus necesidades.

3.1.1. Procesadores programables

El IXP1200 consta de siete maquinas programables, Procesador Embebido RISC (*StrongARM*) y seis micromáquinas o procesadores de E/S, todos en el chip NP.

StrongARM es un procesador de propósito general ARM a 32 bits con endian configurable, tiene un puerto serial incorporado, soporta memoria virtual, un cache de instrucciones de 16 KB, cache de datos de 8KB, un conjunto de instrucciones reducido (RISC), aritmética de 32 bits, vectores de punto flotante provisto por el coprocesador, memoria direccionable por byte, y soporte de un sistema operativo kernelizado.

Micromáquinas (uE): Son procesadores RISC multi-hilo, las cuales llevan a cabo el movimiento y procesamiento de los datos sin asistencia del núcleo o el Strong Arm. Estos procesadores se utilizan para implementar aplicaciones de red (networking) ya que permiten una manipulación rápida de los datos, específicamente con ellas se puede examinar el contenido de los paquetes a varios niveles de la pila de protocolos de red tales como reenvío y conmutación de paquetes, pero también se pueden realizar aplicaciones que requieran inspecciones y manipulaciones más profundas del contenido de los paquetes.

Dentro de las funciones que se pueden realizar están verificación de checksum, procesamiento y clasificación de los headers, almacenamiento de los paquetes en la memoria, verificación en tablas, reenvío IP, modificación de cabeceras, cálculo del Checksum, salida de los paquetes hacia la capa física del hardware, etc..

Las uEs se programan por medio de microcódigo, el cual fue desarrollado con el Kit Intel IXA Software Development (SDK). Cada micromáquina tiene cuatro hilos de hardware independientes y a su vez cada hilo tiene su *Program Counters* (PC). Esto hace un total de 24 contextos (o hilos) de microcódigo en el NP. Las micromáquinas manejan operaciones con bits, bytes, word (palabra), y longwords.

Cada micromáquina contiene su propio almacenador de instrucciones (Program Control Store), junto con 256 registros y otros registros de control y estado (CSRs). Como todas las micromáquinas son idénticas las funciones pueden ser intercambiadas entre ellas, es decir no hay ninguna función fija para cualquiera de las micromáquinas porque son totalmente programables.

Las micromáquinas funcionan como un procesador pipelined[1] que pasa por cinco estados

1- La instrucción es obtenida de su lugar de almacenamiento
2- La instrucción es decodificada y se obtiene la dirección o direcciones de los registros
3- Se obtienen los operandos de los registros
4- Los operandos son pasados a la unidad lógico aritmética (ALU)
5- El resultado se escribe en el registro destino.

Hay dos tipos de registros dentro de las uE, los registros de propósito general (GPRs) y los registros de transferencia. Son 128 registros de 32 bits de propósito general y pueden ser accedidos de dos formas, en la primera cada hilo puede acceder a 32 registros o la segunda en forma global, donde los GPRs se dividen a su vez en dos bancos (subregistros), un banco A y un banco B y todos pueden acceder a esos registros b. El tener los dos bancos o subregistros permite que la ALU tome dos operandos al mismo tiempo.

Existen Interfaces a unidades funcionales en el IXP1200 que son interfaz SDRAM y SRAM, bus de interfaz FIFO (utilizada por una unidad llamada FBI) y la interfaz PCI.

[1] *Procesador que contiene funciones especificas acomodadas en forma de tubería, por las que pueden pasar todos los datos que entran al procesador.*

3.2. Arquitectura de Software

La arquitectura de software en el IXP1200 consta de un Kit de desarrollo (SDK) que provee de las herramientas necesarias para el diseño de sistemas de red, en la Tabla 3.3 se muestran todo el KIT.

Software	**Propósito**
Compilador C	Compilar programas en C para el StrongARM
Compilador NCL	Compilar Programas en NCL para el StrongArm
Compilador Micro C	Compilar programas de micro C para las micromáquinas
Ensamblador	Ensamblar programas para las micromáquinas
Simulador	Simular aplicaciones
Cargador	Carga software en el IXP1200
Monitor	Se comunica con el NP e interactúa con el software en ejecución
Bootstrap	Inicia el NP
Código de Referencia	Ejemplos

Tabla 3.3 Arquitectura de Software del IXP1200

Para trabajar con el Kit se utiliza un ambiente de desarrollo híbrido, ya que los compiladores para el StrongARM se utilizan bajo un sistema operativo Linux o VxWorks y; los compiladores y ensamblador para las uE trabajan bajo plataforma Windows NT/200x/XP profesional.

Además del software disponible, se cuenta con un paradigma de programación que consiste utilizar un marco de trabajo llamado *Elemento de Computo Activo* (ACE – Active Computing Element). Un ACE se puede definir como la unidad básica de programación definida por el SDK de Intel para sus Procesadores de Red, el cual incluye código que se ejecuta en las micromáquinas o el StrongARM.

La arquitectura esta compuesta de dos tipos de ACES:

- *ACE Núcleo o ACE Convencional:* Estos ACEs se ejecutan en el núcleo o Strong ARM del IXP1200. Se interconectan en forma de pipe (tubería) para mover paquetes entre los ACEs y desarrollar dos funciones.

  Clasificación: Durante esta fase el ACE aplica una secuencia de reglas a los paquetes y los coloca en colas asociadas con una cierta acción.

  Acción: Durante esta fase, se ejecuta la acción de cada cola, la acción involucra algún procesamiento sobre los paquetes y después son reenviados a otros ACEs.

  Los Core ACEs son desarrollados en C o C++ y un lenguaje especial llamado NCL (Network Classication Language).

- *MicroACE o ACE Acelerado:* Hay dos tipos de MicroACE, el primero llamado componente del núcleo o de rutas lentas, los cuales son ACEs que se ejecutan en el StrongArm; y el segundo componentes de microbloque o de ruta rápida que se ejecutan en las micromáquinas.

  Los componentes de microbloque contienen las funciones básicas de procesamiento que se le quieran hacer al paquete mientras que los paquetes con excepciones son entregadas a componentes del núcleo y tienen una ruta mas larga y lenta, sí es que se pretenden manejar excepciones.

La arquitectura IXP1200 utiliza un mecanismo de excepciones “IX_EXCEPTION” para el manejo de excepciones enviadas al núcleo y en el núcleo se determina como serán procesados los paquetes subsecuentemente. Este mecanismo puede o no utilizarse

Para estructurar una aplicación se consideran conceptos importantes que relacionan a los ACEs. Estos son:

Microbloque: Como se menciono antes, es un componente que se ejecuta en una micromáquina, un ejemplo es un microbloque que realiza el reenvió IP o Bridging . Un microbloque puede ser escrito en código ensamblador o en Micro C para las micromáquinas.

Grupo de Microbloques: Varios microbloques que se pueden combinar y descargarlos como una sola imagen en un sola micromáquina, es decir es un Conjunto de microbloques ejecutandose en una micromáquina.

Ciclo de Despacho (Dispatch Loop): Mecanismo general para controlar el flujo de paquetes entre microbloques y grupo de microbloques, debido a que no todos lo paquetes siguen la misma ruta. De hecho un paquete no puede ser procesado por todos los microbloques presentes en la micromáquina.

Manejador de Recursos (Resource Manager): Este modulo se ejecuta en el StrongARM y provee de una interfaz al componente de núcleo de un microACE para manejar su microbloque. Usando el manejador de recursos, el componente del núcleo y las micromáquinas pueden compartir memoria para las estructuras de datos, y pasar paquetes de regreso y hacia delante. Si se utiliza el cargador del WorkBench del SDK desde Windows, él realiza ésta función.

Componente del Núcleo: El componente de núcleo de un microACE de ejecuta en el StrongARM, y los demás ACEs de sistema lo ven como un ACE convencional.

Colas de paquetes: Como el movimiento de los paquetes no se puede predecir y los microbloques necesitan continuar el procesamiento sin esperar, entonces los paquetes se almacenan en “Colas de Comunicación” o FIFOs. Estas colas son unidireccionales y si se necesita comunicación bi-direccional se utilizan dos colas.

Los microACEs pueden ser combinados con ACEs núcleo para formar un procesamiento en forma de entubamiento (pipeline) para implementar aplicaciones de red. En la figura 3.2, se muestra un ejemplo de una aplicación de reenvío IP usando MicroACEs y ACEs convencionales.

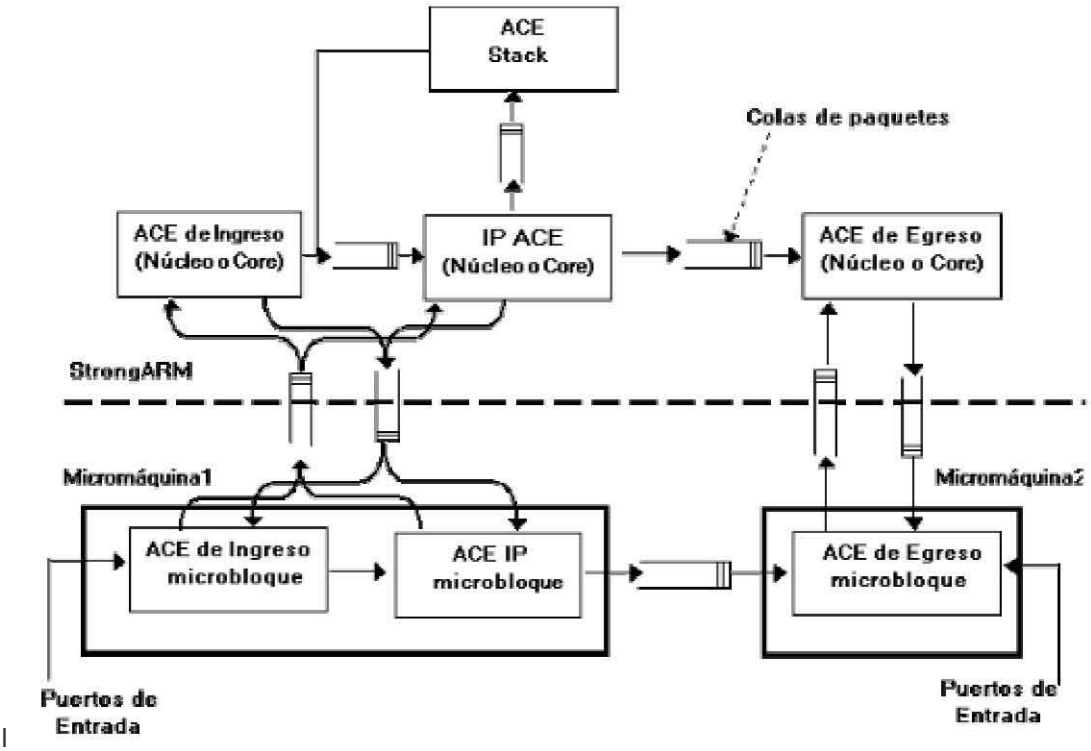

Figura 3.2 MicroACEs y ACEs convencionales

CAPÍTULO 4

En este capítulo se explica la propuesta de diseño del regulador de trafico "Traffic Shaper". Separando las funciones necesarias que se consideraron para su funcionamiento.

4. Diseño del sistema regulador de tráfico (Traffic Shaper) para un procesador de red

4.1. Regulador de Tráfico (Traffic Shaper)

Los *traffic shapers* se utilizan para prevenir la congestión y suavizar el tráfico, juega un papel importante en la realización de esquemas de control de tráfico en redes de alta velocidad para asegurar requerimientos de calidad de servicio.

El control de tráfico se define como un conjunto de acciones tomadas por la red para evitar una posible congestión [26]. También toma medidas para adaptar fluctuaciones de tráfico y otros problemas dentro de la red [26]. El control de congestión se refiere al conjunto de acciones tomadas por la red para minimizar la intensidad, extensión y duración de la congestión.

Los *traffic shapers* realizan la función de red llamada *traffic shaping,* a continuación se da una explicación breve de esta función.

4.1.1. Traffic Shaping

Traffic shaping es una técnica o mecanismo importante para garantizar calidad de servicio en una red que provee servicios integrados. Es a menudo usado en un nodo de acceso, para regular el tráfico entrante a una red. Es decir provee un mecanismo para controlar la cantidad de tráfico y la tasa promedio (tasa limite) que se envía hacía una red.

Actualmente los algoritmos más importantes para la función *Traffic shaping* son los conocidos como *Leaky bucket* y *Token bucket*, ambos son usados para diferentes propósitos.

Para el algoritmo *Leaky Bucket* (Cubeta con goteo) se puede hacer una analogía con una cubeta (bucket) de agua con un pequeño agujero en la parte de abajo a través del cual ésta se gotea. La caída de las gotas de agua es análoga a que los paquetes salgan de la cubeta a una tasa fija. Por ultimo los paquetes que llegan a la cola (*shaper leaky bucket*) en ráfagas es análogo a agregar algo de agua a la cubeta.

En si, la cubeta de goteo consiste de una cola finita. Si cuando llega un paquete hay espacio en la cola, éste se agrega a ella; de lo contrario se descarta. En cada pulso de reloj se transmite un paquete, (a menos que la cola este vacía). El algoritmo de cubeta de goteo envía un número fijo de paquetes por segundo, la cantidad de datos que se envían cada segundo varía debido a que el tamaño del paquete puede variar. Por lo que el algoritmo tiende a suavizar las ráfagas y permite que el tráfico saliente sea conformable a límites predecibles pero no garantiza una tasa fija. Para mas información se pueden consultar [3][4][5].

Un algoritmo modificado es el llamado *token bucket* o cubeta de tokens, el cual mejora al anterior en el aspecto en que tiene un mejor control de la tasa de datos de salida. En lugar de enviar paquetes cada N unidades de tiempo, un *shaper token bucket* puede reenvíar arriba de k octetos de datos en cada unidad de tiempo, en esencia el *shaper* examina el tamaño del siguiente paquete en la cola para determinar cuanto tiempo mantendrá el paquete antes de enviarlo. Cuando el *token bucket* ha acumulado suficientes *tokens* para múltiples paquetes la salida puede consistir en una pequeña ráfaga que no sobrepasa el tamaño máximo de un paquete.

Para el diseño del sistema planteado se opto por el *token bucket*, por lo que en la sección 4.2.4 se profundiza más el algoritmo.

4.2. Bloques Funcionales

En este trabajo se considera un sistema de red, a aquel que permite realizar varias funciones sobre paquetes. Para el sistema de red propuesto en este trabajo se necesito tomar en cuenta varios aspectos. Primeramente la arquitectura donde se implementara así como su funcionamiento y manejo, y segundo las funciones de red o por llamarlo de alguna manera el procesamiento necesario sobre los paquetes para conseguir la regulación de tres tipos de flujo de trafico.

Para el sistema de red planeado se requieren las siguientes funciones sobre los paquetes.

- Reenvío de paquetes
- Clasificación de paquetes:
- Encolar y descartar paquetes
- Regulación de tráfico (*traffic shaping*)

Además se requirió del planteamiento de varios bloques funcionales que permitan a la arquitectura de hardware (NP) llevar a cabo el procesamiento de paquetes requerido. Y algo importante, se necesita una asignación de recursos de hardware para dichos bloques.

Los bloque funcionales planteados son la recepción de paquetes por el procesador de red de acuerdo al funcionamiento de la tarjeta para el correcto manejo de los paquetes, la clasificación de tres tipos de paquetes, los correspondientes al encolamiento y des-encolamiento de paquetes almacenados en memoria, la regulación del tipo trafico utilizando el algoritmo de *token bucket* correlacionado y por ultimo la transmisión de paquetes fuera del la tarjeta.

A continuación se explica con más detalle cada uno de estas funciones.

4.2.1. Recepción de paquetes o entrada de paquetes al NP

Para que el NP pueda recibir un paquete debe haber una interacción entre las micromáquinas dedicadas a esta función, unidades de FBI y el hardware externo al chip IXP. Todo es coordinado por las micromáquinas.

El bus de datos externo en el IXP1200, llamado bus IX, que esta conectado a las MACs de la tarjeta, transfiere pedazos de datos de 64 bytes. Esos pedazos de 64 bytes son llamados también mpaquetes. Esos mpaquetes pueden tener las etiquetas de SOP, EOP o MOP que indican inicio, final y parte intermedia de un paquete.

Para que una micromáquina pueda realizar la recepción de un solo mpaquete debe seguir los siguientes pasos [4].

1- Determinar si hay un mpaquete disponible, revisando los registros de control y estado (CSR) *rcv_rdy_lo* y *rcv_rdy_hi.* Dentro de la unidad FBI hay un pequeño procesador llamado secuenciador *ready-bus*, el cual se programa para revisar los dispositivos físicos conectados al bus de datos externo, para verificar en cual de ellos hay datos disponibles; basándose en esa información el secuenciador actualiza los registros *rcv_rdy_lo* y *rcv_rdy_hi.*

2- Una vez que esta disponible el mpaquete, la micromáquina emite una petición de recepción, la cual se lleva a cabo escribiendo al CSR *rcv_req* en el FBI. Esto permite mover uno o más mpaquetes desde el hardware físico a un elemento RFIFO. El FBI contiene 16 elementos RFIFO (Receive FIFO), cada uno puede mantener un mpaquete.

3- Cuando termina la transferencia de un mpaquete debida a la petición de recepción, el FBI emite una señal para confirmar la recepción y se actualiza el CSR de control de recepción (*rcv_cntrl*). El registro *rcv_cntrl* contiene información del estado del mpaquete recibido (notificaciones de error, estado del reensamblaje y tamaño del mpaquete).

Como los paquetes entrantes pueden ser mayores o menores a 64 bytes, las micromáquinas tienen que coordinar el reensamblado de los mpaquetes en el paquete completo. Los mpaquetes son reensamblados en la memoria SDRAM, la cual es dividida en buffers. Los mpaquetes son colocados contiguamente en memoria utilizando uno de los LIFOs de la SRAM, así que el buffer representa al paquete completo entrante en memoria, figura 4.1.

En la figura 4.2 se muestra un algoritmo utilizado para el reensamblado de un paquete [4], donde el estado de reensamblaje es información que se va actualizando conforme se va analizando cada uno de los mpaquetes que conforman un paquete, tal como un contador que lleva el control del número de mpaquetes, banderas que indican si el mpaquete es valido o fue descartado.

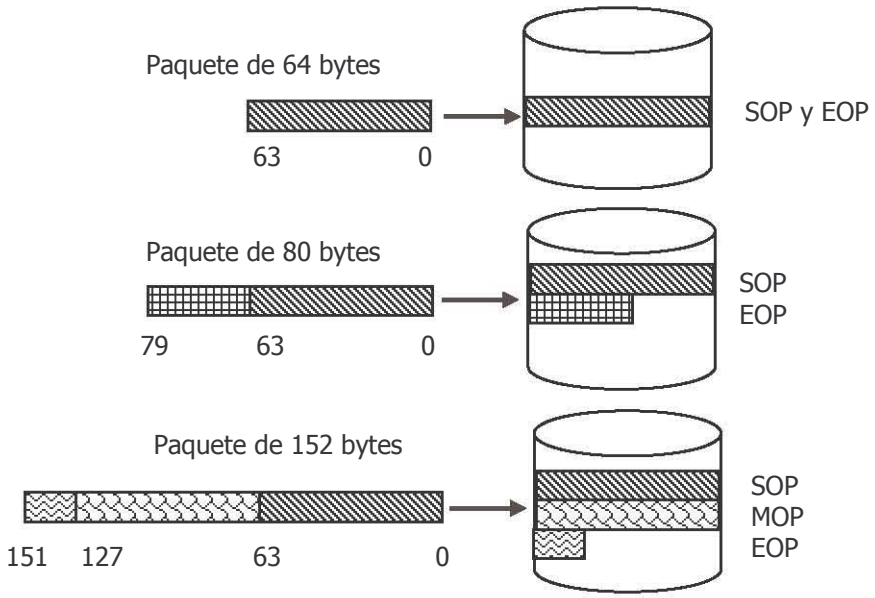

Figura 4.1 Los mpaquetes son reensamblados en memoria SDRAM. Al espacio asignado en esta memoria

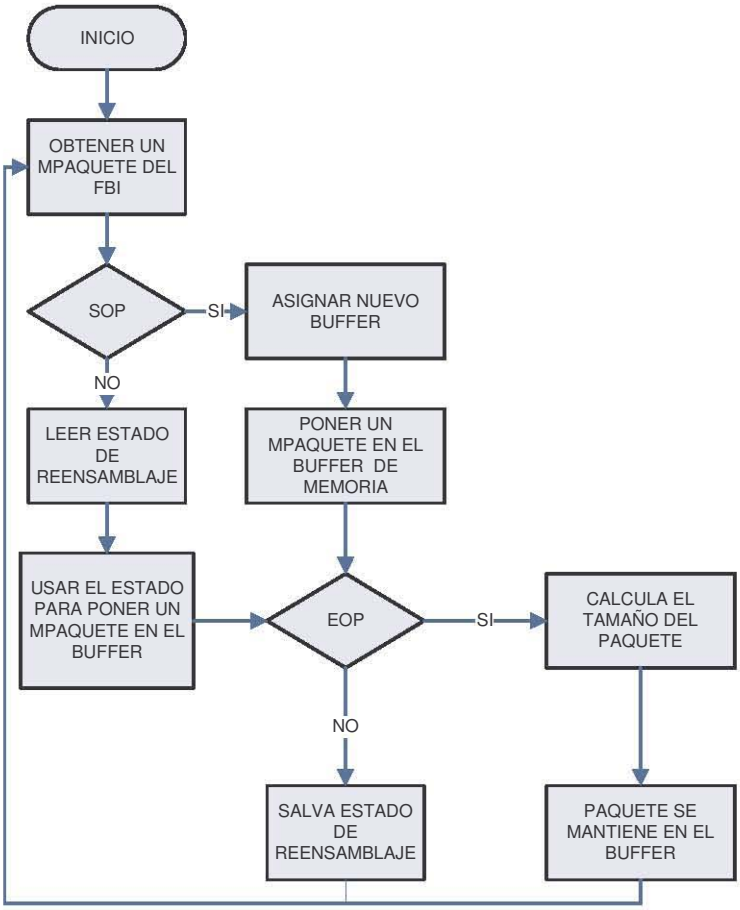

Figura 4.2 Algoritmo de reensamblado de paquetes

4.2.2. Clasificación de paquetes

La clasificación de paquetes es una tarea importante que es desarrollada por muchas aplicaciones basadas en redes IP.

El término clasificación de paquetes se refiere al proceso de "mapear" un paquete a uno de varios tipos de flujos o categorías. La clasificación de paquetes se puede dividir en estática y dinámica, la primera se refiere a un conjunto de categorías que son determinadas con anticipación y que no cambian, mientras que las segundas a que las categorías cambian todo el tiempo [1].

Para clasificar se hace uso de las cabeceras correspondientes a las capas del protocolo. Algunos de los identificadores que se utilizan para la clasificación son direcciones IP, direcciones MAC, tipo de protocolo y número de puerto.

Para este trabajo la clasificación es estática, los *frames* que se manejan son Ethernet (*Figura 4.3*) y las reglas definidas a priori sirven para identificar tres tipos de flujo, estos son:

- Paquetes
  - no Ip, e Ip no TCP ni UDP correspondientes a Ethernet
  - paquetes 802.3
- Paquetes IP UDP Ethernet
- Paquetes IP TCP Ethernet

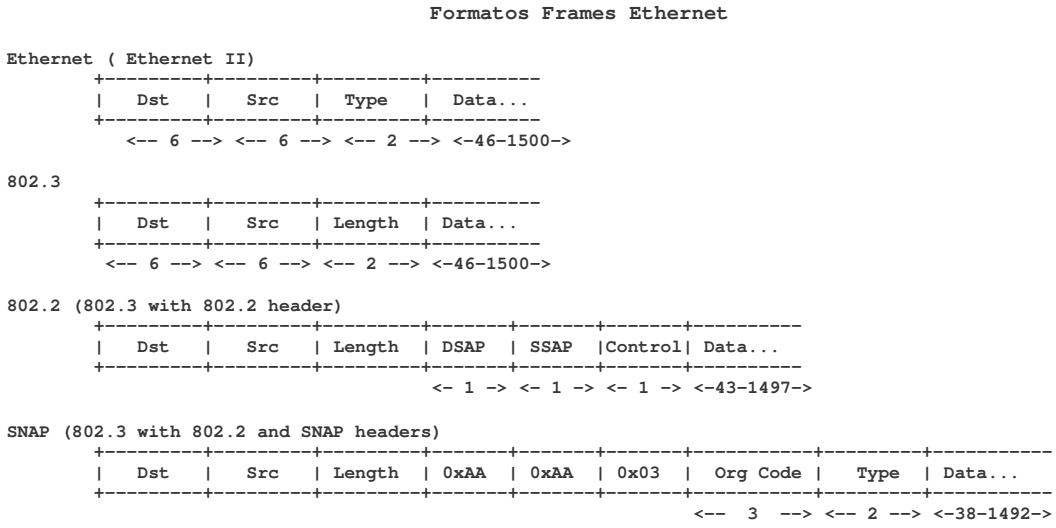

Figura 4.3 Frames Ethernet

Para la clasificación de paquetes se utiliza el mpaquete SOP que es leído de un RFIFO (antes de asignarle en espacio en memoria). Esto porque de ahí se puede extraer la cabecera Ethernet y el datagrama IP.

Para identificar los tipos de paquetes primeramente se hace uso del treceavo y catorceavo byte de la cabecera Ethernet los que indican o el campo tipo para especificar al protocolo que es transportado en la trama en caso de un paquete Ethernet o el campo longitud de la trama, el cual es utilizado para indicar la cantidad de *bytes* que hay en el campo de datos en 802.3 (Figura 4.3). Estos dos bytes son extraídos y almacenados temporalmente en una variable que se denomina *ether_header.protocol_length_type* que sirve para hacer una distinción entre los paquetes. [19]

- Ethernet IP
- Ethernet ARP y Ethernet 802.3

El pseudocodigo es el siguiente.

Si *(ether_header.protocol_length_type > ETHER_MTU Y*
ether_header.protocol_length != ETHER_ARP) ***Entonces***
Ethernet IP
En otro caso
Ethernet ARP y Ethernet 802.3
Fin Si

ETHER_MTU = 1500 o en hexadecimal 05 DC
ETHER_ARP = 2054 o en hexadecimal 0X806

Una vez identificados los paquetes IP Ethernet se tiene que hacer una separación entre paquetes TCP y UDP. Entonces se extrae la cabecera IP V4, de donde se analizara el campo de protocolo. La información de este campo es almacenada en una variable que se le asigno el nombre de *ipv4_header.protocol* . Este campo y por lo tanto la variable pueden tener diferentes valores dependiendo del protocolo , los valores que interesan conocer son.

0x06 en hexadecimal para protocolo TCP
0x11 en hexadecimal para protocolo UDP

El pseudocodigo completo para la clasificación mencionada anteriormente es:

Si *(ether_header.protocol_length_type > ETHER_MTU Y*
ether_header.protocol_length != ETHER_ARP) **Entonces**
Paquetes IP
Extraer de la cabecera IP el campo de protocolo

Si *(ipv4_header.protocol es igual a TCP_PROTOCOL*) **Entonces**
Paquete Ethernet IP / TCP
En otro caso
Si (*ipv4_header.protocol == UDP_PROTOCOL*) Entonces
Paquete IP/UDP
En otro caso
Paquete Ethernet IP no TCP ni UDP
Fin Si
Fin Si

En otro caso
Paquetes no IP (ARP) y 802.3
Fin Si

Una vez identificado el tipo de paquete, este se envía a su correspondiente cola.

4.2.3. Encolamiento y desencolamiento

El manejo de colas de paquetes es la parte esencial de muchos sistemas de red, ya que permite proveer diferentes calidades de servicio a esos flujos de paquetes.

Para este trabajo el manejo de colas de paquetes se utiliza para poder manejar tres tipos de paquetes y asignarle a cada flujo una regulación en bits por segundo.

Para el sistema diseñado se utilizan las colas lógicas para encolar el paquete dependiendo de su clasificación. Se le llaman colas lógicas porque en realidad ahí no esta almacenado el paquete sino información que permite localizar un paquetes que se coloco en memoria SDRAM.

La función que se encarga de recibir un paquete en el NP, retorna un identificador de paquete llamado manejador del paquete o buffer (buffer handle), que sirve para saber a que paquete nos estamos refiriendo al momento de encolar y desencolar. Entonces las colas lógicas permiten encolar y desencolar los identificadores de los paquetes.

Para encolar y desencolar los tres tipos de flujo se utiliza una estructura de datos basada en arreglos, es decir tres arreglos son las tres colas lógicas donde se almacena el manejador del buffer (de cada paquete).

Para su control se utiliza un índice productor que apunta al siguiente elemento del arreglo donde se pondrá el paquete encolado y un índice consumidor que indica el elemento del arreglo que contiene el paquete a desencolar. Figura 4.4 .

Los algoritmos de encolamiento y des-encolamiento se tomaron de [4] y se muestran en el apéndice A.

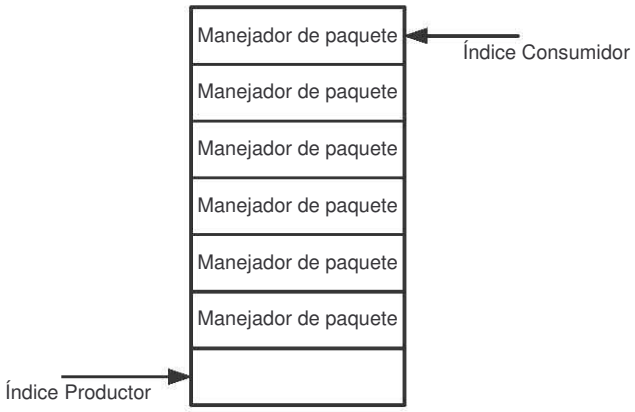

Figura 4.4 Representación de los elementos de una cola basada en arreglos

4.2.4.Regulación (Shaping)

Uno de los principales problemas encontrado en el diseño fue la manera de implementar la función de regulación (*traffic shaping*) en una de las interfaces de red del IXP1200. Para la solución, lo primero fue elegir que algoritmo utilizar para la regulación, los más comunes son el llamado *leaky bucket* y el *token bucket*. Se opto por el algoritmo *token buket* modo *shaping*, debido a que lo que se almacena en el *bucket* son *tokens* y no paquetes y por lo tanto lo que se tiran son *tokens* y no paquetes.

A continuación se explica el algoritmo y posteriormente las modificaciones hechas al mismo para adaptarlo a lo que se desea.

4.2.4.1. Algoritmo *token bucket* modo *shaping*

Para la explicación del algoritmo se considera un flujo de paquetes entrante a una red cada uno de ellos de tamaño L, un buffer para almacenar los paquetes, un *bucket* de tamaño B donde se almacenan *tokens* y C el número de *tokens* dentro del *bucket*.

Si llega un paquete de tamaño L (bytes) y este es menor o igual al número de *tokens* dentro del *bucket* C, entonces el paquete pasa y se tiran L *tokens* del *bucket*. En caso contrario el paquete espera en el buffer hasta que sea conformable con C, entonces el paquete pasa y se tiran L *tokens* del *bucket*. Figura 4.5.

En el algoritmo original el *token* es generado a una tasa constante y almacenado en un *bucket*. Aquí se hizo la modificación de poder guardar un tamaño fijo de *tokens* en el *bucket* periódicamente. Figura 4.6.

De acuerdo al RFC 2215 y 2211 la tasa a la cual se llena *bucket* es medida en bytes por segundo, y el valor esta en un rango de 1 byte por segundo a 40 terabytes por segundo. La profundidad del *bucket* es medido en bytes y el valor de este parámetro tiene un rango de 1 byte a 250 gigabytes.

El pseudocódigo del algoritmo descrito y utilizado para el código que realiza esta función es el siguiente [21]:

```
If ( L< = C){
        C=C-L;
Pasa paquete();
          }
En otro caso {
     While (C<L) {
espera; }
if (C = = L)     {
          C=C-L;
Pasa paquete(); }
          }
```

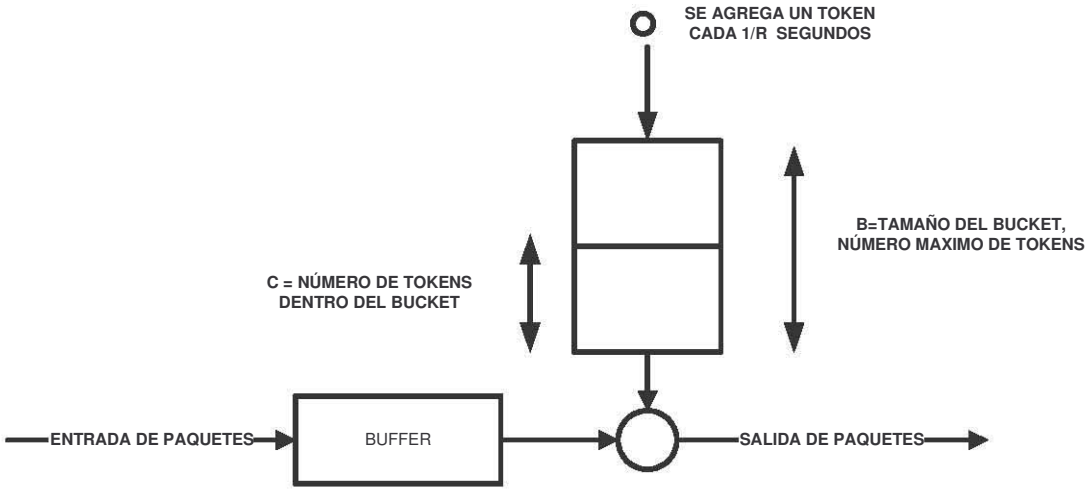

Figura 4.5 Algoritmo Token bucket modo shaping

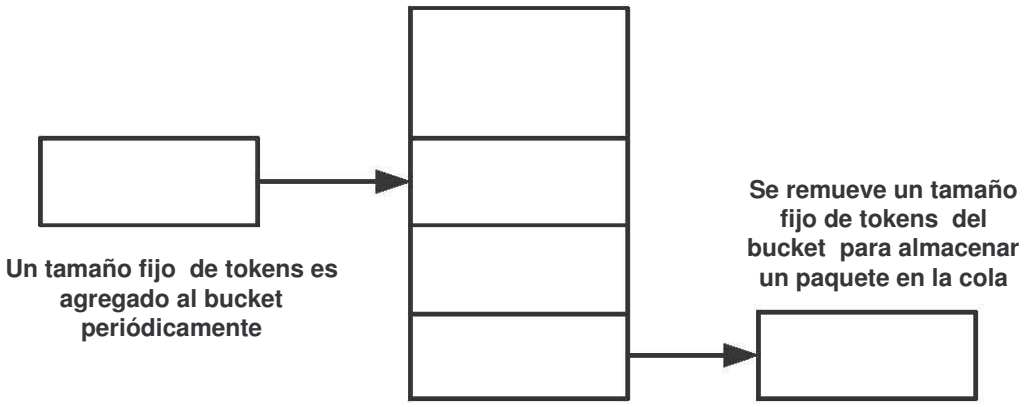

Figura 4.6 Entrada y salida de los tokens en el bucket

Este algoritmo permite a la salida ráfagas, pero limitadas a una longitud máxima regulada, es decir permite ahorrar permisos de envió de n paquetes hasta un tamaño máximo de la cubeta B (bucket), lo que significa que todas las ráfagas de hasta n paquetes con un total de B bytes se pueden enviar a la vez.

Entonces para este algoritmo se tiene:

- A largo plazo la tasa de salida es limitada a ρ, que es la tasa de llegada de los token al bucket.
- A corto plazo, se puede enviar una ráfaga de tamaño B
- La cantidad de tráfico entrante a la red en un intervalo T esta limitada por

  Tráfico = B + ρ *T

4.2.4.2. Algoritmo *token bucket* correlacionado.

El algoritmo *token bucket* modo *shaping* descrito anteriormente es aplicable para la regulación de un sólo tipo de tráfico o bien no considerando el tipo. En esta parte se explica un algoritmo que permite la regulación de varios flujos de paquetes de acuerdo a *n* clases de tráfico. Este algoritmo consiste en tener varios *token bucket* correlacionados y su objetivo es poder utilizar el ancho de banda no usado por clases de tráfico consideradas más "caras", para transmitir datos de clases más "baratas".

La base del diseño de este algoritmo se obtuvo de [5], El aquí presentado tiene modificaciones, ya que permite el manejo de paquetes de tamaño variable.

Para este algoritmo se asume que se tienen *n* clases de tráfico y por lo tanto se necesitan *n* reguladores *token bucket* donde cada uno funcionara de igual forma que el explicado antes, por lo que se aceptan paquetes de tamaño variable. Para cada clase de tráfico *i*, hay un *bucket*, llamado *bucket i*, que es usado para garantizar el mínimo ancho de banda para cada clase de tráfico *i*. Además hay una cola asociada llamada cola *i* que almacena los paquetes de clase *i* cuando el *bucket i* esta vacío.

Para cada tipo de tráfico *i* se agregan un número fijo de *tokens* al *bucket i*. La generación del número fijo de *tokens* para la clase de trafico *i*, se le llama generador de *tokens i* y el número fijo de *tokens* que éste genera son llamados *tokens* de clase *i*.

Si se asume que se está generando un número de *tokens* de tipo *i*, ese número es colocado en el *bucket i* si éste no ésta lleno. En el caso de que el *bucket i* este lleno, éste es colocado en el *bucket j*, donde *j* es el siguiente número más pequeño que sea un poco mayor a *i*.

Cuando la cola *i* no esta vacía implica que el *bucket i* esta vacío si se considera una capacidad de enlace infinita, y por lo tanto cuando el *bucket i* no esta vacío la cola *i* esta vacía.

Si todas las colas están vacías, entonces se coloca el número fijo de *tokens* en el *bucket j*, donde $j = i + 1$, tal que el *bucket j* no esta lleno. Si el *bucket j* esta lleno para toda $j \geq i$, entonces el número de *tokens* fijo se pierde. En la figura 4.7 se aprecia la idea del algoritmo.

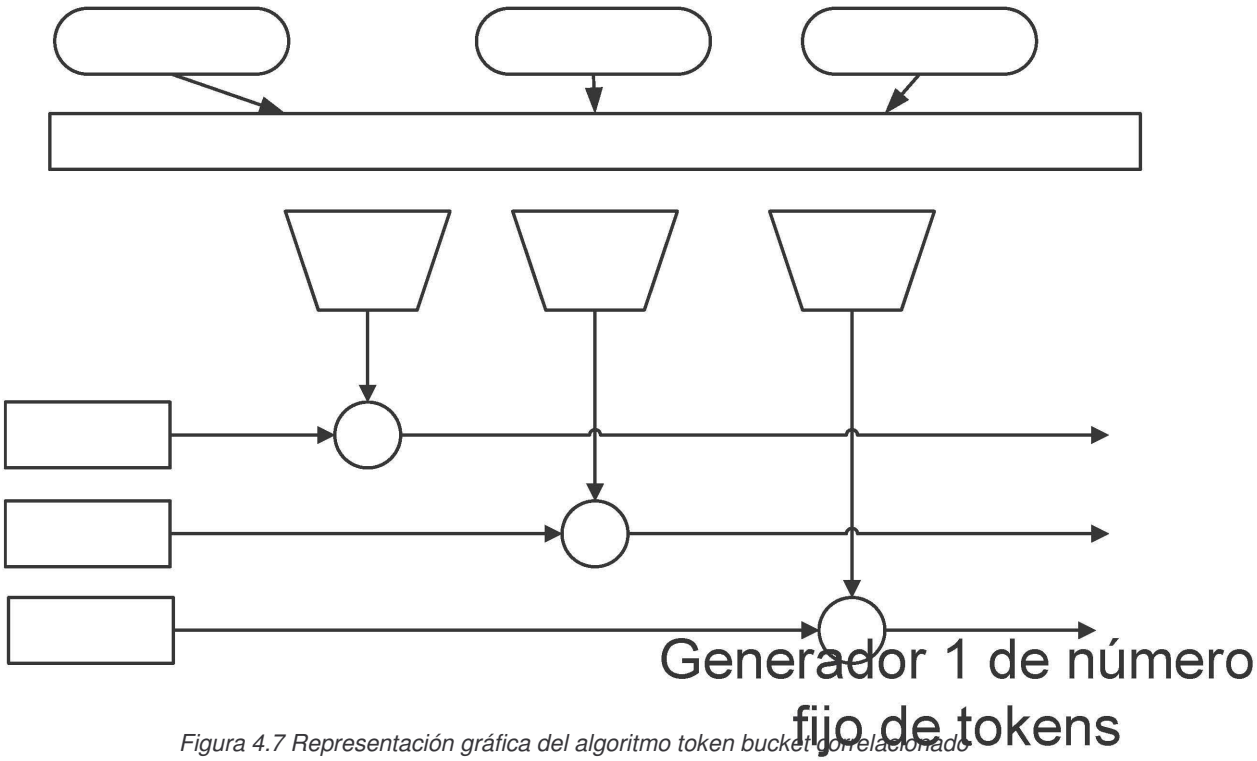

Figura 4.7 Representación gráfica del algoritmo token bucket correlacionado

En el diseño para el procesador de red, una de las micromáquinas se dedica a la generación de *tokens* para los tipos de tráfico y asigna un número fijo de *tokens* a cada *bucket* de acuerdo al algoritmo. En un inicio los *buckets* están llenos.

En específico para el sistema a implementar se tomaran en cuenta sólo tres tipos de tráfico y en consecuencia tres *buckets* que se llenaran con un número de *tokens* asignado.

El primer *bucket* que es asignado a la clase más cara, se llena aproximadamente agregando *k bytes* cada 584 ns, el segundo a *k* bytes cada 786 ns y el tercero a *k* bytes cada 983 ns.

4.2.5.Transmisión de paquetes fuera del NP

Una vez que el paquete ha sido desencolado y regulado este necesita salir del dispositivo al medio. Al proceso de enviar fuera el paquete se denomino transmisión del paquete y se considera el proceso inverso al de recepción.

El secueciador *ready-bus* al igual que para la recepción, revisa el estado del o los puertos utilizados para la transmisión para ver si están disponibles y de a cuerdo a la información obtenida modifica el registro de control y estado CSR llamado *xmit_rdy,* el cual es leído por el código asignado a las o la micromáquina que se encarga del bloque de transmisión.

Antes de describir los pasos que se consideran para la transmisión de un mpaquete, es importante mencionar que cada uno los elementos TFIFO contiene los siguientes campos:

- El campo que mantiene los mpaquetes de 64 bytes del buffer de datos para que sean enviados al dispositivo MAC.

- Ocho bytes de control que permiten obtener información sobre cual puerto esta disponible para enviar por ahí los datos, así como también el orden de los mpaquetes, en dado caso que el paquete sea mayor a 64 bytes, es decir si es SOP o EOP.

El proceso de transmisión de un sólo mpaquete consiste de seis pasos:

1. Seleccionar el elemento TFIFO en el cual se escribirán datos de acuerdo al diseño.

2. Esperar a que el elemento TFIFO sea marcado como vacío, es decir que no contenga datos anteriores, para evitar sobre escribir datos.

3. Transferir datos al elemento TFIFO seleccionado desde la SDRAM y/o registros de transferencia de la micromáquina.

4. Escribir a la palabra de control (control word) del elemento TFIFO. La palabra de control contiene información necesaria para la transmisión. Tal información es a cual MAC y puerto enviar los datos, la cantidad de datos a enviar y si el paquete es SOP, EOP, ambos o ninguno de ellos.

5. Esperar a que el dispositivo MAC este listo para más datos. Se debe estar seguro que la MAC esta lista para recibir datos en su buffer antes de enviarlos del TFIFO a la MAC. Las MAC tienen señales que son leídas por el secuenciador *ready-bus* que ayudan al código de las micromáquinas a decidir cuando validar las entradas a los TFIFO. Las señales MAC son llamadas *transmit ready bits* a las cuales se puede acceder desde el código a través de los CSRs *xmit_rdy_hi* y *xmit_rdy_lo*

6. Validar el elemento TFIFO. Una vez que se determino que el dispositivo MAC esta listo para recibir datos, éste valida el elemento TFIFO. Esto es escribir al registro xmit_validate.

Una vez que el TFIFO ha sido validado, termina el trabajo del código asignado a la micromáquina. La transmisión de un paquete depende de un proceso realizado por el hardware llamado TSM hardware (Transmit State Machine Hardware) [4][11]. El proceso consiste en ir procesando los TFIFO en orden y mover un puntero. Cuando el puntero se mueve a un TFIFO validado, se copian los datos al puerto MAC indicado en la palabra de control, después el hardware marca al TFIFO como invalido.

Para el diseño se tomo como base el algoritmo proporcionado en [4] donde un hilo de la micromáquina se encarga de realizar el ciclo de despacho (dispatch loop). Este ciclo desencola los paquetes realiza la regulación e inicia el bloque de transmisión.
Los demás hilos llenan los TFIFOs, para lo que se considera un ciclo en donde cada iteración se maneja un mpaquete.

4.3. Asignación de recursos de hardware.

Como se menciono en el capítulo 3, los Procesadores de Red, son procesadores programables que usan *multihilaje*, su arquitectura multiprocesador explota un paralelismo a nivel de paquetes, tiene un conjunto de instrucciones diseñadas específicamente para la implementación de tareas relacionadas con *networking*. Específicamente el NP IXP1200 como ya se menciono, tiene 6 CPUs RISC llamadas micromáquinas. Cada micromáquina tiene 4 hilos y soporta hardware multihilaje con cambios de contexto.

En este trabajo se utilizan dos de las micromáquinas para procesar la recepción de paquetes por uno de los puertos (puerto 0) Ethernet 100 Mbps. El puerto esta asociado con los elementos RFIFO dependiendo de la micromáquina e hilo que se este utilizando. En el sistema, se asocia el puerto 0 a las micromáquinas 0 y 1 (teniendo en total 8 hilos), y cada hilo de cada micromáquina están asociados a elementos RFIFO (figura 4.8).

Las micromáquinas usadas para la recepción y reensamblado de los llamados m-paquetes también hacen la función de identificación de tres tipos de paquetes y encolamiento a tres colas lógicas para tres tipos de flujo.

Sólo una micromáquina se usa para desencolar, organizar la salida y transmitir los paquetes fuera del dispositivo. De los cuatro hilos de la micromáquina sólo uno de ellos se encarga de desencolar los paquetes y de utilizar el algoritmo de *Token Bucket* correlacionado para organizar la salida de los paquetes (regularlos) y pasarlos al bloque de transmisión. Los tres hilos restantes conforman el bloque de transmisión. Estos hilos están asociados a elementos TFIFOs, donde se colocan los mpaquetes que conforman un paquete completo y para ser enviados al puerto 1 (MAC1). Lo anterior se muestra en la figura 4.8.

Otra de las micromáquinas se dedico al llenado de los *buckets*. Las restantes se dejan para posteriores modificaciones al sistema.

Con respecto a las memorias (SRAM, SDRAM y Scratch) estas son compartidas entre las micromáquinas y son usadas para almacenar la arquitectura de colas (SRAM), y paquetes entrantes (SDRAM), así como el reensamblado de los mpaquetes en un paquete (SRAM).

La memoria SRAM se utiliza también para almacenar y modificar las banderas necesarias para el control del sistema.

La arquitectura de colas tiene básicamente la función de encolar y desencolar.

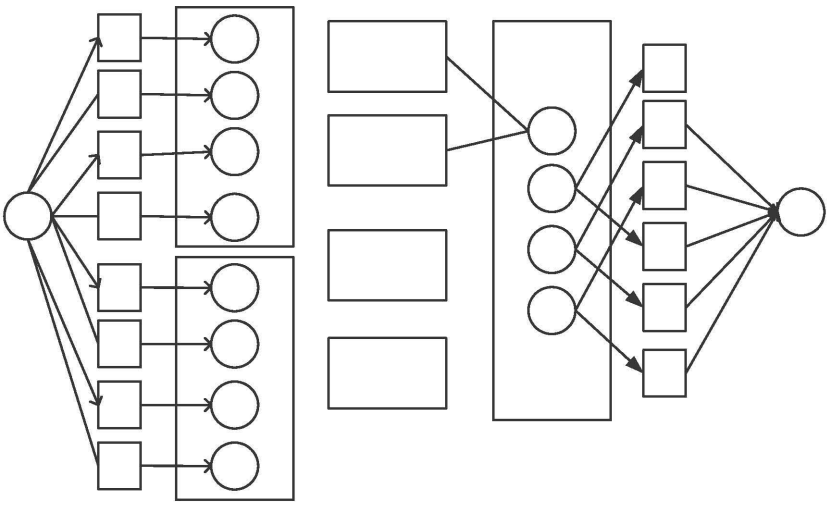

Figura 4.8 Representación de la asignación de recursos de hardware en el diseño del sistema

Puerto 0

CAPÍTULO 5

En esta parte se evalúa y analiza el sistema de red diseñado en el capitulo 4 dando algunas justificaciones de los resultados obtenidos. Todos los experimentos se realizaron en el NP IXP1200.

5. Evaluación y análisis del sistema de red

Lo que se pretende evaluar del sistema diseñado en el capitulo 4, es la salida regulada para los tres tipos de tráfico planteados así como un análisis de los resultados obtenidos.

5.1. Ambiente de red para el análisis y evaluación

Para el análisis y evaluación del sistema se cuenta con un ambiente de red que consta de una PC que tiene instalada la tarjeta con el procesador de red IXP200; otras dos PCs, una utilizada para generar los flujos de paquetes entrantes al sistema y conectada al puerto 0 Ethernet de la tarjeta, y otra para capturar la salida y conectada al puerto 1 del NP. Figura 5.1.

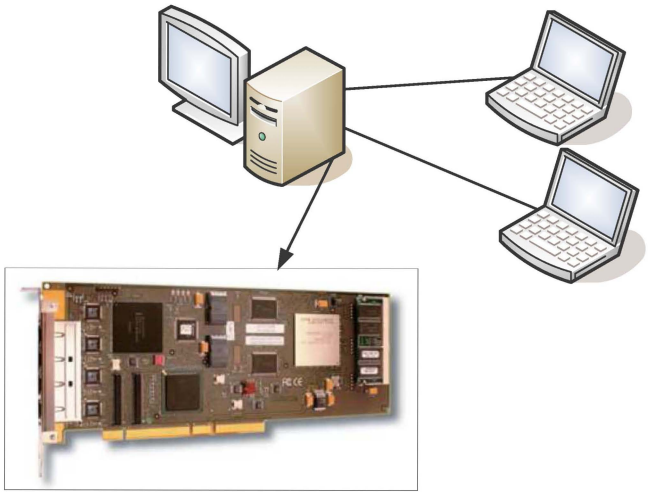

Figura 5.1 Ambiente de red para el análisis y evaluación

Para la corrida de los experimentos el tamaño de los tres *token buckets* es de 1.5 K . Su llenado es; *token bucket* 1 se llena a 1 byte cada 584 ns , el *token bucket* 2 a 1 byte cada 786 ns y el *token buket* 3 a 1 byte cada 983 ns . Lo que respectivamente representa las tasas de 1.7 MB/s, 1.3 MB/s y 1 MB/s.

5.2. Evaluación y análisis de la regulación

Para el análisis de cada tipo de tráfico por separado, se analizan tres escenarios. Cada uno de ellos consta de inyectar flujos de1000 paquetes de 256 bytes con una duración de 20 ms. La diferencia entre los escenarios 1, 2 y 3 es el retardo entre flujos, el cual es de 40 ms, 80 ms y 160 ms respectivamente. En la tabla 5.1 se muestras todos los escenarios introducidos a la entrada de la tarjeta.

Escenario	Paquetes	Cantidad de paquetes por flujo	Retardo entre paquetes [μs]	Retardo entre flujo [ms]
1	IP-UDP	1000	20	40
2	IP-UDP	1000	20	80
3	IP-UDP	1000	20	160
4	IP-TCP	1000	20	40
5	IP-TCP	1000	20	80
6	IP-TCP	1000	20	160
7	802.3	1000	20	40
8	802.3	1000	20	80
9	802.3	1000	20	160

Tabla 5.1 Escenarios con distinto retardo entre flujos

En las figuras 5.2a - 5.10a se muestran las entradas al sistema de las tres secuencias de flujo planteadas y en las figuras 5.2b - 5.10b las salidas correspondientes.

En la tabla 5.2 se tiene una relación de la cantidad de paquetes entrantes y salientes así como la tasa promedio de salida por flujo. De estos resultados y de las gráficas se observa que hay una pérdida de paquetes y una tasa de salida para cada tipo de tráfico.

La presencia y diferencia de pérdida de paquetes en lo diferentes escenarios se debe a:

-Los buffers (colas) puede almacenar a lo más 512 paquetes.

-Si el retardo entre flujos es pequeño no se alcanzan a drenar todos los paquetes almacenados cuando llega la siguiente ráfaga, y como no se pueden almacenar muchos paquetes, entonces se pierden. Por lo que en los escenarios 1, 4 y 7 hay más pérdida de paquetes que en los escenarios 2,3,5,6,8 y 9.

Escenario	Paquetes	Paquetes entrada	Paquetes salida	Tasa salida promedio por flujo
1	IP-UDP	10000	5590	19.00 Mbps | 2.3775 MB/s
2	IP-UDP	10000	6156	19.16 Mbps | 2.3950MB/s
3	IP-UDP	9953	6100	19.27 Mbps | 2.400 MB/s
4	IP-TCP	9697	3949	12.5 Mbps | 1.5600 MB/s
5	IP-TCP	9715	6149	12.514 Mbps | 1.5642 MB/s
6	IP-TCP	10000	6547	12.600 Mbps | 1.5700 MB/s
7	802.3	10000	3074	9.3 Mbps | 1.16 MB/s
8	802.3	10000	4719	9.3 Mbps | 1.16 MB/s
9	802.3	10000	6208	9.4 Mbps | 1.17 MB/s

Tabla 5.2 Resultados de los escenarios 1-9

Teóricamente al inicio de cada flujo debe existir una ráfaga limitada a la longitud máxima regulada que depende del tamaño del *token bucket*, para después drenarse a la tasa de llegada de los *tokens*, que permiten una salida de 1.77 MB/s, 1.3 MB/s y 1 MB/s respectivamente para cada tipo de tráfico. De acuerdo a la tasa promedio de salida obtenida esta es cercana a la tasa de drenado, ya que como se menciono en el capítulo 4 (sección 4.2.4.1) a largo plazo predomina una tasa ρ que es la lata a la cual se llena el *token bucket*.

En general se puede decir que la salida para cada tipo de tráfico si esta siendo regulada y que a la salida se obtiene una tasa cercana a ρ.

Es importante mencionar que el cálculo de la tasa de llegada de los *tokens* al *bucket* depende de la arquitectura de hardware utilizada, en particular en el NP utilizado los cálculos se hicieron en base a los ciclos de reloj de mismo y dados por el simulador WorkBench SDK 2.1, por lo que la cuenta de ciclos puede tener errores.

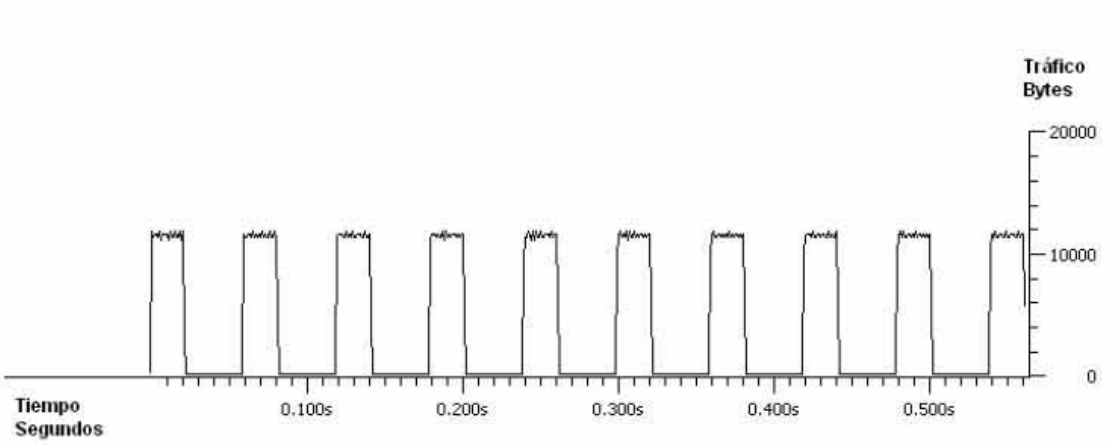

Figura 5.2a. Gráfica que muestra la entrada del escenario 1, no regulada

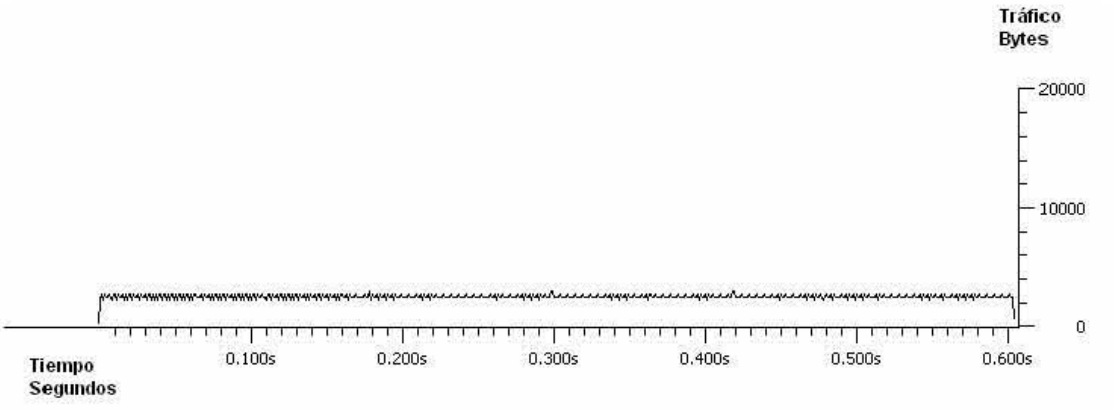

Figura 5.2b. Grafica que muestra la salida del escenario 1, regulada

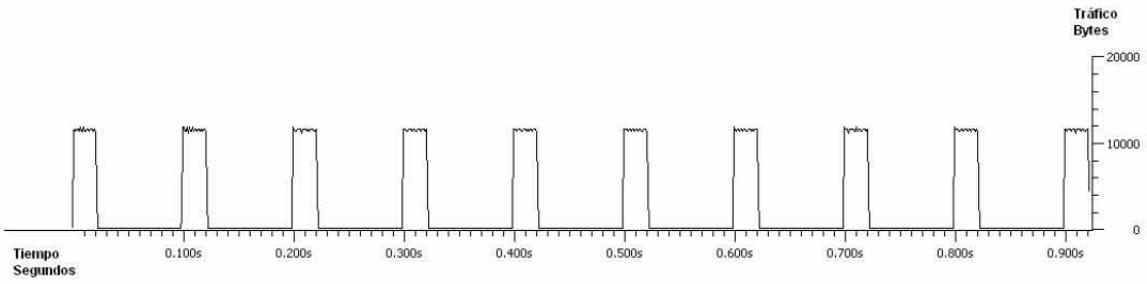

Figura 5.3a. Gráfica que muestra la entrada del escenario 2, no regulada

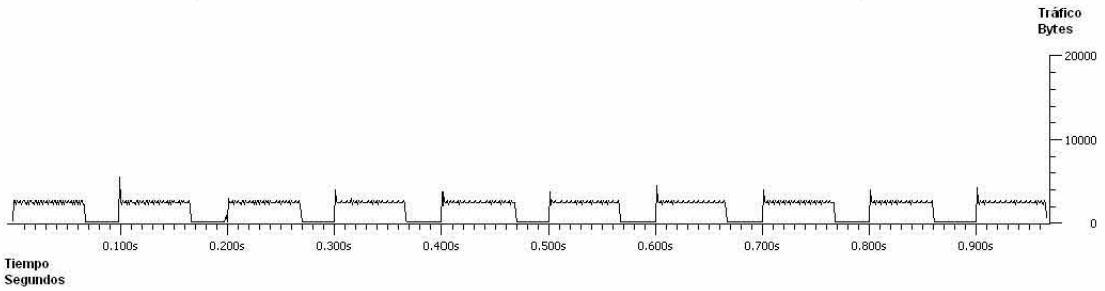

Figura 5.3b. Grafica que muestra la salida del escenario 2, regulada

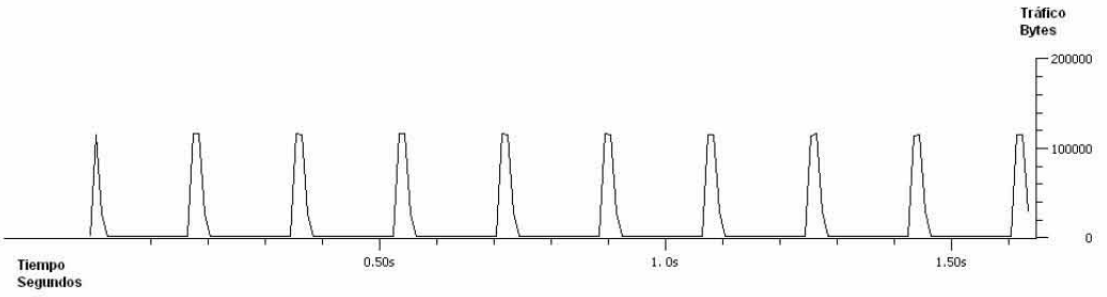

Figura 5.4a. Gráfica que muestra la entrada del escenario 3, no regulada

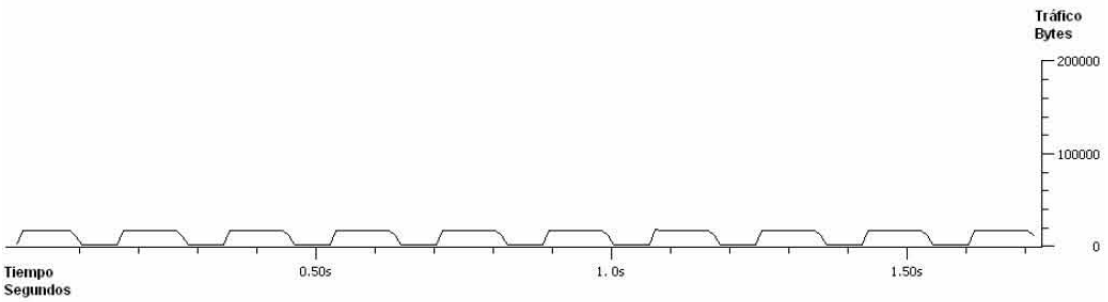

Figura 5.4b. Grafica que muestra la salida del escenario 3, regulada

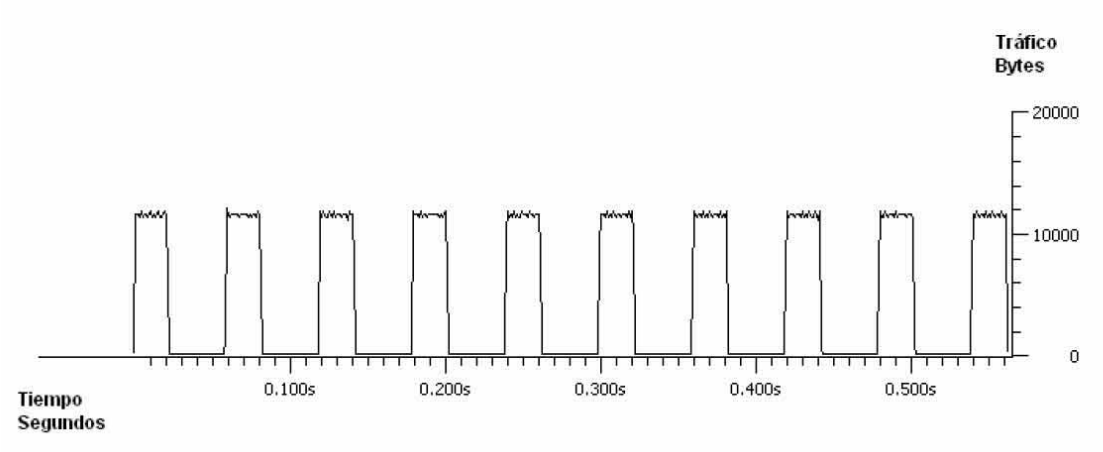

Figura 5.5a. Gráfica que muestra la entrada del escenario 4, no regulada

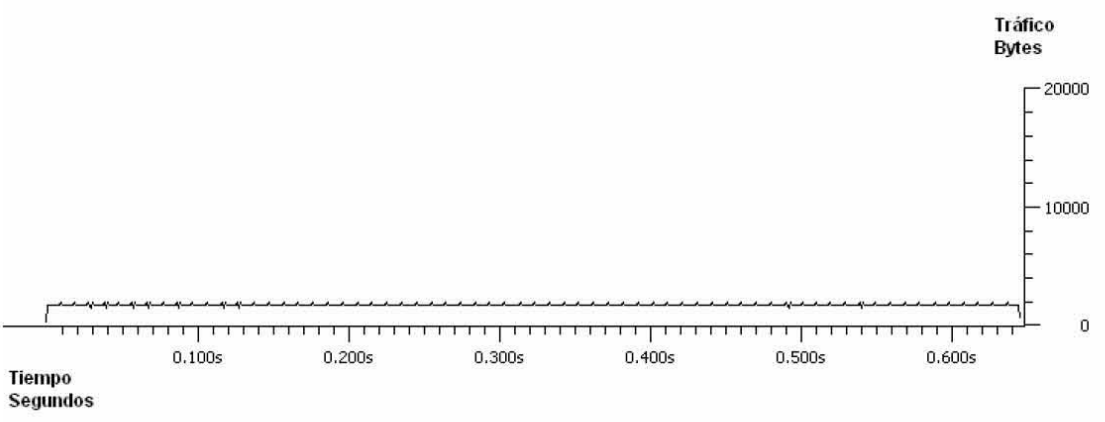

Figura 5.5b. Grafica que muestra la salida del escenario 4, regulada

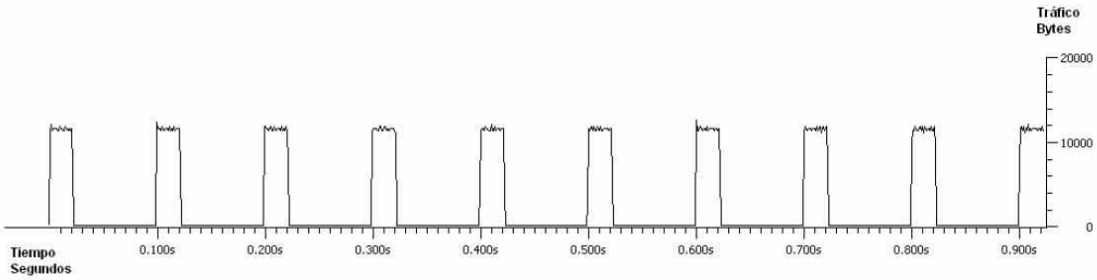

Figura 5.6a. Gráfica que muestra la entrada del escenario 5, no regulada

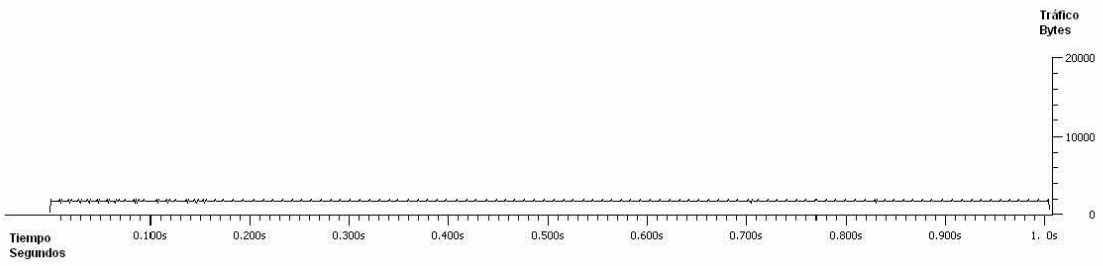

Figura 5.6b. Grafica que muestra la salida del escenario 5, regulada

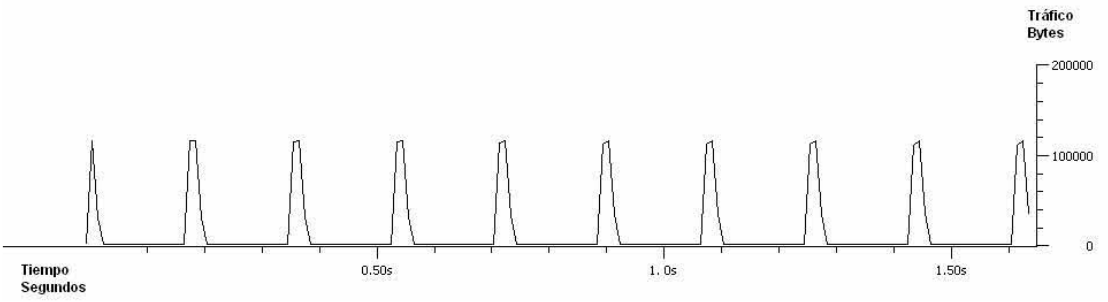

Figura 5.7a Gráfica que muestra la entrada del escenario 6, no regulada

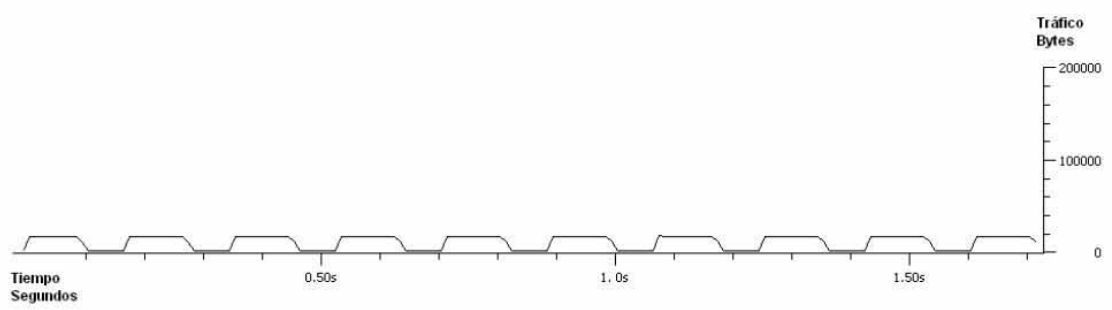

Figura 5.7 b Grafica que muestra la salida del escenario 6, regulada

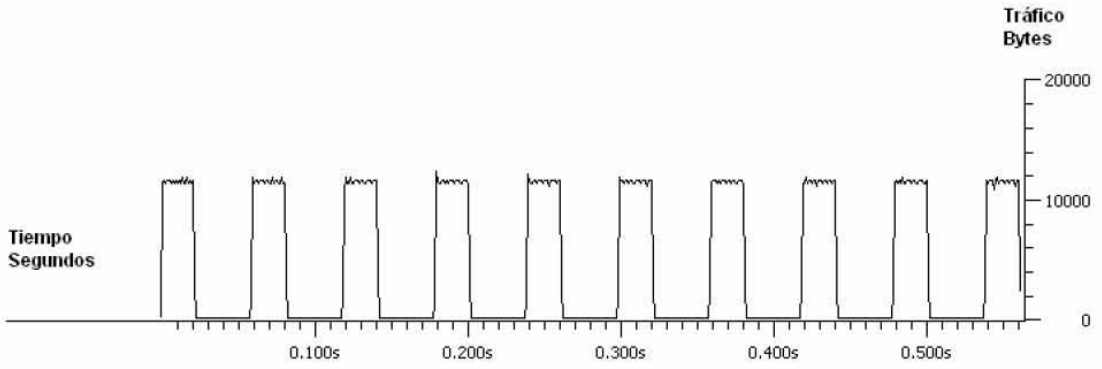

Figura 5.8 a Gráfica que muestra la entrada del escenario 7, no regulada

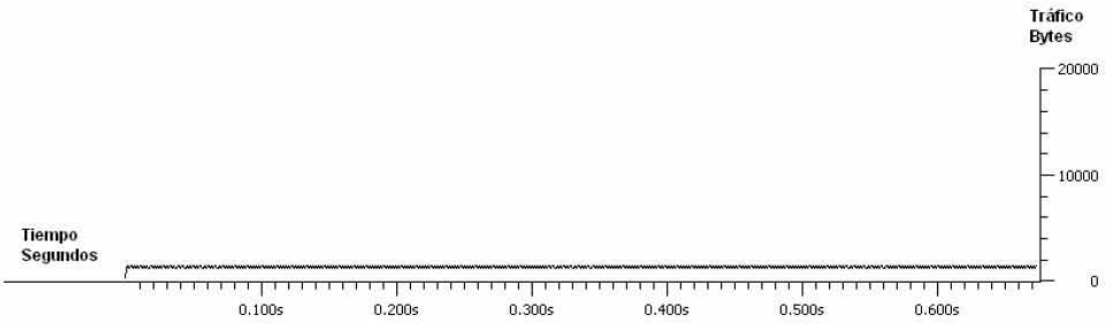

Figura 5.8 b. Grafica que muestra la salida del escenario 7, regulada

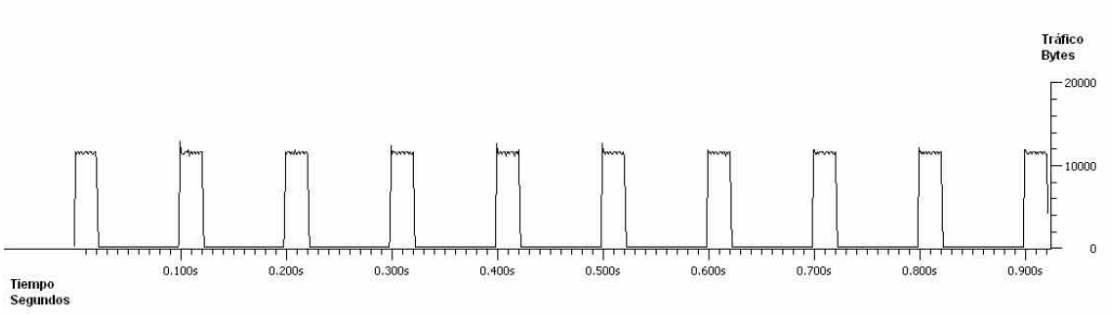

Figura 5.9 a. Gráfica que muestra la entrada del escenario 8, no regulada

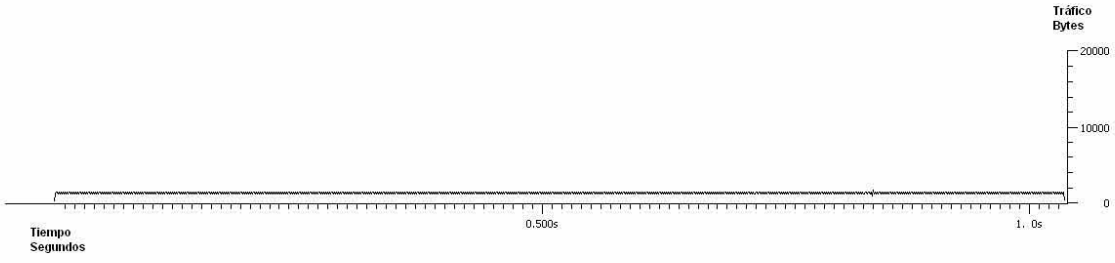

Figura 5.9 b. Grafica que muestra la salida del escenario 8, regulada

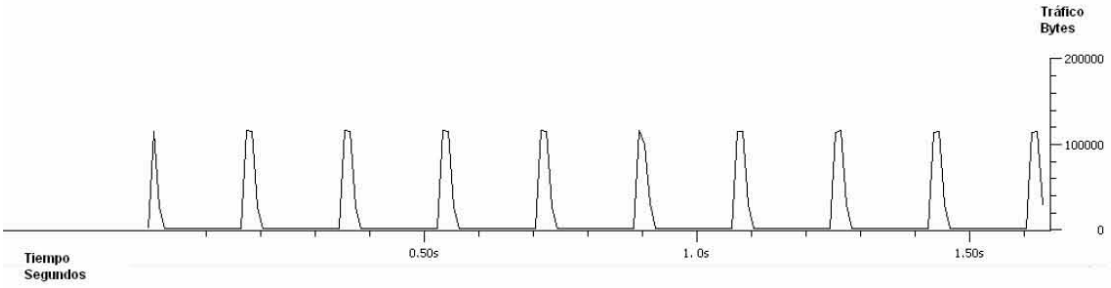

Figura 5.10 a. Gráfica que muestra la entrada del escenario 9, no regulada

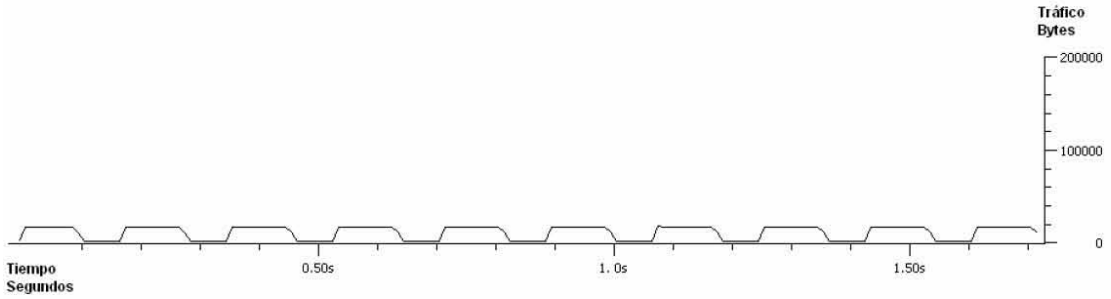

Figura 5.10 b. Grafica que muestra la salida del escenario 9, regulada

Ahora si se inyectan flujos de los tres tipos de tráfico con los escenarios de la tabla 5.3, donde los paquetes son de 256 bytes. Se nota que las tasas de salida son diferentes para los tres tipos de tráfico debido a que el llenado de los *token bucket* es diferente. Figuras 5.11a, 511b, 5.12a y5.12b.

Escenario	Paquetes	Cantidad de paquetes por flujo	Retardo entre paquetes [μs]	Retardo entre flujo [ms]
10	892.3	1000	20	80
	IP-TCP	1000	20	80
	IP-UDP	1000	20	80
11	892.3	1000	20	160
	IP-TCP	1000	20	160
	IP-UDP	1000	20	160

Tabla 5.3 Escenarios con flujos de los distintos 3 tipos de tráfico

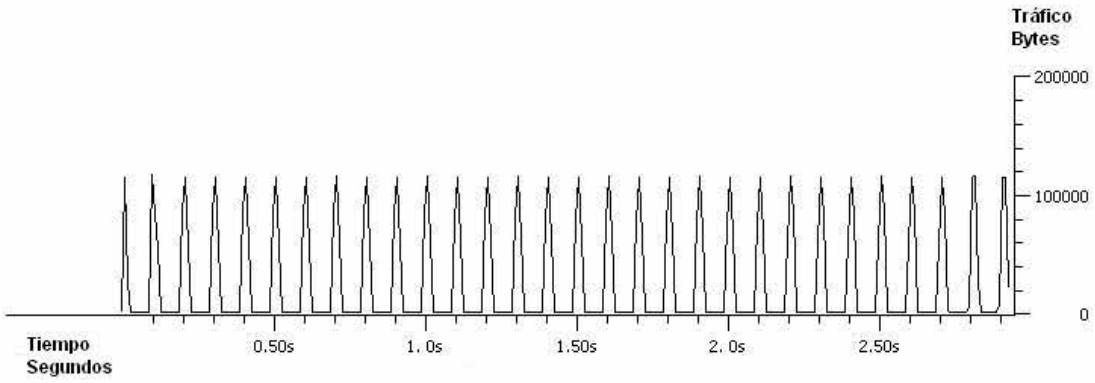

Figura 5.11a Gráfica que muestra la entrada del escenario 10, no regulada

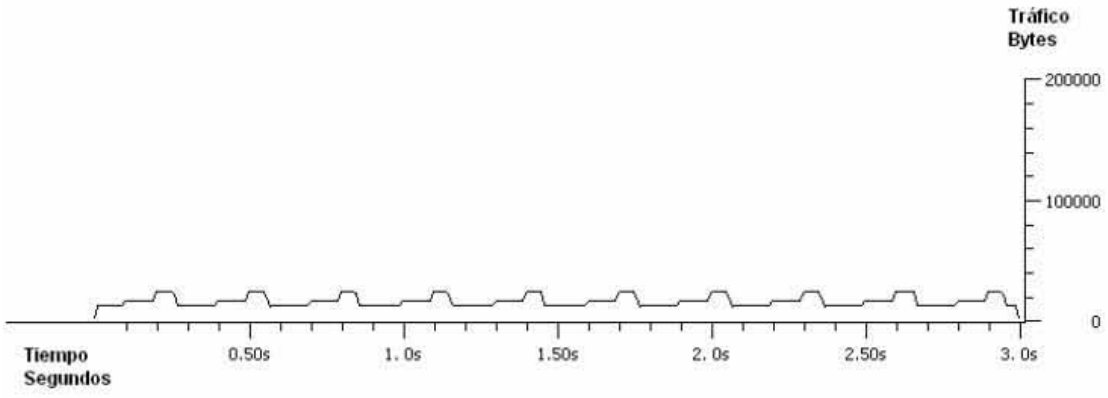

Figura 5.11 b Gráfica que muestra la salida del escenario 10, regulada

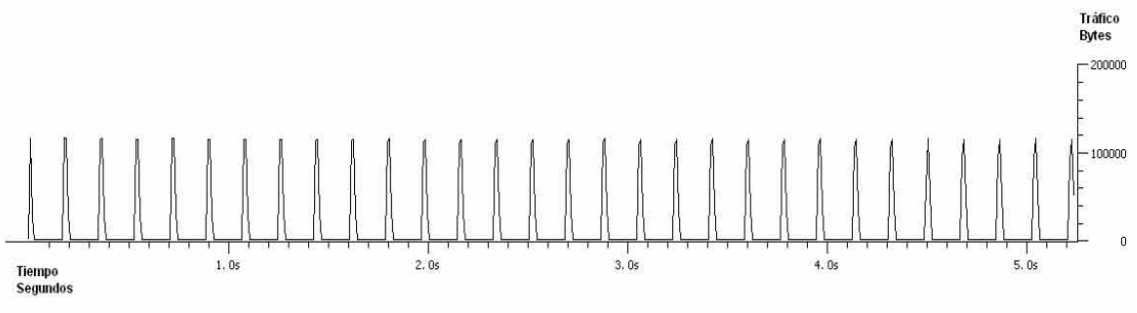

Figura 5.12 a Gráfica que muestra la entrada del escenario 11, regulada

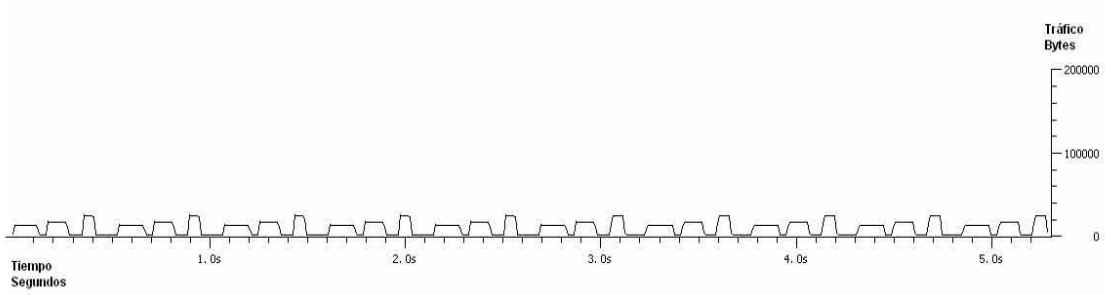

Figura 5.12 b Grafica que muestra la salida del escenario 11, no regulada

5.3. Tasa promedio máxima de salida

Como las tasas promedios de salida para cada tipo de tráfico es diferente, se plantea el siguiente experimento, que consiste en inyectar flujos con 2500, 5000, 10000, 15000, 30000 y 40000 paquetes de 256 bytes de los tres tipos de tráfico en un segundo, dejando un retardo entre flujos de 0.1 segundo, Esto representa diferentes tasas de entrada de 5.12Mbps, 10.2 Mbps, 20.4Mbps, 30.7 Mbps, 62Mbps, y 81 Mbps.

El resultado obtenido es una tasa promedio máxima de salida de 13Mbps. Figura 5.13. Otra observación importante es que si la tasa de entrada promedio es menor a la máxima de salida, no habrá modificaciones en la salida ni pérdida de paquetes.

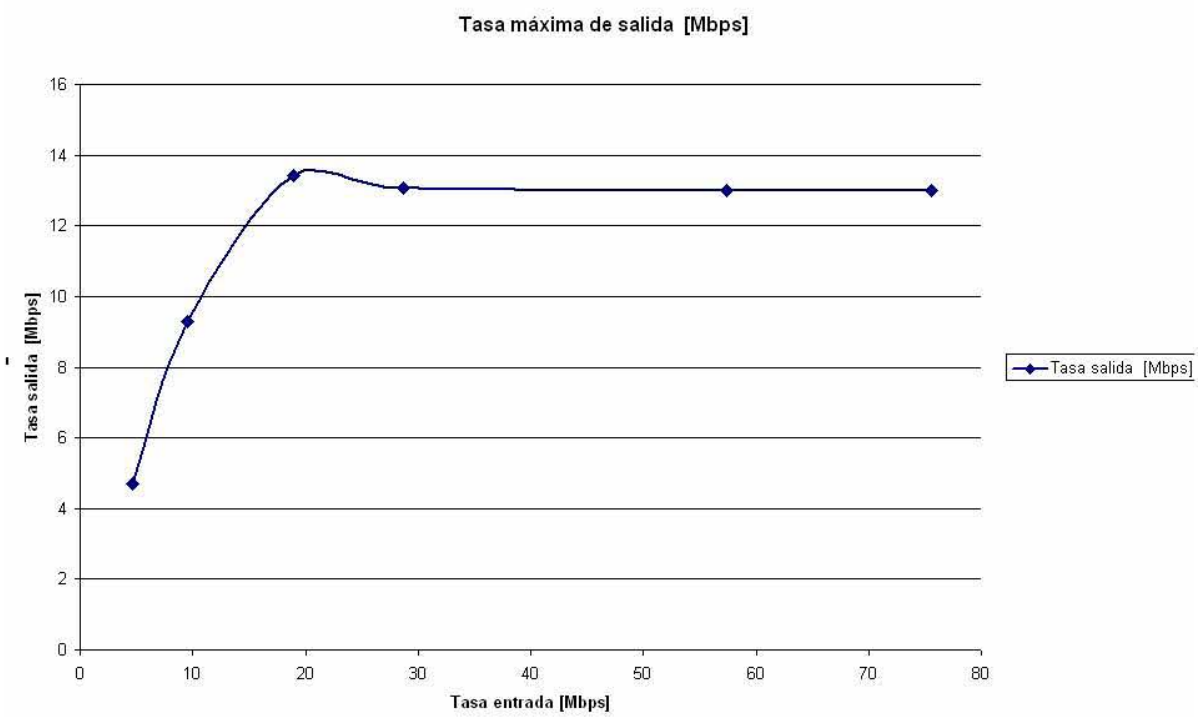

Figura 5.13 Tasa Máxima Promedio a la salida del regulador de tráfico

5.4. Pérdida de paquetes

Si existe una pérdida de paquetes debidas al tamaño de los buffers o colas asignadas a cada tipo de tráfico, y si las colas tienen prioridad entonces es lógico pensar que las pérdidas serán mayores en las de prioridad menor que en las de prioridad mayor, cuando se tengan flujos de tráfico alternado.

En la Figura 5.14 se muestra el porcentaje de paquetes perdidos cuando la tasa de entrada se va incrementando. Donde se observa que para el tráfico 1 (IP-UDP) se tiene un porcentaje menor de pérdida de paquetes, siguiéndole el segundo tipo 2 (IP-TCP), y por último donde hay más pérdidas, el tipo de tráfico 3 (No IP, IP no TCP ni UDP y 802.3). Estos resultados son los que se esperaban, debido a la calendarización del algoritmo *token bucket* correlacionado.

También se nota que a mayor tasa de entrada más pérdida de paquetes se tiene, entonces si el tráfico agregado no sobrepasa la tasa promedio de 13 Mbps no existirá perdida de paquetes.

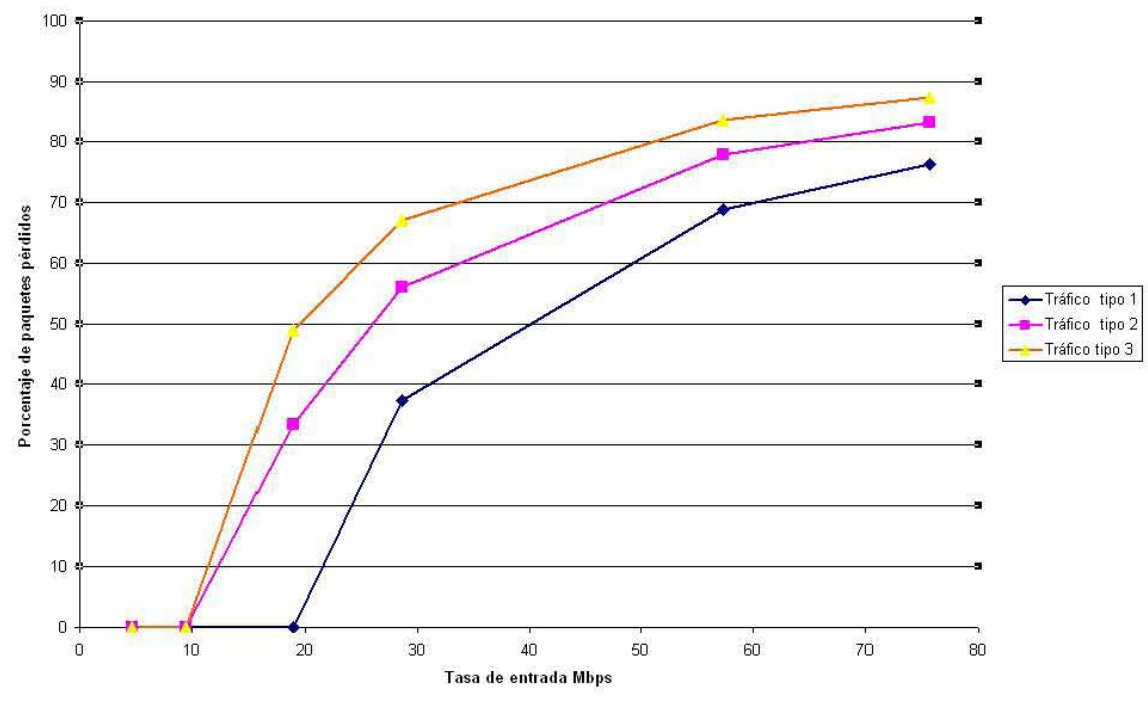

Figura 5.14 % de pérdida de paquetes por tipo de tráfico

5.5. Análisis General

El último experimento es inyectar paquetes de tamaño variable de todos los tipos y analizar su comportamiento con el escenario mostrado en la tabla 5.4.

Escenario	Paquetes	Tamaño [bytes]	Cantidad de paquetes por flujo	Retardo entre paquetes [μs]	Retardo entre flujo [ms]
12	ARP	60	3000	5	80
	IP-TCP	100	3000	5	80
	IP-UDP	150	2500	6	80
	892.3	256	1000	20	80
	IP-TCP	64	3000	5	80
	IP-UDP	64	3000	5	80
	ARP	60	3500	5	80
	IP-TCP	200	2000	10	80
	IP-UDP	200	2000	10	80

Tabla 5.4 Escenario con tamaño de paquetes variable

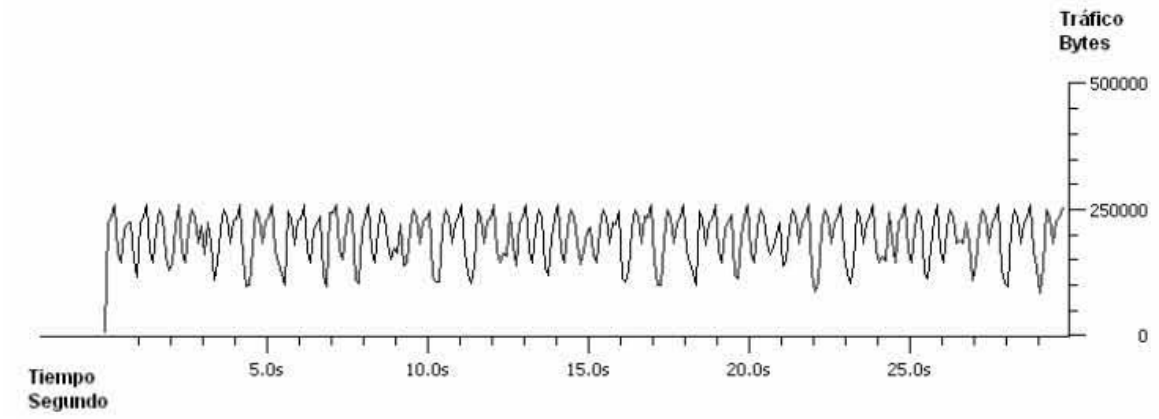

Figura 5.15a Gráfica que muestra la entrada del escenario 12

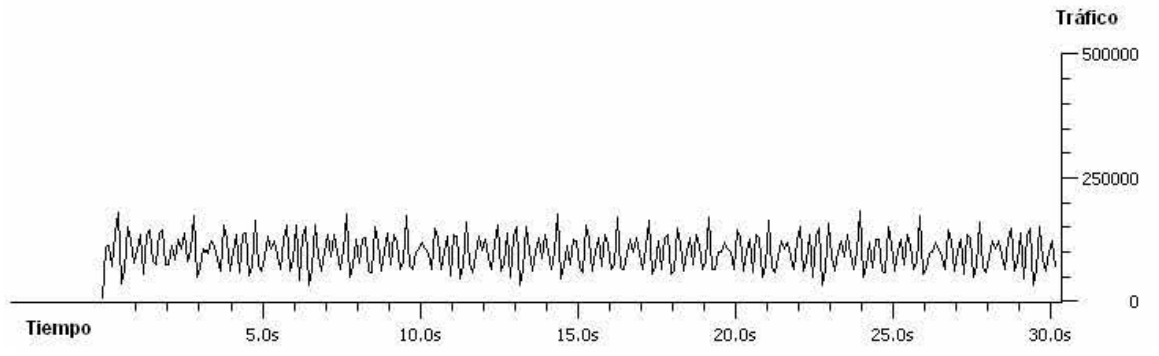

Figura 5.15b Salida escenario 12

Donde la tasa promedio de salida de este escenario es de 7.8 Mbps, que no sobrepasa la tasa máxima promedio de salida obtenida anteriormente y aparte es menor a la de entrada que es de 15.4 Mbps.

La salida del escenario 12 no es tan conformada como la de los primeros (1-11) porque el tamaño variable de paquetes causa pequeñas ráfagas a la salida.

CAPÍTULO 6

En este último capítulo se describen las conclusiones recopiladas durante la elaboración del trabajo. También se mencionan alguna posibles mejoras o modificaciones al sistema de red.

6. Conclusiones

Durante la elaboración de este trabajo surgieron bastantes detalles e inconvenientes para poder llegar hasta este punto. Así que las conclusiones se dividieron en dos aspectos que se refieren a:

- El diseño e implementación
- Evaluación y análisis

En lo que se refiere al diseño e implementación de un sistema de red en un NP se puede decir que tiene su complicación, si se piensa en la administración de los recursos de hardware de acuerdo a las funciones que se pretende realice el sistema. Es decir, para el diseño se tiene que tomar en cuenta la necesidad de conocer el hardware y su funcionamiento para una correcta implementación de los algoritmos planteados, además de aprovechar los beneficios que ofrece la tecnología hibrida de los NP,

Y no se debe olvidar el objetivo principal, el cual es la creación del sistema de red planteado. Se deben escoger o proponer los algoritmos que se crean más convenientes y se ajusten a las necesidades.

En el trabajo no se necesitaba ajustar el sistema a requerimientos específicos, simplemente es una propuesta que puede ser mejorada y usada para el diseño de otros sistemas de Red.

De lo anterior se puede decir que aunque los fabricantes proporcionan códigos y aplicaciones de ejemplo cada diseñador puede optimizar y reutilizar esos recursos siempre y cuando se entienda y conozca tanto el hardware como los algoritmos referentes a las varias funciones de red.

El poder tener una tecnología que sea maleable permite poder crear infinidad de soluciones para el diseño de un sistema de red, debido a que depende de la creatividad e ingenio del diseñador o diseñadores.
En cuanto a la evaluación y análisis del regulador de tráfico se puede concluir lo siguiente:

- El algoritmo *traffic shaping* correlacionado aparentemente funciona de una manera adecuada.
- El sistema se realiza regulaciones por clase de tráfico.
- La calendarización si respeta la prioridad de los tipos de tráfico,
- Uno de los problemas principales del sistema es la perdida de paquetes. La cual se incrementa mientras más grande sea la tasa de entrada,
- La tasa de salida para cada tipo de tráfico se ajusta más a la tasa de llenado de los *buckets*.

6.1. Posibles Mejoras al Regulador de Tráfico

Como posibles mejoras o modificaciones al diseño del sistema se propone lo siguiente:

- El llenado de los *buckets* sea realizado por el procesador núcleo y no por las micromáquinas, para aprovechar ese recurso en expansiones.
- Realizar una pequeña interfaz, que permita modificar los parámetros del llenado de los t*okens* y el tamaño de la cubeta sin necesidad de entrar al código.
- Analizar otros tipos de sincronización y comunicación entre mas micromáquinas que tal vez permitan un mejor rendimiento.
- Analizar la implementación de otro algoritmo de encolamiento.
- Analizar el manejo de más de 3 colas, y si hay otra forma de poder almacenar más paquetes en espera.

Estas mejoras se proponen en base a los resultados obtenidos, los problemas que se presentaron durante la elaboración del trabajo y en trabajos pensados a futuro.

Referencias

[1] Douglas E.. Comer. Network Systems Design Using Network Processors. Prentice-Hall, 2004.

[2] A.S. Tanenbaum, Computer Networks (3rd ed)Prentice-Hall

[3] A.S. Tanenbaum Redes de Computadoras (4ta ed) Prentice-Hall

[4] Erik J. Johnson, Aaron Kunze. IXP1200 Programming. The Microengine Coding Guide for the Intel® IXP1200 Network Processor Family
Ed. Intel Press IXP1200 Programming

[5] Tsern-Huei, Lee, " Correlated Token Bucket Shapers for Multiple Traffic Classes", IEEE, 0-7803-8521-7/04/ 2004

[6] Wenjiang Zhou, Chuang Lin, Yin Li and Zhangxi Tan,"Queue Management for QoS Provision Build on Network Processor", Computer Society, Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems 2003

[7] Sorin Cotofana Stamatis Vassiliadis, Stephan Wong, Network Processors: Issues and Prospectives, in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA-2001) (2001).

[8] Daniel Hedberg , Master Thesis Report," Network Processor based Exchange Terminal – Implementation and evaluation", Department of Microelectronics and Information Technology,Royal Institute of Technology (KTH)

[9] Francis Chang, Wu-chang Feng, Kang Li, OGI School of Science and Engineering at OHSU, Systems Software Laboratory, University of Georgia pp

[10] Niraj Shah. Understanding Network Processors. September 2001. " Approximate Caches for Packet Classification",

[11] Intel "IXP1200 Network Processor Family Hardware Reference Manual", Intel 2000

[12] Intel "IXP1200 Network Processor Programmer's Reference Manual", Intel 2000

[13] Intel Corp, Intel IXA SDK ACE Programming Framework: IXA SDK 2.01 Developer's Guide. CD-ROM. Revision 3.4. Intel Corp. December 2001

[14] Intel Corp., “The IXP1200 Network Processor Datasheet.” Intel Networking and Communications Design Components.Dec. 2001. Intel Corp. 17 May 2002 <http://www.intel.com/design/network/datashts/27829810.pdf>

[15] Intel Corp, “Intel Microengine C Compiler Language Support Reference

[16] Manual.” Intel Networking and Communications Design Components. March 2002. Intel Corp. Feb. 2003.

[17] http://www.lancs.ac.uk/postgrad/asloglou/

[18] http://www.linktionary.com Enciclopedia de Networking.

[19] http://www.mit.edu/~map/Ethernet/Ethernet.txt

[20] Mike Banahan, Declan Brady, Mark Doran . The C Book,1991, Addison Wesley. http://publications.gbdirect.co.uk/c_book/

[21] QOS IP Concepts et Algorithmes
http://www.prism.uvsq.fr/~mea/cours/pdf/dea/RM_qos3.pdf

[22] J. Postel, RFC 0791 - Internet Protocol, September 1981

[23] F. Baker, RFC 1812 - Requirements for IP Version 4 Routers, June 1995

[24] Wroclawski, J., "Specification of the Controlled Load
Quality of Service", RFC 2211, September 1997.

[25] Shenker, S., and J. Wroclawski, "General Characterization, Parameters for Integrated Service Network Elements", RFC 2215, September 1997.

[26] Ghani, Nasir “ATM Traffic Management Consideration for Facilitating BroadBand Acces, IEEE Comunications Magazine, November 1998