UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

FACULTAD DE INGENIERÍA

"*IMPLEMENTACIÓN DE UN SISTEMA DE POLÍTICAS DE TRÁFICO CON EL PROCESADOR DE RED IXP1200*"

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: OSCAR RENÉ VALDEZ CASILLAS

TUTOR:
DR. VICTOR RANGEL LICEA

2006

JURADO ASIGNADO:

Presidente: **Dr. Landeros Ayala Salvador**

Secretario: **Dr. Ramón Gutiérrez Castrejón**

Vocal: **Dr. Víctor Rangel Licea**

1er. Suplente: **Dr. Miguel Moctezuma Flores**

2do. Suplente: **Dr. Javier Gómez Castellanos**

Ciudad Universitaria, México, D.F.

TUTOR DE TESIS:

DR. VÍCTOR RANGEL LICEA

FIRMA

Agradecimientos.

Para llevar a buen término este trabajo de tesis, debo agradecer a:

- Al Consejo Nacional de Ciencia y Tecnología por el apoyo económico brindado.
- A la Facultad de Ingeniería de la U.N.A.M., por recibirme en su comunidad, tanto académica como estudiantil.
- Al Dr. Victor Rangel por su invitación a participar en este proyecto, así como al Dr. Javier Gómez por su apoyo.
- A Karén, que mucho me ha ayudado a seguir adelante.
- A mis padres y mis hermanos, por su apoyo.
- Y a los profesores y compañeros que han estado apoyandome.

Dedicatoria:
Le dedico este trabajo en especial a Karén, a mis padres, mis hermanos, agradeciendo a Dios el poder llegar a este punto.

En especial, quiero dejar un recuerdo a Mabe, que se nos adelanto en el camino, pero que siempre su recuerdo esta entre nosotros. Descansa en paz.

Índice.

Capítulo 1 Introducción.
- 1.1 Definición del problema. 1-1
- 1.2 Objetivo y contribuciones. 1-2
- 1.3 Estructura del documento 1-3

Capítulo 2 Estado del Arte.
- 2.1 Descripción de los sistemas de red. 2-1
- 2.2 ¿Que es un Traffic Policer? 2-4
- 2.3 ¿Porque hay congestión? 2-6
  - 2.3.1 Control de Admisión. 2-8
  - 2.3.2 Políticas de red. 2-9
  - 2.3.3 Algoritmos de bucket. 2-9
- 2.4 ¿Porque procesadores de red? 2-11
- 2.5 Soluciones de Hardware, de software y soluciones híbridas. 2-12
- 2.6 ¿Que son los procesadores de red? 2-12
- 2.7 Conclusiones. 2-19

Capítulo 3 IXP1200.
- 3.1 Características generales de la arquitectura. 3-3
- 3.2 Características del procesador de red de intel. 3-3
- 3.3 IXP en Hardware. 3-5
- 3.4 Características generales de la metodología de programación. 3-7
- 3.5 Conclusiones. 3-8

Capítulo 4. Diseño del traffic policer.
- 4.1. Bloque de entrada o Ethernet. 4-2
- 4.2. Bloque de clasificación. 4-4
- 4.3. Bloque Token Bucket. 4-5
- 4.4. Bloque de encolamiento. 4-7
- 4.5. Bloque de Calendarización. 4-8
- 4.6. Bloque de salida. 4-10
- 4.7. Conclusiones. 4-10

Capítulo 5. Resultados y análisis.
- 5.1. Resultados obtenidos. 5-3
  - 5.1.1. Escenario 1. 5-3
  - 5.1.2. Escenario 2. 5-5
  - 5.1.3. Escenario 3. 5-6
  - 5.1.4. Escenario 4. 5-7
  - 5.1.5. Escenario 5. 5-9
  - 5.1.6. Escenario 6. 5-10

5.1.7. Escenario 7. 5-11
5.2. Conclusiones. 5-20

Capítulo 6 Conclusiones.
6.1 Observaciones finales. 6-1
6.2 Trabajo a futuro 6-3

Índice de figuras

Figura 2-1 ***Traffic Policer*** de políticas suaves. 2-6
Figura 2-2 Idea general de los algoritmos de ***Bucket***. 2-9
Figura 2-3 Esquema de un procesador de red. 2-18
Figura 3-1 Arquitectura de una micromáquina. 3-3
Figura 3-2 Arquitectura del Procesador IXP1200. 3-4
Figura 3-3 Esquema por bloques de la tarjeta ENP-2505. 3-6
Figura 3-4 Tarjeta ENP-2505. 3-7
Figura 4-1 Diagrama a bloques que describe al sistema de red. 4-2
Figura 4-2 Esquema del algoritmo de ***Token Bucket*** en modo ***policer.*** 4-6
Figura 4-3 Esquema Productor – Consumidor. 4-8
Figura 4-4 Diagrama del algoritmo ***Round Robin***. 4-9
Figura 5-1 Esquema con la tarjeta ENP-2505 5-2
Figura 5-2 Conexión PC a PC 5-3
Figura 5-3 Gráfica del tráfico de Pc a Pc 5-4
Figura 5-4 Gráfica del tráfico después de la tarjeta 5-5
Figura 5-5 Gráfica del tráfico de Pc a Pc 5-6
Figura 5-6 Gráfica del tráfico después de la tarjeta 5-6
Figura 5-7 Gráfica del tráfico de Pc a Pc 5-7
Figura 5-8 Gráfica del tráfico después de la tarjeta 5-7
Figura 5-9 Gráfica del tráfico de Pc a Pc 5-8
Figura 5-10 Gráfica del tráfico después de la tarjeta 5-9
Figura 5-11 Gráfica del tráfico de Pc a Pc 5-10
Figura 5-12 Gráfica del tráfico después de la tarjeta 5-10
Figura 5-13 Gráfica del tráfico de Pc a Pc 5-11
Figura 5-14 Gráfica del tráfico después de la tarjeta 5-11
Figura 5-15 Gráfica del tráfico de Pc a Pc 5-13
Figura 5-16 Gráfica del tráfico después de la tarjeta 5-13
Figura 5-17 Gráfica del tráfico de Pc a Pc 5-14
Figura 5-18 Gráfica del tráfico después de la tarjeta 5-14
Figura 5-19 Gráfica del tráfico de Pc a Pc 5-15
Figura 5-20 Gráfica del tráfico después de la tarjeta 5-15
Figura 5-21 Gráfica del tráfico de Pc a Pc 5-16
Figura 5-22 Gráfica del tráfico después de la tarjeta 5-16
Figura 5-23 Gráfica del tráfico de Pc a Pc 5-17
Figura 5-24 Gráfica del tráfico después de la tarjeta 5-17
Figura 5-25 Gráfica del tráfico de Pc a Pc 5-18
Figura 5-26 Gráfica del tráfico después de la tarjeta 5-18
Figura 5-27 Tasa de entrada vs Porcentaje de paquetes perdidos 5-19
Figura 5-28 Tasa de entrada vs Tasa de salida 5-20

Índice de tablas

Tabla 2.1. Comparación entre procesadores de red 2-18
Tabla 3-1. Características de los procesadores IXP 3-4
Tabla 3-2. Características de los puertos externos. 3-4
Tabla 3-3. Características generales de la tarjeta ENP-2505 3-6
Tabla 5-1 Escenario 1 5-3
Tabla 5-2. Escenario 2 5-5
Tabla 5-3. Escenario 3 5-6
Tabla 5-4. Escenario 4 5-8
Tabla 5-5. Escenario 5 5-9
Tabla 5-6. Escenario 6 5-10
Tabla 5-7. Escenario 7 5-12

Resumen.

La congestión en las redes se presenta por exceder la capacidad del medio en que se transmite. Al haber congestión, los recursos destinados para el tráfico de la red se agotan, o en caso de que sea necesario volver a enviar la información el canal se volverá a saturar.

Sin embargo, el manejo en tiempo real de los paquetes requiere de un poder de cómputo capaz de manejar la velocidad del medio y en caso de que haya más de una interfaz de red, debe poder manejar el tráfico agregado que se obtiene.

Para lograr esto, se propone la utilización de un sistema de red llamado ***traffic policer***, que mantiene el tráfico dentro de los límites establecidos, por medio del algoritmo de cubetas de tokens. Este tipo de algoritmo descarta los paquetes que no están dentro de lo establecido, limitando la salida de forma tal que el medio de transmisión no se sature.

El separar el tráfico es también útil, ya que permite aplicar las reglas que se definan en cuanto al tráfico, dando prioridad a un tipo de tráfico y dejando que el otro se retrase, ya que es menos importante.

El hardware utilizado es la tecnología hibrida, conocida como procesadores de red y en este caso el procesador de red ***IXP1200*** de Intel, conjunción de procesador dedicados y diseñados en hardware; y procesador de propósito general, que puede ser programada de forma flexible.

Capítulo 1 Introducción.

1.1 Definición del problema

Al inicio del Internet, su uso solo se limitaba a ambientes académicos y de investigación. Las redes que lo soportaban no eran mucho más rápidas que los procesadores desarrollados hasta ese entonces. En la década de los 90's, cuando el Internet empieza a penetrar en las industrias, comercios y sobre todo, hogares, la demanda de un mejor servicio y las capacidades multimedia que se empezaron a brindar, exigieron un incremento en el uso de las redes ya existentes. La capacidad de estas últimas fue menor a la demanda y los desarrolladores se vieron obligados a mejorar las tasas de transmisión así como la cantidad de datos que se podían transportar. En los últimos 5 años el tráfico de datos se ha doblado en tamaño cada año. A mediados de esta misma década de los 90's, aparecen los dispositivos ópticos, los cuales están diseñados para poder tener la transmisión de decenas de giga bits/seg., en cada enlace físico.

El límite físico de la fibra es alrededor de 100 Tb/s, cuando se usan técnicas de codificación eficientes, pero el equipo que procesa la información enviada en forma de paquetes no ha crecido con la misma rapidez.

Esto en combinación con la necesidad de elementos de red con características más avanzadas, como calidad de servicios, manejo de tráfico y seguridad en las redes ha llevado al desarrollo de nuevas arquitecturas de hardware para el equipo de red. Algunas tecnologías ya son usadas comercialmente, mientras que otras se encuentran en la fase de desarrollo.

En una red, la creciente necesidad de un manejo inteligente del procesamiento de paquetes ha llegado a varias de las aplicaciones basadas en el protocolo IP, como ruteadores con calidad de servicio (***QOS routers***), ***firewall*** y aplicaciones relacionadas con la seguridad.

Hay que tomar en cuenta que los recursos para el procesamiento de paquetes son finitos y si las últimas congestiones de la red fueron lo suficientemente largas, los paquetes pueden perderse por un desbordamiento del buffer o por un retraso excesivo (del tiempo real de los paquetes); esto nos lleva a que si un paquete se pierde en la red, la cantidad de recursos que el paquete había usado se pierde también al mismo tiempo.

Por lo tanto, si se requiere hacer una transmisión con calidad en el servicio que se brinda, las congestiones y pérdidas de paquetes se deben de reducir al mínimo posible. Para esto es necesario que las tecnologías que se están planeando y surgiendo al mercado, sean capaces de manejar la información a una velocidad mucho mayor a la del medio, para que este último no note que hay un sistema de red en medio.

Para poder llevar esto a la práctica se han desarrollado arquitecturas hibridas; tecnologías que combinan la velocidad de los desarrollos hechos en hardware y la flexibilidad de los sistemas programados en procesadores de propósito general. A este tipo de hardware se le conoce como procesador de red, ya que la información que procesa son los paquetes que pasan a través de la red. Un ejemplo de ello es la tecnología ***IXA*** de ***Intel***, que contiene a los procesadores de red ***IXP***.

1.2 Objetivo y contribuciones

El objetivo a alcanzar es la implementación de un sistema de red basándose en esta nueva tecnología en desarrollo llamada ***Network Processors*** o procesadores de red. El fin de este sistema de red es solucionar las congestiones que ocurren en la red, implementando alguno de los métodos diseñados para tal fin.

1.3 Estructura del Documento.

El presente documento tiene la siguiente estructura:

En el capítulo 2 se da el estado del arte tanto de los sistemas de red como de los procesadores en genera, y en particular se hace hincapié en las características que debe de llevar un procesador de red.

En el capítulo 3 se describe el procesador de red que se va a utilizar para el desarrollo de este trabajo, el procesador ***IXP1200***, tanto su arquitectura como la metodología de programación que propone ***Intel***.

En el capítulo 4 se describe la estructura por bloques funcionales del sistema de red propuesto, así como una descripción del funcionamiento del hardware y de los algoritmos utilizados para realizar el ***traffic policer***.

En el capítulo 5 se muestran los resultados obtenidos a partir del planteamiento de varios escenarios para probar el sistema de red implementado.

El capítulo 6 muestra las conclusiones a las que se llega en este trabajo, así como las mejoras posibles y el trabajo a futuro que se puede realizar.

Capítulo 2 Estado del arte

2.1 Descripción de los sistemas de red.

Se usará el mismo concepto acerca de los Sistemas de red que contempla Comer [4]: "***el término sistema de red (network system) se usará para hacer referencia a un componente electrónico que maneja datos. Dado que las redes de computadoras están compuestas por diferentes componentes electrónicos, solo nos interesarán aquellos que procesan paquetes***."

De esta forma, dispositivos tales como el repetidor, el hub o el modem no serán de interés, dado que operan a nivel de señal eléctrica, o capa física del modelo OSI, y no a nivel de ***frames*** (tramas) o paquetes.

Así, algunos ejemplos de los sistemas de red serían:

- ***Bridge*** (puente). Es un sistema de red que conecta dos redes individuales y reenvía paquetes entre ellas. Los puentes examinan direcciones de la capa de enlace (capa 2) para dar a los datos el camino a seguir. Para ser precisos, un puente reenvía los ***frames***. A menudo son usados para conectar redes Ethernet.

- ***Switch*** (conmutador). Sistema de red que conecta dos o más computadoras y reenvía los frames entre ellas. Los ***switches*** de red también son llamados ***switches*** de capa 2.

Conceptualmente, un ***switch*** puede verse como un conjunto de ***bridges*** entre un conjunto de computadoras. Las conexiones entre las computadoras y el ***switch*** asemejan las conexiones entre las computadoras y un hub, pero con mayor rendimiento. Como con los ***bridges***, los ***switches*** son usados a menudo para conectar redes Ethernet.

- VLAN ***Switch*** (Conmutador de red virtual): Es un ***switch*** de capa 2 que emula múltiples segmentos de red. Como un ***switch*** convencional, un ***switch*** VLAN conecta múltiples computadoras y reenvía los frames entre ellas.

Aunque, desde el punto de vista del procesamiento de paquetes, también existen en Internet otros sistemas de red que los procesan, como son:

- ***Firewall*** o corta fuegos. Sistema que provee seguridad en la red de acuerdo a las políticas dadas por el administrador. Un ***firewall*** típico verifica los accesos para prevenir entradas no autorizadas de acuerdo con las políticas establecidas, boqueando los paquetes entrantes que no cumplen con ellas. Aunque es posible implementarlo de forma interna dentro de un ruteador o en una computadora dentro de su sistema operativo, también se puede manejar como un sistema por separado.

- ***Virtual Private Network*** (VPN, Red Privad virtual): Un sistema que utiliza cifrado para proveer comunicación a través de Internet. Los sistemas VPN son usados en pares, son necesarios dos sitios que instalan un sistema de este tipo entre ellos e Internet y son configurados para poder conocer uno la existencia del otro. Una vez establecido, las computadoras se comunican entre si de forma privada. Esto es, si se llega a interceptar los paquetes que viajan a través del Internet entre los dos sitios, no será posible decodificar o interpretar el contenido de los paquetes.

- Traductor de direcciones de Red (***Network Adress Translation, NAT***). Permite a diversas computadoras de un solo sitio compartir una sola IP pública valida. El sistema NAT es colocado entre el sitio y el Internet, renombrando a los paquetes para cambiar las direcciones y alguna otra información en las cabeceras.

- Balanceador de Carga. Otro sistema usado en los sitios ***web*** de gran tamaño para incrementar el desempeño y la disponibilidad del mismo al permitir a múltiples computadoras ejecutar una copia del servidor ***web***. Un balanceador de carga acepta los paquetes entrantes y los envía a la copia del servidor que tiene una carga menor. Los términos ***switch*** de capa 4+ y ***switch*** de capa 7 son sinónimos del balanceador de carga.

- ***Set-Top Box***. Usada dentro del sistema de entretenimiento digital, permite a un usuario seleccionar un programa transmitido a todos los usuarios o uno de contenido personalizado. Este sistema se comunica con el proveedor, este cifra el flujo de datos y lo envía al sistema de audio/video local.

Existen sistemas de monitoreo y control de paquetes, reservados para propósitos especiales y que no son del todo implementados. Algunos ejemplos serían

- Monitor de tráfico (***Traffic Monitor***): Puede medir el tráfico pico y el tráfico promedio desde o hacia un conjunto de destinos.

- Políticas de red (***Traffic Policer***): Es usado para descartar el tráfico que excede un predeterminado límite de acuerdo a las especificaciones dadas por el administrador de red.

- Regulador de tráfico o formateador de tráfico (***Traffic Shaper***). Retrasa algunos paquetes y permite a otros el paso de forma rápida de acuerdo a un conjunto de especificaciones. Algunos ruteadores ofrecen este sistema.

- Analizador de paquetes (***Packet Analyzer***): Este sistema esta dentro de la red y captura copias de los paquetes para realizar mediciones del desempeño de la red. Los analizadores de paquetes son usados para detectar problemas y medir la carga que hay en la red.

Sin embargo, la existencia de los sistemas de monitoreo y control se da por la necesidad de regular el tráfico, y prevenir que en la red ocurran demasiadas congestiones, gastando los recursos con los cuales cuenta la red. Algunos de esos sistemas se integran dentro de los mencionados anteriormente, como es el ***firewall***, o los ruteadores.

2.2 ¿Que es un Traffic Policer?

Los sistemas de red como son los analizadores de trafico realizan una ***medición del trafico*** básica. Para hacerlo, estos sistemas obtienen una copia de cada trama que atraviesa la red, examina el contenido de la trama y actualiza contadores y otro tipo de información estadística. Por ejemplo, la medida del tráfico puede producir una cuenta de paquetes, un número promedio de paquetes por unidad de tiempo, un estimado de la actual utilización de la red, el porcentaje de paquetes de ***broadcast***, el número de ***frames*** que llevan mensajes IP, o la duración promedio de una conexión TCP.

La medición del tráfico a menudo cae dentro del concepto de ***Service level agreements (SLAs)*** o acuerdos de nivel de servicio. Un ***SLAs*** es un contrato legal entre dos entidades en las cuales una acuerda proveer el uso de servicios de red a otra. Por ejemplo, un ISP (***Internet Service Provider***, Proveedor de servicios de internet) puede ofrecer a una corporación la conexión a Internet. El ***SLA*** puede especificar el máximo de transmisión y la tasa promedio, o el número de bytes de datos que pueden ser transferidos cada mes.

Cada participante en el ***SLA*** puede ser beneficiado en la medición del tráfico. Un cliente puede beneficiarse debido a que el costo del servicio es relativo a las necesidades que se tengan. Entonces, un cliente al que se le mide el tráfico puede determinar la categoría del servicio dentro del cual la cuenta actual se localiza. Si la categoría anterior es menos costosa, y el servicio que se proporciona es suficiente, entonces se puede renegociar el acuerdo. El proveedor puede usar la medida del tráfico para monitorear el uso del consumidor. Si el uso del consumidor se incrementa, el proveedor puede convencer al cliente de que contrate una categoría más alta de servicio.

El concepto de ***traffic policing*** esta relacionado con el concepto de medida de tráfico y más con el concepto general de ***administración de tráfico***. Como su nombre lo implica, el sistema de red de políticas de trafico (***traffic policing***) se refiere a la ejecución de una actividad en la cual si el trafico excede los limites especificados, es marcado como un candidato para ser descartado o es explícitamente desechado. Las políticas requieren que el tráfico sea medido debido a que la ejecución de estas determina cual tráfico cae dentro de una categoría especificada con ciertos parámetros. También requiere de medidas de grano mas fino que las que se usan para hacer la

facturación por uso del servicio, debido a que están diseñadas para controlar tipos específicos de tráfico. Por ejemplo, se puede necesitar separar las medidas para cada conexión TCP o para el tráfico destinado a un puerto en específico de un protocolo más que la suma de ambos.

Los proveedores usan las políticas de tráfico para asegurar que los clientes no obtengan mas servicio del que se tiene contratado. Por ejemplo, considérese un cliente que realizo el pago de un tasa de transmisión de 1Mbps, y trata de transmitir trafico a 2 Mbps. El proveedor puede usar un sistema de este tipo para tirar todos los paquetes del cliente que excedan 1Mbps en promedio en un corto periodo de tiempo.

El ***Traffic-policing*** permite examinar el flujo de tráfico del cliente y marcar o descartar los paquetes que excedan los ***SLA***s. La función del traffic-policing usa el algoritmo de token bucket, pero la cola de paquetes es reemplazada com un descartador de paquetes o una función de marcado de paquetes. Si la función de policing determina que un paquete en particular esta dentro del perfil, el paquete es admitido dentro de la red.[8]

Si la función de policing determina que el paquete esta fuera del perfil, el paquete es descartado inmediatamente (politica dura, ***hard policing***) o admitido a la red pero marcado como fuera del perfil (politica suave, ***soft policing***).[8]

Marcando el tráfico como fuera del perfil, permite que los paquetes que están dentro del perfil sean manejados de forma diferente a los paquetes que estén fuera del perfil. Por ejemplo, se puede configurar un ruteador de acceso para marcar un paquete y cambiar su tiempo de vida, de tal forma que el siguiente ruteador lo considere descartar primero durante los periodos de congestión mientras sigue manejando el tráfico que está dentro del perfil

Un aspecto importante al realizar el diseño de estos sistemas de políticas de tráfico es la velocidad. Las decisiones que se deben de tomar deben ser hechas de manera rápida debido a que el sistema necesita decidir como manejar los paquetes en tiempo real. Esto es, el sistema de políticas no puede gastar demasiado tiempo en tomar una decisión.

Se conoce que una operación recae en la ruta crítica si la operación debe de ser hecha en tiempo real. Las operaciones del sistema de políticas de tráfico recaen en ruta crítica debido a que el sistema debe terminar antes de que la disposición del paquete sea conocida.

Por lo que el ***traffic policer*** es un mecanismo para asegurar que el flujo de paquetes entrante no exceda un ancho de banda ya definido. La parte que exceda el ancho de banda es marcada (como de baja prioridad) o descartada. Figura 2-1.

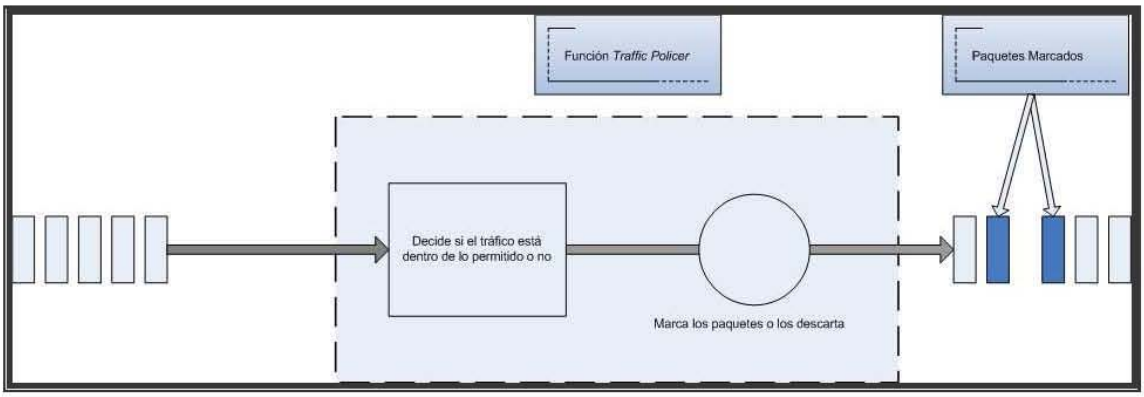

Figura 2-1 *Traffic Policer* de políticas suaves.

2.3 ¿Porque hay congestión?

La congestión puede ocurrir por varias razones. Si de manera repentina comienzan a llegar cadenas de paquetes por tres o cuatro líneas de entrada y todas necesitan la misma línea de salida, se generará una cola. Si no hay suficiente espacio para ***buffers*** en memoria, dado que el espacio que se tiene es finito, y si las últimas congestiones han sido lo suficientemente largas, los paquetes se pueden empezar a perder debido a un desbordamiento del ***buffer*** o por un retraso excesivo.

Los procesadores lentos también pueden causar congestión. Si los procesadores de los ruteadores son lentos para procesar los paquetes y colocarlos en los ***buffers*** correspondientes, administrar las tablas de ruteo, y otras tareas de administración requeridas, los paquetes se encolaran por un tiempo excesivo, aún cuando la línea tenga la capacidad suficiente. El caso contrario (un ancho de banda pequeño y bastante poder de procesamiento) también propicia este problema.

También dentro de una red existe una cierta cantidad de recursos, y si un paquete se llega a perder, entonces los recursos asignados a ese paquete se pierde al mismo tiempo. Otra forma de visualizar el problema que se presenta con la congestión, es que si existen demasiados paquetes en una parte de la red, entonces su desempeño se reduce considerablemente. Sin embargo, la capa de red provee un control de congestión para asegurar envío correcto de los paquetes de la fuente hasta su destino.

En este caso, el control de congestión es parte de la red en su conjunto, al contrario del control de flujo, en el cual solo intervienen los ***hosts*** que están realizando la transmisión y recepción de los paquetes.

Se puede dar una clasificación acerca del tipo de control de congestión, y se divide en dos grandes grupos: métodos de ciclo abierto o preventivo, en los cuales no hay realimentación de la red o del host destino, procurando que con un buen diseño se pueda asegurar que no ocurra la congestión. Dentro de las soluciones de ciclo abierto se incluye el poder decidir cuando se acepta de nuevo tráfico, cuando se descartan paquetes, cuales y como tomar las decisiones de calendarización dentro de la red.

Existen también los métodos de ciclo cerrado o reactivos, dentro de los cuales hay una realimentación explicita o implícita desde la red o el destino. Las soluciones de este tipo se basan principalmente en tres partes:

- Monitorear el sistema para detectar cuándo y dónde ocurren congestiones
- Pasar esta información a lugares en los que pueda llevarse a cabo acciones.
- Ajustar la operación del sistema para corregir el problema.

Dentro de los métodos para controlar la congestión de una red se encuentran los siguientes:

- Control de Admisión
- Políticas de tráfico
- Control de Flujo

En el control de congestión de ciclo abierto se puede tener los siguientes métodos:

-Control de Admisión y políticas de tráfico
-Algoritmos de bucket

y en el control de congestión de ciclo cerrado:

-Control de la ventana de Flujo: Control de flujo TCP
-Control de tasa de Flujo: tasa ABR para redes ATM

Interesándonos en los métodos de ciclo abierto, se realiza una breve descripción de cada uno.

2.3.1 Control de Admisión

Admite un nuevo flujo de paquetes (nueva conexión) solo cuando la red puede manejarlos adecuadamente. Se considera más adecuado para redes de circuito virtual. El usuario provee un conjunto de reglas de tráfico (pico máximo de la tasa de transmisión, promedio de la tasa de transmisión, máximo retardo permitido, etc.) durante la fase de configuración de la conexión; así la red permite al usuario acceder solo si tiene los recursos suficientes disponibles.
De otra forma, la petición de conexión será negada.

Una vez que se tiene esto, la red monitorea el flujo de tráfico para verificar si los usuarios obedecen las reglas de tráfico o hace que el usuario la cumpla aplicando las políticas de red

2.3.2 Políticas de red.

Se establece de antemano las políticas a seguir de a cuerdo a las reglas que se han establecido o los acuerdos dados. Si el flujo de paquete no obedece las reglas ya establecidas, verifica si el flujo de paquetes obedece las reglas, y si no lo hace lo "castiga" de dos formas:

-Tira los paquetes que no obedecen las reglas
-Les da una baja prioridad.

2.3.3 Algoritmos de *bucket*

Los algoritmos de Leaky y Token Bucket son ampliamente usados como técnicas para aplicar las políticas de red.

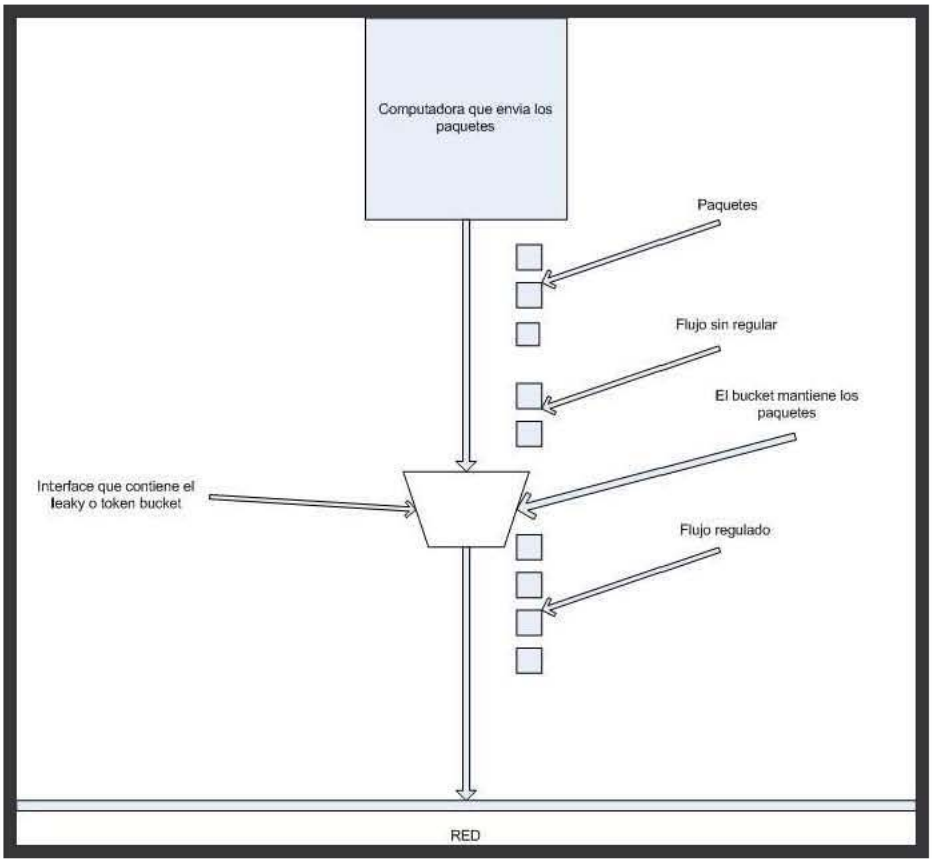

Figura 2-2 Idea general de los algoritmos de Bucket.

Leaky bucket. El host puede enviar ráfagas que son almacenadas en un buffer de la interfaz, la que envía a la red un flujo constante de salida. Si la ráfaga es de tal intensidad o duración que el buffer se llena, los paquetes excedentes son descartados, o bien son enviados a la red con una marca especial que les identifica como de segunda clase. Estos paquetes serán los primeros candidatos a ser descartados en caso de congestión.

El algoritmo ***token bucket*** es una versión mejorada del anterior que compensa al ***host*** que alterna intervalos de tráfico con otros de inactividad frente al que esta siempre transmitiendo.

El algoritmo puede ser entendido conceptualmente como sigue:

- Un ***token*** es agregado al ***bucket*** cada ***1 / r*** segundos, donde ***r*** es la tasa de llenado del ***bucket***.
- El ***bucket*** puede tener al menos ***b tokens***. Si un ***token*** llega cuando el ***bucket*** esta lleno, es descartado.
- Cuando un paquete (PDU de capa de red) de ***n*** bytes llega, ***n tokens*** son removidos del ***bucket***, y el paquete es enviado a la red.
- Si hay menos de ***n tokens*** disponibles, no se remueven ***tokens*** del ***bucket***, y el paquete es considerado como ***no conformable***.

El algoritmo permite ráfagas arriba de ***b*** bytes, pero después de ejecutarse por un tiempo, la salida de los paquetes conformables es limitada a tasa constante ***r***. Los paquetes no conformables pueden ser tratados de diferentes formas:

- Pueden ser desechados
- Pueden ser encolados para su transmisión posterior cuando suficientes ***tokens*** se han acumulado en el ***bucket***.
- Pueden ser transmitidos, pero marcados como no conformables, para que posiblemente sean desechados si la red se satura.

El mecanismo que sigue para ello es el siguiente: cuando el ***host*** no envía datos el ***bucket*** va sumando ***tokens*** hasta un máximo igual a la capacidad del ***buffer***. Los ***tokens*** acumulados pueden utilizarse después para enviar ráfagas con un flujo ***M*** mayor de lo normal. Cuando se agotan los créditos, el flujo vuelve a su valor normal ***r*** y el algoritmo funciona como un leaky ***bucket***.

Los parámetros que definen este algoritmo son el flujo normal ***r***, la capacidad del buffer ***C*** y el flujo máximo ***M*** que normalmente igual a la velocidad de la interfaz física.

Una vez que la conexión es aceptada, la red puede brindar ***QoS*** (Calidad de servicio) mientras que el tráfico siga las características especificadas en las reglas de tráfico.

Los parámetros de las reglas de tráfico y ***QoS*** son un enlace entre dos partes, las aplicaciones y la red, y estos parámetros se deben de respetar por ambas partes. Al realizar las políticas de red se monitorea el tráfico para averiguar si sigue las características previstas por las reglas de tráfico. En general suceden tres situaciones:

1. El tráfico que viole las reglas establecidas es manejado de forma apropiada:
   - Se descarta
   - Es marcado como de baja prioridad: en un punto de congestiones más vulnerable a ser descartado
   - Es reformado de acuerdo a las políticas.

2. Se realizan métricas
   - Son medidas las características en tiempo real del tráfico

3. Es marcado el tráfico
   - El tráfico es comparado con las reglas de tráfico y es marcado o descartado

2.4 ¿Porque procesadores de red?

Con la rápida adopción del Internet para el comercio electrónico y los procesos de negocios, ha aparecido un incremento en la demanda de servicios inteligentes. Los proveedores de servicio buscan tener la habilidad para ofrecer, de manera sencilla, nuevos tipos de servicios basados en una variedad de mecanismos como la prioridad del tráfico, las redes privadas virtuales, y el balanceo de carga. Tales servicios necesitan ser lo suficientemente flexibles para ofrecer nuevas características y funcionalidad sin requerir de una reconstrucción total de la infraestructura de red. Los proveedores de equipos de red reconocen la necesidad de tal flexibilidad. Debido a

esto se ha incrementado el traslado hacia plataformas programables que extiendan el tiempo de vida de sus productos y que permitan una rápida introducción de nuevos servicios. [11]

2.5 Soluciones de Hardware, de software y soluciones híbridas.

Una de las preguntas fundamentales alrededor del diseño de sistemas de redes se centra en la discusión software o hardware: la pregunta consiste en si los sistemas de red consistirán de hardware convencional con todo el procesamiento codificado en software, o deberán ser sistemas construidos por completo en hardware de propósito especial. De un lado, las soluciones de software ofrecen flexibilidad - el software puede ser modificado o actualizado con un costo mínimo. El sistema básico puede sobrevivir a los cambios mayores (p.e. un cambio en el protocolo). Además, la velocidad del hardware sobre el cual esta programado se puede incrementar sin requerir nuevo software. Del otro lado, las soluciones específicas de hardware ofrecen una alta velocidad - hardware de propósito especial puede ser construido para cada función. Las rutas internas de los datos pueden ser diseñadas para mover los paquetes entre los dispositivos de entrada y salida y la memoria sin retraso, y la arquitectura puede ser diseñada para evitar los cuellos de botella de un CPU convencional.

Desafortunadamente, los diseños en hardware o software tienen sus desventajas. Los diseños en software tiene un bajo desempeño; los diseños en hardware son inflexibles y costosos para modificar o actualizar. Consecuentemente los diseñadores de hardware tiene un compromiso: un dispositivo conocido como procesador de red (***network processor***). Como un CPU convencional, un procesador de red puede ser programado. Pero al contrario de un procesador convencional, un procesador de red ha sido optimizado par el procesamiento de paquetes. La meta del procesador de red es una tecnología que combine las ventajas de la velocidad de un diseño en hardware específico y la flexibilidad de la implementación del software.

2.6 ¿Que son los procesadores de red?

Como ya se ha mencionado, el desarrollo del Internet requiere de soluciones cada vez mayores. Dentro de estas soluciones se encuentra la forma en que los paquetes son tratados. Para ello se ha desarrollado una nueva tecnología llamada procesadores de red.

Para empezar a entender que es un procesador de red, habrá primero que conocer que es un procesador y que tipos hay.

En los últimos años, debido a que la cantidad de datos se ha estado incrementando, y los protocolos se han vuelto más dinámicos y sofisticados, además de que los protocolos se han ido introduciendo más rápidamente, se hace necesario el uso de una tecnología que sea flexible.

Actualmente los elementos de procesamiento que existen se pueden clasificar en:

- **Procesadores de propósito general, *GPP***

Procesadores programables, pero no optimizados para aplicaciones de red

En muchas terminales, especialmente en las computadoras de escritorio y portátiles, los procesadores de propósito general son usados para el procesamiento de protocolos. Estos procesadores también están presentes en los primeros ruteadores y en algunas ocasiones en los sistemas embebidos. Los procesadores de propósito general, como su nombre lo indica, tienen la capacidad para manejar cualquier tipo de protocolo, pero lo hacen de forma ineficiente dado a que hay una sobrecarga de control sobre las instrucciones, desde que son buscadas (***fetch***) hasta que son decodificadas. También existe una sobrecarga asociada con las operaciones con los encabezados que son más pequeños que la longitud de palabra que maneja el CPU. También llegan a presentar problemas con el procesamiento en tiempo real debido al manejo de interrupciones, la jerarquía de ***cache*** y el sistema operativo y normalmente se necesita enviar al buffer los paquetes más de una vez. Haciendo referencia al modelo de red OSI, estos procesadores son usados en general para procesamiento de las capas 3 en adelante, ya que las 2 capas anteriores requieren procesamiento en tiempo real.

- **Circuitos integrados de propósito especifico ASIC(*Application Specific Integrated Circuit*)**

Procesadores de alta capacidad de procesamiento, pero con mucho tiempo invertido en su desarrollo, careciendo de flexibilidad.

Los ***ASIC*** son lo totalmente opuesto a los procesadores de propósito general. Los procesadores ***ASIC*** son diseñados para un solo protocolo y lo manejan de manera muy eficiente. El gran problema es que se pierde flexibilidad, sin embargo, este tipo de procesadores han tenido gran éxito con algunos protocolos, por ejemplo, ***Ethernet***. Estos procesadores operan de manera general sobre un flujo de datos y no sufren la sobrecarga de almacenar en buffer parte de los datos.

- **Computadora de conjunto especifico de instrucciones (*Application Specific Instruction Set Computer*).**

Las computadoras de este tipo a menudo son llamadas procesadores con un conjunto específico de instrucciones para una aplicación (***Application Specific Instruction Set Processors, ASIPs***). La idea con un procesador de este tipo es diseñar un conjunto de instrucciones que corresponda a una aplicación. Esto da una alta flexibilidad y al mismo tiempo, las operaciones más importantes pueden tener una sola instrucción y, por lo tanto, ejecutarse eficientemente. En los procesadores ***ASIP*** se utiliza el estándar de la arquitectura de von Neuman. Para encontrar el conjunto de instrucciones correcto, se puede usar un perfilador de operaciones. Esto significa que los algoritmos son ejecutados en un simulador y las estadísticas obtenidas son conjuntadas para saber de que forma se esta utilizando cada operación. Al terminar este procedimiento, se debe de saber cuales operaciones deben llevar instrucciones especializadas y cuales requieren de toda una secuencia de instrucciones. Si alguna secuencia de operaciones aparece de manera frecuente en la ejecución, se puede considerar el agrupar esa secuencia de operaciones en una sola instrucción.

El numero de operaciones requeridas para una cierta tarea puede ser diferente tanto, entre a mayoría de los ***ASIP***s y los procesadores de propósito general, como entre dos ***ASIP*** diferentes. Sin embargo, la medida tradicional de la potencia de computo, millones de instrucciones por segundo (***MIPS***), puede que no de una visión suficiente del desempeño cuando se comparan dos procesadores de red.

- ***RISC* con Conjunto de Instrucciones Optimizado.**

Los ***RISC*** (Computadora con conjunto de instrucciones reducidas) con un conjunto de operaciones optimizados es similar a lo ***ASIP***. La gran diferencia es que se toma como base el conjunto de instrucciones ***RISC*** de la arquitectura y algunas otras instrucciones son agregadas al núcleo de este conjunto de instrucciones. Esto simplifica tanto el desarrollo de la arquitectura, como de las herramientas de soporte, como los compiladores. Sin embargo, esto puede resultar en que sean implementadas algunas funcionalidades que sean innecesarias para el procesamiento de los datos de la red. Este acercamiento puede ser visto como una mezcla entre los procesadores de propósito general y los ***ASIP***.

Las instrucciones agregadas pueden ser implementadas de diferentes formas:

-***Aceleración de Instrucciones.*** Significa que no se agrega nuevas instrucciones, solo las instrucciones que son usadas con mayor frecuencia son implementadas de manera mas eficiente.

-***Extensión de la ruta de los datos.*** La ruta de los datos se extiende con nuevos bloques para manejar las instrucciones agregadas. La ejecución sigue siendo controlada por el conjunto ***RISC*** original. Los procesadores del tipo una instrucción sobre múltiple datos (***Single Instruction – Multiple Data, SIMD***) caen en esta categoría

-***Extensión de procesador esclavo.*** En este caso las instrucciones agregadas son ejecutadas en un procesador secundario o esclavo, haciendo que el procesador ***RISC*** modificado sea una maquina que ejecuta el paradigma de múltiples instrucciones sobre múltiples datos (***Multiple Instruction – Multiple Data, MIMD***). Esto es, las nuevas instrucciones pueden ser ejecutadas en paralelo con el conjunto de instrucciones ***RISC*** original y el control de las rutas del procesador puede ser rediseñado.

- **Arquitecturas de Hardware reprogramables.**

Las arquitecturas de hardware reprogramables han sido usadas exitosamente en muchos tipos de aplicaciones. La gran ventaja sobre los procesadores tradicionales es que permiten mucho más procesamiento paralelo, casi como un ***ASIC***. La diferencia esta en que tienen la flexibilidad que en los ***ASIC***s se pierde. Dos tipos generales de arquitecturas de este tipo están disponibles; los dispositivos lógicos programables (***Programmable Logic Devices, PLD***), los cuales realizan la

suma de productos de manera funcional y los arreglos de campos de compuertas programables (***Field Programmable Gate Arrays, FPGA***s), los cuales tienen una pequeñas tablas de avistamiento que permiten cualquier función lógica. Los ***PLD***s son usados para implementar maquinas de estado finito, las cuales son normalmente usadas en la ruta de control de una arquitectura y las ***FPGA***s son usadas para implementar las rutas de los datos.

Las arquitecturas reprogramables pueden ser usadas para una aplicación mas especifica. Este tipo de arquitecturas sacrifican algo de flexibilidad para lograr un mejor desempeño y un bajo consumo de potencia.

Sin embargo, cada uno de los procesadores mencionados anteriormente tiene sus limitaciones, sobretodo si hablamos de flexibilidad y desempeño. Los procesadores ***ASIC*** son los menos flexibles, ya que son los que están desarrollados para soluciones de hardware, pero ofrecen un desempeño muy alto. En la orilla opuesta se encuentran los procesadores ***GPP***, que son los más flexibles pero con la desventaja de un bajo desempeño.

- **Procesadores de red (*NP, Network Processor*)**

Un procesador de red es un ***ASIP*** para aplicaciones de red - un dispositivo programable con características en su arquitectura o circuiteria especiales para el procesamiento de paquetes. Mientras que los procesadores no cubren todas las soluciones para las aplicaciones de red, se cree que estros procesadores de red cubrirán la parte más interesante y de alto crecimiento. Se da una definición amplia con intención de reflejar el amplio rango de las arquitecturas programables propuestas para el procesamiento de red. Como resultado, los procesadores de red comparten características con muchas otras opciones de implementación:

- co-procesadores
- procesadores de comunicaciones usados para aplicaciones de red
- reconfigurable a nivel de Hardware. (p. e. ***FPGA***)
- El procesador de propósito general (***GPP***) es usado para tareas de enrutamiento.

Al comparar las implementaciones de los sistemas basados en los diferentes tipos de procesadores, se observa que los procesadores ***ASIP*** son el mejor acercamiento a la mayoría de las implementaciones de sistemas de red. Un ***ASIP*** para redes, o ***Network Processor*** (Procesador de red, NP), provee el balance correcto entre hardware y software, para llegar a cumplir con los siguientes requerimientos:

- Desempeño: Al ejecutar las principales rutinas de cómputo en hardware, los NPs pueden ejecutar diferentes aplicaciones a la velocidad del medio.

- Flexibilidad: Teniendo software como la mayor parte del sistema, permite al equipo de red la rápida adaptación a los cambios de estándares y aplicaciones.

- Rápida salida al mercado: El diseño de software es mucho más rápido (y barato) que el diseño del hardware con la funcionalidad equivalente.

- Potencia: Mientras que los NPs no pueden estar en dispositivos sensitivos a la energía (como serían handhelds), su consumo de potencia es importante por razones de costo.

- Lograr un alto desempeño en el procesamiento; tienen flexibilidad para la programación y son más económicos que un procesador de propósito general.

Los procesadores de red son de diferente naturaleza, algunos son productos comerciales, otros son proyectos en etapa de investigación aun y solo algunos han sido anunciados recientemente, pero no hay mucha información disponible

Aparte de la arquitectura de hardware del procesador de red, también el soporte para programación es muy importante. Algunos procesadores de red tiene compiladores para C y C++, otros tiene soporte para un tipo especial de lenguaje de programación de red y algunos otros son programados usando micro código de bajo nivel.

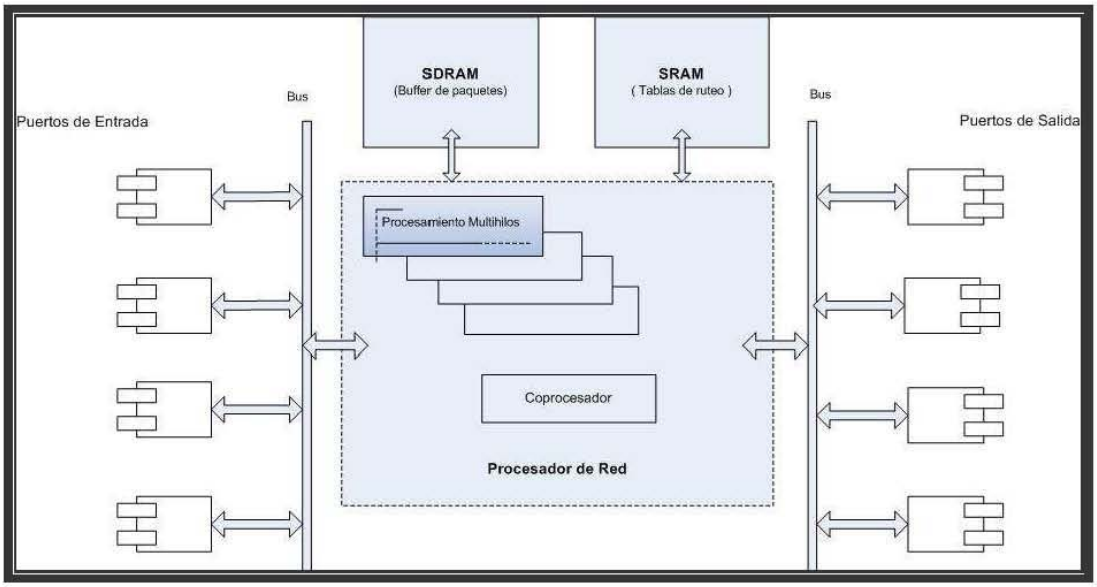

Figura 2-3 Esquema de un procesador de red.

En el mercado hay más de un tipo de procesadores de red, y de acuerdo con lo visto anteriormente, se puede dar una clasificación aproximada de ellos en la siguiente tabla:

Tabla 2.1: Comparación entre procesadores de red

PROCESADOR	Aspecto del Sistema	Aplicaciones	Arquitectura
Motorota C-5	Ruteador	Capas 2-7	RISC + ASIP
Intel IXP 1200	**Ruteador**	**Capas 2-7**	**RISC + ASIP**
Agere FPP, RSP, ASI	Tarjeta	Ruta de datos	ASIP + ASIC
Agere NP10 y TM10	Tarjeta	Ruta de datos	ASIP + ASIC
AMCC nP7120	Tarjeta	Capas 3	RISC + ASIC
AMCC nP7250	Tarjeta	Capas 3	
AMCC nP3400	Ruteador	Capas 3-7	RISC
IBM PowerNP 4GS3	Ruteador	Capas 2-4	RISC+ASIP+ASIC
Silicon Access iAP	Tarjeta	Address Lookup	ASIC + ALU
SwitchCore CXE-16	Ruteador	Capas 2-4	ASIC
Coresma 6001	Terminal	Capas 2-3	RISC
PMC-Sierra PM7388	Tarjeta	Capas 2	ASIC
PMC-SierraPM2329	Tarjeta	Clasificación	ASIP + ASIC
Broadcom BCM 5680	Ruteador	Capas 2-7	ASIP + ASIC
Broadcom BCM 5632	Ruteador	Capas 2	ASIC
Broadcom BCM 1250	Ruteador	Capas 4-7	Propósito general

Solidum Systems PAX1100	Tarjeta	Clasificación	Reconfigurable
Sitera (Vitesse) IQ 2000	Ruteador	Capas 2-7	ASIP + ASIC
Xelerated X40	Tarjeta	Capas 2-4	ASIP
Lantronix DSTini	Terminal	Capas 2-4	Propósito general
ClearwaterNetworks	Ruteador	Ruta de control	RISC
ClearSpeed Platform	Tarjeta	Clasificación+forwarding	SIMD
Lexra LX8000	Tarjeta	Capas 3-7	RISC
STM Network Pro-Cessors			ASIP + ASIC
Infineon + Dresden University	Ruteador/Terminal	Capas 2-4	RISC

2.7 Conclusiones.

El hardware que maneje un sistema de red deberá ser capaz de manejar los paquetes entrantes, aplicar las políticas establecidas y obtener las estadísticas deseadas a la misma velocidad con que transmite el medio. Para ello dicho hardware tiene que tener un desempeño como si se tratara de una solución implementada directamente en hardware. Sin embargo, este tipo de tecnología no es flexible y una vez diseñada e implementada es difícil modificarla. Por otro lado, las soluciones de software nos dan la flexibilidad requerida, pero no el desempeño deseado. Es por esto que una tecnología que mezcle lo mejor de ambos estaría en lo óptimo que se desea. Es ahí en donde entran los Procesadores de red.

Capítulo 3 IXP1200

Intel ha introducido la Intel® ***Internet Exchange Architecture*** (IXA, Arquitectura de intercambio sobre Internet) que provee los "bloques" necesarios, tanto en software como en hardware, para construir plataformas de red programables y reusables. El principal componente de la arquitectura ***IXA*** de Intel® es la familia de procesadores de red ***IXP***. El procesador de red ***IXP*** es una plataforma completamente programable con una arquitectura de almacenamiento de datos distribuido consistente de múltiples procesadores que trabajan en hardware con multihilos (***multithreading***). Cabe mencionar que diferentes universidades de todo el mundo han adoptado la plataforma ***IXP*** tanto para su docencia como para realizar investigación. [3]

La arquitectura ***IXP*** es una innovación tecnológica que es extremadamente flexible, pero al programar en la plataforma ***IXP*** requiere que el programador deje a un lado el paradigma de la programación secuencial, para pasar al modelo de la programación distribuida y paralela, lo cual no es siempre sencillo.[3]

Los procesadores de ***INTEL IXP1200*** combinan el desempeño de un procesador de de propósito general de alta velocidad o procesador de núcleo, con seis micromáquinas programables con manejo de multihilos (***multithread***). Esto significa, que el procesador central no debe gastar ciclos del reloj procesando los paquetes. El procesador de red esta optimizado para manejar

paquetes normales usando la ruta de paquetes de cada micromáquina, mientras que los paquetes excepcionales son enviados al procesador de núcleo para procesamiento extra. Cuando es usada apropiadamente esta característica, hace que se tenga un mejor desempeño si es comparado con una computadora personal actuando como un ruteador.

3.1 Características generales de la arquitectura.

El procesador ***StrongARM*** es un procesador ***RISC*** de 32 bits de propósito general.

Las seis micro maquinas también son procesadores tipo ***RISC***, pero están optimizados para un procesamiento veloz de los paquetes. Su conjunto de instrucciones esta específicamente desarrollado para el procesamiento de datos de la red. Por ejemplo, las instrucciones existen para operaciones a nivel de bits, a nivel de bytes, o en operaciones combinadas de desplazamiento o rotación en una sola instrucción. Cabe señalar que las micromáquinas no manejan operaciones de punto flotante.

Usar el procesador ***StrongARM*** o las micromáquinas es escoger entre velocidad y flexibilidad; el procesador ***StrongARM*** es un procesador de propósito general con instrucciones flexibles y puede tomar decisiones complicadas. Las micromáquinas son más simples, pero pueden realizar operaciones sobre los paquetes de manera rápida. Sin embargo, se puede utilizar las micromáquinas para realizar trabajos rutinarios, como el reenvío de los paquetes (***forwarding***), y el procesador ***StrongARM*** puede ser usado para realizar decisiones más complicadas, por ejemplo, la actualización y mantenimiento de la tabla de ruteo.

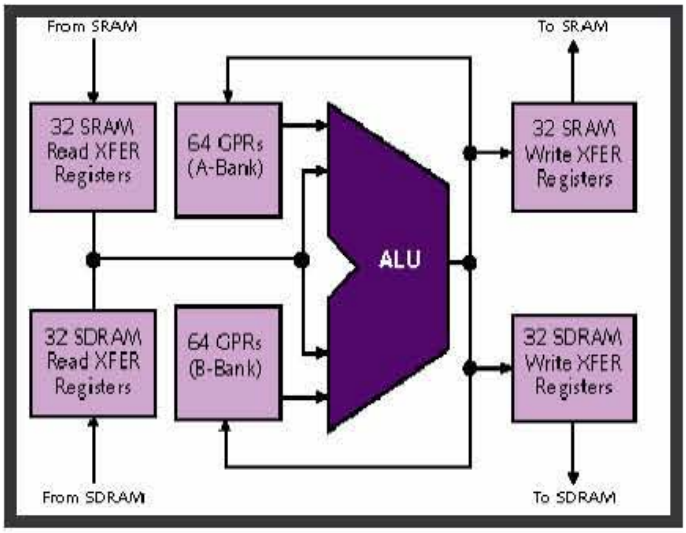

Figura 3-1 Arquitectura de una micromáquina.

La figura anterior muestra la arquitectura básica de una micromáquina. Cada micromáquina soporta hasta cuatro hilos en ejecución por hardware.

Cada micromáquina tiene un gran número de registros. 64 registros de propósito general (***General Purpose Register, GPR***) en un banco A y 64 en un banco B, para un total de 128 ***GPR*** de 32 bits. Cada micromáquina tiene 64 registros de transferencia SDRAM y 64 registros de transferencia SRAM. Cuando una micromáquina quiere leer datos de la SDRAM o de la SRAM, los datos deben ser cargados dentro de estos registros de transferencia primero, y después son enviados a las micromáquinas. El propósito de tener estos registros de transferencia es que las micromáquinas puedan continuar procesando con los ***GPR***s mientras otras unidades funcionales del procesador ***IXP*** pueden leer o escribir a los registros de transferencia. Esto es exactamente igual que escribir o leer en un buffer de lectura o de escritura.

3.2 Características del procesador de red de intel.

El procesador de red de Intel que se va a manejar recibe el nombre de Intel ***IXP1200***. Es el nombre de la primera generación de chips de procesador de red, y existen 4 modelos disponibles, como lo muestra la siguiente tabla:

Tabla 3-1 Características de los procesadores IXP

Numero de modelo	Número de pines	Soporte para CRC	Soporte para tasa ECC	Posible velocidad del reloj
IXP1200	**432**	**No**	**No**	**166, 200, o 232**
IXP1240	**432**	**Si**	**No**	**166, 200, o 232**
IXP1250	**520**	**Si**	**Si**	**166, 200, o 232**
IXP1250	**520**	**Si**	**Si**	**Solo 166**

Cada procesador de red de esta serie cuenta con:

- Un procesador ***RISC*** embebido
- Seis procesadores de paquetes programables
- Múltiples e independientes buses internos
- Mecanismos de sincronización del procesador
- Una pequeña cantidad de memoria interna
- Una interfase serial de baja velocidad
- Múltiples interfases externas para memoria externa
- Múltiples interfases para buses de entrada y salida externos
- Un coprocesador para el calculo de la función hash
- Otras unidades funcionales.

Los puertos de conexión externos operan a diferentes velocidades:

Tabla 3-2. Características de los puertos externos.

Tipo	Ancho del Bus	Velocidad del reloj	Tasa de Transmisión
Línea Serial	No Aplica	No Aplica	38.4
Bus PCI	32 bits	33-66 Mhz	2.2 Gbps
Bus IX	64 bits	66-104 Mhz	4.4 Gbps
Bus SDRAM	64 bits	≤ 232 Mhz	9.28.0 MBps
Bus SRAM	16 o 32 bits	≤ 232 Mhz	464.0 MBps

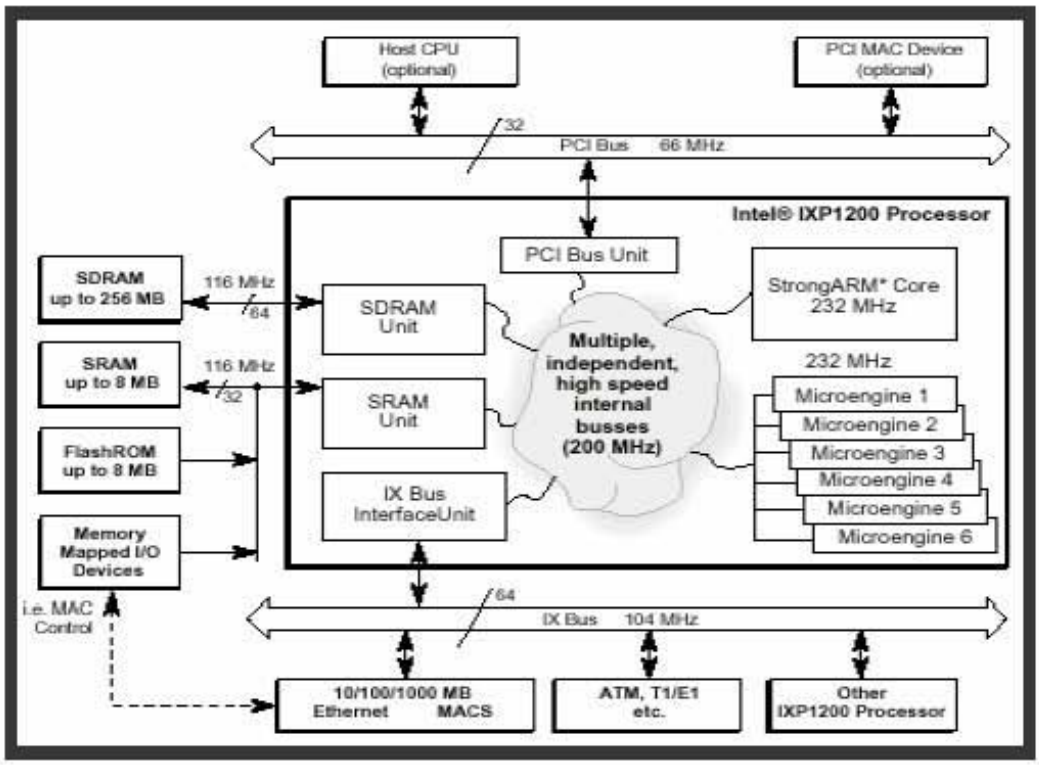

Figura 3-2 Arquitectura del Procesador IXP1200

3.3 IXP en Hardware.

La arquitectura anteriormente descrita es considerada como una computadora dentro de otra de mayor tamaño y potencia. Por lo tanto, las unidades descritas de memoria, interfaces externas y conexiones a bus, están contenidas en una tarjeta de expansión para PC, la cual utiliza un puerto PCI de la computadora anfitrión para su funcionamiento.

Dicha tarjeta es desarrollada por la compañía Radisys y tiene como modelo el número ENP-2505.

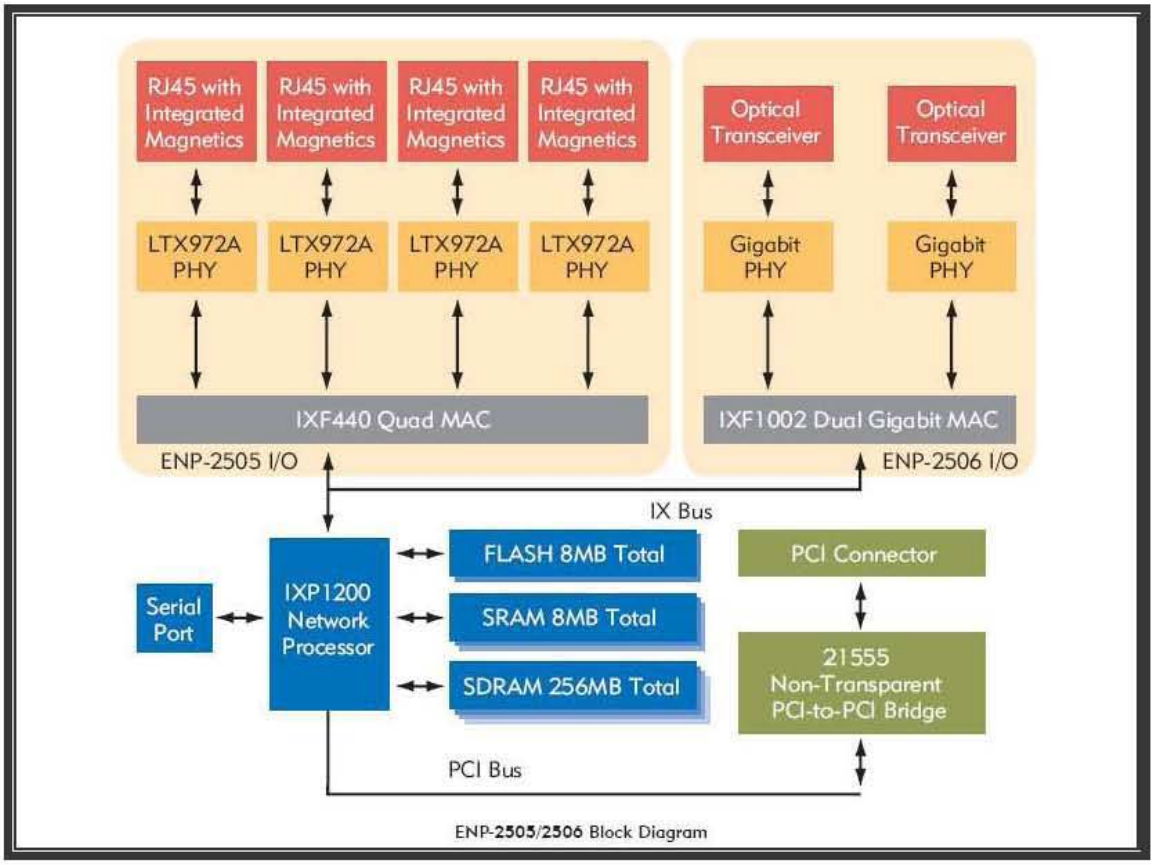

Figura 3-3 Esquema por bloques de la tarjeta ENP-2505

Las características generales que presenta esta tarjeta son las siguientes:

Tabla 3-3 Características generales de la tarjeta ENP-2505

Característica	Descripción
Procesador	Intel IXP1200 con un procesador StrongARM de 232 MHz como núcleo y 6 micromáquinas
Memoria SDRAM	256 Mbytes
Memoria SRAM	8 Mbytes
Memoria Flash	8 Mbytes
Puertos de entrada	4 10/100 Ethernet
PCI-to-PCI Bridge	21555 non-transparent
Ancho del Bus PCI	32 o 64 bits
Velocidad del Bus PCI	33 MHz o 66 MHz
Requerimientos de consumo	13.5 W Típicamente

Figura 3-4 Tarjeta ENP-2505

3.4 Características generales de la metodología de programación.

Intel propone una forma de programación, llamada elemento activo de computo (***Active computing element, ACE***) siendo el componente de software primario en la arquitectura ***IXP***. Los ***ACE***s son generalmente asignados a una tarea. Se pueden organizar como un procesamiento serial o pipeline. Los ***ACE***s convencionales solo se ejecutan en el procesador ***StrongArm***. Las micromáquinas ejecutan solo ***ACE***s acelerados llamados ***microACE***s. Un microbloque es un conjunto de ***microACE***s en serie y ejecutándose en una misma micromáquina, donde cada ***microACE*** es por separado una hilo. El objetivo de los ***microACE***s es el procesamiento de paquetes tan rápido como sea posible de forma de ***pipeline***. Para cada ***microACE*** ejecutándose hay un componente convencional en el núcleo en el ***ACE*** en el procesador núcleo o principal. Cuando hay paquetes especiales o de excepción que requieren mas procesamiento, y esto puede hacer que el ***microACE*** tarde demasiado en el procesamiento de su parte del pipeline, el paquete es enviado hacia el componente del núcleo el cual lo redirige hacia otro ***ACE*** convencional que maneja el paquete, pero que no corresponde al mismo ***microACE***.

Los ***ACE***s convencionales en el procesador ***StrongARM*** son codificados en C/C++. Los ***microACE***s son específicos para el hardware y son escritos en el código de máquina específico para el ***IXP1200***. En el microcódigo, el programador puede tomar ventaja de las instrucciones especiales de hardware y, por ejemplo, escoger entre la ***SRAM*** y la ***SDRAM***. El ***Intel Developer Workbench*** puede convertir el código en C en micro código, pero puede ser que no se tome por completo la ventaja que proporcionan las instrucciones especiales. Esto se puede dar porque no se

tome ventaja de las múltiples formas de la memoria RAM o sea forzado a transferir datos hacia los registros para operaciones que pueden ser realizadas directamente en memoria.

Los ***microACE***s son descargados hacia las micromáquinas como archivos de imagen con la extensión .uof. Además de las imágenes de los ***microACE***s, los micro bloques deben tener un ciclo de despacho (usualmente convertido del lenguaje C a microcódigo) el cual recibe los paquetes entrantes y un archivo de configuración el cual le dice a las micromáquinas como manejarlos ***microACE***s juntos.

Sin embargo, aunque es la forma se propone sea usada por defecto, también se toma en cuenta que se puede realizar la programación basándose en el subconjunto de instrucciones que contiene el ***MicroEngine C***, así como las librerías y programas de ejemplo que Intel proporciona.

Con estas librerías y programas de ejemplo, es posible realizar modificaciones en un menor tiempo, ya que contienen macros y rutinas que ya identifican las cabeceras de los paquetes de acuerdo a los RFC correspondientes y algunas otras funciones que permiten tener los ***ACE***s de entrada y salida.

3.5 Conclusiones.

Como ya se menciono en el capitulo 2, el usar una tecnología hibrida permite desarrollar sistemas de red que puedan adaptarse o modificarse. Para esto, la tecnología ***IXP*** de Intel es una tecnología hibrida, que además proporciona herramientas de depuración y desarrollo, como son librerías que se pueden tomar de base para realizar la programación del sistema de red, adecuando solamente los bloques necesarios.

Capítulo 4. Diseño del traffic policer

El sistema de red que se presenta a continuación tiene la siguiente estructura basada en la definición de traffic policer mencionada anteriormente, la cual indica que se limita la cantidad de paquetes que van a pasar por el medio. Los paquetes que sobrepasen este límite, son descartados, por lo que solamente el tráfico que se tiene acordado pasará.

El sistema planteado se puede descomponer en varios bloques, los cuales se pueden plantear de la siguiente forma:

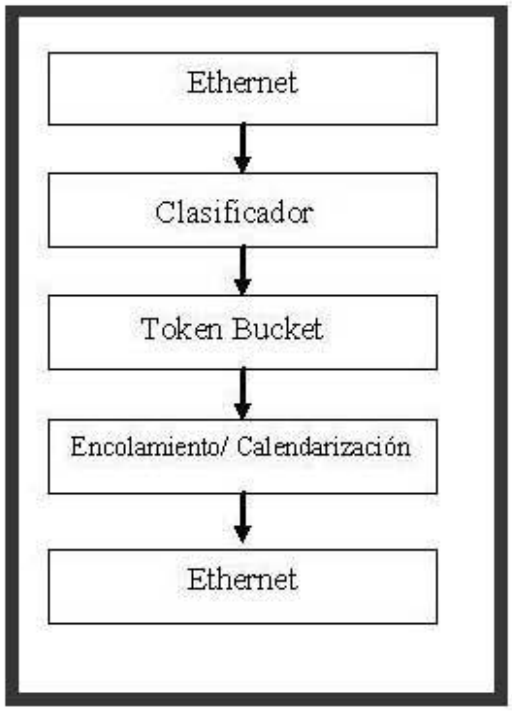

Figura 4-1 Diagrama a bloques que describe al sistema de red

4.1. Bloque de entrada o Ethernet.

Las funciones que se realizan en este bloque son las relacionadas con las capas 1 y 2 del modelo OSI.

Tanto el bloque de entrada como el de salida tienen de forma general la misma función, dado que en el caso de la entrada, recoge los paquetes del medio físico y los alista para que sean usados dentro de la aplicación, mientras que el bloque de salida realiza la función a la inversa. Sin embargo, dentro del IXP1200 se tienen las siguientes características para la parte de entrada [3]:

1) El paquete es recogido por uno de los dispositivos MAC asociados al bus IX, siendo que este dispositivo puede ser del tipo de Ethernet, ATM o SONET.

2) La unidad de interfase con el bus IX, la cual se le denomina FBI (***First In First Out(FIFO) Bus Interface, FBI***) contiene un procesador programable llamado ***ready-bus sequencer*** (secuenciador que notifica que el bus esta listo), el cual se encarga de preguntar por poleo a los diferentes dispositivos MAC para verificar quien tiene ya datos listos. Sin embargo, el bus IX transmite solo fragmentos de datos de 64 bytes, llamados mpaquetes (***mpackets***). En el caso de que el medio físico maneje tamaños de frames mayores a 64 bytes, las micromáquinas tienen que coordinarse para reensamblar los mpaquetes en un solo frame del medio. Sin embargo, se presenta el problema de cómo identificar que parte del paquete es la que se esta manejando. Para esto se recurre a marcar los mpaquetes de acuerdo a su posición en el paquete original. Si el mpaquete se encuentra al inicio, se le marca como inicio de paquete (***Start of Packet, SOP)***, si se encuentra al final, se le marca como final del paquete (***End Of Packet, EOP***). Los paquetes intermedios no son marcados, aunque, existe tambien la posibilidad de tener un paquete que tenga ambas marcas (cuando el paquete original tiene un tamaño menor o igual a 64 bytes).

3) Una vez que ya se han recibido datos, el secuenciador actualiza los registros contenidos en el FBI, llamados CSR (***control and status registers,*** registros de control y estado). Hay tres de estos registros que intervienen en la recepción, ***rcv_rdy***, ***rcv_req*** y ***rcv_cntl***. Los primeros registros indican el puerto en el cual esta listo el dato, y se encuentra divididos en dos partes:

    1. ***rcv_rdy_lo*** , que representa los puertos 0 a 31;
    2. ***rcv_rdy_hi***, que representa a los puertos 32 a 55

Una vez que uno de los bits que indican el puerto es activado, se inicia la transferencia de datos, haciendo una petición de recepción. Esto se logra al escribir al registro ***rcv_req*** la petición de mover los mapquetes a un elemento ***Receive FIFO (RFIFO)***. El elemento RFIFO es en realidad un buffer usado para guardar los datos de llegada. La unidad FBI contiene 16 elementos de este tipo, cada uno de los cuales puede contener un mpaquete.

Cuando la transferencia de datos hacia los RFIFOs se ha acompletado, la unidad FBI indica a las micromáquinas que actualicen el tercer registro: ***rcv_cntl***. Este último CSR contiene información acerca de fallas, el estado del mpaquete en el reensamblaje y el tamaño de los datos en el mpaquete.

4) Las micromáquinas mueven los mpaquetes de los RFIFOs hacia la memoria SRAM.

Esto se repite para cada mpaquete que compone al paquete originalmente recibido. En nuestro caso, de este bloque se encargan dos de las micromáquinas que contiene el IXP1200.

La parte correspondiente a la salida de los paquetes desde el procesador de red se verá como bloque final, para hacer hincapié en las partes que son diferentes en cuanto a este bloque de entrada.

4.2. Bloque de clasificación.

La clasificación que actualmente se tiene se basa en identificar los paquetes tipo ARP e IP. Debido a que hay diferencias entre Ethernet y IEEE 802.3 se recurre a verificar que el contenido sea un campo de tipo de trama siguiente y no uno de longitud de trama. Principalmente se trabaja con tramas de tipo Ethernet, por lo que el campo de tipo de trama siguiente es el que se busca que tenga un valor por encima a 1500 en decimal o 05DCH en hexadecimal. Esto es porque se asume que si este campo tiene un valor mayor a un 600H se considera de tipo Ethernet, de lo contrario es IEEE802.3 [7].

Por lo anterior, se planteo que la condición para que sea un paquete de tipo ARP o un paquete IP sea la siguiente:

```
if (ether_header.protocol_length > ETHER_MTU && ether_header.protocol_length != ETHER_ARP)
                        {
                         *tip=2; //IP
    }
    else{
                         *tip=3; //ARP
                         }
```

El valor que toma ETHER_ARP es de acuerdo a los números para el campo de tipo en Ethernet, el valor hexadecimal de 0806H.

Las micromáquinas, para hacer el procesamiento de los bytes que están en la cabecera y realizar la clasificación, utilizan los mpaquetes que se encuentran en los RFIFOs. De hecho, solo es necesario leer el primer mpaquete -el cual tendrá la marca SOP-; dado que los mpaquetes tienen una longitud de 64 bytes y la información necesaria se encuentra en esta parte.

Así, esta clasificación se puede extender no solo a lo que son protocolos de capa 3, si no también a los puertos correspondientes en capa 4, planteando para cada caso la forma de extracción de los datos necesarios del primer mpaquete.

Las micromáquinas que realizan el bloque de entrada intervienen en esta parte del sistema, ya que son las que pueden leer o modificar las cabeceras de los paquetes.

4.3. Bloque Token Bucket.

En este bloque se aplica el algoritmo de Token bucket en la modalidad de policer, este algoritmo se encarga de limitar la entrada y salida de paquetes, descartando los paquetes que sobrepasen lo establecido. Una sola micromáquina se encarga de llevar a cabo la función de llenado del bucket. De forma general, se tiene lo siguiente [6]:

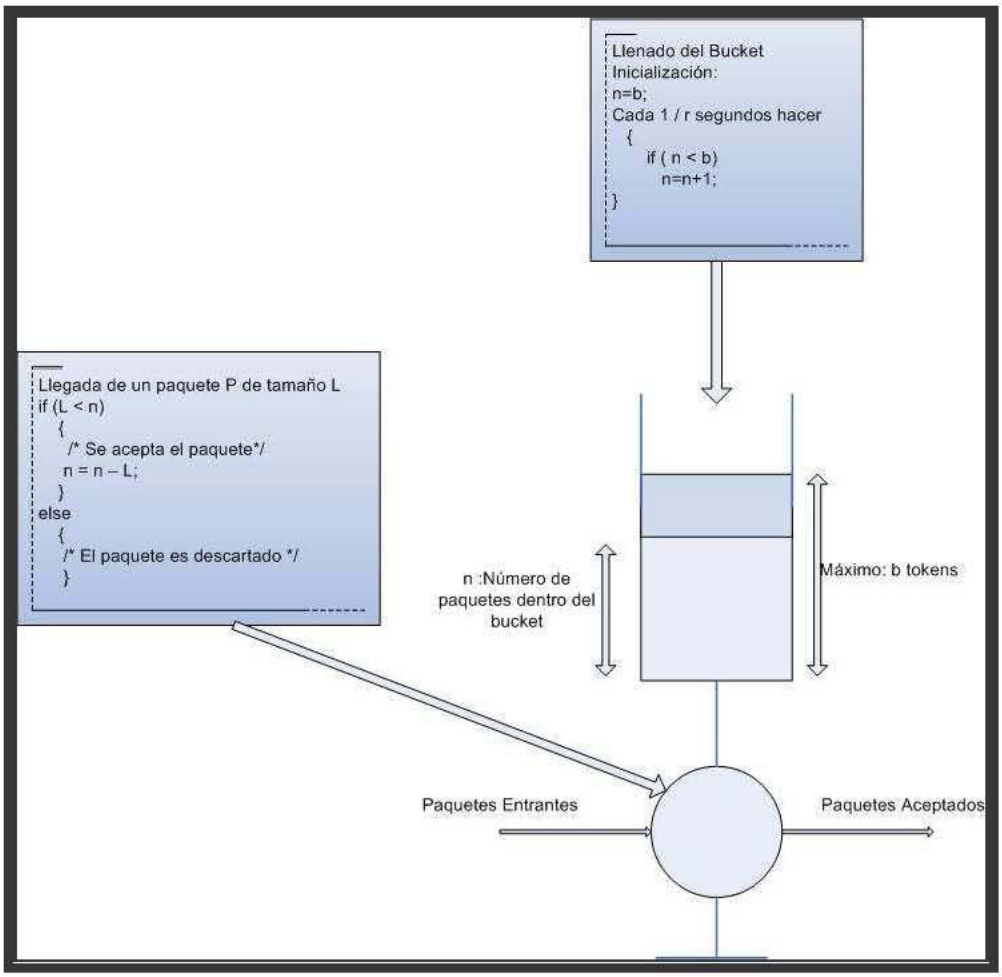

Figura 4-2 Esquema del algoritmo de Token Bucket en modo policer

Donde:

- r: tasa de llenado del bucket [bytes / seg]
- b: tamaño del bucket [bytes]
- n: número de tokens dentro del bucket [bytes]
- L: longitud del paquete en bytes [bytes]

Durante un periodo de T segundos, el token bucket acepta por lo menos b + (T x r) bytes de tráfico.

El token puede estar dado en paquetes o en bytes. Para el primer caso, se verificaría la cantidad de paquetes que hay dentro, para que de acuerdo a la cantidad de tokens se dejen salir.

En el segundo caso, que es el usado para este trabajo, los tokens representan bytes, por lo que se hará una comparación entre los tokens dentro del bucket y se eliminaran tantos como mida el paquete.

La finalidad de usar este bloque es el poder limitar la entrada de los paquetes al sistema, se colocó un ***bucket*** para cada cola, de manera que el llenado y vaciado de paquetes sea de acuerdo a la clasificación propuesta.

4.4. Bloque de encolamiento.

Dado que el hardware del IXP1200 puede realizar tareas en paralelo, se escogió un algoritmo de productor – consumidor basado en un arreglo. Otra opción viable seria una lista ligada de tipo FIFO (***First In First Out,*** primero entra primero sale), pero se tendría que cuidar que la estructura de la lista no se corrompiera, y dado que los apuntadores al principio y fin de la lista son los que encolan y liberan, es necesario asegurar la lista para que solo una operación de encolamiento o desenconamiento se haga a la vez. Esto representa una operación a la vez se puede realizar dentro de la lista, desperdiciando las propiedades del hardware que puede trabajar en forma paralela.

En el caso del productor - consumidor, se tienen dos índices que apuntan hacia los paquetes que pueden ser sacados (productor) y las localidades en el arreglo que pueden ser ocupadas (consumidor). Para mantener la integridad del arreglo, es necesario que solo una operación de encolamiento se haga a la vez y solo una de desenconamiento al mismo tiempo, ya que los índices se actualizan conforme se van realizando cada una de las operaciones. Sin embargo, se puede realizar una operación de encolamiento y una de desenconamiento al mismo tiempo, con lo que se puede aprovechar el paralelismo existente.

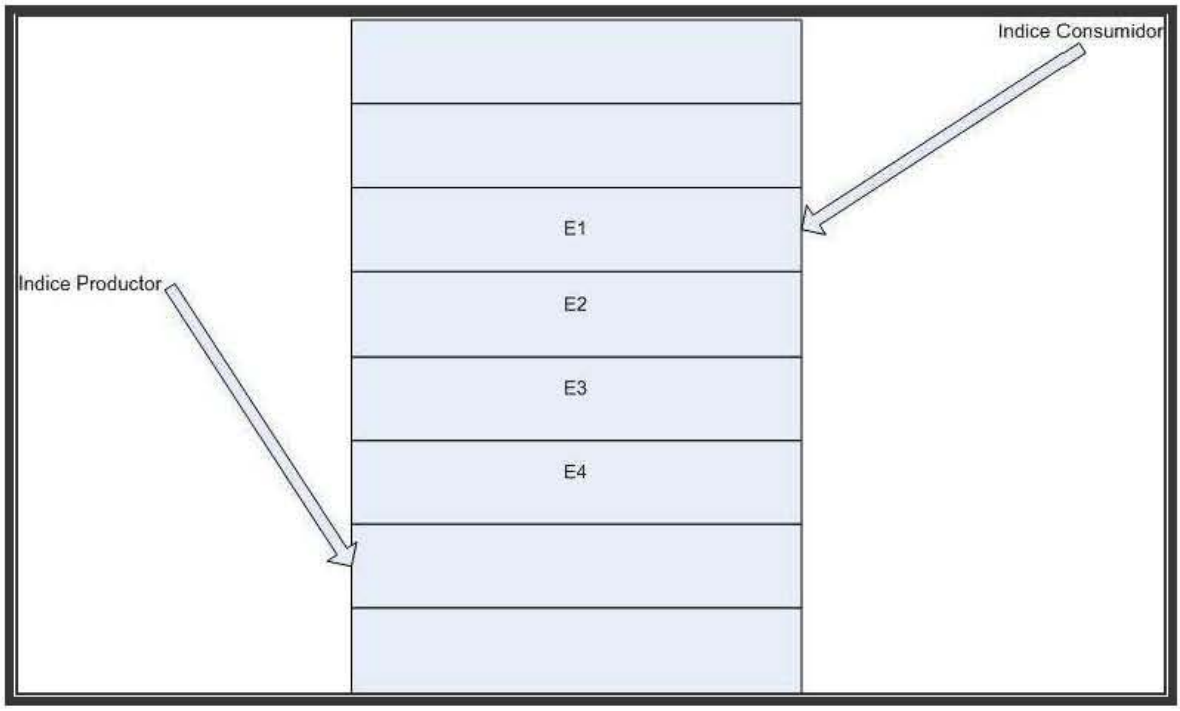

Figura 4-3 Esquema Productor - Consumidor

Para el caso que nos ocupa, se tienen dos colas, una para los paquetes de tipo ARP y otra para los paquetes de tipo IP, y de igual forma que el bloque anterior, las micromáquinas que se encargan de esto son las dedicadas a la parte de la entrada.

4.5. Bloque de Calendarización.

Se utiliza el algoritmo de Round Robin, el cual distribuye las peticiones o la carga a las colas en forma rotativa. La primera petición que llega es dirigida a una de las colas, y las peticiones siguientes las asigna de forma circular. Una vez que es asignada la petición, la cola correspondiente es movida al final de la lista, manteniendo así una igualdad de asignaciones.

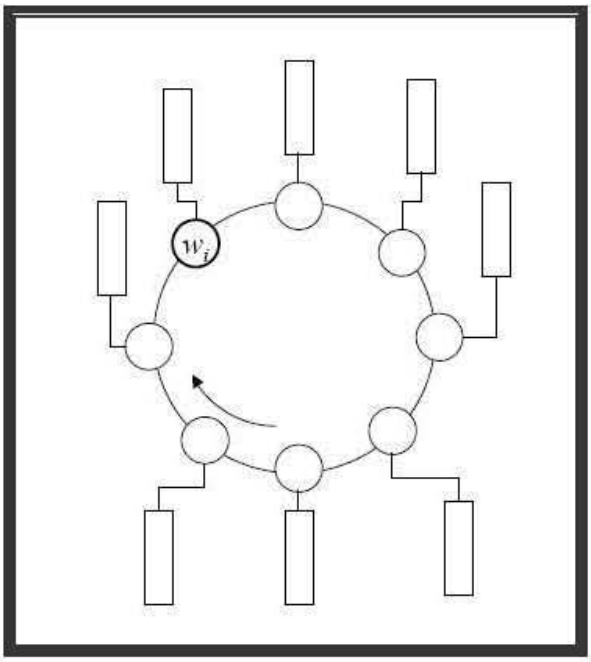

Figura 4-4 Diagrama del algoritmo Round Robin.

Este es uno de los algoritmos, de forma general, más sencillos y justos en el reparto de carga. Cada cola tiene asignado un intervalo de tiempo llamado cuantum o cuanto. Si se agota, se elige al proceso siguiente. Si se libera antes de terminar su tiempo, se libera el recurso y se pasa a la siguiente cola.

Una variante de este algoritmo es el ***Weighted Round Robin*** (WRR). Esta es una versión avanzada del algoritmo de round-robin que elimina las deficiencias del algoritmo original. En el caso del weighted round-robin, se puede asignar un peso a cada cola del grupo, de tal forma que si se quiere que una de las colas libere a los paquetes de forma más rápida que las otras, se le asignara un peso mayor. Por lo general se utiliza para los balanceadores de carga, de tal forma que se aproveche la capacidad de los servidores que se tiene.

4.6. Bloque de salida.

Una vez que el paquete ha sido modificado por las micromáquinas, una de las restantes se encarga de la salida de los paquetes. Como se menciono con anterioridad, el bloque de salida, conceptualmente debe realizar la función inversa al de entrada.

Las diferencias que pueden existir entre ambos se mencionan a continuación [3]:

1) Antes de la transmisión del paquete, internamente se debe de haber asignado el puerto por el cual saldrán.

2) Los dispositivos MAC verifican que el mpaquete quepa en el buffer de salida, un ***Transmit FIFO*** o ***TFIFO***. Cada TFIFO tiene un buffer de 64 bytes para enviar datos al dispositivo MAC y 8 bytes de control que le dicen al secuenciador que puerto se puede usar para la salida del paquete mientras los mpaquetes son reensamblados para formar el paquete original. De igual manera que en el caso de la recepción, el secuenciador ***ready-bus*** realiza las mismas operaciones. Recoge el estado de los puertos y los coloca en el CSR ***xmit_rdy***, El campo de valido indica si los datos y las secciones de control de la estructura estan completas. Si el campo de valido no es correcto, no se maneja ese TFIFO. Una vez transmitido, el TFIFO se invalida y se esperan los nuevos datos.

3) De la igual manera que la parte de la recepción, este proceso se debe de repetir para cada mpaquete que conforma el paquete. El dispositivo MAC detecta el final del paquete a traves de la información de control que es proporcionada por las micromáquinas.

4.7. Conclusiones.

El diagrama a bloques muestra la idea general acerca de estos sistemas de red. Las consideraciones cada uno llevan a cabo una función en específico, tales que pueden ser usadas por otros sistemas de red. Gran parte del código usado se tomó y adecuo de las librerías de ejemplo que se proporcionan, no sin antes hacer una serie de pruebas para ver cual es más conveniente para lo propósitos que se buscaban.

Aunque los bloques fundamentales están ya dados, es necesario entender la arquitectura y la forma de programación que propone ***Intel***.

Capítulo 5. Resultados y análisis.

Una vez descrito el sistema de red y la forma de implementación se procedió a realizar pruebas en el hardware ***ENP-2505.*** Para ello se plantearon varios escenarios. Se presentan gráficas que representa el tráfico medido a la salida de la tarjeta con la entrada definida para cada escenario.

Para el análisis se utilizaron un analizador de red como lo es ***Ethereal*** [13] y un generador de flujo de paquetes, ***packETH*** [12]. El esquema de conexiones utilizado se muestra en la figura 5-1:

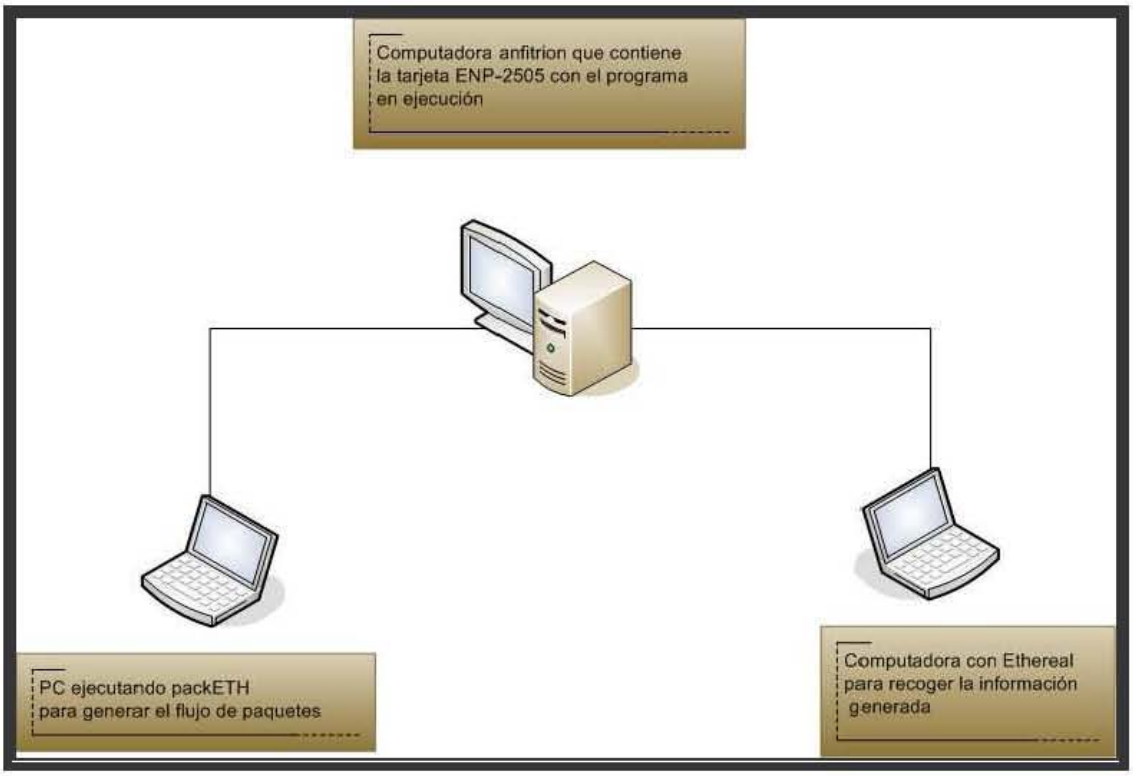

Figura 5-1 Esquema con la tarjeta ENP-2505

Sin embargo, para tener una referencia de los flujos de paquetes utilizados, se estableció una conexión entre los dos equipos terminales, a fin de obtener datos de la transmisión sin la tarjeta en medio. Figura 5-2.

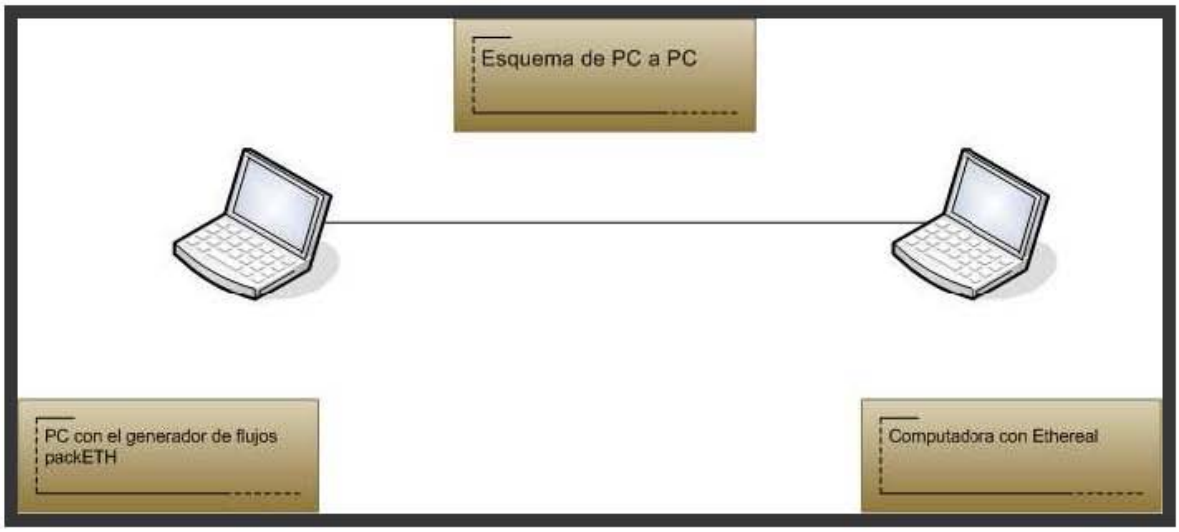

Figura 5-2 Conexión PC a PC

El tipo de flujos de paquetes que se alimentan a la tarjeta, de acuerdo a la clasificación mencionada en el apartado 4.2, son tramas 802.3, ARP y tramas IP con protocolos TCP, UDP de diversos tamaños.

Se considero un ***bucket*** de tamaño de 260 bytes para las tramas 802.3 y ARP, y de 520 para los paquetes IP. La tasa de llenado del ***bucket*** esta definida de acuerdo al número de ciclos de máquina que se tarda en incrementar los tokens. El procesador StrongARM tiene un reloj interno de 233 MHz, por lo que cada ciclo de reloj sera equivalente a 4.29 ns, por lo que los ciclos toma sumarle 1 a C=C+1 es la tasa con la que se llena de tokens el ***bucket***. Para el ***bucket*** de 260 bytes se tienen 240 ciclos (971 [kB/s]) y para el ***bucket*** 520, 238 ciclos (979 [kB/s]); aproximadamente 1[MB/s] en cada caso.

Los primeros escenarios 1 a 3 tienen la finalidad de verificar con un flujo constante de paquetes la regulación que presenta el sistema. Se considera un flujo combinado, con paquetes de ambos ***buckets***., para después probar ambos flujos por separado.

5.1. Resultados obtenidos

5.1.1.Escenario 1

En este escenario, se creo un flujo de paquetes 802.3 de tamaño 256 bytes, 1000 paquetes con un intervalo entre ellos de 20[μS], y una separación al siguiente de 20000 [μS].

También se agrego un flujo de paquetes IP – TCP con las mismas características.

Tipo	Tamaño	Cantidad	Retraso entre paq. [μS]	Espacio al sig. flujo [μS]
802.3	256	1000	20	20000
IP-TCP	256	1000	20	20000

Tabla 5-1 Escenario 1

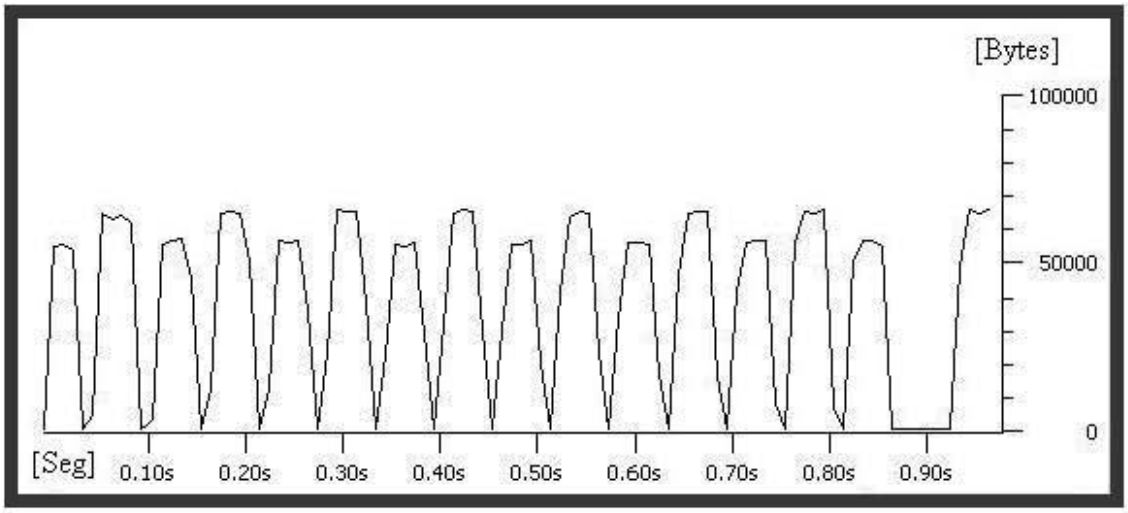

Figura 5-3 Gráfica del tráfico de Pc a Pc

La tasa promedio de transmisión es de 31.001 [MBit/s] y se transmitieron un total de 4720000 bytes.

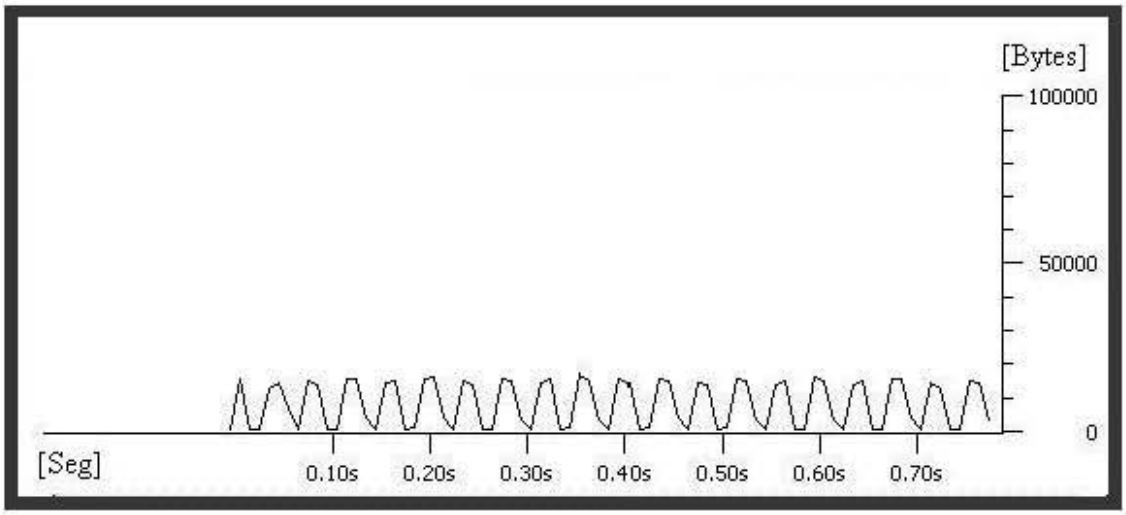

Figura 5-4 Salida de la tarjeta

La salida que se obtiene de la tarjeta se muestra en la figura 5-4. La tasa promedio que se obtiene es de 6.506 [MBits/s]. El total de bytes recibidos es de 636040, lo que representa el 13.47% del total de paquetes enviados.

5.1.2.Escenario 2

Para este escenario se enviaron solo los paquetes 802.3 con las mismas características del escenario anterior, Figuras 5-4 y 5-5.

Tipo	Tamaño	Cantidad	Retraso entre paq. [µS]	Espacio al sig. flujo [µS]
802.3	256	1000	20	20000

Tabla 5-2 Escenario 2

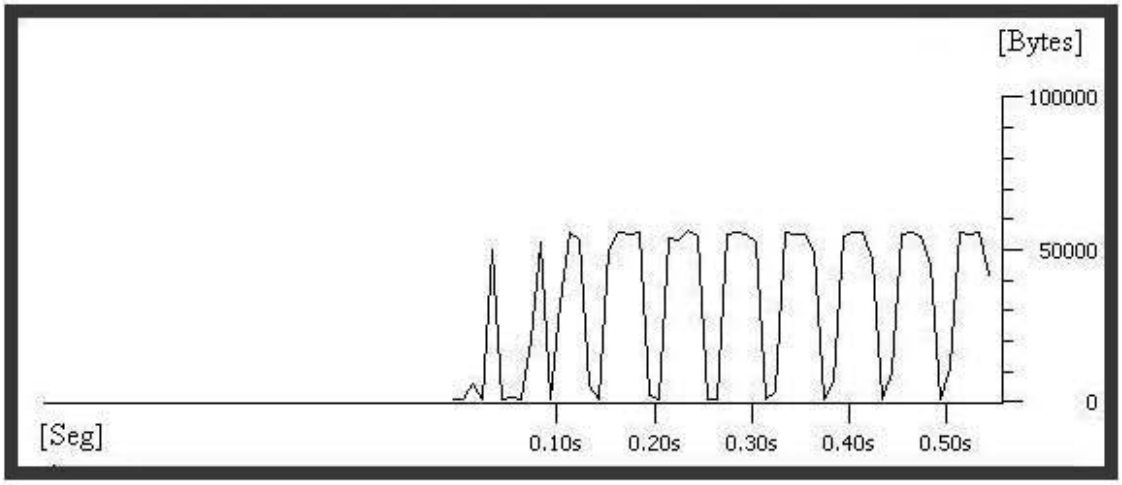

Figura 5-5 Gráfica del tráfico de Pc a Pc

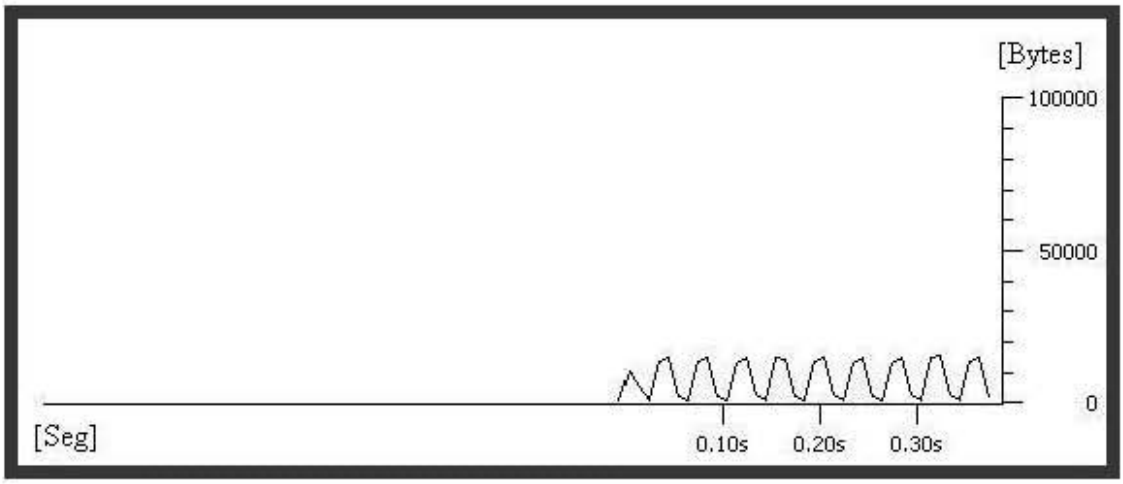

Figura 5-6 Salida de la tarjeta

La salida que se observa en la figura 5-5 es aproximadamente el 15.797% de la entrada anterior.

5.1.3.Escenario 3

El flujo de paquetes estuvo compuesto por IP – TCP de 256 bytes, en un total de 1000 paquetes con un espaciado de 20 [µS] entre paquetes y un espacio al siguiente flujo de 20000 [µS].

Tipo	Tamaño	Cantidad	Retraso entre paq. [µS]	Espacio al sig. flujo [µS]
IP - TCP	256	1000	20	20000

Tabla 5-3 Escenario 3

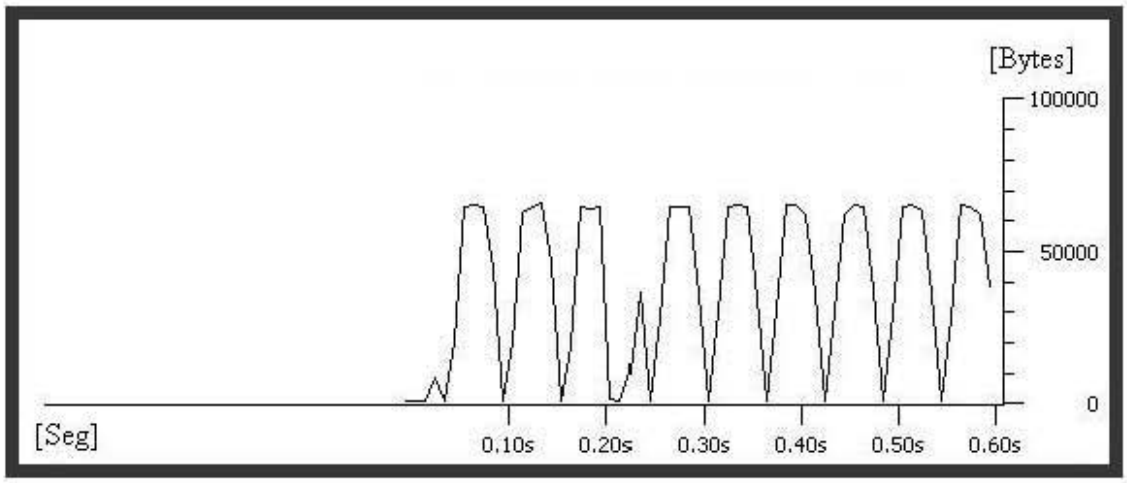

Figura 5-7 Flujo de Pc a Pc

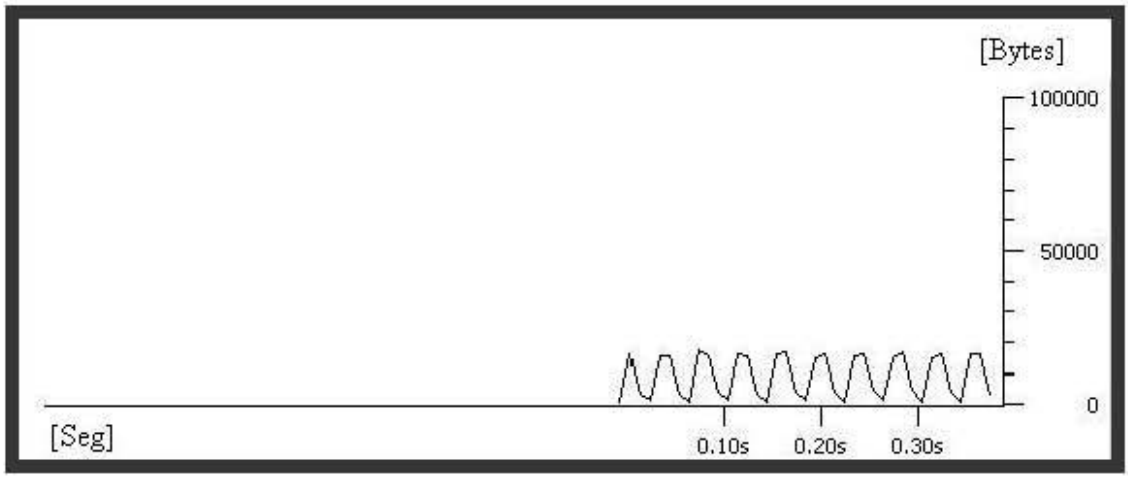

Figura 5-8 Tarjeta

De forma similar a las anteriores, figura 5-7, el flujo recibido es de aproximadamente el 15%, con un total de 14.63 %

5.1.4.Escenario 4

En este escenario se hizo una mezcla de paquetes, no solo de los dos anteriores que se venían manejando.

La configuración se muestra en la Tabla 4.

Tipo	Tamaño [Bytes]	Cantidad	Retraso entre paq. [µS]	Espacio al sig. flujo [µS]
802.3	256	20000	50	10000
IP-UDP	150	30000	33	15000
IP-TCP	256	10000	100	10000
ARP	60	50000	20	10000
IP-UDP	256	30000	33	10000
IP-TCP	200	3000	200	10000

Tabla 5-4 Escenario 4

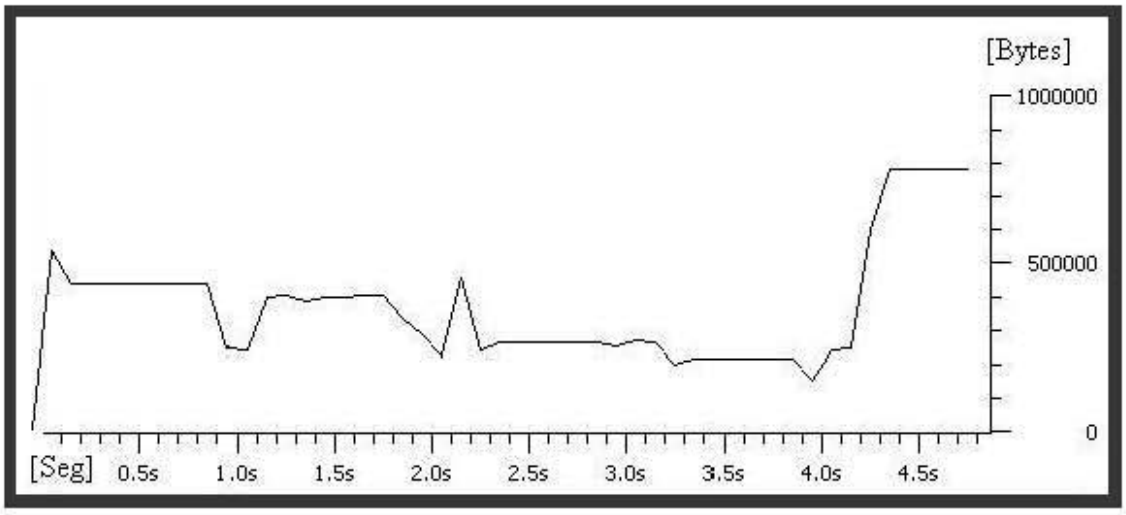

Figura 5-9 PC a PC

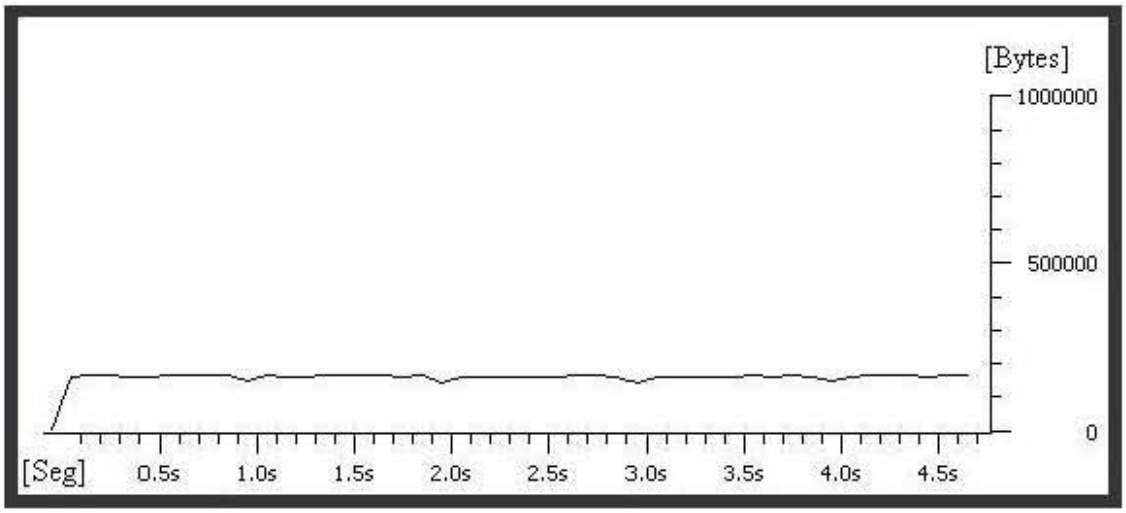

Figura 5-10 Salida de la tarjeta

En esta ocasión, la velocidad que se tuvo de PC a PC fue de 30.096 [Mbits/s], y la velocidad después de la tarjeta se redujo a 11.940 [Mbits/s], lo cual representa que se limito a un 39.67% la velocidad original con la que entraron los paquetes.

5.1.5.Escenario 5

En este escenario se preparo para verificar la atenuación y pérdidas de paquetes que se tienen al tener ráfagas grandes en intervalos de tiempo corto, así como un espaciamiento con la siguiente ráfaga bastante grande. En este caso se considera solo un tipo de tráfico.

Tipo	Tamaño	Cantidad	Retraso entre paq. [µS]	Espacio al sig. flujo [µS]
IP - TCP	256	40000	25	1000000

Tabla 5-5 Escenario 5

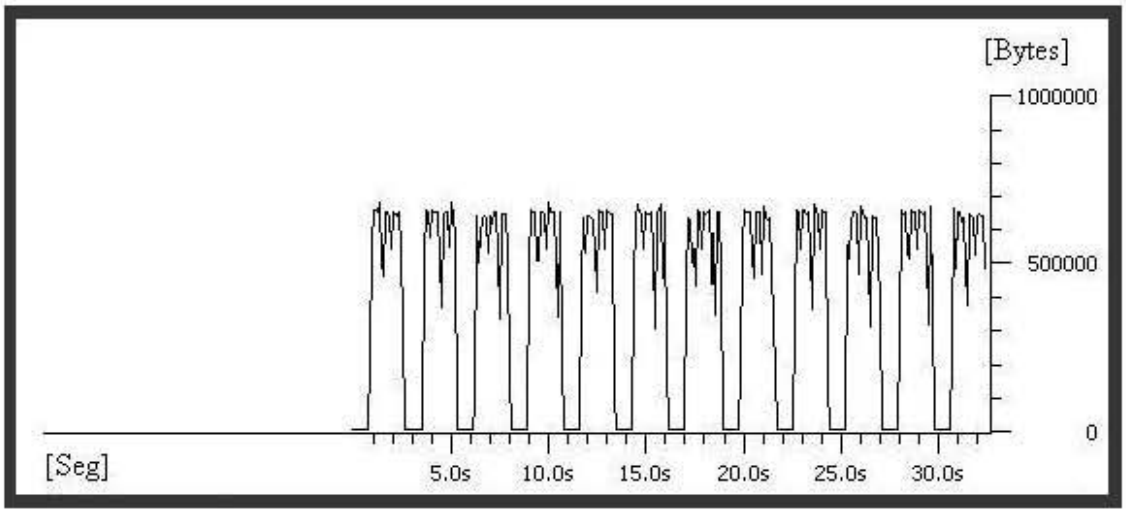

Figura 5-11

En esta primera parte se tiene una velocidad promedio de 30.182 [MBits/s], con 480000 paquetes de 256 bytes enviados.

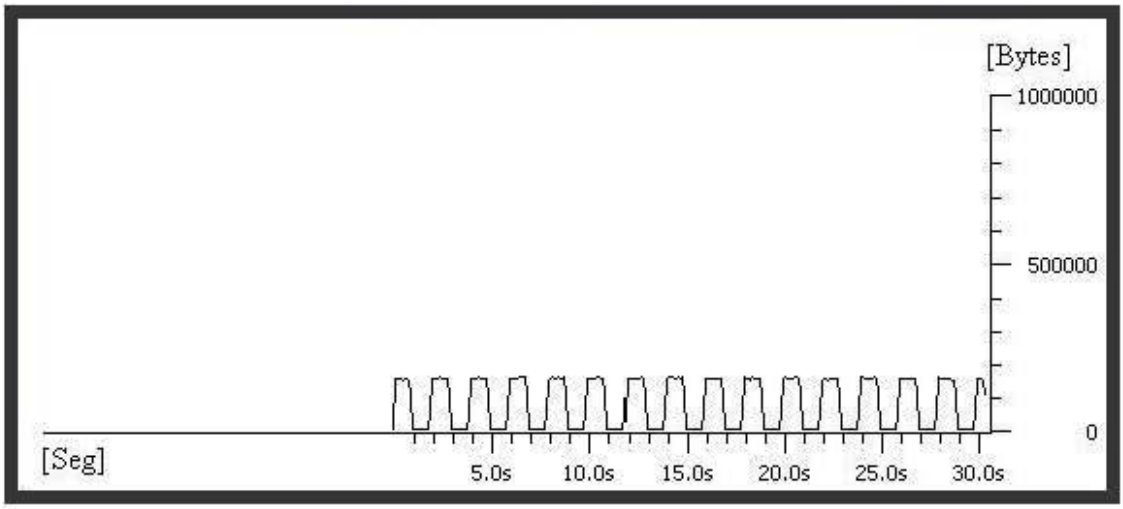

Figura 5-12

Después de la tarjeta se encontró que los paquetes se transmitían a una velocidad de 6.306 [MBits/s], con un total de 93814 paquetes, lo que representa el 19.54% del flujo original.

5.1.6.Escenario 6

Similar al escenario 5, pero con paquetes de la segunda clasificación.

Tipo	Tamaño	Cantidad	Retraso entre paq. [µS]	Espacio al sig. flujo [µS]
802.3	256	40000	25	1000000

Tabla 5-6 Escenario 6

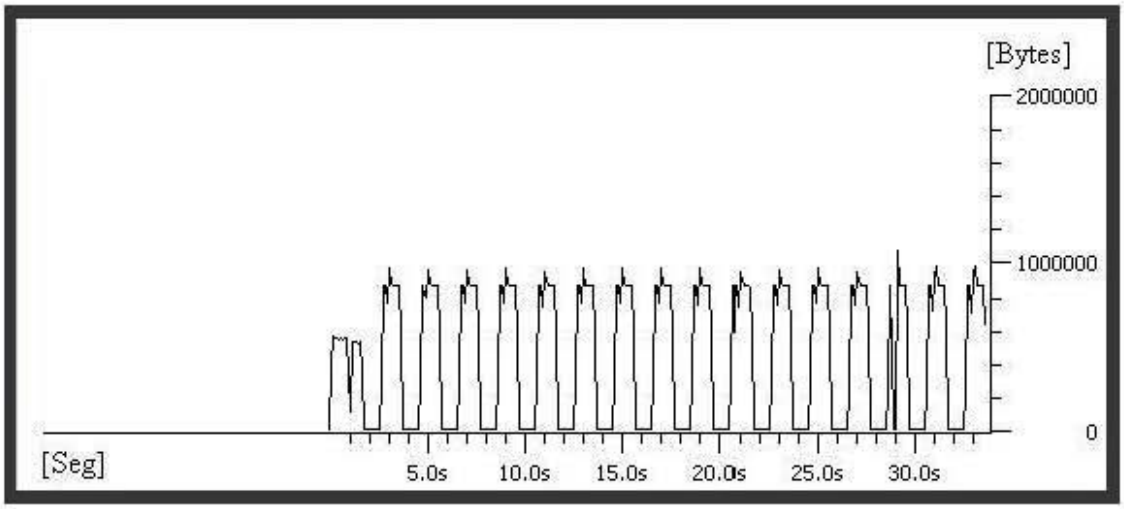

Figura 5-13

Se tiene una tasa de transmisión de 34.305 [MBits/s] con un total 670493 paquetes.

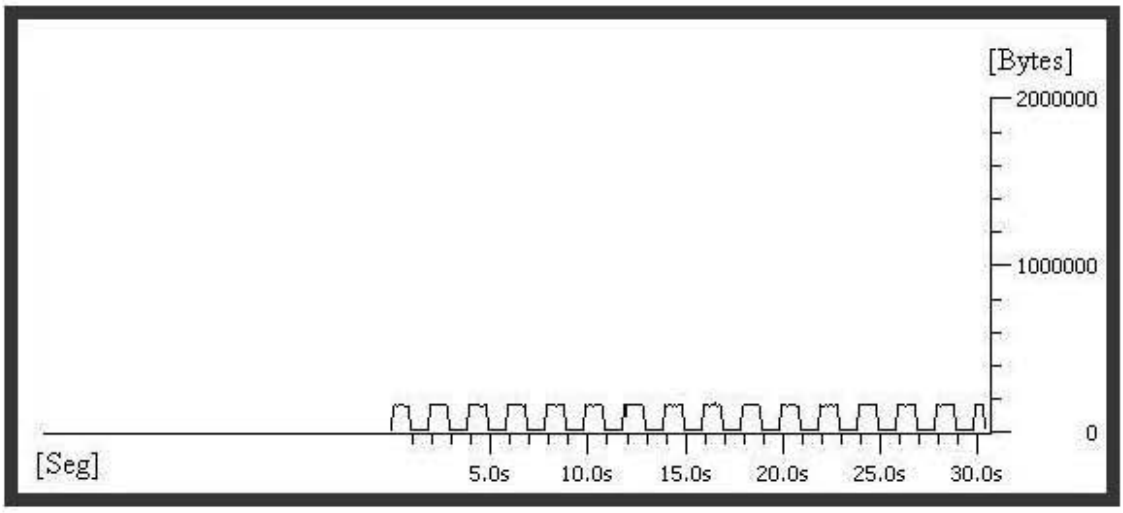

Figura 5-14

En la figura 5-14 se muestra lo que se obtuvo después de pasar por el ***policer***, teniendo una tasa de 6.251 [MBit/s], y un total de 110367 paquetes transmitidos, lo que representa el 16.46% del flujo total.

5.1.7.Escenario 7

En este escenario se preparo más de un flujo de paquetes, con la finalidad de verificar el comportamiento del sistema con distintas cargas de paquetes y a diferentes velocidades.

Dada la configuración del sistema, es de esperarse que conforme se aumente la cantidad de paquetes y la velocidad de los mismos, las perdidas de paquetes sean más notorias, ya que se descartan y no se almacenan los que exceden la capacidad del ***bucket***.

N.	Tipo	Tamaño	Cantidad	Retraso entre paq. [µS]	Espacio al sig. flujo [µS]
1					
	802.3	256	2500	400	5000
	TCP	256	2500	400	5000
2					
	802.3	256	5000	200	5000
	TCP	256	5000	200	5000
3					
	802.3	256	10000	100	5000
	TCP	256	10000	100	5000
4					
	802.3	256	15000	66	5000
	TCP	256	15000	66	5000
5					
	802.3	256	30000	33	5000
	TCP	256	30000	33	5000
6					
	802.3	256	40000	25	5000
	TCP	256	40000	25	5000

Tabla 5-7 Escenario 7

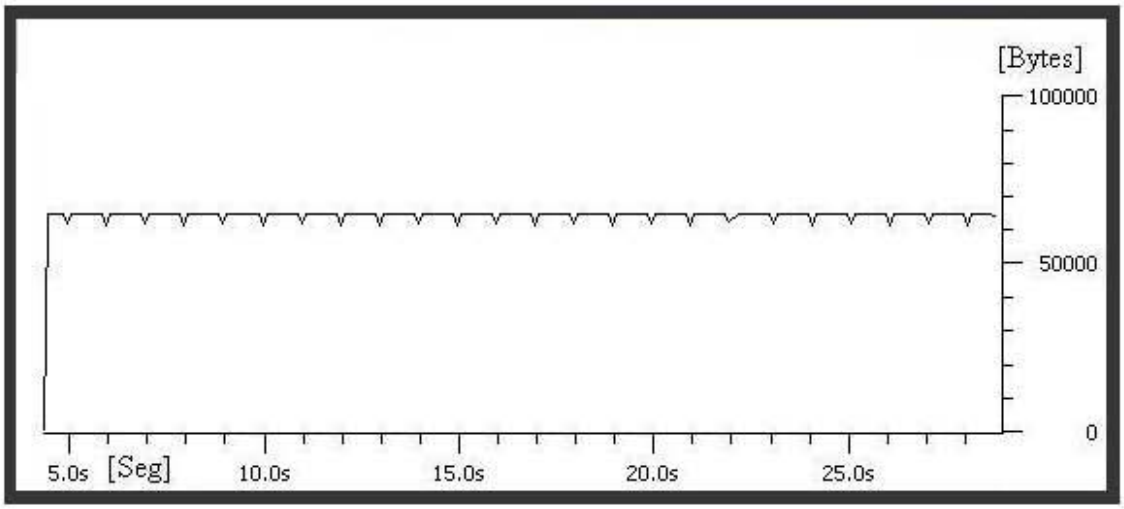

Figura 5-15 Pc a PC

Tasa de transmisión promedio: 5.097 [MBits/s]

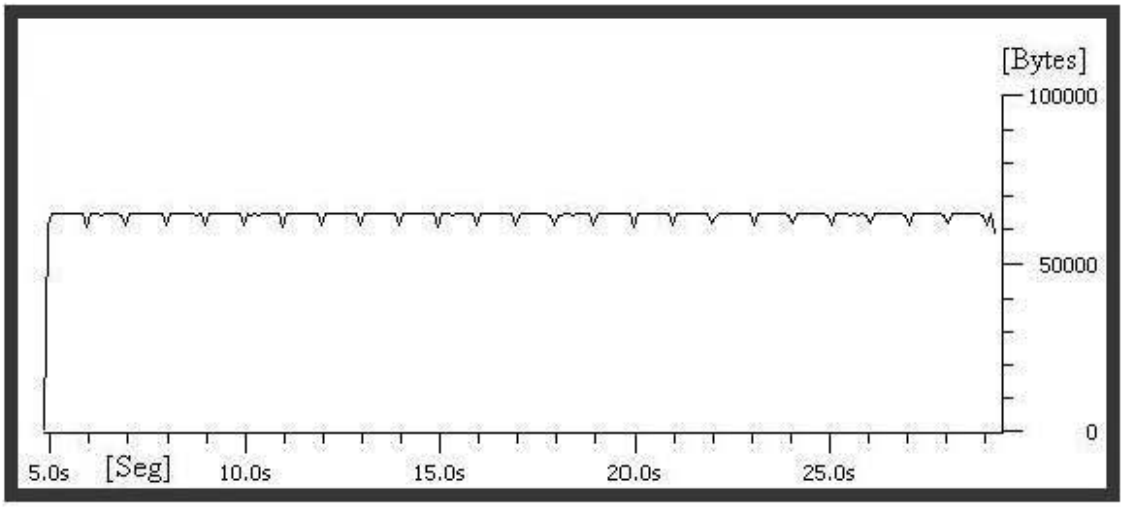

Figura 5-16 Tarjeta

Tasa de transmisión promedio: 5.092 [MBits/s]. En este caso no se nota una degradación muy grande en la gráfica

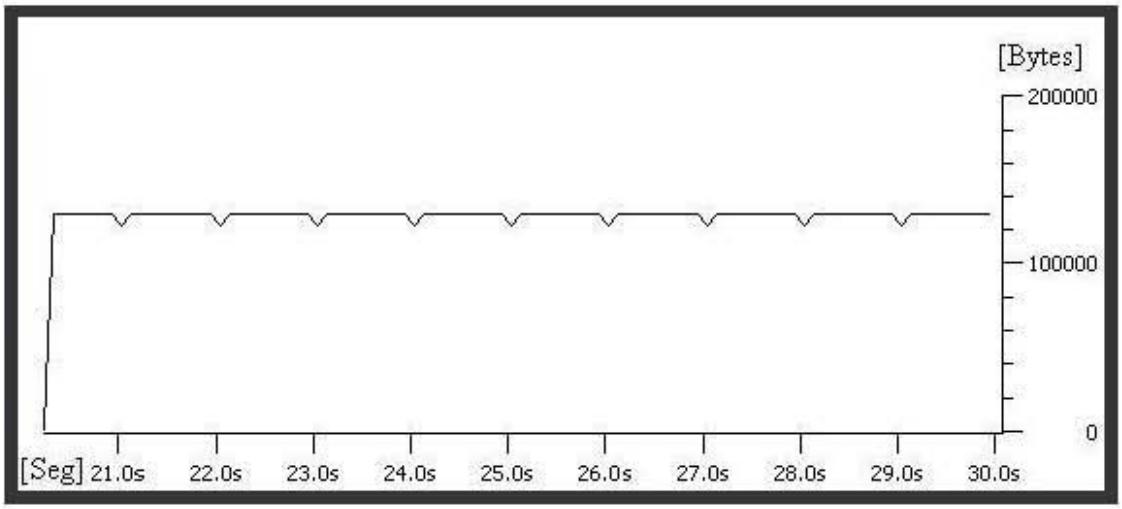

Figura 5-17 Pc a Pc

Tasa de transmisión promedio: 10.189 [MBits/s]

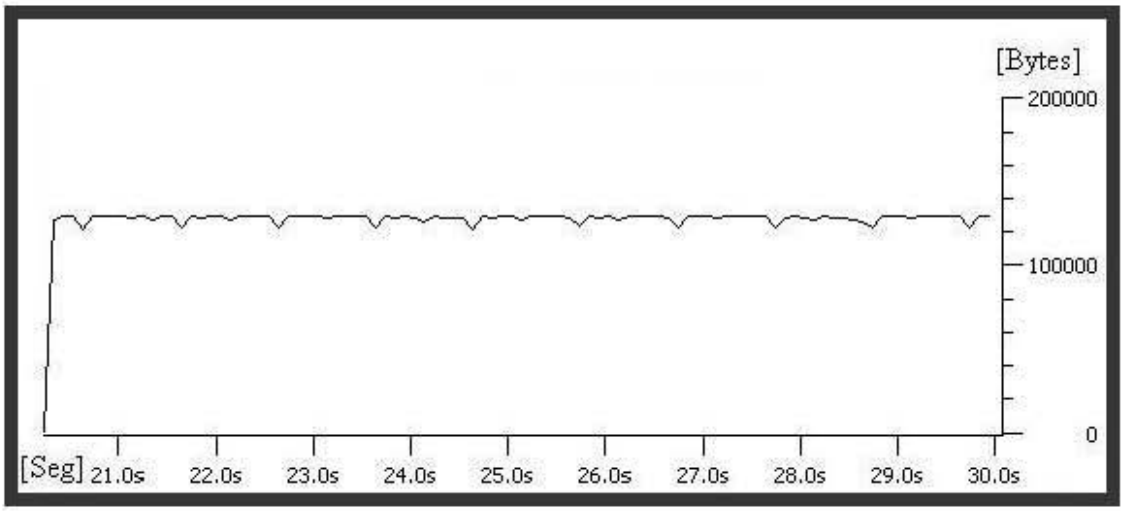

Figura 5-18 Tarjeta

Tasa de transmisión promedio: 9.606 [MBits/s]. Se empieza a notar la perdida de paquetes en la gráfica.

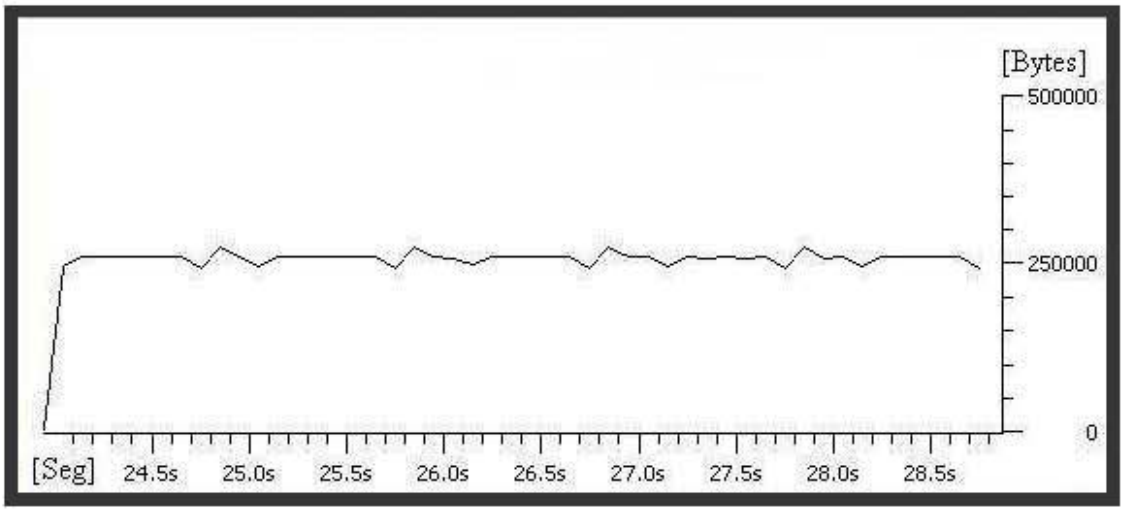

Figura 5-19 Pc a Pc

Tasa de transmisión promedio: 20.328[MBits/s]

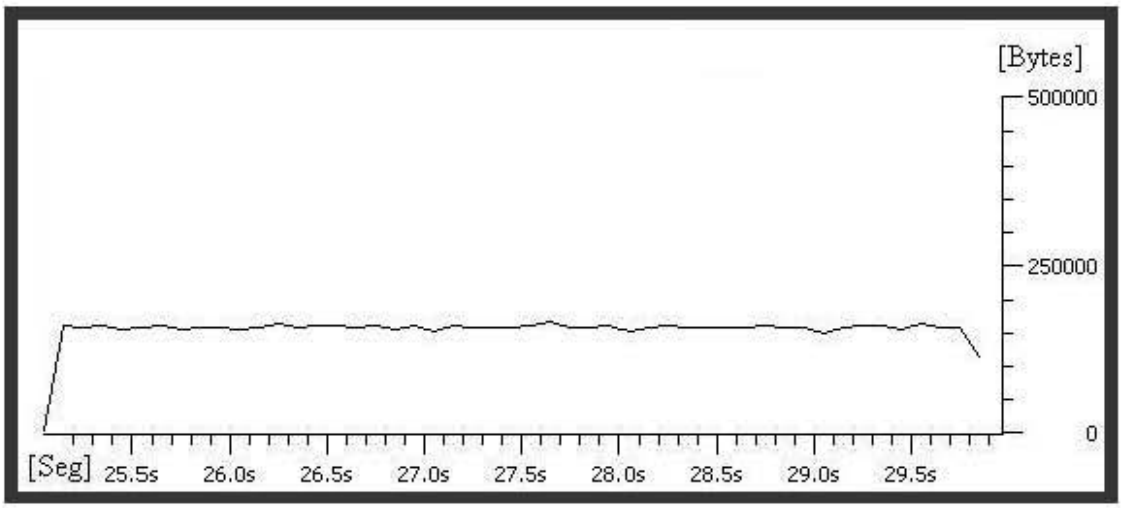

Figura 5-20 Tarjeta

Tasa de transmisión promedio: 12.468[MBits/s]. Conforme se aumenta el tráfico, la tas de salida se empieza a estabilizar y se pierden paquetes.

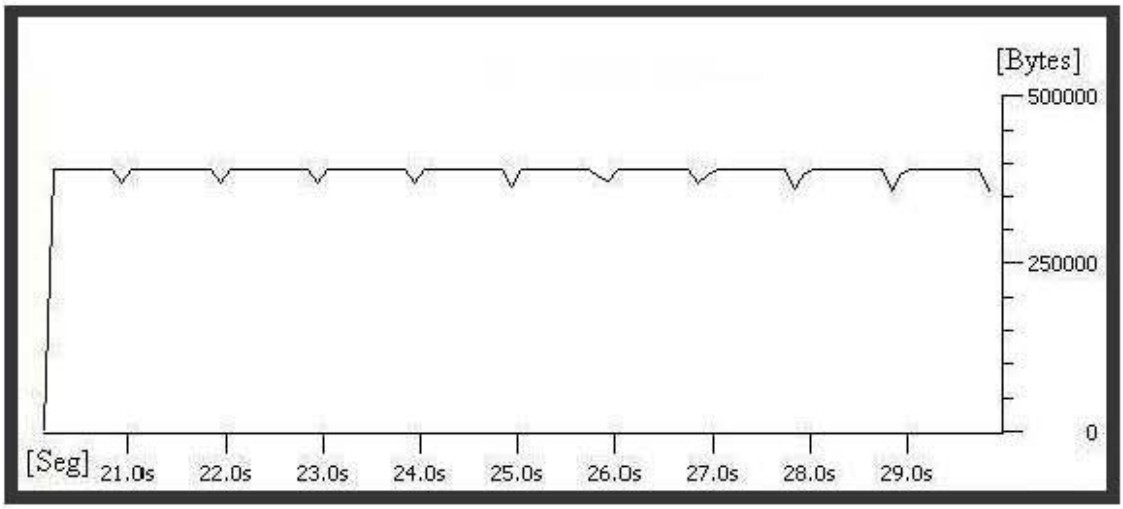

Figura 5-21 Pc a Pc

Tasa de transmisión promedio: 30.528[MBits/s]

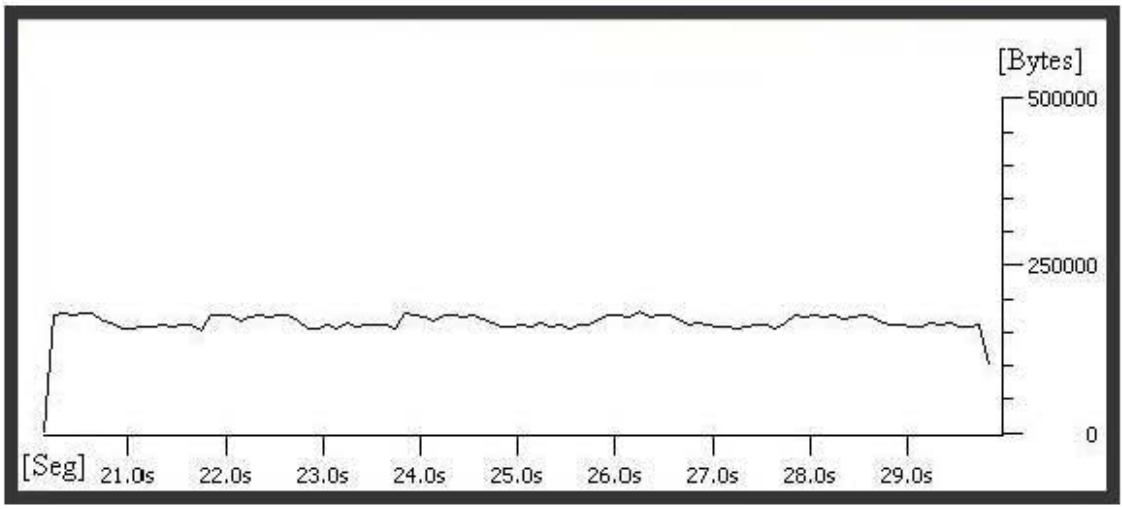

Figura 5-22 Tarjeta

Tasa de transmisión promedio: 13.083[MBits/s]. Continúa la pérdida de paquetes debido al aumento de la tasa de entrada.

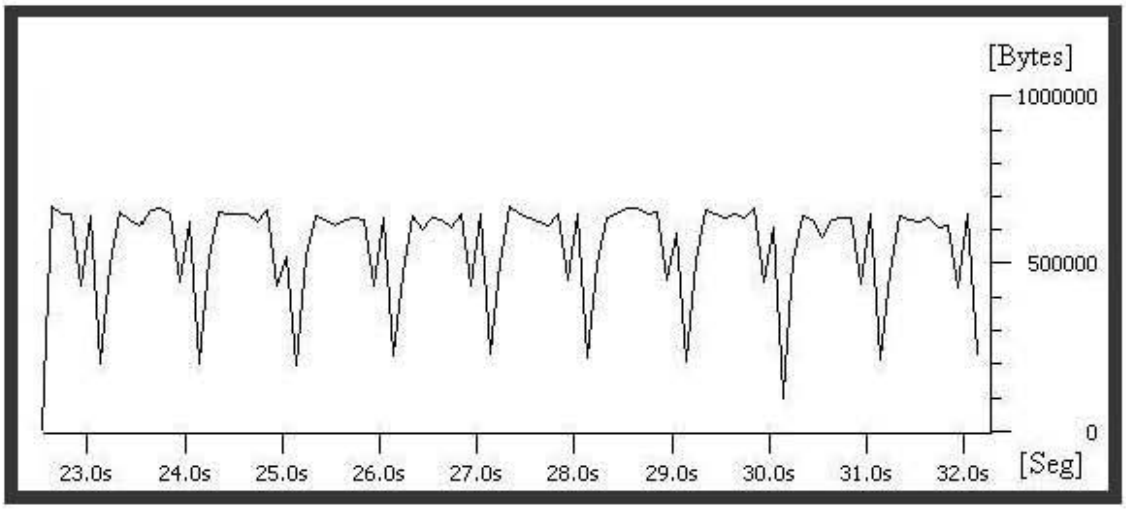

Figura 5-23 Pc a Pc

Tasa de transmisión promedio: 44.029 [MBits/s]

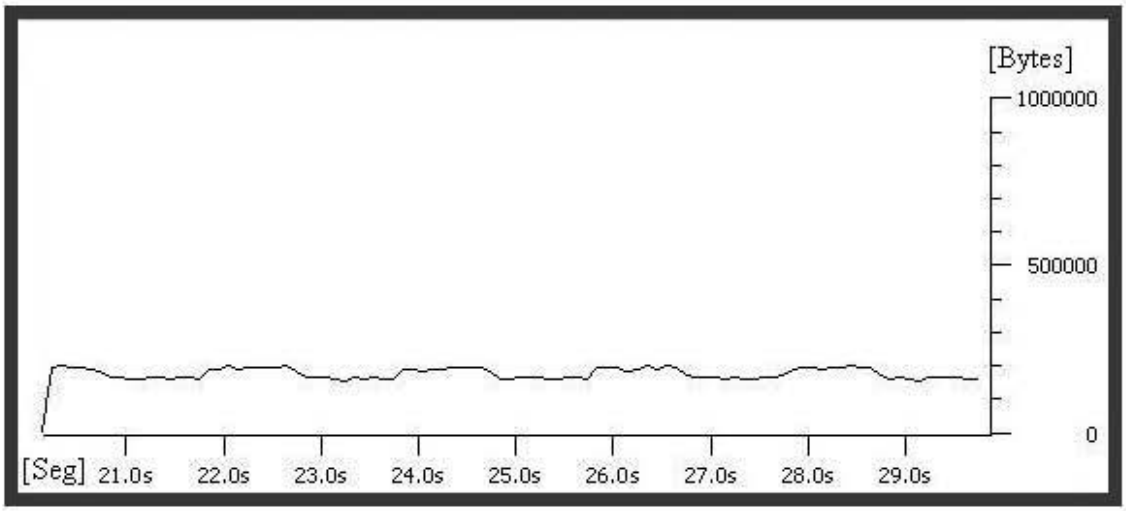

Figura 5-24 Tarjeta

Tasa de transmisión promedio: 13.708 [MBits/s]. La tasa de salida se estabiliza cercano a 13[MBits/s].

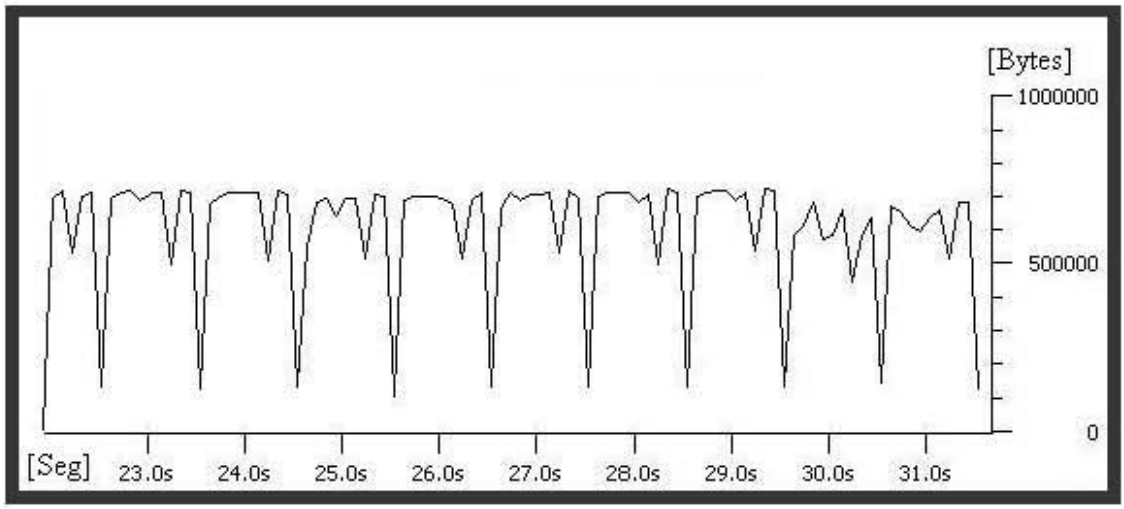

Figura 5-25 Pc a Pc

Tasa de transmisión promedio: 46.373 [MBits/s]

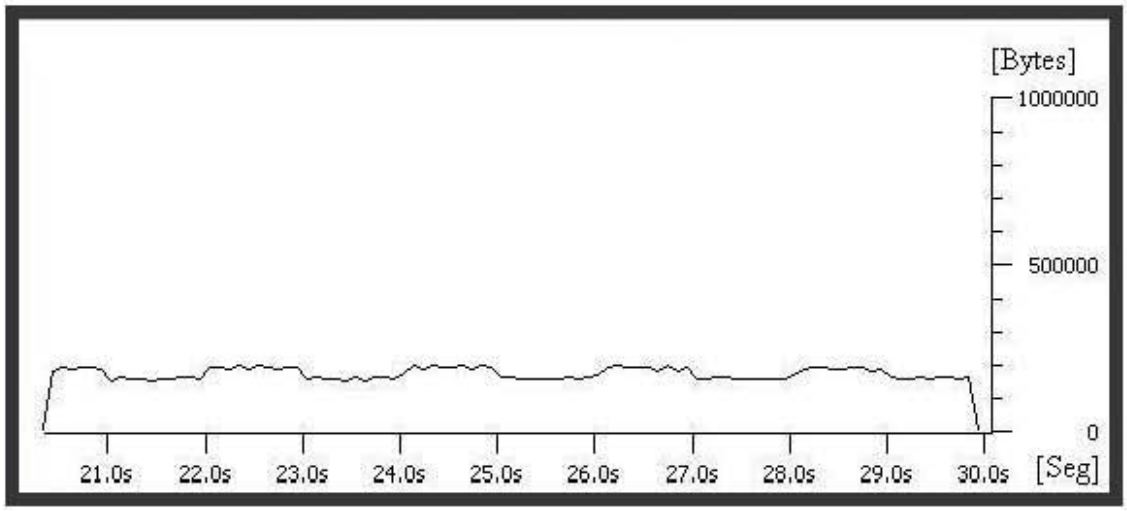

Figura 5-26 Tarjeta

Tasa de transmisión promedio: 13.674 [MBit/s]. En esta prueba final se tiene que se estabiliza arriba de los 13[MBits/s].

De los experimentos anteriores se tienen las siguientes gráficas:

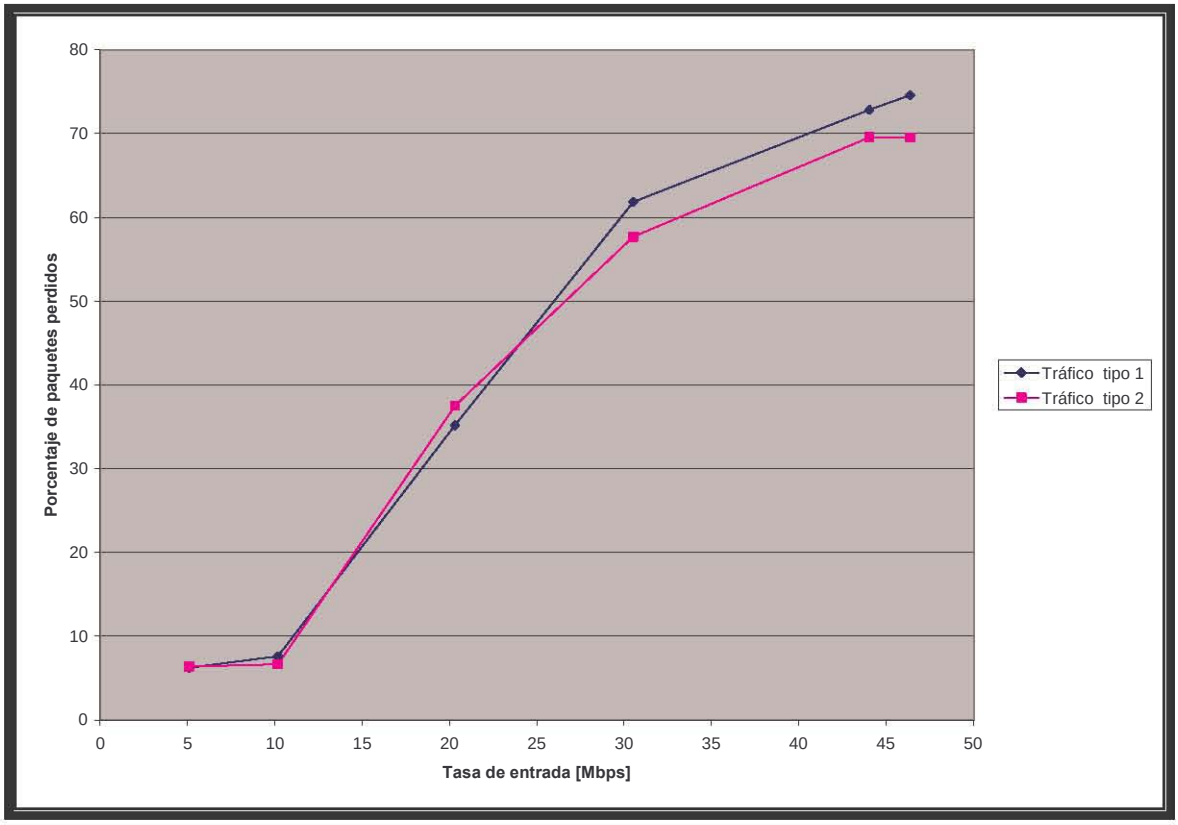

Figura 5-27 Gráfica Tasa de entrada vs Porcentaje de paquetes perdidos

En esta gráfica se consigna el porcentaje de perdidas de paquetes contra la tasa de entrada aplicado. Se verifica que al aumentar la tasa de entrada, la cantidad de paquetes perdidos aumenta, no solo en uno de los buckets, si no em ambos, hasta llegar a casi el 80 % de paquetes perdidos.

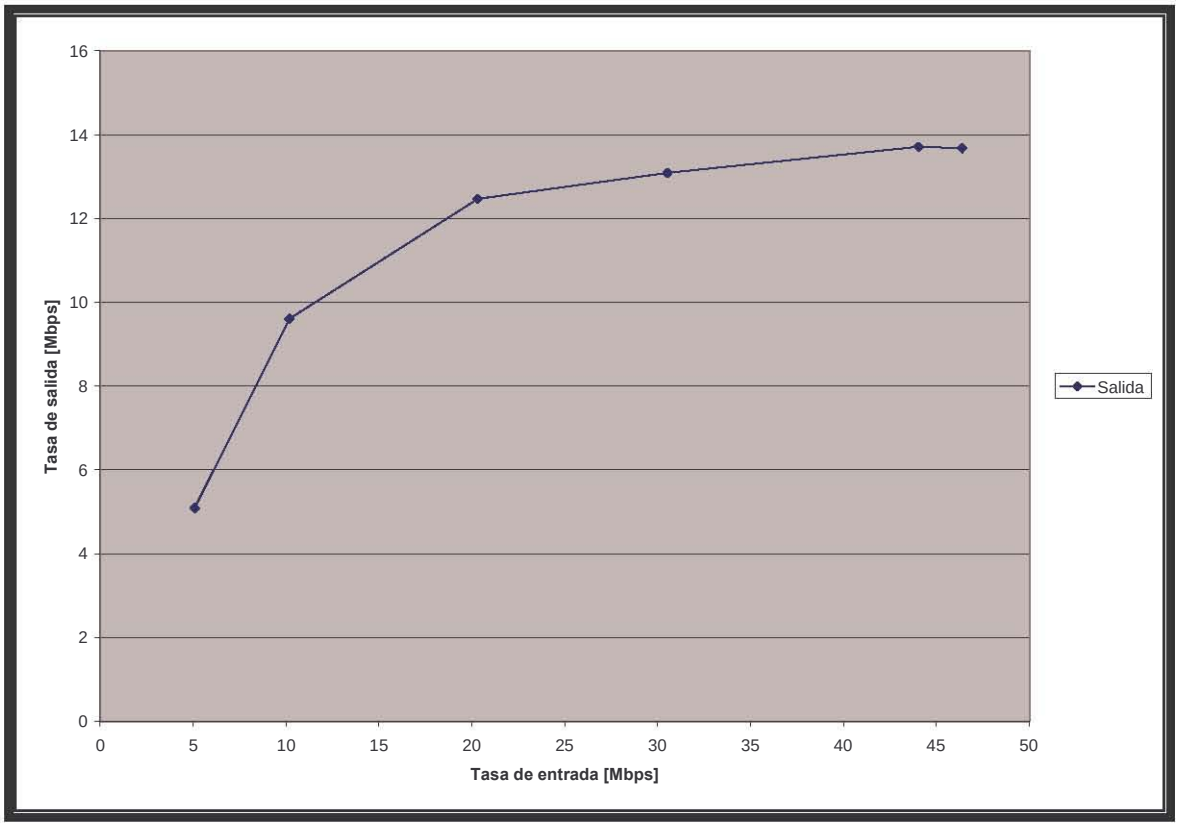

Figura 5-28 Tasa de entrada vs Tasa de salida

En esta gráfica se observa que la tasa de salida se estabiliza entre los 13 y 14 [MBits/s], tal y como se venia observando en las gráficas precedentes.

5.2. Conclusiones.

De los primeros experimentos, se observa que al tener un solo flujo con las mismas características, el sistema de red lo limita siempre a aproximadamente el 15% del flujo original. Aun si el flujo es combinado, pero con las mismas características, se conserva la relación dada. (Escenarios 1 a 3).

Sin embargo, al tener un escenario con diferentes tamaños de paquetes, ya no solo uno como en los escenarios anteriores, el comportamiento varia hasta aproximadamente un poco más del doble, 39%. (Escenario 4). Esto se debe a que al momento de llenar cada uno de los ***buckets***,

no se mantiene una entrada uniforme como en los anteriores, si no a que al haber paquetes de diversos tamaños, los tokens se agotan de una manera no uniforme.

En los escenarios 5 y 6, la entrada de paquetes fue mucho mayor que las anteriores y en un tiempo menor, aunque se le dio más tiempo al ***bucket*** para que se desahogara y volviera a llenar. Sin embargo, se mantuvo un porcentaje a la salida de entre 15 y 20% del flujo original.

El escenario 7 muestra de manera gradual lo que se ha venido observando en los anteriores. En los primeros flujos se ve que la pérdida de paquetes no es muy grande, sin embargo, al ir aumentando la tasa de entrada se empiezan a perder paquetes en mayor cantidad.

También se observa que aunque se aumente la tasa de salida, el sistema de red llega a un punto en que la salida se mantiene. En este caso se encuentra alrededor de los 14 [MBits/s]. Aunque se le sigue inyectando tráfico en mayor cantidad y tasa de transmisión, no se incrementa de igual forma la salida que se tiene después de la tarjeta, si no que se atenúa con la respectiva pérdida de paquetes.

Capítulo 6 Conclusiones.

6.1 Observaciones finales.

El implementar un sistema de red como lo es el ***traffic policer*** lleva a profundizar en diferentes aspectos, tales como los algoritmos relacionados y su posible forma de implementación. Además, hay que ampliar la visión hacia los diferentes sistemas de red y como pueden ayudar a resolver el problemas de la congestión.

El ***traffic policer*** ayuda a resolver los problemas de congestión por medio de las reglas establecidas. Estas reglas se deben de establecer de acuerdo a un análisis del flujo de tráfico que se tiene y como se quiere regular o limitar. Para esto, hay que entrar más a fondo en lo que son los diferentes protocolos y su posible forma de clasificación.

Sin embargo, para la implementación de un sistema que sea flexible y que con el tiempo pueda ser modificable, sin comprometer su eficiencia, no es cosa sencilla. Con las actuales arquitecturas de procesadores, se tienen algunas de las características mencionadas anteriormente pero se sacrifican otras. Al aparecer las tecnologías hibridas, como el caso de los procesadores de red, permiten trabajar de manera mas sencilla en la implementación del sistema de red.

Y aunque se tiene la ventaja de tener una tecnología que se encuentra en el punto medio entre flexibilidad y desempeño, al ser hibrida necesita de un estudio más en detalle. Por principio de cuentas, la arquitectura que se maneja no es igual a lo ya conocido, siendo que el procesador de red maneja una arquitectura Harvard, donde la memoria no se concentra en un solo lugar, si no que se tienen varias unidades, y no la tradicional von Neumann, donde la memoria se concentra en una sola unidad.

De ahí uno que uno de los primeros retos a enfrentar, teniendo ya la idea del sistema, es entender esta nueva propuesta de hardware, además de la forma de programación que se maneja para obtener el mayor rendimiento del mismo. Ahí es donde se encuentra otro reto, ya que la programación no es la usual. Además de usar un conjunto reducido del lenguaje C, adaptado para el hardware del procesador de red, hay que cambiar el paradigma de programación hacia uno nuevo que maneja ejecución en paralelo.

El poder tener un simulador y librerías de referencia de esta combinación de hardware y software auxilió de manera importante en el desarrollo de este trabajo, permitiendo examinar antes de probar directamente en el hardware los programas realizados el estado de registros, memorias, cambios de contexto, etc. y poder realizar una depuración más guiada de los mismos.

Los resultados obtenidos muestran la flexibilidad de este hardware, al poderse colocarse como un elemento en medio de dos computadoras sin que afectara al rendimiento de las mismas. Su presencia se hacia notar al hace funcionar el sistema de red, realizando las tareas para las que esta diseñado. Las gráficas presentadas muestran como el ***traffic policer*** realiza a cabo su tarea, descartando paquetes y manteniendo una relación entre el tráfico que llega y el que sale. Dicha relación esta en función del tamaño de la cubeta, del tamaño de y manejo de los tokens que estarán dentro de la misma y de la tasa de llenado de tokens, ya que esa tasa será parte de la forma en que los paquetes saldrán al medio. Cabe hacer notar, que si se el tráfico aumenta en demasía, el ***policer*** descartará una mayor cantidad de paquetes.

Para que se de prioridad a cierto tipo de tráfico, se planteo la característica de clasificación. Con ello se asegura que se puede discernir entre el tráfico que se considere más importante y se le quiera dar prioridad, y el que no cumpla con las características deseadas.

Hay que hacer notar que algunas de las herramientas utilizadas para este trabajo forman parte de la comunidad de software libre, como lo es el generador de paquetes, ***packETH*** y el analizador de red ***Ethereal***, y algunas aotras son plataformas comerciales que se proporcionaron junto con la donación del hardware ***ENP-2505*** por parte de ***Intel,*** como es el caso del SDK 2.0, ambiente de desarrollo, simulación y depuración para el procesador de red ***IXP1200*** que tiene la tarjeta ya mencionada.

6.2 Trabajo a futuro

El sistema mostrado solo trabaja de forma unidireccional, y dado que la tarjeta cuenta con 4 puertos disponibles, de los cuales solo se utilizaron dos, seria conveniente poder llevarlo a una forma bidireccional, estableciendo las políticas respectivas en cada caso. También hay que considerar que el procesador de red ***IXP1200*** es la primera generación de este tipo de combinación hardware software, por lo que sería de gran interés conocer las mejoras y facilidades que presenten las siguientes generaciones de esta tecnología, además que permitirían entrar a otros medios de transmisión diferentes, como lo sería el medio inalámbrico o el medio óptico.

Referencias:

[1] RTP Packet Classifier Tutorial

Nick Oliver, Cheng Lai, Raymond Liu

http://www.cs.ucla.edu/classes/fall02/cs218/l1/uploads/ixp1200.doc

[2] Hardware Architecture for

Protocol Processing

Tomas Henriksson

Linköping Studies in Science and Technology

http://www.da.isy.liu.se/pubs/tomhe/tomhe-lic.pdf

[3] IXP1200 Programming. The Microengine Coding Guide for the Intel® IXP1200 Network Processor Family

Erik J. Johnson and Aaron Kunze

Ed. Intel Press

[4] Network Systems Design Using Network Processors IXP1200 Version

Douglas Comer

Ed. Prentice Hall

[5] Computer Networks, Fourth Edition

Andrew S. Tanenenbaum,

Ed. Prentice Hall, 2003

[6] QOS IP: Concepts et Algorithmes

http://www.prism.uvsq.fr/~mea/cours/pdf/dea/RM_qos3.pdf

[7] Documento referente a los números Ethernet

http://www.mit.edu/~map/Ethernet/Ethernet.txt

[8] Internet Processor II ASIC: Rate-limiting and Traffic-policing Features
Chuck Semeria, Juniper Networks White paper Part Number : 200005-001 09/00

[9] Traffic Policing in Academic Environments
Brian Kerkhoff, Matt Kolon, Juniper Networks Application Note, Part Number: 350031-001

[10] What is Traffic Policing?

Chikalimba-Gama, Christopher http://cnx.rice.edu/content/m13373/latest

[11] 1160_04_1202_ENP-2505_datasheet.pdf

http://www.radisys.com/products/datasheet_page.cfm?productdatasheetsid=1055&

CFID=2303074&CFTOKEN=23906499

[12] packETH

http://packeth.sourceforge.net/

[13] Ethereal

http://www.ethereal.com/