TITULO

MODULO DE GESTION ACTIVA DE COLAS.

ESTUDIANTE

LAURA MARTINEZ RAMIREZ

TRABAJO DE GRADO

1329

DIRECTOR

Ing .GUSTAVO RAMIREZ M.Sc

MAYO DEL 2014

BOGOTÁ

Índice General

Lista de Acrónimos

- QoS: Calidad y Servicio
- TCP: Protocolo de control de Transmisión
- UDP: Protocolo de intercambio de datagramas
- AQM: Gestión activa de colas
- FIFO: Primero que entra y primero que sale
- FQ: Encolamiento justo
- WFQ: Encolamiento justo ponderado
- DRR: Déficit Round Robin
- CSMA/CD: Sensor portador de múltiple acceso con detección de colisiones
- LAN: Área de red local
- MAC: El control de acceso a medios
- IP: Protocolo de internet
- ARP: Protocolo de resolución de direcciones
- RARP: Protocolo de resolución de direcciones inversas
- NSAP: Dirección de servicio de red de punto de acceso
- NIDS: Sistema de detección de intrusos de una red
- ACPI: Interfaz Avanzada de configuración y energía

INTRODUCCION GENERAL

En la actualidad, las tecnologías en telecomunicaciones y las redes en general avanzan rápidamente, incrementándose el número de usuarios, el tráfico y la complejidad total.
Uno de los problemas más grandes a lo que se enfrentan el desarrollo y la investigación en este campo es el del rápido crecimiento del volumen, tamaño o necesidades de mayor velocidad en los datos transferidos así como las limitaciones tecnológicas de aumentar la velocidad de procesamiento en los routers y los demás nodos intermedios en la red. Todos esos problemas llevan a situaciones de congestión produciendo una pérdida de paquetes y retraso en la entrega de información lo que conlleva finalmente al deterioro de la calidad de servicio (QoS) demandada por el usuario.

Cuando una aplicación necesita enviar información a través de la red, lo primero que hace es encapsular esa información en un protocolo de transporte (TCP, UDP), a partir de ese momento la información pasa a llamarse datagrama. Un protocolo de transporte sirve para especificar cómo quiere que viaje el paquete. La razón para elegir un método u otro depende de lo importante que sea la información, el costo, o de la rapidez requerida.

Existen varias técnicas que pueden solucionar o reducir estos problemas de congestión. Entre esas técnicas se encuentran mecanismos de control de flujo como el que proporciona el protocolo TCP [1], o algoritmos de control y la gestión activa de colas o AQM (Active Queue Management) [2]. Los algoritmos AQM consisten en la detección temprana de congestión basándose en criterios determinísticos o probabilísticos realizando la gestión y diferenciación en la transmisión.

Los routers utilizan algoritmos de control de colas o gestión activa de colas (AQM) para decidir en qué momento y según la prioridad de los paquetes se puede colocar en una cola o pasar a otra interfaz de salida, también poseen un gestor de memoria de cola el cual controla el número de paquetes en cada una.

El dispositivo TurboCap es una tarjeta de red con puertos Gigabit Ethernet que puede enviar la información de paquetes en alta velocidad a cualquier analizador de red con un controlador para Windows.

Este proyecto de grado consiste el desarrollo de una herramienta de evaluación de calendarización de paquetes para que próximos estudiantes de Maestría en telecomunicaciones o estudiantes de la carrera en ingeniería electrónica de la Pontificia Universidad Javeriana puedan utilizarla como herramienta de realización de pruebas en congestión de datos o en proyectos encaminados a calidad y servicio.

Este proyecto de grado implementará los protocolos AQM [3] más utilizados como los FIFO [4], FQ (Fair Queueing) [5], WFQ (Weighted Fair Queueing) [6] y DRR (Deficit Round Robin) [7], con la opción de agregar más protocolos en caso de necesitarlo, para ser utilizados en pruebas de congestión de datos, desarrollo de algoritmos de

aprovisionamiento de QoS, ingeniería de tráfico y proyectos afines con fines educativos o de investigación.

OBJETIVO GENERAL

Implementar un módulo de gestión de colas para análisis de QoS basado en algoritmos FIFO, Fair Queuing, Weighted Fair Queuing y Deficit Round Robin.

OBJETIVOS ESPECIFICOS

- Describir algorítmicamente los procesos de gestión de colas de los algoritmos FIFO, Fair Queing, Weigted Fair Queuing y Deficit Round Robin.
- Diseñar el modelo de obtención de datos para el manejo estadístico de las variables características en la gestión de colas (Retardo, Diferencia de Retardo, promedio de paquetes encolados y Throughput).
- Desarrollar una interfaz de usuario que controlen las variables de operación de los algoritmos de gestión de colas propuestos.
- Implementar y validar la funcionalidad de los algoritmos de gestión de colas propuestos sobre la plataforma de desarrollo de TurboCap.

Capítulo 3

Capítulo 1

MARCO CONCEPTUAL

1. Gestión Activa de Colas

1.1 Introducción

Las redes de comunicación han sufrido durante mucho tiempo la degradación del rendimiento debido a la congestión. El rápido crecimiento, tanto en el uso y el tamaño de las redes informáticas ha provocado un renovado interés en los métodos de control de la congestión. La fuente es un método donde los algoritmos de control de flujo pueden variar la velocidad a la que el origen envía paquetes. Algunos algoritmos de control de flujo están diseñados principalmente para asegurar la presencia de buffers libres en el host de destino.

1.2 Ethernet

Todas las máquinas de una misma red tienen asociado un identificador único en la red. Este identificador se llama dirección IP y es un direccionamiento lógico, es decir, independiente de la arquitectura sobre la que este montada la red (Ethernet [7],Token Ring,Token Bus etc...).

Todos los equipos de una red Ethernet están conectados a la misma línea de transmisión y la comunicación se lleva a cabo por medio de la utilización un protocolo denominado *CSMA/CD* (*Carrier Sense Multiple Access with Collision Detect* que significa protocolo de acceso múltiple que monitorea la portadora.

Con este protocolo cualquier equipo está autorizado a transmitir a través de la línea en cualquier momento y sin ninguna prioridad entre ellos. Esta comunicación se realiza de manera simple:

- Cada equipo verifica que no haya ninguna comunicación en la línea antes de transmitir.
- Si dos equipos transmiten simultáneamente, entonces se produce una colisión (o sea, varias tramas de datos se ubican en la línea al mismo tiempo).
- Los dos equipos interrumpen su comunicación y esperan un período de tiempo aleatorio, luego una vez que el primero ha excedido el período de tiempo, puede volver a transmitir.

Este principio se basa en varias limitaciones:

- Los paquetes de datos deben tener un tamaño máximo.
- Debe existir un tiempo de espera entre dos transmisiones.

Una vez añadida la cabecera IP al paquete, solo falta por añadir las cabeceras propias del tipo de red sobre el que va a moverse el paquete, en este caso la cabecera Ethernet.

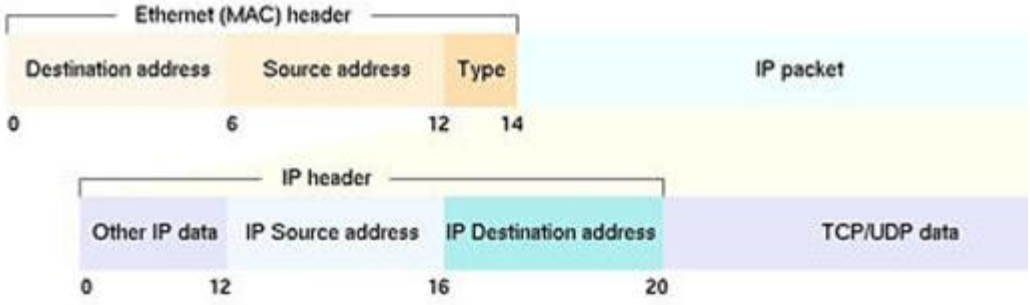

Figura 1.1. Paquetes en IP y TCP/UDP (1) tomado de documentación LIBPCAP

En una LAN Ethernet todo dispositivo de red direccionable tiene asociada una dirección Ethernet (normalmente llamada dirección MAC) que viene grabada de serie por el fabricante. Una de las cosas con las que se informa la cabecera Ethernet es la MAC de cada extremo de la comunicación. Cada máquina tiene una relación de equivalencias IP-MAC para que la red conozca cuál es la dirección MAC asociada a una IP. Esta solución existe en sistemas reales y tiene el inconveniente de ser muy costoso de mantener y poco flexible. De ahí que las redes Ethernet cuenten con un mecanismo de resolución de direcciones IP a MAC y viceversa, estos protocolos se llaman ARP y RARP. Una vez que tenemos unida toda esa información Payload + TCP + IP + Ethernet lo que en su conjunto se llama trama Ethernet se puede comenzar con la transmisión.

Un ejemplo del modelo es cuando el paquete no se dirige directamente a su destinatario sino que es copiado por el puerto en todos los cables de Red (Ver Fig. 3.2). Todos los sistemas comenzarán a leer los primeros 6 bytes de la trama en los que esta codificada la MAC destino, pero solo aquel cuya MAC coincida con la del destinatario, leerá el resto del paquete.

Como puede verse, la red no tiene ningún mecanismo para asegurar que un envió solo pueda ser leído por su legıtimo destinatario, basta un pequeño cambio de configuración en el modo en que el manejador de red captura los datos para que la maquina reciba todos los paquetes que circulan por la red.

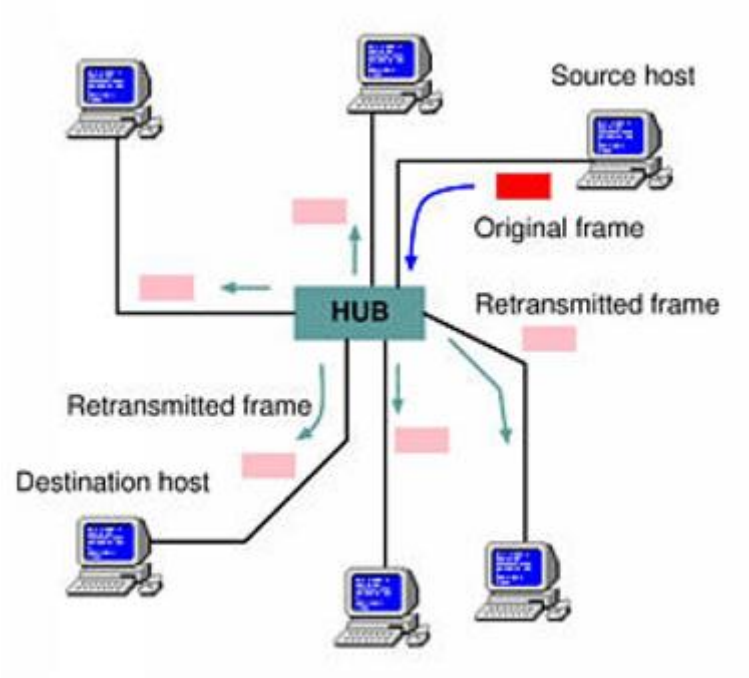

Figura 1.2 Envío de una trama ETHERNET a una LAN (2) Tomado de documentación LIBPCAP

1.3 Protocolo TCP/UDP

Son el conjunto de protocolos básicos de comunicación, en la capa de transporte, por medio de él se logra la transmisión de información en redes de ordenadores.

TCP/IP [7] proporciona la base para muchos servicios útiles, incluyendo correo electrónico, transferencia de ficheros, también pueden proporcionar chequeos de seguridad controlando las transferencias. Permite un intercambio de paquetes entre aplicaciones y puede elegirse para las que no demanden una gran cantidad de paquetes y poder operar óptimamente.

Protocolo TCP (en inglés Transmision control protocol) es orientado a la conexión y ofrece una transmisión de datos confiable. El TCP es el encargado del ensamble de datos provenientes de las capas superiores hacia datos estándares, asegurándose que la transferencia de datos se realice correctamente.

Conceptualmente, enviar un mensaje desde un programa de aplicación en una maquina hacia un programa de aplicaciones en otra, significa transferir el mensaje hacia abajo, por las capas sucesivas del programa en la maquina emisora, transferir un mensaje a través de la red y luego, transferir el mensaje hacia arriba, a través de las capas sucesivas del programa en la maquina receptora.

En la práctica, el programa es mucho más complejo de lo que se muestra en el modelo. Cada capa toma decisiones acerca de lo correcto del mensaje y selecciona

una acción apropiada con base en el tipo de mensaje o la dirección de destino. Por ejemplo, una capa en la máquina de recepción debe decidir cuándo tomar un mensaje o enviarlo a otra máquina, otra capa debe decidir qué programa de aplicación deberá recibir el mensaje.
Sé entiende que cualquier maquina conectada hacia dos redes debe tener dos módulos de interfaz de red, aunque el diagrama de la Figura 1.3 de estratificación por capas muestra sólo una capa de interfaz de red en cada máquina.

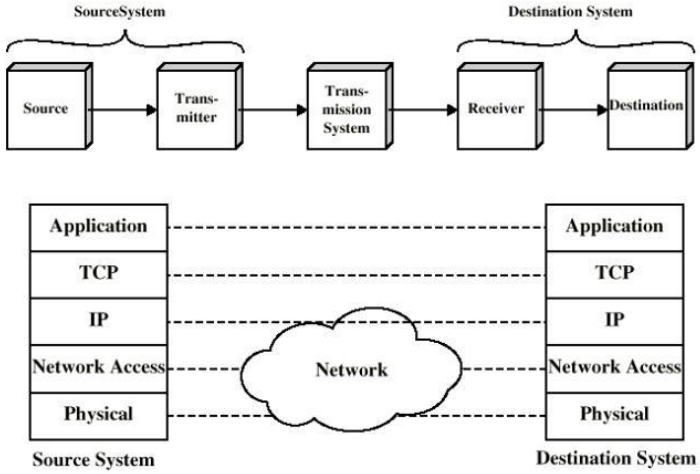

Figura 1.3 Capas de red. (6) Tomado de investigación TCP/UDP

1.4 Protocolo IP

El protocolo IP es parte de la capa de red del conjunto de protocolos TCP/IP, este permite el transporte de paquetes, aunque sin garantizar su entrega. En realidad, el protocolo IP procesa paquetes de IP de manera independiente al definir su ruta y envío en la cabecera del mensaje.
El protocolo IP determina el destinatario del mensaje mediante 3 parámetros:

- El campo de dirección IP: Dirección del equipo.
- El campo de máscara de subred: una máscara de subred que le permite al protocolo IP establecer la parte de la dirección IP que se relaciona con la red.
- El campo de pasarela predeterminada: le permite al protocolo de Internet saber a qué equipo enviar un datagrama, si el equipo de destino no se encuentra en la red de área local.

Los datos circulan en Internet en forma de paquetes, estos son datos encapsulados, es decir, datos a los que se les agrega un encabezado que contiene información sobre su transporte (como la dirección IP de destino).
Los routers analizan (y eventualmente modifican) los datos contenidos en un paquete para que puedan transitar.

En la Figura 3.4 se observa claramente cómo se encuentra distribuido en bits un paquete

<table>
<tr><td colspan="2"><--</td><td colspan="2">32 bits</td><td>--></td></tr>
<tr><td>Versión
(4 bits)</td><td>Longitud del encabezado
(4 bits)</td><td>Tipo de servicio
(8 bits)</td><td colspan="2">Longitud total
(16 bits)</td></tr>
<tr><td colspan="3">Identificación
(16 bits)</td><td>Indicador
(3 bits)</td><td>Margen del fragmento
(13 bits)</td></tr>
<tr><td colspan="2">Tiempo de vida
(8 bits)</td><td>Protocolo
(8 bits)</td><td colspan="2">Suma de comprobación del encabezado
(16 bits)</td></tr>
<tr><td colspan="5">Dirección IP de origen (32 bits)</td></tr>
<tr><td colspan="5">Dirección IP de destino (32 bits)</td></tr>
<tr><td colspan="5">Datos</td></tr>
</table>

Figura 3.4 Encabezado de un paquete

- **Versión** (4 bits): Es la versión del protocolo IP que se está utilizando IPv4 o IPv6 para verificar la validez del paquete.
- **Longitud del encabezado** o *IHL* por *Internet Header Length* (4 bits): Es la cantidad de palabras de 32 bits que componen el encabezado
- **Tipo de servicio** (8 bits): Indica la forma en la que se debe procesar el paquete.
- **Longitud total** (16 bits): Indica el tamaño total del datagrama en bytes. El tamaño de este campo es de 2 bytes, por lo tanto el tamaño total del datagrama no puede exceder los 65536 bytes.
- **Identificación:** Son campos que permiten la fragmentación de paquetes.
- **TTL** o **Tiempo de vida** (8 bits): Especifica el número máximo de routers por los que puede pasar un paquete. Por lo tanto, este campo disminuye con cada paso por un router y cuando alcanza el valor crítico de 0, el router destruye el paquete.
- **Protocolo** (8 bits): Este campo, en notación decimal, permite saber de qué protocolo proviene el datagrama.
- **Suma de comprobación del encabezado (16 bits)**: este campo contiene un valor codificado en 16 bits que permite controlar la integridad del encabezado para establecer si se ha modificado durante la transmisión. Esto se realiza de tal modo que cuando se suman los campos de encabezado, se obtenga un número con todos los bits en 1.
- **Dirección IP de origen** (32 bits): Este campo representa la dirección IP del equipo que envía y permite que el destinatario responda.
- **Dirección IP de destino** (32 bits): dirección IP del destinatario del mensaje.

El tamaño máximo de un datagrama es de 65536 bytes. Sin embargo, este valor nunca es alcanzado porque las redes no tienen suficiente capacidad para enviar paquetes tan grandes.

Además, las redes en Internet utilizan diferentes tecnologías por lo tanto el tamaño máximo de un datagrama varía según el tipo de red.

Tipo de red	MTU (en bytes)
Arpanet	1000
Ethernet	1500
FDDI	4470

Tabla 3.1 Tamaño máximo de paquetes según tipo de red

2. Protocolos para la Gestión Activa de colas

2.1 Introducción

La técnica tradicional para controlar el tamaño de las colas en los routers fue sustituida por la gestión activa de colas (AQM) la cual proporciona un mejor rendimiento en la red para la probabilidad de pérdidas de paquetes y del retardo extremo a extremo respecto a su predecesor. El esquema de la gestión activa de colas tiene un impacto significativo en la calidad de servicio de aplicaciones con requerimientos de tiempo real. A continuación se encuentra la explicación y el funcionamiento de algunos protocolos de la gestión activa de colas, estos protocolos específicos son los que se implementarán en este proyecto.

2.2 Firs In Firs Out

Describe el principio de una fila técnica de procesamiento en donde los paquetes hacen fila en el orden en el que llegan al router para el control de tráfico y en el que cada paquete se trata por igual sin tener en cuenta prioridad o tamaño.

Este protocolo tiene las siguientes restricciones:

- Debido a que trata todos los flujos de la misma manera no puede dar un buen control pues no tiene en cuenta tamaño ni prioridad.
- En los periodos de congestión este protocolo beneficia a los flujos UDP por encima de TCP debido a que el ultimo baja su tasa de paquetes en cambio UDP sigue mandando paquetes con la misma tasa.
- También existe la posibilidad de que un solo paquete que sea muy grande congestione todo el flujo y se pierda información.
- El retardo de la cola dependerá del tamaño de la cola

Por otra parte este protocolo es fácil de implementar y presenta sobrecarga baja del sistema para routers basados en software. La ventaja de una cola FIFO [1] es que proporciona un retardo predecible que los paquetes puedan experimentar a medida que pasan a través del router.

Si c es la velocidad de enlace y B es el tamaño máximo del buffer, entonces el retraso obligado, D es

$$D \leq \frac{B}{c}$$

2.3 Fair Queueing

El objetivo es obtener una repartición equitativa de todos los recursos de la red haciendo que no existan flujos dominantes que puedan ocupar todo el ancho de banda y se pierdan más paquetes. Este protocolo consiste en hacer una clasificación de cada paquete y encolarlo en su respectivo flujo. El encabezado de cada cola se elige según el algoritmo Roun Robin [2], que consiste en seleccionar todos los elementos en un grupo de manera equitativa y en un orden racional, normalmente comenzando por el primer elemento de la lista hasta llegar al último y empezando de nuevo desde el primer elemento.

Este protocolo le da la oportunidad a todos los paquetes de clasificarse en una cola y continuar con la comunicación sin importar lo dominante que sea respecto a otros paquetes de la misma red.

La principal desventaja es que cada clasificación reserva el mismo ancho de banda, por lo que en el caso de que un paquete sea muy pequeño se perderá espacio de transmisión, y en el caso en el que el paquete sea muy grande este no será enviado y se perderá la información.

Si *c* es la velocidad de enlace, cada flujo *i* obtiene un porcentaje mínimo razonable de *c* / *i* bits por segundo a la salida. Se debe tener en cuenta que si el enlace está completamente cargado, cada cola no puede conseguir más de *c* / *i* bits por segundo. Sin embargo, no todos los flujos tienen paquetes a enviar en un instante particular, una cola individual puede llegar más alta que *c*/ *i* para sí mismo.

Se supone que el tiempo de llegada de k paquetes de un flujo es Ak, el tiempo de inicio para la transmisión es Sk, y el tiempo final es Ek, respectivamente. Entonces, el tiempo necesario para enviar el paquete k_i es Ek-Sk, a se denota por Pk, es decir

$$Pk = Ek - Sk$$

Sin embargo, la necesidad de determinar la hora de inicio, depende del tiempo de llegada Ak si no hay ningún paquete que esté realizando el envío, de lo contrario, la hora de inicio tendrá que esperar hasta que la hora de finalización del paquete anterior (k - 1) se ha terminado. Eso significa que la hora de inicio del paquete k_i es el máximo de Ak y Ek-1. Por lo tanto, se puede escribir

$$Ek = \max[Ak - (Ek - 1)] + Pk$$

Mientras que el protocolo de encolamiento puede garantizar una participación equitativa mínima y con destino en la demora para cada flujo, se enfrenta a un problema diferente desde el punto de vista de aplicación debido a la variación del tamaño de los paquetes.

Consideremos por ejemplo la asignación de un único recurso entre N usuarios. Suponga que hay una cantidad μ_{total} de este recurso y que cada uno de los usuarios de solicitar una cantidad (P_i), en virtud de una asignación particular, recibe una cantidad (μ_i). La asignación justa según el criterio de equidad max-min establece que para poder realizar una asignación de encolamiento solo se puede lograr si ningún usuario recibe más de sus solicitud, esta condición se reduce a ($\mu_i = MIN\ (\mu_{fair}, P_i)$ en el ejemplo simple con (μ_{fair}), la parte justa que se establece de modo que ($\mu_{total} = \sum_{i=1}^{N} \mu_i$) Este concepto de equidad generaliza fácilmente al caso de múltiples recursos. Se debe tener en cuenta que en la definición max-min de la equidad está implícito que es la suposición de que los usuarios tienen los mismos derechos a los recursos.

2.4 Weighted Fair Queueing

Fue diseñada para mejorar Fair Queueing, específicamnte para satisfacer las demandas de distintas aplicaciones. WFQ [3] hace una repartición de flujos teniendo en cuenta el ancho de banda de cada paquete y dandole prioridad a los que menos consumen para disminuir el retraso en el buffer, luego reparte equitativamente el resto de ancho de banda en los demas paquetes.

Este protocolo realiza dos funciones al mismo tiempo: Controla los tiempos de tráfico en la parte delantera de la cola para reducir el tiempo de respuesta y comparte o distribuye el ancho de banda restante entre los flujos que lo necesitan asegurando la eficiencia de transmisión y estabilizando automaticamente la congestión de la red entre los flujos .

El protocolo WFQ tiene en cuenta los siguientes criterios:

- Cada mensaje se clasifica según tiempo de retardo, tiempo de inicio y final para dirigirse una cola o flujo distinto.
- La asignación de ancho de banda entre todos los flujos se realiza de manera distribuida para reducir el tiempo de retardo
- La dirección IP de origen por ejemplo, se utiliza como parámetro en la asignación de ancho de banda aunque éste se asigna equitativamente entre todos los flujos, entonces se realiza una repartición proporcional en el ancho de banda para los flujos con mayor precedencia IP o menor peso.

Este protocolo clasifica los flujos individuales utilizando la siguiente información la cual se toma de las cabeceras IP/TCP/UDP. Estos parámetros se utilizan como entrada para un algoritmo que produce un número de longitud fija para usarlo como índice en la cola:

- Dirección IP de origen
- Dirección IP de destino
- Número de protocolo para identificar TCP o UDP
- Tipo de servicio
- Número de la fuente TCP / UDP
- Número de destino del puerto TCP/UDP

Hay un número fijo de colas por flujo o clase, el algoritmo traduce los parámetros de flujo en un número de cola. El número de colas asignadas dinámicamente se basa en la configuración de ancho de banda de la interfaz y se puede configurar en un rango de 16 a 4096 en multiplos de 2. En la Tabla 3.2. Se encuetra la lista de colas por omisión dinámica basadas en el ancho de banda.

Configuración de ancho de banda	**Numero de colas dinámicas**
>= 64k	16
65k-128k	32
129k-256k	64
257- 512	128k
<512k	256

Tabla 3.2. Colas dinámicas según ancho de banda

2.5 Déficit Round Robín

A menudo existe un conflicto entre retraso y ancho de banda para garantizar o reducir el tiempo de procesamiento. El déficit round-robín (DRR) es acertado desde el punto de vista del proceso, logrando convertir el tiempo en una constante, es decir, *O (1)* que es el tiempo de espera la cola en lugar de una complejidad cuadrática que es el caso de los algoritmos anteriores.

DRR [4] puede ser ilustrado a través de tres parámetros clave:

- Qi, quantum representa la transmisión en bytes en proporción a la cola *i* en cada ronda
- El contador del déficit, C_i^{def} que hace el seguimiento de la diferencia entre el número de bytes que deberían haber sido enviados y el número de bytes realmente enviados en cada ciclo.
- El tamaño de buffer, Bi para la cola *i*.

Si se asigna una cola de mayor tamaño el ancho de banda del enlace de salida es mayor, En este esquema, para cada cola *i*, el algoritmo mantiene una constante Qi su (quantum) y una variable de C_i^{def} (su déficit).

En cada ciclo, el algoritmo visita cada cola no vacía en secuencia. Para cada cola *i* no vacía, el algoritmo va a transmitir tantos paquetes como sea posible de tal manera que su total *Bi* de tamaño es menor o igual a *Qi* + C_i^{def}. Si la cola *i* se vacía, el algoritmo restablece C_i^{def} a cero. Si no se restablece a cero, C_i^{def} acumulará créditos indefinidamente. Así, el déficit de cola C_i^{def} se actualiza a

$$Qi + C_i^{def} - Bi$$

Y el algoritmo se mueve en el servicio de la siguiente cola no vacía.

Como puede verse a partir del algoritmo, *Qi* + C_i^{def} representa el número máximo de bytes que se puede transmitir durante un ciclo de round-robin. Las colas que no pueden enviar con *Qi* + C_i^{def} para el siguiente ciclo se enviará con *Qi* + C_i^{def} - Bi además de la habitual *Qi* cuántica. El C_i^{def} actualizado representa los bytes a ser compensado.

Uno de los inconvenientes del DRR es cuando existe un mayor número de colas vacías de las no vacías pues se desperdicia mucho tiempo al verificar si la cola está vacía o no. Para eliminar este tipo de residuos, el algoritmo mantiene una cola separada llamada ActiveList. El ActiveList contiene una lista de los índices de colas que contienen por lo menos un paquete. Cada entrada en la ActiveList, además del índice de cola, realiza un seguimiento de su déficit no utilizado también.

3. Herramientas

3.1 Introducción

Para conseguir un óptimo procesamiento de los paquetes que viajan por las redes es necesario tener las aplicaciones y programas adecuados para así lograr una exportación e importación de la información que se encuentra en las cabeceras. Para la implementación de este proyecto se utilizó una interfaz que logra extraer con éxito la información necesaria para el análisis de la comunicación, estas herramientas son la tarjeta de red RTL8931D, la extracción de los paquetes se llevó a cabo con la librería de WinPcap y la lectura de los paquetes se hizo por medio del programa Wireshark, Todos estos conceptos se explicarán a continuación.

3.2 Turbocap

La tarjeta programable TurboCap soporta simultáneamente velocidades completas Gigabit Ethernet en dos puertos con tiempos precisos, también soporta multiples tablas de gestión; se puede utilizar para recibir datos en varias interfaces como puertos de agregación o archivos obtenidos utilizando aplicaciones en software.

TurboCap está integrado con WinPcap (Windows) y libpcap (Linux), el diagrama general de funcionamiento se observa en la Figura 1.4.

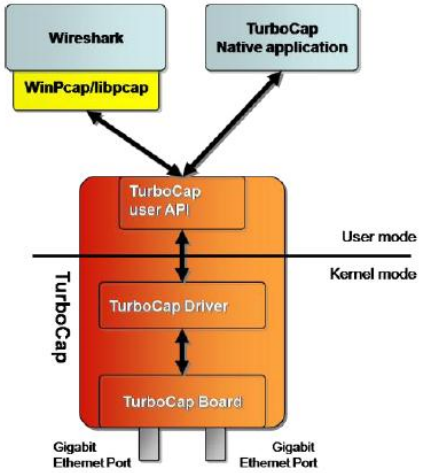

Figura 1.4. Módulos de la tarjeta TurboCap (4) Tomado de USER GUIDE

El panel de control TurboCap (Figura 1.5) proporciona una forma intuitiva para configurar los parámetros de las tablas TurboCap conectados en ese momento. Los cambios realizados en un adaptador utilizando el panel de control TurboCap se reflejarán en todas las aplicaciones que utiliza el adaptador.

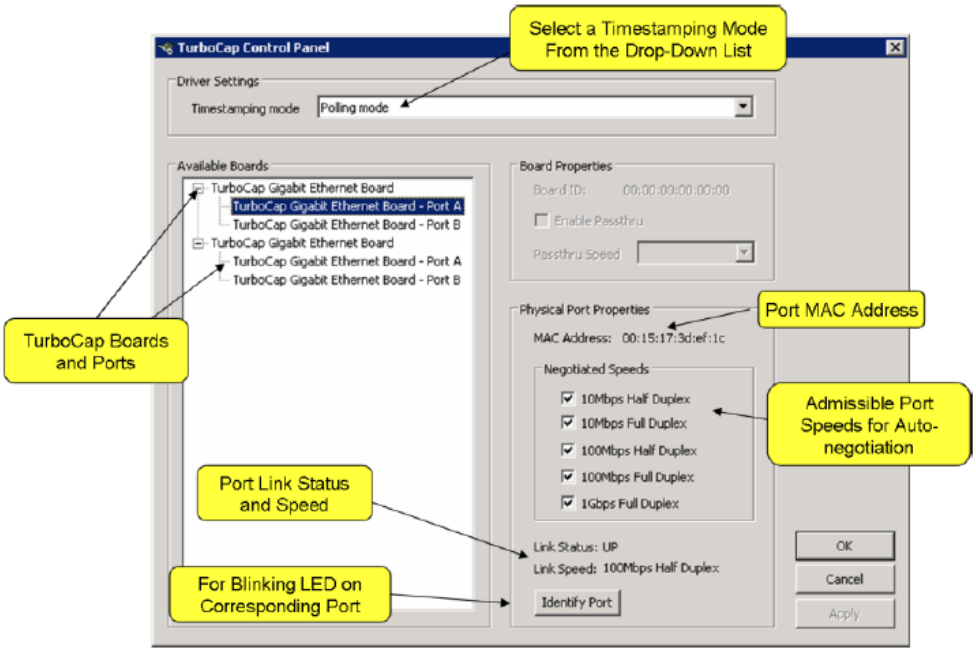

Figura 1.5 Plataforma de tarjeta TurboCap. (5) Tomado de User Guide

En la figura anterior, muestra la dirección MAC para el puerto seleccionado se muestra en el cuadro de propiedades del puerto físico, las velocidades no negociadas se eligen entre las velocidades del puerto seleccionado. Cada puerto puede tener un conjunto diferente de velocidades negociadas, para que un LED en el puerto escogido parpadee, se debe

seleccionar el botón identificar Puerto (Identify Port), de esta manera se puede asociar un puerto físico en particular con el nombre del puerto en el panel de control.

El aplicativo de la plataforma (API) permite escribir olfateando o "sniffing" herramientas de inyección de paquetes, así como el uso de las aplicaciones existentes basadas en WinPcap (Windows) o libpcap (Linux).

3.3 WinPcap

Es una biblioteca de código abierto para la captura de paquetes y análisis de red para las plataformas Win32.

Winpcap [5] es una librería de fuente abierta escrita en C que ofrece al programador una interfaz desde la que capturar paquetes en la capa de red. En la Figura 1.6 se encuentra el diagrama de bloques general para realizar cualquier implementación en esta interfaz.

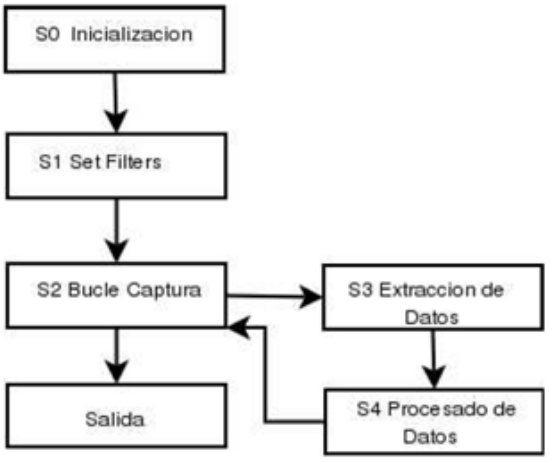

Figura 1.6. Esquematización de un programa con Libpcap (3) Tomado de Documentación LIBPCAP

Es fácil acceder a los datos en la red con este enfoque, ya que el sistema operativo hace frente con los detalles de bajo nivel (manejo de protocolo, reenvió de paquetes) y proporciona una interfaz familiar que es similar al que se utiliza para leer y escribir archivos. A veces, sin embargo, algunas aplicaciones requieren acceso directo a los paquetes de la red. Es decir, que necesitan acceso a los datos "en bruto" en la red sin la intervención de procesamiento del protocolo por el sistema operativo.

El propósito de WinPcap es dar éste tipo de acceso a las aplicaciones Win32; que proporciona facilidades para: capturar los paquetes, tanto los destinados a la máquina donde

se está ejecutando como los que se dirigen a otros Host, también filtra los paquetes de acuerdo a las normas especificadas por el usuario antes de enviarlos a la aplicación, transmite paquetes a la red recopilando información estadística sobre el tráfico de la red.

Este conjunto de capacidades se obtiene por medio de un controlador de dispositivo, que se instala dentro de la porción de red de núcleos de Win32, todas estas características se exportan a través de una interfaz de programación, fácil de explotar por las aplicaciones y disponibles en diferentes sistemas operativos.

La interfaz de programación de WinPcap puede ser utilizado por muchos tipos de herramientas de red para el análisis, solución de problemas, seguridad y vigilancia de los paquetes. En particular, las herramientas clásicas que se basan en WinPcap son:

- Analizadores de red y protocolos
- Monitores de red
- Registradores de tráfico
- Generadores de tráfico
- Puentes a nivel de usuario y routers
- Sistemas de detección de intrusiones de red (NIDS)
- Escáneres de red
- Herramientas de seguridad

WinPcap recibe y envía los paquetes independientemente de los protocolos TCP-IP. Esto significa que no es capaz de bloquear, filtrar o manipular el tráfico generado por otros programas en la misma máquina: simplemente identifica (sniff) los paquetes que se reciban por una interfaz de red. Por lo tanto, no proporciona el apoyo adecuado para aplicaciones como conformadores de tráfico QoS, programadores y firewalls personales.

3.4 Wireshark

Es un analizador de protocolos utilizado para realizar análisis y solucionar problemas de redes de comunicaciones, también funciona para el desarrollo de software y protocolos, y como una herramienta didáctica para investigación académica, pues añade una interfaz gráfica y muchas opciones de organización y filtrado de información. Así, permite observar todo el tráfico que pasa a través de una red estableciendo una configuración en modo promiscuo.

Al ejecutar wireshark [6] se abre una ventana con diferentes opciones como la lista de interfaz, de acuerdo a lo que se va a realizar se eligen las opciones como se muestra en la Figura 1.7.

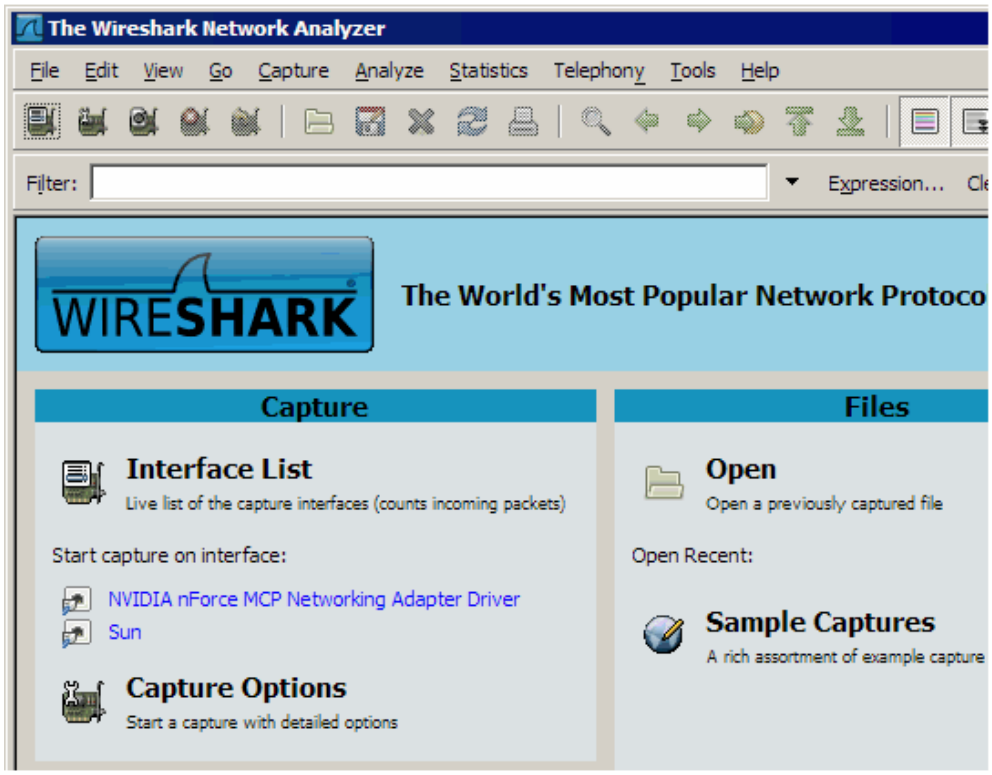

Figura 1.7 Menú principal de Wireshark (6) Tomado de Manual para usar Wireshark

Wireshark está disponible para Linux y Windows, captura paquetes de datos en tiempo real de cualquier interfaz de red, muestra la información del protocolo detalladamente abriendo y guardando los paquetes capturados. Este programa es capaz de importar y exportar datos de paquetes desde y hacia muchos otros programas de captura en el caso de este trabajo de grado Winpcap.

En la Figura 1.8 se encuentra un ejemplo de como muestra el programa los paquetes encontrados en la red, también muestra como despliega y guarda la información de paquetes.

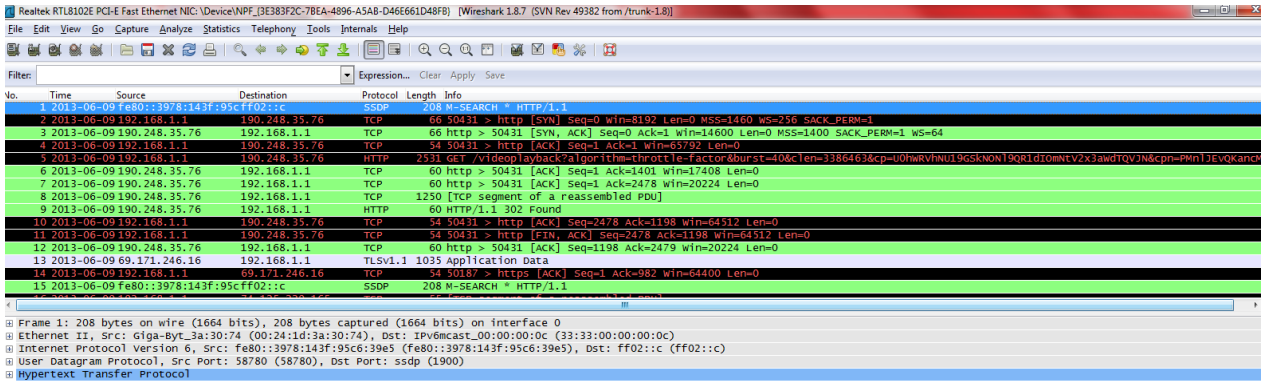

Figura 1.8 Ejemplo de implementación del programa Wireshark (7) Tomado de Manual

3.5 Tarjeta de red RTL8139D

La tarjeta RTL8139D es un controlador altamente integrado y rentable de un solo chip Fast Ethernet que proporciona un rendimiento de 32 bits, y el pleno cumplimiento con los estándares IEEE 802.3u. Posee una interfaz de gestión de configuración y energía (ACPI), también es compatible con los pins ROM de arranque compartido. Además de la función de ACPI, la RTL8139D también es compatible con la reactivación remota, es capaz de realizar un reinicio interno a través de la aplicación de potencia auxiliar.

Para proporcionar un bajo costo, la RTL8139D es capaz de utilizar un cristal de 25 MHz u OSC como su fuente de reloj interno. Mantiene los costos de mantenimiento de redes bajos y elimina las barreras de uso. Es la manera más fácil de actualizar una red de 10 a 100Mbps. Para mejorar la compatibilidad con productos de otras marcas.

Función de transmisión: La CPU inicia una transmisión mediante el almacenamiento de un paquete completo de datos en uno de los descriptores en la memoria principal. Cuando todo el paquete se ha transferido a la memoria intermedia de Tx, el RTL8139D es instruido para mover los datos desde el buffer de Tx a la FIFO de transmisión interna en el modo maestro de bus PCI. Cuando la transmisión FIFO contiene un paquete completo o se llena hasta el nivel umbral programado, el RTL8139D comienza la transmisión de paquetes.

Función de recepción: El paquete de entrada se coloca en el RTL8139D Rx FIFO. Al mismo tiempo, la RTL8139D realiza el filtrado de paquetes de multidifusión de acuerdo a sus algoritmos. Cuando la cantidad de datos en el Rx FIFO alcanza el nivel definido en la recepción del registro de Configuración, la RTL8139D pide al PCI que comience a transferir los datos a la memoria intermedia de Rx en el modo maestro de bus PCI.

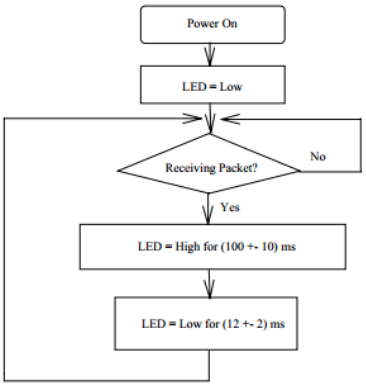

Figura 1.9 Diagrama de flujo RTL8931D (8) Tomado de Datasheet

Capítulo 2

ESPECIFICACIONES

Teniendo en cuenta que en el momento de implementar cualquier tipo de aplicación de envío de datos, a través de routers es necesario conocer el tipo de control de colas y cuál de los algoritmos es el adecuado para el tipo de aplicación, se implementaron los protocolos de control de colas en una tarjeta RTL8931D

La tarjeta de red admite la recepción y transmisión de paquetes a partir de cada puerto Ethernet y ofrece estos paquetes a una aplicación que se ejecuta en el sistema operativo del Host, luego se transmite a uno de los puertos.

Cada paquete recibido se entrega con una marca de tiempo que corresponde con el momento en que se completa la recepción total del paquete, es decir, cuando el último byte del paquete fue recibido. La marca de tiempo se representa como un valor entero de 64 bits sin signo, que representa el número de nanosegundos transcurridos

La implementación de colas se realizó en software, en donde se encuentran todos los algoritmos de gestión activa de colas (AQM) es decir, FIFO, FQ (Fair Queueing), WFQ (Weighted Fair Queueing) y DRR (Deficit Round Robin).

Para obtener información valiosa en el control de colas es necesario conocer algunos parámetros respecto a tiempos, tamaños de paquete e información relevante de las cabeceras de los paquetes, es por esta razón que el algoritmo arroja algunos resultados para que los usuarios realicen un análisis completo de la congestión existente en el router. En el caso de la validación del algoritmo implementado se realizó la comparación de los resultados con el modelo matemático de los algoritmos ya existentes.

Los resultados que arroja el programa son: retardo por cola , diferencia en el retardo por cola , throughput por cola throughput total tasa de paquetes entrantes por cola , tasa de bits entrantes por cola, tamaño promedio de cola, tamaño promedio de paquetes y clasificación por flujos.

El usuario a través de una interfaz virtual tiene la opción de escoger cuál de los protocolos desea probar, sin embargo, la configuración del algoritmo será modificado directamente desde el software siendo transparente para el usuario y notificando los resultados anteriormente mencionados.

En la Figura 2.1 Se encuentra representado el diagrama en bloques del proyecto en donde inicialmente está la interacción directa con el usuario, también controla el proceso de aplicación, ubicada en la siguiente etapa la cual gestiona las colas siguiendo por una base de datos que posee las estadísticas de los paquetes, estos son trasmitidos pasando por las colas y también almacena las estadísticas que validan los algoritmos de AQM. Luego los paquetes se direccionan a la interfaz de aplicación de la tarjeta de red.

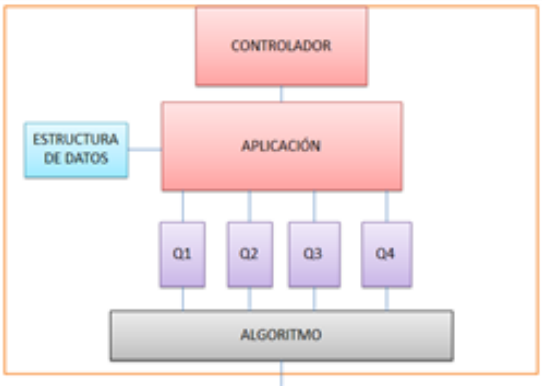

Figura 2.1. Diagrama en bloques del Proyecto

Las pruebas que se realizaron para garantizar el buen funcionamiento del proyecto se representan en la Figura 2.2 La entrada consiste en un generador de tráfico que va a la tarjeta, de máximo 4 colas luego se realiza la medición.

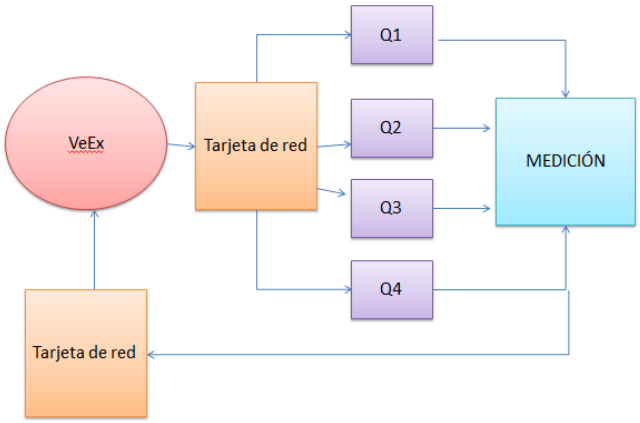

Figura 2.2. Diagrama de pruebas

Para obtener información valiosa en el control de colas es necesario conocer algunos parámetros respecto a tiempos, tamaños de paquete e información relevante de las cabeceras de los paquetes por ejemplo, es por esta razón que el algoritmo que se desarrolló en este proyecto arrojó algunos resultados para que los usuarios realicen un análisis completo de la congestión existente en el puerto. En el caso de la validación del algoritmo implementado se realizó la comparación de los resultados con el modelo matemático de los algoritmos ya existentes.

Los resultados que arroja el programa son: retardo por cola, diferencia en el retardo por cola, throughput por cola, throughput total, tasa de paquetes entrantes por cola, tasa de bits entrantes por cola, tamaño promedio de cola , tamaño promedio de paquetes y clasificación por flujos.

Capítulo 3

DESARROLLO

1. Introducción

En este capítulo se realizará una explicación detallada de cómo se desarrolló cada bloque de este proyecto junto con parte del algoritmo diseñado de cada protocolo, demostración de la interfaz de usuario, las funciones utilizadas para el envío y captura de paquetes y un panorama general de todo el algoritmo.

2. Panorama General

En la Figura 3.1 se encuentra representado todo el proceso que sigue el algoritmo de este proyecto iniciando por la escogencia del protocolo a probar, siguiendo por la captura de paquetes, su debido análisis según las especificaciones del proyecto, el encolamiento de estos y el envió de los paquetes según el protocolo escogido en el inicio.

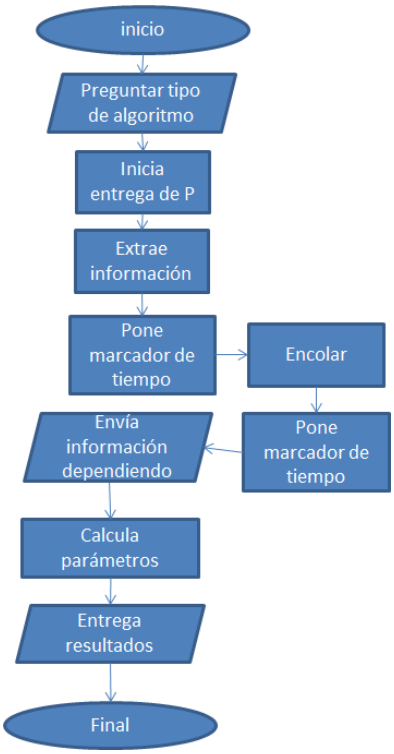

Figura 3.1. Diagrama de flujo de la Aplicación

Para que el usuario pueda manejar correctamente la implementación fue necesario desarrollar una interfaz gráfica adecuada, en donde se especificarán los parámetros necesarios según las necesidades del proceso de investigación o análisis de cualquier

protocolo especificado en los objetivos de este proyecto. En la Figura 3.2 está representado el menú principal de la interfaz de usuario realizada en este proyecto.

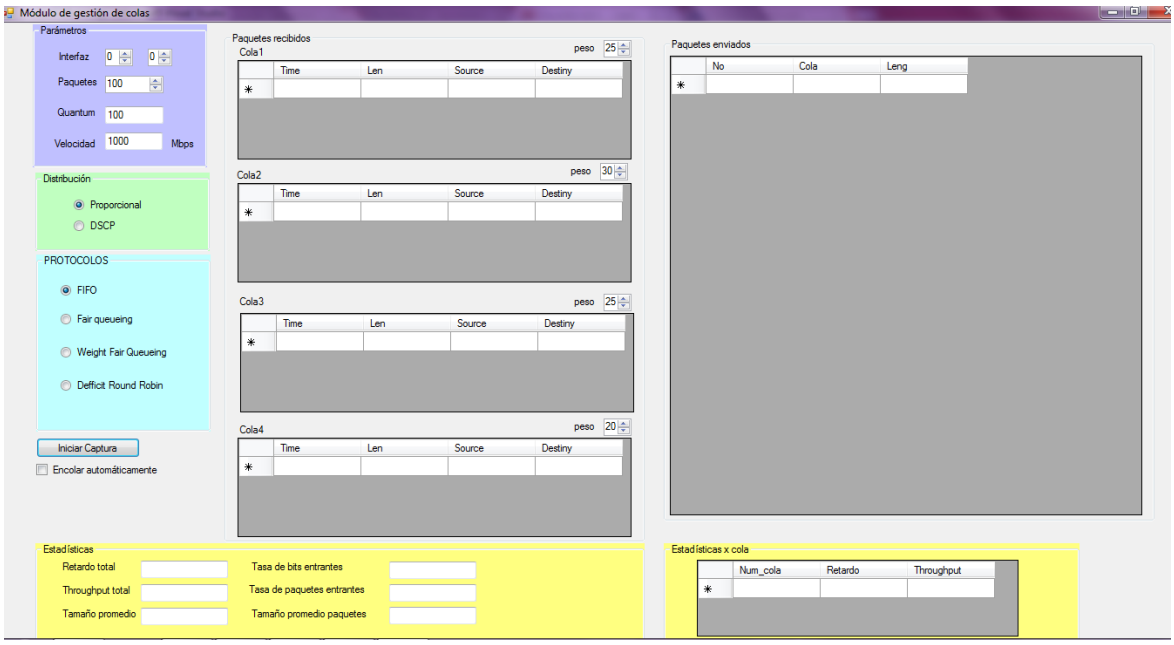

Figura 3.2 Interfaz gráfica para el usuario

Parámetros: En el cuadro de color azul en la parte superior izquierda se encuentran los parámetros que el usuario debe variar según sea su escogencia, los parámetros son el número de la interfaces a escoger para la captura de datos y para el envío de estos. Para tener la seguridad del número de interfaz que corresponde a los puertos es necesario revisar las interfaces existentes con el programa Wireshark. También está el número de paquetes que el usuario desea capturar y el valor aproximado del quantium que solamente se tendrá en cuenta cuando se esté realizando la prueba del Deficit Round Robin, y la velocidad con la que se desea enviar los paquetes. Por último en caso de escoger el protocolo Weighted Fair Queueing es necesario que el usuario realice una repartición de ancho de banda en la esquina superior derecha de cata cola.

Protocolos: El siguiente cuadro en la imagen es un listado de cuatro protocolos para que el usuario tenga la posibilidad de escoger entre estos, es importante aclarar que solo es posible escoger un protocolo a la vez.

Botones: Justo debajo del cuadro de protocolos se encuentran los botones, el primero es el de "Iniciar captura" el cual oprimirá el usuario cuando desee iniciar la captura de los paquetes, este botón debe iniciarse después de escoger los parámetros y el protocolo. Debajo de este botón se encuentra una selección para que inicie el envío de paquetes inmediatamente después de terminar su captura.

Estadísticas: En el cuadro de color amarillo se encuentran todas las estadísticas que necesita el usuario para realizar un buen análisis de cada protocolo: las estadísticas son Retardo por cola el cual es una tabla en donde muestra el retardo de cada paquete en cada

cola y se observa en la tabla del costado derecho en la parte inferior, Troughput total, Tamaño promedio, Retardo total, Tasa de paquetes entrantes, Tamaño promedio de paquetes, Troughput por cola que también está ubicado en la tabla, Tasa de bits entrantes y clasificación por flujos.

Paquetes recibidos: El siguiente cuadro son cuatro tablas en donde se encuentra alguna información útil de cada paquete recibido después de la captura para poder realizar el análisis, en la tabla se encuentra el tamaño del paquete, la dirección de destino y la dirección fuente. Es importante resaltar que cada tabla representa cada una de las colas, por ejemplo en el protocolo FIFO solamente se maneja una cola por lo que todos los paquetes capturados se ubicarían en la primera tabla nada más.

El proyecto tiene dos opciones de filtrado en la captura de paquetes, el primero simplemente captura los paquetes y los distribuye en tamaños iguales, es decir si se capturan 100 paquetes en cada tabla se ubicarán 25 paquetes en cada cola. La siguiente opción de filtrado se realiza con los 6 bits de DSCP y esta distribución se explicará más adelante en la sección 3.7.

Paquetes enviados: Este último cuadro ubicado al lado derecho es una tabla que se llenará cuando empiece la transmisión de paquetes y mostrará el orden en el que se envían los paquetes según el protocolo escogido por el usuario.

3. FIFO

La Figura 5.3. Es la representación gráfica del proceso que debe seguir un paquete si se ejecuta el algoritmo FIFO en el programa. El estado inicial (Estado 1) esperará a que llegue un paquete para pasar al siguiente estado, de lo contrario permanecerá allí; el algoritmo permanecerá en el estado 2 hasta que se complete la transferencia del paquete, en el caso en el que el paquete sea de mayor tamaño que la memoria o el espacio de almacenamiento este se eliminara y pasara al estado inicial para esperar a que llegue el siguiente paquete, en caso de que el paquete llegue por completo pasa al Estado 3 en donde se toman los datos necesarios para realizar el análisis del algoritmo especificado en los objetivos y en la sección de especificaciones, almacena los datos y se dirige al siguiente estado para enviar el paquete a su destino y volver al estado inicial con el siguiente paquete.

Es importante tener en cuenta en esta máquina de estados que el segundo paquete debe esperar a que el proceso se concluya con el primer paquete y el tercero debe esperar a que se envíen los primeros dos por lo que el tiempo promedio de espera que tengan los procesos depende por completo del orden en que llegan los paquetes a ejecutarse, en caso de que los paquetes no se encolen estos se descartan.

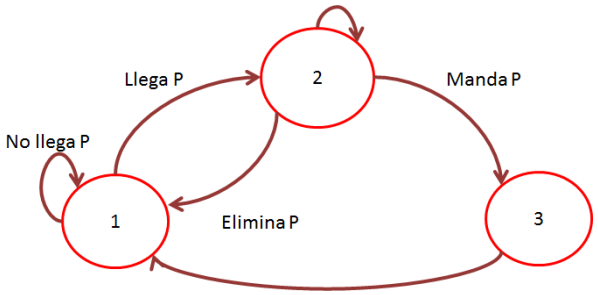

Figura 3.3 Máquina de estados FIFO

En la Figura 3.4 Se encuentra el diagrama de flujo utilizado para realizar el algoritmo del protocolo First In First Out

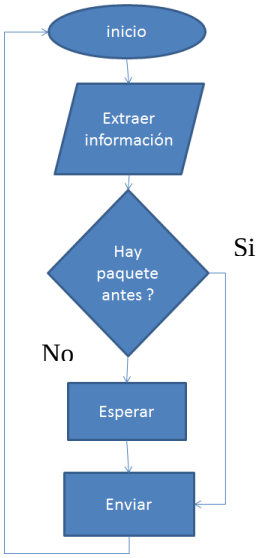

Figura 3.4 Diagrama de flujo de FIFO

Para la realización del algoritmo primero se realiza la formación de 2 tablas en donde guarda la información más relevante de cada paquete entrante como lo es el tiempo, el tamaño y la direción IP de la fuente y el destino. Las direcciones IP se guardan en una tabla oculta denominada "dataGridView1" para utilizar esa informaión más adelante. La informaión que se muestra en pantalla para el usuario será guardada en "dataGridView3" y "radioButton1" significa que el código trabaja sobre el botón 1 es decir sobre la selección del protocolo FIFO.

Inicialización:

```
if (radioButton1->Checked == true) {
        cantp=dataGridView1->RowCount;
         enviados=0;
        promediop=0;
         for(p=0;p<cantp;p++){
```

```
promediop = promediop + Convert::ToInt32(dataGridView1->Rows[p]->Cells[1]-
>Value);
        }
}
```

En este proceso simplemente guarda la infrmación necesaria para realizar el análisis y procede al armado del paquete y llenado para comenzar con el envío.

```
Llenar el  resto del paquete
for(i=12;i<len;i++){
        packet[i]=i%256;
}
/* Enviar el paquete */
if (pcap_sendpacket(fp, packet, 100 /* Tamaño */) != 0){
MessageBox::Show("\nError sending the packet: \n");
}
else{
        enviados++;
    }
}
```

En este fragmento se realiza un llenado del paquete a enviar, con la dirección de destino , direción de origen y tamaño guardada en la tabla se reconstruye el paquete para enviarse.

Como se puede observar en ambos fragmentos el paquete capturado es enviado en el orden de llegada y solamente se extrae la información necesaria para realizar las estadísticas.

4. Fair Queueing

La Figura 3.5 es la máquina de estados del protocolo Fair Queueing en donde en el Estado inicial espera a que llegue el paquete, cuando esto sucede pasa al estado 2, en donde el algoritmo realiza una clasificación de tamaño con el sistema round robin, y pasa al siguiente estado para encolarlo según los resultados de la clasificación, después envía los paquetes en el estado 4.

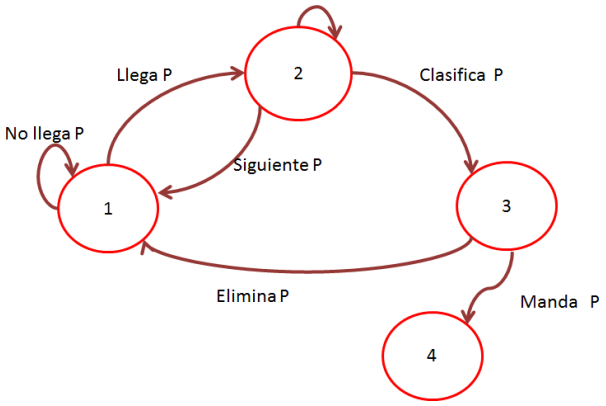

Figura 3.5 Máquina de estados Fair Queueing

En la Figura 3.6 se encuentra el diagrama en bloques del protocolo FQ, en este diagrama se especifica los pasos a seguir para el encolado más no las ecuaciones o cálculos que realiza el protocolo para lograr la clasificación del encolamiento.

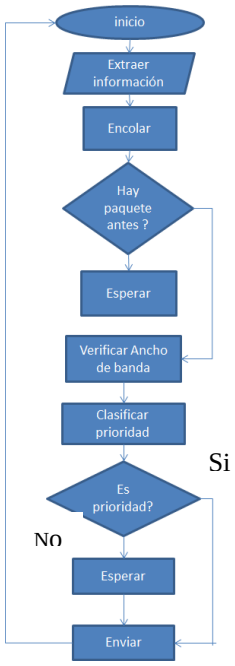

Figura 3.6 Diagrama de flujo de FQ

El servicio de Fair queueing proporciona una repartición equitativa de los paquetes a enviar, pero no consigue una justa asignación de ancho de banda debido a las variaciones en los tamaños de paquetes. Sea *R (t)* el número de rondas realizadas en la disciplina round-robin hasta el momento *t*, sea $N_{ac}(t)$ el número de conversación activa. Entonces $\partial R/\partial t = \mu/N_{ac}(t)$, donde μ es la velocidad de la línea de salida (suponemos μ = 1). Un paquete de tamaño P cuyo primer bit empieza a un tiempo (t_0) tendrá sus rondas P de servicio pasado poco más tarde, en el tiempo *t* tal que $R(t) = R(t_0) + P$. Sea t_i^{α} el tiempo que el paquete *i*

perteneciente a la conversación α llega a la puerta de entrada, y se define los números S_i^α y F_i^α como el valor de *R (t)* cuando el servicio terminó. P_i^α Denota el tamaño del paquete, la siguiente relación: $F_i^\alpha = S_i^\alpha + P_i^\alpha$ y $S_i^\alpha = MAX(F_{i-1}^\alpha, R(t_i^\alpha))$, el orden de los valores F_i^α es el mismo que el orden de los tiempos de envío de los paquetes.

Para emular este algoritmo en un esquema práctico de transmisión se realizará paquete por paquete. Se debe tener en cuenta que la función $N_{ac}(t)$ y las cantidades S_i^α y F_i^α dependen sólo de los tiempos de llegada de paquetes t_i^α y no en los tiempos de transmisión de paquetes reales, siempre y cuando se define una conversación siendo activa sí ($R(t) \leq F_i^\alpha para\ i = MAX(j|t_i^\alpha \leq t|$).

El algoritmo de transmisión de paquetes por paquete en este proyecto se define simplemente por la regla de que siempre que un paquete termina la transmisión, el siguiente paquete enviado es el que tiene el valor más pequeño de F_i^α. El requisito de continuidad es la demanda de que las prioridades de transmisión relativas dependen continuamente en los tiempos de llegada de paquetes. El hecho de que la F_i^α depende de forma continua en el t_i^α significa que el algoritmo satisface este requisito de continuidad.

La asignación de la prontitud debe basarse únicamente en datos que ya están en la puerta de enlace. Una de estas estrategias de asignación es dar menos retraso a los usuarios que utilizan menos de su cuota justa de ancho de banda.

Cuando la cola está llena y llega un nuevo paquete, el último paquete de la conversación que actualmente utiliza más espacio de búfer se deja caer. Se ha optado por dejar las cantidades F_i^α y S_i^α sin cambios cuando se cae un paquete.

Calculo de Si y Fi por cola

cola1

```
if(radioButton2->Checked == true){
        r=(float)(Convert::ToDouble(textBox12->Text)/4)
}
filas_fq=0;
promediop =0;
enviados=0;
for (i=0;i<tam_cola[0];i++){
        if (r>f_ant) {
        s=r;}
        else{
        s=f_ant;
        }
        pi=Convert::ToInt32(dataGridView1->Rows[i]->Cells[1]->Value);
        f=s+pi;
guarda los datos  de f y s en una tabla
        f_ant = f;
        filas_fq++;
```

}

Ordenamiento de f de menor a mayor

Inicialemente guarda los datos extraidos del paquete en una variable, después se dispone a realizar el calculo de *Fi* y *Si*, en el cual realiza un recorrido con el condicional de que si el tamaño del paquete es menor a *r* entonces *s* será igual a *r*, de lo contrario *s* será igual al *Fi* anterior, luego concluye con el calculo de $Fi = Si + Pi$ en donde *Pi* es el tamaño del paquete. Este proceso se realiza de la misma manera para cada una de las colas guardando en un listado el número resultante de *Fi* .La siguiente linea es el ordenamiento de *Fi* de menor a mayor, dependiendo del orden de este listado será el orden en el que se envíen los paquetes.

Es importante tener en cuenta que *r* en este protocolo es igual en todas las colas y es la velocidad de transmisión dividido el número de colas c/N .

5. Weighted Fair Queueing

Este algoritmo de la Figura 3.7 realiza dos funciones al mismo tiempo, controla los tiempos de trafico en la parte delantera de la cola para reducir el tiempo de respuesta y comparte o distribute el ancho de banda restante depeniendo de la escogencia del usuario, en el estado 2 a 3. En el estado inical espera a que llegue el paquete y luego de realizar la clasificacion continúa con el siguiente paquete para su clasificación y envio.

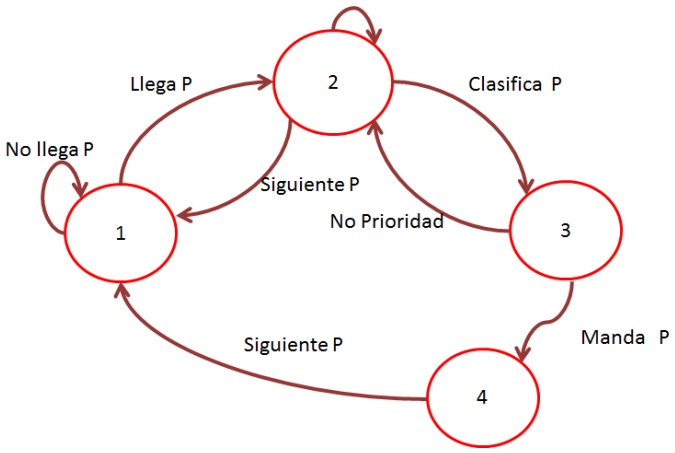

Figura 3.7 Máquina de estados de Weight Fair Queueing

En WFQ, cada paquete está asociado con una etiqueta de inicio S_i^α y etiqueta de salida F_i^α, que corresponden a los tiempos virtuales en que el primero y los últimos bits del paquete, respectivamente. En el momento que el paquete *i* del flujo, llega a la cola, se marca su etiqueta de inicio y el final de la siguiente etiqueta:

$$S_i^\alpha(p_k^i) = MAX\{V(A(p_k^i), F_i^\alpha(p_{k-1}^i))\} \quad (1)$$

$$F_i^{\alpha}(p_k^i) = S_i^{\alpha}(p_k^i) + \frac{l_k^i}{r_i} \quad (2)$$

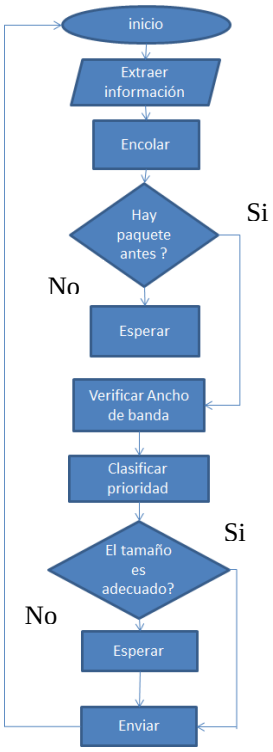

Figura 3.8 Diagrama de flujo de WFQ

```
calculo de Si y Fi por cola
cola1
if(radioButton3->Checked == true){
        peso1 = (int)numericUpDown4->Value;
        r = (float)(Convert::ToDouble(textBox12->Text)* peso2)/400;
}
        f_ant=0;
for (i=0;i<tam_cola[1];i++){
        if (r>f_ant) {
        s=r;
        }
        else{
        s=f_ant;
        }
        pi=Convert::ToInt32(dataGridView4->Rows[i]->Cells[1]->Value);
        f=s+p;
        f_ant = f;
        filas_fq++;
        }
```

En este protocolo la distribución del ancho de banda varía en cada cola y se dá por escogencia del usuario, por lo que antes de empezar el calculo de *Fi* y *Si* de cada cola el algoritmo revisa cual fue el porcentaje escogido por el usuario, lo convierte en porcentaje y esta distribución de ancho de banda dá como resultado el *r* utilizado después para los siquientes cálulos.

Inicialemente guarda los datos extraidos del paquete en una variable, despues se dispone a realizar el calculo de *Fi* y *Si,* en el cual realiza un recorrido con el condicional de que si el tamaño del paquete es menor a *r* entonces *s* será igual a *r*, de lo contrario *s* será igual al *Fi* anterior, luego concluye con el calculo de *Fi = Si+ Pi* en donde *Pi* es el tamaño del paquete. Este proceso se realiza de la misma manera para cada una de las colas guardando en un listado el número resultante de *Fi* .La siguiente linea es el ordenamiento de *Fi* de menor a mayor, dependiendo del orden de este listado será el orden en el que se envíen los paquetes.

6. Déficit Round Robin:

Se va a suponer que las cantidades F_i^{α}, que indican el porcentaje dado a fluir *i*. Se asume que cada flujo *i* es asignado Q_i por valor de bits en cada ronda.

Para ilustrar mejor este algoritmo se desarrollará un ejemplo con algunos valores y tener más claro de cómo funciona el proceso [10].

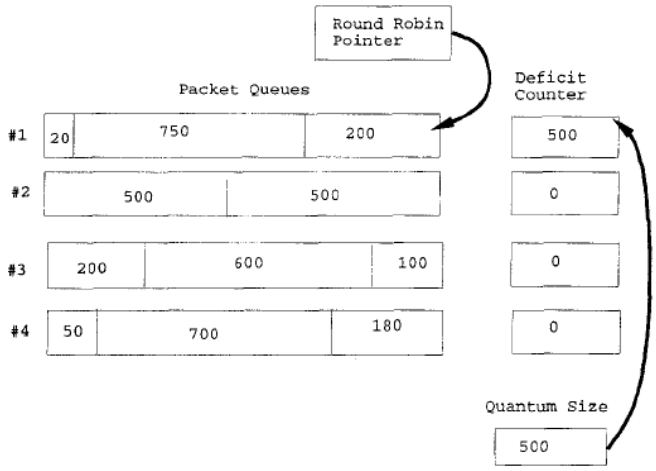

Figura 3.9 Ejemplo Deficit Round Robin (5)

Al principio, todas las variables se inicializan a cero. El puntero round-robin apunta a la parte superior de la lista activa. Cuando se atiende la primera cola, se añade el valor Q de 500 en este ejemplo. El resto después de servicio de la cola se deja en la variable.

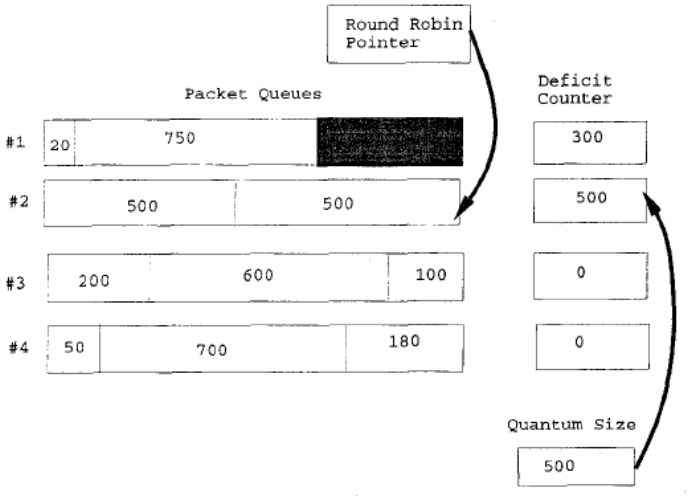

Figura 3.10 Ejemplo Deficit Round Robin (5)

Después de enviar un paquete de tamaño 200, la cola tenía 300 bytes de su valor en la izquierda. No podría usarlo la ronda actual, ya que el próximo paquete en la cola es de 750 bytes. Por lo tanto, la cantidad de 300 se trasladará a la siguiente ronda cuando se puede enviar paquetes de un tamaño total de 300 (déficit de la vuelta anterior) + 500 (quantum).

$Q = Min(Q_i)$. La acción F_i^{α}, que asignó al flujo *i* es simplemente Q_i/Q. Finalmente, dado que el algoritmo funciona en las rondas, se puede medir el tiempo en términos de rondas, los paquetes que llegan de los diferentes flujos se almacenan en diferentes colas.

El número de bytes enviados por la cola i en la ronda k, permite a cada cola enviar paquetes en la primera ronda sujeto a la restricción de que los bytes $1 < Q_i$. Si no hay más paquetes en la cola *i* después de que la cola se ha mantenido, una variable de estado denominada se inicializa a cero. En las rondas siguientes, la cantidad de ancho de banda utilizable por este flujo es la suma de los valores de la ronda anterior.

El diagrama de flujo del algoritmo se muestra en la Figura 3.10 y representa el proceso que desarrolla el algoritmo.

Figura 3.11 Diagrama de flujo del Déficit Round Robin

A continuación se mostrará cómo se programó el algoritmo DRR para este proyecto:

```
if (radioButton4->Checked == true){
  if (cola == 1) {
        bytes = bytes + header->len;
guarda los datos del paquete en tablas
        tcola = tcola+1;
        }
  if (cola == 2) {
        bytes = bytes + header->len;
guarda los datos del paquete en tablas
        tcola = tcola+1;
        }
  if (cola == 3) {
```

```
        bytes = bytes + header->len;
guarda los datos del paquete en tablas
        tcola = tcola+1;
        }
  if (cola == 4) {
        bytes = bytes + header->len;
guarda los datos del paquete en tablas
        tcola = tcola+1;
        }
  if (tcola >=lcola){
        cola = cola + 1;
        tcola = 0;
        contl = -1;
  }
}
```

Inicialmente como en todos los protocolos se debe guardar todos los datos relevantes de cada paquete en las tablas.

```
if (radioButton4->Checked == true) {
  q1=q2=q3=q4 = Convert::ToDouble(textBox10->Text);
  cola_act = 1;
  enviados=0;
  promediop=0;
  active_list = 1;

        for (j=0;j<4;j++){
         env_cola[j]=0;
        }
        for (j=0;j<4;j++){
         estado_cola[j]=0;
        }
Guarda los tamaños en unatabla para realizar unacomparación con el quantium
        while(active_list == 1){
                if ((cola_act == 1)) {
                  if (env_cola[0]<tam_cola[0]){
                long_pk=Convert::ToInt32(dataGridView1>Rows[env_cola[0]]->Cells[1]-
>Value);
                   }
                  else{
                        long_pk=q1+1;
                  }
                  if (long_pk <= q1){
Comienza con el llenado del paquete
LLenar el resto del paquete
        for(i=12;i<long_pk;i++){
```

```
                packet[i]=i%256;
        }
/* Envía el paquete */
        if (pcap_sendpacket(fp, packet, 100 /* size */) != 0){
        MessageBox::Show("\nError sending the packet: \n");
        }
        else{
        promediop = promediop + long_pk;
        dataGridView2->Rows->Add(1);
        dataGridView2->Rows[enviados]->Cells[0]->Value= enviados;
        dataGridView2->Rows[enviados]->Cells[1]->Value= "1";
        dataGridView2->Rows[enviados]->Cells[2]->Value= long_pk;
        enviados++;
        env_cola[0] = env_cola[0]++;
        cola_act=2;
        q1=q1-long_pk+q1;
        }
}
        else{
        q1=q1+Convert::ToInt32(Convert::ToDouble(textBox10->Text));
        cola_act=2;
        }
}
```

En esta parte del algoritmo se encuentra el envío de los paquetes y el encolamiento según el protocolo DRR. Por facilidad solamente se mostrará el procedimiento de una sola cola, sin embargo esto debe repetirse para cada cola.

Inicialmente se realiza un recorrido por el primer paquete de cada cola y lo compara con el valor del quantium para saber si se puede enviar o no, en caso de que este sea menor se realiza el proceso de envío del paquete, se resta el tamaño del paquete al valor del quantium para tenerlo en cuanta en la siguiente ronda, y verifica si la cola se encuentra activa, después se procede al fragmento para el llenado del paquete con la información guardada en las tablas y posterior envío.

7. Recibir paquetes

Por lo general, lo primero que hace una aplicación basada en WinPcap es obtener una lista de adaptadores de red conectados. Tanto libpcap y WinPcap proporcionan la función pcap_findalldevs_ex () para este propósito: esta función devuelve una lista enlazada de estructuras pcap_if, Cada uno de los cuales contiene información completa acerca de un adaptador adjunto. En particular, la descripción de los campos contiene el nombre y las personas del dispositivo correspondiente.

En primer lugar, pcap_findalldevs_ex (), al igual que otras funciones libpcap, tiene un parámetro de errbuf o error en el buffer. Este parámetro apunta a una cadena de llenado por

libpcap con una descripción del error, si algo sale mal. No todos los sistemas operativos soportados por libpcap proporcionan una descripción de las interfaces de red.

Como obtener información de los dispositivos de instalación:

WinPcap ofrece también otra lista de estructuras pcap_addr, con:

- Una lista de direcciones para esa interfaz.
- Una lista de máscaras de red (cada uno de los cuales corresponde a una entrada en la lista de direcciones).
- Una lista de direcciones de difusión (cada uno de los cuales corresponde a una entrada en la lista de direcciones).
- Una lista de direcciones de destino (cada uno de los cuales corresponde a una entrada en la lista de direcciones).

Además, pcap_findalldevs_ex () puede también devolver adaptadores remotos y una lista de archivos pcap que se encuentran en una carpeta local determinado.

La función que abre un dispositivo de captura es pcap_open (). snaplen especifica la porción del paquete a capturar .

Banderas: la bandera más importante es la que indica si el adaptador se pone en modo promiscuo. En el funcionamiento normal, un adaptador sólo captura los paquetes de la red que están destinados a ella; Por lo tanto, los paquetes intercambiados por otros hosts se ignoran.
En cambio, cuando el adaptador está en modo promiscuo que captura todos los paquetes si están destinados a ello o no. El modo promiscuo es el valor predeterminado para la mayoría de aplicaciones de captura.

to_ms especifica el tiempo de espera de lectura, en milisegundos. Una lectura en el adaptador siempre devolverá después de to_ms milisegundos, incluso si no hay paquetes disponibles en la red . to_ms también define el intervalo entre los informes estadísticos si el adaptador está en modo de estadística.

- Un tiempo de espera to_ms a 0 significa que no hay tiempo de espera
- Un tiempo de espera de -1 en el otro lado hace que una lectura en el adaptador para volver siempre inmediatamente.

Una vez que se abre el adaptador, la captura se puede iniciar con pcap_dispatch ( ) o pcap_loop ( ). Estas dos funciones son muy similares, la diferencia es que el envío pcap_ () devuelve cuando el tiempo de espera expira mientras pcap_loop () no regresa hasta que se hayan capturado los paquetes contantes.

Ambas funciones tienen un parámetro de devolución de llamada, packet_handler, apuntando a una función que va a recibir los paquetes. Esta función es llamada por libpcap para cada paquete nuevo que viene de la red y recibe un estatus genérico (que corresponde

al parámetro de usuario de pcap_loop () y pcap_dispatch ()), una cabecera con un poco de información sobre el paquete como la fecha, hora, la duración y los datos reales del paquete incluyendo todas las cabeceras de protocolo.

Busqueda de interfaces de red

```
if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL, &alldevs, errbuf) == -1){
        MessageBox::Show("Error in pcap_findalldevs");
        Application::Exit();
}

for(d=alldevs; d; d=d->next) {
        if (d->description)gcnew String(d->description);
                else
        }
        if(i==0)    {
        printf("\nNo interfaces found! Make sure WinPcap is installed.\n");
        Application::Exit();
        }
        if(numericUpDown1->Value < 1 || numericUpDown1->Value > i){
        }
```

Lista de dispositivos libres

```
        pcap_freealldevs(alldevs);
        Application::Exit();
        }
        for(d=alldevs, i=0; i< numericUpDown1->Value-1 ;d=d->next, i++);
        if ( (adhandle= pcap_open(d->name,
                    65536,
            PCAP_OPENFLAG_PROMISCUOUS,
                    1000,
                    NULL,
                   errbuf
                    ) ) == NULL){
                pcap_freealldevs(alldevs);
                Application::Exit();
        }
```

Abre el archivo dump

```
        const char * archivo;
        archivo = "winpcap";
        dumpfile = pcap_dump_open(adhandle, archivo);
 if(dumpfile==NULL)
   {

   }
        pcap_freealldevs(alldevs);
        cpu_time_entrada= (float)clock ();
        bytes = 0;
```

Recupera los paquetes

```
        contl = 0;
        contp=0;
        tcola = 0;
while(contp< numericUpDown2->Value ){
        res = pcap_next_ex( adhandle, &header, &pkt_data);
if(res == 0)
```

Tiempo de espera transcurrido

```
continue;
```

convertir la fecha y hora en formato legible

```
        local_tv_sec = header->ts.tv_sec;
        ltime=localtime(&local_tv_sec);
        strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);
```

Recupera la direccion IP de la cabecera

```
        ih = (ip_header *) (pkt_data + 14); //Longitud de la cabecera
```

Recupera la direccion UDP de la cabecera

```
        ip_len = (ih->ver_ihl & 0xf) * 4;
        uh = (udp_header *) ((u_char*)ih + ip_len);
```

Convierte los bytes de la red y los del

```
        sport = ntohs( uh->sport );
        dport = ntohs( uh->dport );
```

El proceso demostrado inicialmente busca todos los dispositivos de red disponibles, luego selecciona la interfaz escogida por el usuario para la captura de los paquetes, guarda los datos de cada paquete en un archivo teniendo en cuenta algunos valores máximos en caso de que no sean compatibles con el tipo de paquetes que se busca y se dispone a extraer alguna información de tiempos y direcciones IP en las cabeceras para convertirlos en datos legibles.

```
if ((radioButton6->Checked == true)&&(radioButton1->Checked == false)){
res = pcap_next_ex( adhandle, &header, &pkt_data);
        if(res == 0)
        continue;
```

Convertir los tiempos en datos legibles

```
        local_tv_sec = header->ts.tv_sec;
        ltime=localtime(&local_tv_sec);
        strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);
```

Extrae la posición de la cabecera ip

```
        ih = (ip_header *) (pkt_data + 14); //length of ethernet header
```

Extrae la posiión de la cabeera udp

```
ip_len = (ih->ver_ihl & 0xf) * 4;
uh = (udp_header *) ((u_char*)ih + ip_len);
```

Convierte la información en formato legible

```
sport = ntohs( uh->sport );
dport = ntohs( uh->dport );
b=Convert::ToInt32(ih->tos);
binario="";
s[0]= b/2;r[0]=b%2;
s[1]=s[0]/2;r[1]=s[0]%2;
s[2]=s[1]/2;r[2]=s[1]%2;
binario= r[2]+ binario ;
s[3]=s[2]/2;r[3]=s[2]%2;
binario= r[3]+ binario ;
s[4]=s[3]/2;r[4]=s[3]%2;
binario= r[4]+ binario ;
s[5]=s[4]/2;r[5]=s[4]%2;
binario= r[5]+ binario ;
s[6]=s[5]/2;r[6]=s[5]%2;
binario= r[6]+ binario ;
s[7]=s[6]/2;r[7]=s[6]%2;
binario= r[7]+ binario;
```

Realiza el filtrado según los 6 bits de DSCP

```
if ((binario == "001010")||(binario == "001100")||(binario == "001110")){
    dataGridView1->Rows->Add(1);
    dataGridView3->Rows->Add(1);
    dataGridView1->Rows[contl]->Cells[0]->Value=gcnew String(timestr);
    dataGridView1->Rows[contl]->Cells[1]->Value=header->len;
    bytes = bytes + header->len;
```

Es importante tener en cuenta que para realizar las pruebas de los protocolos será necesario utilizar un generador de paquetes por lo que en el momento de la captura de estos deberá exisitir un filtrado para ubicarlos en las colas denominado DSCP. El algoritmo mostrado anteriormente muestra como extrae del paquete el byte correspondiente a los primero 6 bits de fragmento ToS y convierte el número en binario para clasificarlos en las colas, Aquí solo está demostrado un tipo de clasificación pero los diferentes tipos de DSCP se peden observar en la Figura 3.12

Cola 1	Cola 2	Cola 3	Cola 4
AF11 = 10 (001010)	AF21 = 18 (010010)	AF31 = 26 (011010)	AF41 = 34 (100010)
AF12 = 12 (001100)	AF22 = 20 (010100)	AF32 = 28 (011100)	AF42 = 36 (100100)
AF13 = 14 (001110)	AF23 = 22 (010110)	AF33 = 30 (011110)	AF43 = 38 (100110)

Figura 3.12 Clasificación por DSCP

8. Enviar paquetes

Para el envío de paquetes inicialmente se consideró trabajar sobre la API de la tarjeta TuboCap pues según el manual de este, es compatible con todas las librerías de WinpCap las cuales fueron usadas para la realización de todo el proyecto, sin embargo, se encontraron algunas dificultades al momento de las pruebas pues esta tarjeta al parecer no entendía la totalidad de las librerías utilizadas y no era posible la búsqueda de las interfaces.

En solución a este inconveniente se realizaron todas las pruebas utilizando simplemente un computador con dos tarjetas de red de las cuales una de ellas la RTL8931D es la encargada del envío de los paquetes al analizador de redes. Es importante aclarar que este cambio no interfiere en absoluto con la validación e implementación de los algoritmos a probar en este proyecto, por el contrario esta alternativa permite al usuario utilizar este proyecto de grado en cualquier computador que tenga una tarjeta de red adicional la cual es mucho más fácil de adquirir y utilizar que la tarjeta TurboCap y además se tiene la posibilidad de capturar paquetes de la red etherneth o de redes inalámbricas.

El envío de un único paquete con pcap_sendpacket ()

Después de abrir un adaptador, pcap_sendpacket () es llamada para enviar un paquete. pcap_sendpacket () toma como argumentos un buffer que contiene los datos a enviar, la longitud de la memoria intermedia y el adaptador que se envíe. El búfer se envía a la red como es, sin ninguna manipulación. Esto significa que la aplicación debe crear las cabeceras de protocolos correctos para enviar algo significativo.

Mientras pcap_sendpacket () ofrece una forma simple e inmediata para enviar un único paquete. Una cola de envío es un contenedor para un número variable de paquetes que serán enviados a la red, esta tiene un tamaño, que representa la cantidad máxima de bytes que puede almacenar.

Una cola de envío se crea llamando al pcap_sendqueue_alloc función ( ), especificando el tamaño de la nueva cola de envío.

Una vez creada la cola de envío, pcap_sendqueue_queue () se puede utilizar para añadir un paquete a la cola de envío, esta función toma un pcap_pkthdr con la marca de tiempo y la longitud y un tapón con los datos del paquete, estos parámetros son los mismos que los recibidos por pcap_next_ex ( ) y pcap_handler ( ) por lo tanto hace que un paquete que se acaba de capturar en un archivo pase estos parámetros para pcap_sendqueue_queue ( ).

Para transmitir una cola de envío, WinPcap proporciona la función pcap_sendqueue_transmit() . Tenga en cuenta que la transmisión de una cola de envío con pcap_sendqueue_transmit () es mucho más eficiente que la realización de una serie de pcap_sendpacket (), pero puede hacer que salgan más paquetes de los requeridos en este algoritmo.

Cuando ya no se necesita una cola, se puede eliminar con pcap_sendqueue_destroy () que libera todos los buffers asociados a la cola de envío.

9. Estadísticas

Las estadísticas serán sacadas en diferentes momentos del algoritmo, a continuación se mostrará el proceso de sacar cada una de ellas.

Inmediatamente después de que el algoritmo empieza a "escuchar" o a capturar los paquetes se realiza una marca de tiempo en cada paquete.

Estadisticas de entrada
cpu_time_entrada= (float)clock ();
bytes = 0;

Después de que se han capturado los paquetes y se ha extraído la información de los paquetes y se ha guardado en las correspondientes tablas se entregan las primeras estadísticas con la marca de tiempo de entrada "cpu_time_entrada" guardado anteriormente y el tiempo actual "clock()". Las estadísticas son **Tasa de bits entrantes** y **Tasa de paquetes entrantes.**

```
cpu_time_entrada = (clock() - cpu_time_entrada-cpu_time_entrada)/CLK_TCK;
textBox5->Text =  Convert::ToString(contp/cpu_time_entrada) + " paquetes/s";
textBox6->Text =  Convert::ToString(bytes/cpu_time_entrada) + " bytes/s";
```

Cuando se ha enviado el paquete se realiza el cálculo del **Retardo Total, Throughput por cola, Throughput Total** y el **Tamaño promedio de paquetes.** Estas estadísticas se toman teniendo en cuenta la marca de tiempo al iniciar la captura y la hora actual.

```
cpu_time = (clock() - cpu_time-cpu_time_entrada)/CLK_TCK;
textBox1->Text =  Convert::ToString(cpu_time) + " s";
```

textBox3->Text = Convert::ToString(enviados/cpu_time) + " paquetes/s";
textBox4->Text = Convert::ToString(enviados/cpu_time) + " paquetes/s";
textBox7->Text = Convert::ToString(promediop/enviados);
textBox8->Text = Convert::ToString(promediop/enviados)}

Para el llenado de la tabla de retardo por cola fue necesario realizar una marca de tiempo cuando empieza el encolamiento en cada protocolo y cuando termina de enviar cada paquete.

/marca de tiempo cuando empieza a encolar
cpu_time = (float)clock ();

Capitulo 4

Análisis de Resultados

1. Introducción

En esta sección se presentará, los resultados de todo el proyecto, su explicación y análisis. Para guardar la coherencia se mostrarán las pruebas de cada protocolo con distintas cantidad de paquetes a enviar, y distintas configuraciones del generador de paquetes, los resultados de las estadísticas en el proyecto y los resultados que leerá el analizador de redes llamado VeEx.

2. Configuración para las pruebas.

El primer paso para comenzar la configuración del protocolo y la inicialización de las pruebas, es necesario reconocer las interfaces a usar, saber cuál de ellas será el puerto de captura de la red y cuál será el puerto que se conectará al analizador de redes para el análisis de las estadísticas.
Para la obtención de estos datos es necesaria la utilización del programa Wireshark, en la Figura 4.1 se muestra cómo captura las interfaces de red.

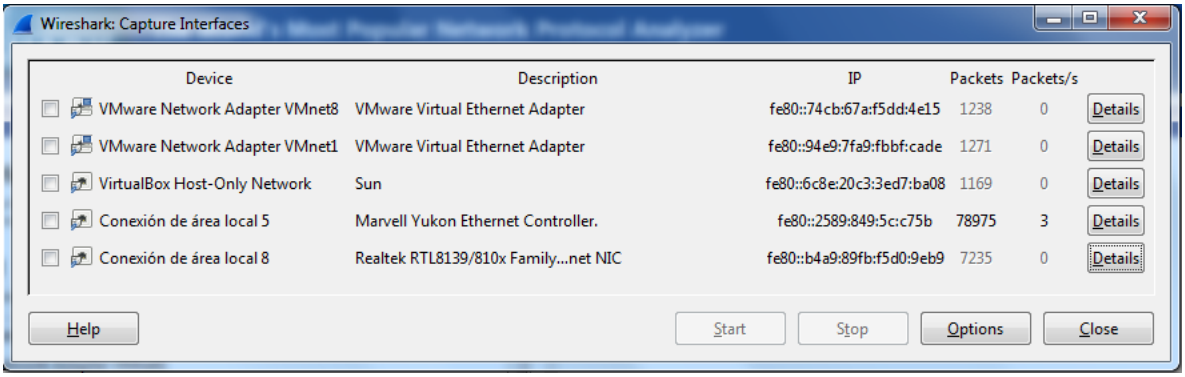

Figura 4.1 Captura de Interfaces en Wireshark

Como se puede observar existen 5 interfaces, para este proyecto se utilizará las dos últimas las cuales son tarjetas de red, la primera denominada “Conexión de área local 5” es la única que está transmitiendo paquetes debido a que está conectado directamente a la red de la universidad. La última interfaz de red denominada “Conexión de área local 8” es la tarjeta de red RTL8139d que se conectará al analizador de redes.

Para tener completa seguridad de que el algoritmo del proyecto está leyendo las interfaces correctamente, se realizó una comparación de los detalles de cada interfaz de red que se obtienen del programa Wireshark junto con la enumeración y nombres de las interfaces en el programa. En la Figura 4.2 se muestran los detalles de lo que es la interfaz número 4, también se muestra al costado derecho que la interfaz número 4 que lee el algoritmo es la misma que muestra Wireshark.

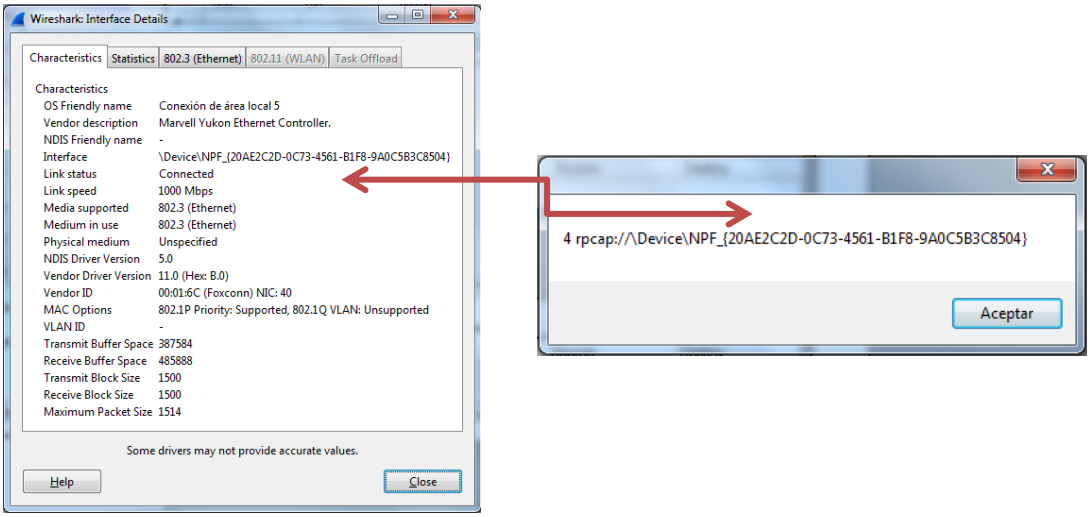

Figura 4.2 Búsqueda de la interfaz Número 4 (Puerto de la red)

En la Figura 4.3 se muestra la comparación y búsqueda de la interfaz Número 5 el cual es el puerto de la tarjeta de red RTL8139 que se conecta al analizador de redes.

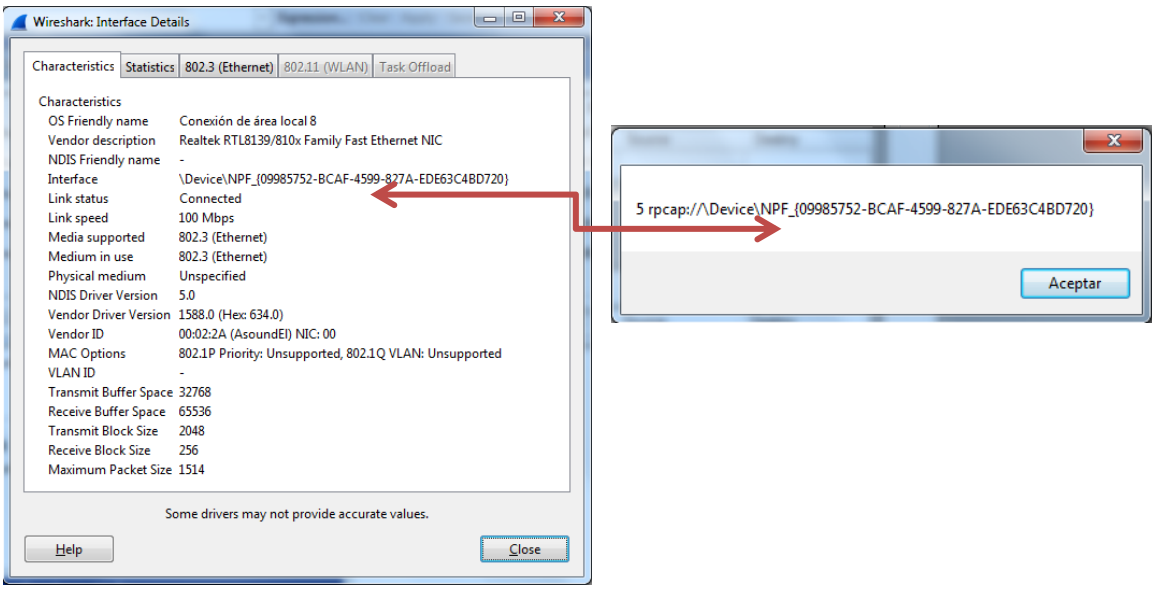

Figura 4.3 Búsqueda de la interfaz Número 5 (Puerto para el análisis de estadísticas)

La tarjeta de red utilizada para enviar los paquetes hacía el analizador de redes es compatible con cualquier computador que tenga la posibilidad de adaptarle más de una tarjeta de red, además puede procesar el envío a altas velocidades y gracias a la versatilidad del algoritmo desarrollado en este proyecto es posible que capture paquetes de la red de internet o inalámbrica.

Ahora bien, ya teniendo claro las interfaces que se utilizan para probar los protocolos de encolamiento se procede a realizar las debidas pruebas de cada protocolo.

3. Pruebas de las estadísticas

Las pruebas realizadas para esta sección se hicieron generando paquetes del analizador de redes VeEx y recibiendo los paquetes a este mismo equipo después de pasar por el encolamiento desarrollado en este proyecto.

Cabe aclarar que la generación de paquetes se realizó por el puerto 2 del VeEx y su recibimiento desde la tarjeta de red RTL8931D se hizo por el puerto 1.

En estadística, un proceso estocástico es un concepto matemático que sirve para caracterizar una sucesión de variables aleatorias que evolucionan en función de otra variable, en este caso el tiempo. Cada una de las variables aleatorias del proceso tiene su propia función de distribución de probabilidad y, entre ellas, pueden estar correlacionadas o no. Cada variable o conjunto de variables sometidas a influencias o efectos aleatorios constituye un proceso estocástico, sin embargo el cálculo de las estadísticas para realizar el análisis en la aplicación es un proceso determinístico en donde las mismas entradas producirán invariablemente las mismas salidas, no contemplándose la existencia del azar ni el principio de incertidumbre.

La inclusión de mayor complejidad en las relaciones con una cantidad mayor de variables a calcular y elementos ajenos al modelo como la captura de paquetes de la red conlleva a que el cálculo de las estadísticas se aproxime a un modelo estocástico.

El analizador de redes VeEx tiene la posibilidad cambiar parámetros importantes para el envío de paquetes por lo que escogió enviarlos en modo continuo y con un tamaño constante con el fin de aproximarse en lo posible a capturar paquetes fijos y poder realizar el cálculo de las variables con modelo determinístico.

Se realizaron diferentes tipos de pruebas:
Para cada protocolo se hizo correr el algoritmo capturando 10000 paquetes con un flujo de tráfico constante, el analizador de redes envía los paquetes con el mismo tamaño, a continuación se muestra una tablas con algunos de los resultados obtenidos tanto de las estadísticas dadas por el algoritmo como datos extraídos del analizador de redes y la velocidad de transmisión de la generación de paquetes.

Todos los datos que se observan en las tablas se pueden verificar en los anexos de este proyecto como imágenes de las pruebas en el analizador de redes y el algoritmo.

		RETARDO PROMEDIO (s)	TRANSMISION (bps)	THROUGHPUT (bps)
FIFO	ALGORITMO	486E-03	2.59M	2.47M
	VeEx	12.47E-03		2.52M
FQ	ALGORITMO	474E-03	2.49M	2.58M
	VeEx	11.83E-03		2.59M
WFQ	ALGORITMO	184E-06	2.59M	2.25M
	VeEx	7.095E-03		2.66M
DRR	ALGORITMO	176E-06	2.59M	3.09M
	VeEx	12.02E-03		2.56M

Tabla 4.1 Resultados de estadísticas en modo Continuo

Retardo

El retardo promedio que se observa en la primera columna es el tiempo que se demora todo el proceso de encolamiento, este retardo en la aplicación es un tiempo estimado de lo que se demora en enviar un paquete, esto se debe a que el algoritmo fue desarrollado de manera lineal por lo que existe un tiempo muerto en el que los paquetes deben esperar a que se capturen es su totalidad.

Tasa de bits

Tanto en la transmisión como en la recepción de paquetes la velocidad con la que se están enviando los paquetes es aproximadamente 2.5Mbps por lo que se puede concluir que no se está perdiendo información en la transmisión y que esta se realiza a la misma velocidad, y la aplicación está realizando el cálculo correctamente pues concuerda con el real leído en el analizador de redes.

Para realizar los cálculos teóricos de tiempo de retardo y throughput se utilizaron las siguientes ecuaciones:

Para el algoritmo First In First Out:

$$D \leq \frac{B}{c} \quad (1)$$

D, es el tiempo de retardo
B, son el total de bytes enviados
c, es la velocidad de transmisión

Para el algoritmo Fair Queueing y Déficit Round Robin se uzarons las mismas ecuaciones pues su ancho de banda está distribuido de la misma manera, es decir mismo porcentaje para cada flujo:

$$R = \frac{c}{N} \quad (2)$$

$$D \leq \frac{B}{R} \quad (3)$$

D, es el tiempo de retardo
B, son el total de bytes enviados
c, es la velocidad de transmisión
N, es el número de flujos
R, es la tasa máxima que puede conseguir un flujo

Para el algoritmo Weigted Fair Queueing se tiene la misma ecuación (3) pero la tasa R difiere pues la distribución de ancho de banda es distinta para cada flujo.

$$R = \left(\frac{w_i}{\sum_{n=1}^{N} w_n}\right) * c \quad (4)$$

En donde w es el peso en cada flujo el cual es escogido por el usuario antes de realizar la captura.
En la Tabla 4.2 se encuentran los resultados de los tiempos de retardo calculados con las ecuaciones nombradas anteriormente y los valores de retardo promedio que entrega la aplicación después de capturar, encolar y enviar 10000 paquetes.

Retardo	**B (Bytes)**	**Tiempo promedio (s)**	
FIFO	1,20E+08	4,63E-01	**TEORICO**
		4,86E-01	**ALGORITMO**
FQ	2,40E+08	2,68E-03	**TEORICO**
		4,74E-01	**ALGORITMO**
WFQ	1,20E+08	1,08E-02	**TEORICO**
		1,84E-04	**ALGORITMO**
DRR	1,20E+08	2,6E-03	**TEORICO**
		1,76E-04	**ALGORITMO**

Tabla 4.2 Comparación de tiempos de retardo

Se observa en la tabla que los tiempos de retardo calculados teóricamente varían con respecto a los entregados por el algoritmo pues como se mencionó anteriormente estos son una estimación de lo que se demora en enviarse un paquete pero no se tiene en cuenta velocidad de procesamiento del equipo, sin embargo con estos resultados es posible que el usuario pueda darse cuenta cuál de los algoritmos es el más apropiado para la aplicación que quiera hacer y puede realizar el respectivo análisis llegando a las mismas conclusiones de los resultados teóricos.

En la Tabla 4.3 se encuentran los resultados de los cálculos realizados teóricamente y los que se obtienen del aplicativo para los valores de throughput en cada algoritmo, el cálculo

de este se realizó dividiendo la velocidad de transmisión (2.5Mbps) en las distintas distribuciones de ancho de banda según sea el caso.

	cola 1 (Mbps)	**cola 2 (Mbps)**	**cola 3 (Mbps)**	**cola 4 (Mbps)**	**TOTAL (Mbps)**	
FIFO	2.52				2.59	**TEORICO**
	2.47				2.47	**ALGORITMO**
FQ	0,6475	0,6475	0,6475	0,6475	2,59	**TEORICO**
	0,646	0,646	0,647	0,646	2,585	**ALGORITMO**
WFQ	1,125	0,675	0,225	0,225	2,25	**TEORICO**
	0,975	0,664	0,306	0,305	2,25	**ALGORITMO**
DRR	0,77425	0,77425	0,77425	0,77425	2,56	**TEORICO**
	0,775	0,774	0,774	0,774	3,097	**ALGORITMO**

Tabla 4.3 Resultados del Throughput

Como se observa en la tabla anterior los valores de throughput teóricos y los que arroja la aplicación concuerdan por lo que se puede concluir que el algoritmo está realizando una correcta repartición del ancho de banda para los 4 protocolos.

4. Validación de los Protocolos

Para la validación de los protocolos se capturarán paquetes de la red inalámbrica y así obtener un número mucho más variado de tamaños en los paquetes y poder realizar el análisis y la verificación de los protocolos con más facilidad. Las imágenes que se mostraran serán extraídas de los resultados del algoritmo al usuario en donde se encuentra el orden en el que se envían los paquetes, por facilidad de cálculos solamente se tendrá en cuenta los resultados de los primeros 5 paquetes enviados pero se capturarán 100 en cada prueba.

4.1 FIFO

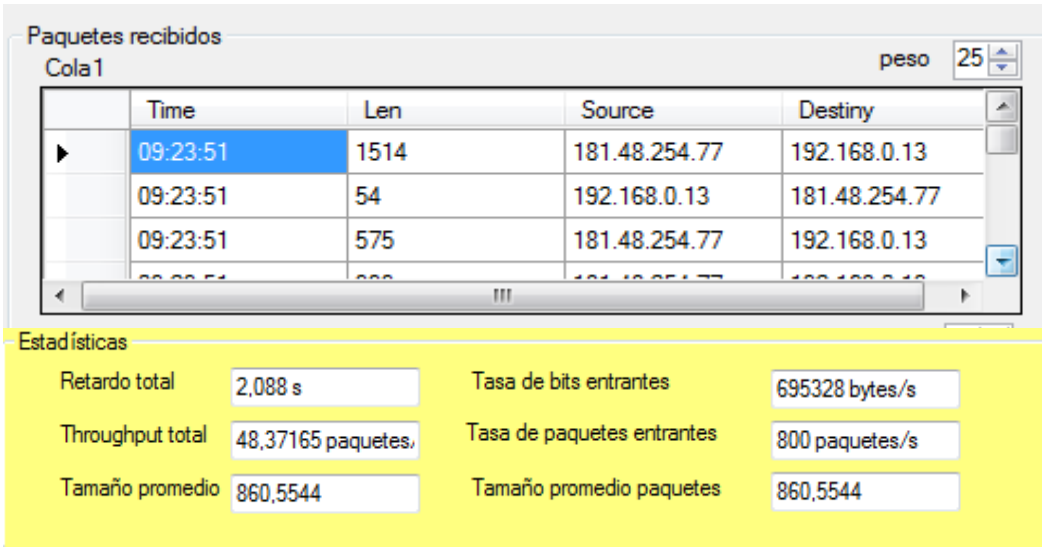

Figura 4.4 Resultados de FIFO

Este protocolo solamente tiene una cola pues no necesita hacer ningún tipo de clasificación por lo que en la Figura 4.4 se observa la tabla con los 100 paquetes y la información importante. En el cuadro de estadísticas se observa el tiempo que se demora en realizar la captura y el envío, en este caso es 2 seg con un Throughput total de 48 paquetes por segundo. Observando este resultado se puede analizar que pasan muy pocos paquetes por segundo en el puerto y es por esta razón que el tiempo de retardo es tan grande sin contar con que el tamaño promedio de paquetes es de 860 bytes entonces necesita de bastante tiempo para transmitir paquetes tan grandes a tan poco throughput.

4.2 Fair Queueing

Para la verificación del protocolo se usarán los primeros paquetes de cada cola con el fin de realizar los debidos cálculos y verificar si el orden en el que se enviaron los paquetes es correcto según la prioridad dada por el protocolo.

Para este protocolo la repartición del ancho de banda es igual para todos los puertos es decir 25% para cada cola y sabiendo que la velocidad de transmisión es 1000Mbps y el número de flujos es 4 se puede decir que la variable *r* es:

$$r = 250$$

Teniendo en cuenta la Ecuación (5) y la Ecuación (6) se mostrará en la Tabla 4.4 los valores calculados de *Si* y *Fi* para cada paquete en cada una de las colas.

$$S_i^{\alpha} = MAX(F_{i-1}^{\alpha}, R(t_i^{\alpha})) \quad (5)$$

$$Fi = Si + Pi \quad (6)$$

paquete	*Fi*	*Si*	*Fi-1*	*Tamaño*
1 (cola1)	1764	250	0	1514
1(cola2)	304	250	0	54
1(cola3)	1764	250	0	1514
1(cola4)	1764	250	0	1514
2(cola1)	3278	1764	1764	1514
2(cola2)	880	304	304	576
2(cola3)	3278	1764	1764	1514
2(cola4)	1818	1764	1764	54
3(cola1)	3332	3278	3278	54
3(cola2)	2394	880	880	1514
3(cola3)	3332	3278	3278	54
3(cola4)	2844	1818	1818	1026

Tabla 4.4 Calculo de Fi y Si

En la Tabla 4.5 se encuentra organizado los tamaños de Fi de menor a mayor el cual será el orden en el que se enviarán por el puerto, también se puede observar en la Figura 6.5 la imagen de los paquetes enviados en el proyecto.

Paquete	*Fi*	*Tamaño*
1(cola2)	304	54
2(cola2)	576	576
1 (cola1)	1764	1514
1(cola3)	1764	1514
1(cola4)	1764	1514
2(cola4)	630	54
3(cola2)	3278	1514
3(cola4)	2790	1026
2(cola1)	3278	1514
2(cola3)	2144	1514
3(cola1)	3332	54
3(cola3)	2844	54

Tabla 4.5 Resultados de los cálculos Fair queueing

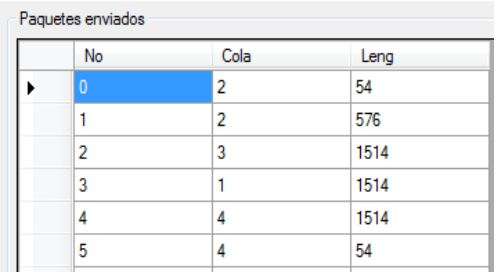

Figura 4.5 Envío de paquetes para Fair queueing

Observando el orden en el que se envían los paquetes y el orden en el que está organizado Fi es claro concluir que el algoritmo está cumpliendo a cabalidad la clasificación por el protocolo Fair queueing y así obtener una repartición equitativa de todos los recursos de la red haciendo que no existan flujos dominantes que puedan ocupar todo el ancho de banda y se pierdan más paquetes.

Este algoritmo le da la oportunidad a todos los paquetes de clasificarse en una cola y continuar con la comunicación sin importar lo dominante que sea respecto a otros paquetes de la misa red.

4.3 Weighted Fair Queueing

Para este protocolo se realizaron los cálculos de Fi y Si de la misma manera que el protocolo anterior, con la variación de que el ancho de banda se dividirá entre las colas de manera distinta:

cola 1	25%
cola2	30%
cola3	25%
cola4	20%

Tabla 4.5 Distribución de ancho de banda para Weighted Fair Queueing

Debido a esta distribución el valor de *r* cambiará para cada cola, en la Tabla 6.5 se observan los cálculos respetivos para la validación del código teniendo en cuenta los tamaños de los paquetes capturados por el algoritmo desde la red inalámbrica, se pueden observar estos resultados en los anexos.

paquete	*Fi*	*Si*	*Fi-1*	*Tamaño*	*r*
1 (cola1)	1764	250	0	1514	250
1(cola2)	1814	300	0	1514	300
1(cola3)	304	250	0	54	250
1(cola4)	260	200	0	60	200
2(cola1)	3278	1764	1764	1514	250
2(cola2)	1868	1814	1814	54	300
2(cola3)	1818	304	304	1514	250
2(cola4)	536	260	260	276	200
3(cola1)	3332	3278	3278	54	250
3(cola2)	3382	1868	1868	1514	300
3(cola3)	3332	1818	1818	1514	250
3(cola4)	614	536	536	78	200
4(cola4)	671	614	614	57	200

Tabla 4.6 Cálculos correspondientes al protocolo Weighted Fair Queueing

En los primeros paquetes de cada cola los Fi anteriores con cero porque se está comenzando con la transición, a partir del segundo paquete ya se tiene un valor numérico mayor que cero para esta variable. Para realizar los cálculos se tuvieron en cuenta las Ecuaciones (7), (8), y (0).

$$S_i^{\alpha} = MAX\{R(t), F_{i-1}^{\alpha}\} \qquad (7)$$

$$F_i^{\alpha} = S_i^{\alpha} + Pi \qquad (8)$$

$$r_i = c * (W_i) \qquad (9)$$

paquete	*Fi*	*Tamaño*
1(cola4)	260	60
1(cola3)	304	54
2(cola4)	276	276
3(cola4)	614	78
4(cola4)	671	57
1 (cola1)	1764	1514
1(cola2)	1814	1514
2(cola3)	2128	1514
2(cola2)	725	54
2(cola1)	3278	1514
3(cola1)	1868	54
3(cola3)	3642	1514
3(cola2)	2239	1514

Tabla 4.7 Resultados de los cálculos para Weighted Fair Queueing

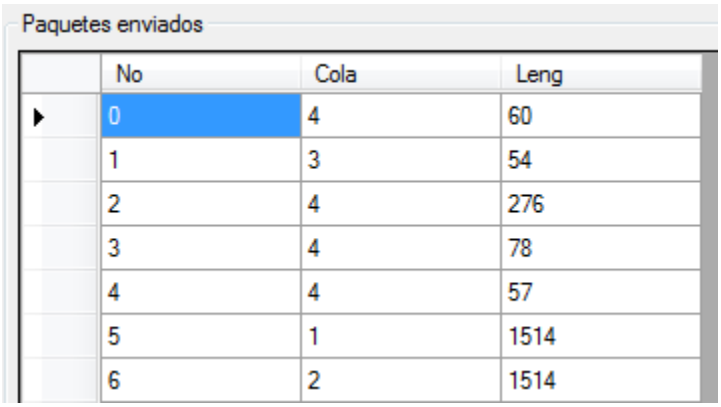

Figura 4.6 Envío de paquetes para Weighted Fair queueing

Se puede observar en la Tabla 4.6 y la Figura 4.6 que el orden en el que se envían los paquetes y el orden en el que está organizado los valores calculados de Fi son iguales por lo que la clasificación del protocolo es completamente válida para este proyecto.

4.4 Déficit Round Robin

paquete	***Tamaño***	***quantium***	***Ronda***
1 (cola1)	60	100	2
1(cola2)	284	300	6
1(cola3)	77	100	2
1(cola4)	60	100	2
2(cola1)	56	90	3
2(cola2)	60	66	7
2(cola3)	60	73	3
2(cola4)	57	43	3

Tabla 4.7 Calculo de rondas

En la Tabla 4.7 se encuentran los resultados obtenidos al capturar 100 paquetes con un quantium de 50, la tercera columna representa el quantium que necesita para que el paquete pueda ser enviado y la columna cuatro representa el número de la ronda en la que se puede enviar el paquete sumando los cuantium cada vez que pasa por esa cola.

paquete	*Tamaño*	*quantium*	*Ronda*
1 (cola1)	60	100	2
1(cola3)	77	100	2
1(cola4)	60	100	2
2(cola1)	56	90	3
2(cola3)	60	73	3
2(cola4)	57	43	3
1(cola2)	284	300	6
2(cola2)	60	66	7

Tabla 4.8 Envío de paquetes según el déficit Round Robin

En la Tabla 4.8 se muestra la ronda en la que se puede enviar el paquete dependiendo del quantium, aunque algunas de las colas tienen el mismo número de ronda para poder enviar el orden en el que se envía depende de la cola en la que se encuentra el paquete, como se observa en la tabla la cola 2 tiene los paquetes más grande por lo que el primer paquete de esa cola solamente se pondrá enviar hasta la ronda 6, sin embargo como se realizó el cálculo con los primeros 3 paquetes de cada cola los paquetes de la cola 2 se enviarán hasta el lugar 13 y 17 según los cálculos.

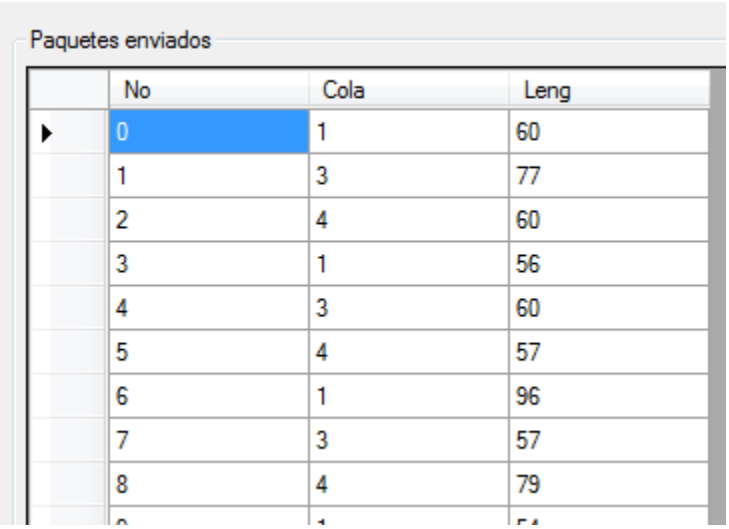

Figura 4.8 Envío de paquetes para Déficit Round Robin

En la Figura 4.8 se muestra los resultados que entrega el algoritmo desarrollado en este proyecto, y como se puede observar el orden en el que se envían los paquetes coincide por completo con el orden en el que se calcularon las rondas según el protocolo, como se mencionó anteriormente los paquetes de la cola 2 no se muestran en la figura pues estos solo pueden ser enviados hasta el lugar 13 y 17 debido a su gran tamaño en comparación con el resto.

Uno de los inconvenientes del DRR es cuando existe una mayor número de colas vacías de las no vacías pues se desperdicia mucho tiempo al verificar si la cola está vacía o no, en

este caso en específico la colas se llenan por completo y no existe el caso de que haya una cola no activa pues solamente se manejan 4 colas en el algoritmo y con la velocidad de transmisión siempre estarán llenas.

Capítulo 5

CONCLUSIONES

Es posible analizar los protocolos FIFO, Fair Queuing, Weighted Fair Queuing y Deficit Round Robin para el análisis de calidad y servicio y gestión activa de colas en una interfaz de usuario cómoda a los ojos y entendible, con un modelo de obtención de datos estadísticos desde cualquier computador con dos puertos de red.

1. Conclusiones de los Algoritmos

El orden en el que son enviados los paquetes en el algoritmo desarrollado en este proyecto coincide con la teoría dada para la clasificación y encolamiento de paquetes en cada uno de los protocolos a evaluar, por lo que se puede concluir que la validación de los algoritmos es acertada en el momento del encolamiento.

El protocolo FIFO posee menos complejidad de procesamiento en el algoritmo no tiene en cuenta prioridad por lo que en el caso que el primer paquete sea lo bastante grande para ocupar todo el ancho de banda los paquetes siguientes se retrasarán.

El protocolo Fair queueing permite una clasificación más justa tiene en cuenta la prioridad de los paquetes enviando primero el paquete con menor tamaño de las cuatro colas para hacer más eficiente la transmisión, sin embargo no tiene en cuenta la distribución de ancho de banda.

La distribución de ancho de banda escogida por el usuario que se realiza en el protocolo Weighted Fair Queueing permite un aprovechamiento más eficiente del ancho de banda del puerto, teniendo en cuenta la prioridad de estos para realizar la clasificación y encolamiento, sin embargo no es eficiente si los tamaños de los paquetes varían en gran proporción.

En el Déficit Round Robin la prioridad con la que envíe los paquetes no sea necesariamente la más eficiente debido a que la escogencia del quantum la realiza el usuario, sin embargo es el protocolo con menos tiempo de retardo para su clasificación y envío.

2. Conclusiones de las estadísticas

Los retardos totales de cada uno de los protocolos son más grandes al valor real pues el proyecto está desarrollado en su gran mayoria en software y este requiere un tiempo de procesamiento considerable para realizar la clasificación, además el proceso de envío solo comienza cuando temina de capturar todos los paquetes, sin embargo este tiempo permite al usuario análizar qué protocolo posee más eficiencia al momento de clasificar y puede realizar un análisis completo de la gestíon activa de colas.

Las tasa de bits entrantes permite al usuario conocer con que velocidad esta capturando los paquetes el algoritmo y darse una idea de la velocidad de transmisión que puede influir considrablemente en la eficiencia de cualquier clasificación.

El tamaño promedio de los paquetes es una estadística útil para el usuario pues le permite darse cuenta si existe la posbilidad de que alguno de los paquetes se pudo haber perdido en el proceso de clasificación o si por el contrario en los resultados arrojados muestra pérdida de paquetes y si este valor no concuerda con esa consecuencia poder análizar la razón de esta pérdida.

La implementación de este algoritmo permite una ejecución fácil sobre cualquier dispositivo que tenga dos tarjetas de red gracias al uso de las librerías WinPcap que es la misma que utiliza Wireshark para leer las interfaces, la aplicación también puede implementarse capturando paquetes de la red inalámbrica.

3. Conclusiones de la Interfaz gráfica

En el caso de utilizar algun generador de paquetes en el cual se pueda realizar un cambio o filtrado en el DSCP al momento de generar, existe en el algoritmo la posibilidad de hacer una preclasificación en las colas siguiendo este protocolo cuyos parámetros se muestran en la sección 2.7 del capítulo 2 de este informe, sin embargo debido a que los datos obtenidos de la red inalambrica tienen en su gran mayoría un de DSCP de 0 por facilidad se trabajo en modo proporcional la distribucion de paquetes en cada cola.

La interfaz de usuario fue desarrollada con el fin de que fuera sencilla la asignación de especificaciones requeridas para el análisis de los protocolos de gestión de colas, esta interfaz muestra la información relevante de cada paquete, junto con las estadisiticas necesarias para verificar algunos parámetros de funcionamiento y por último muestra el orden exacto en el que fueron enviados los paquetes, esto significa que el usuario tiene oportunidad de probar la funcionalidad del del protocolo, que toma en cuenta para su encolamiento y cómo realiza su clasificación.

La utilización de la tarjeta de red normal (en este caso RTL8931D) en lugar de la tarjeta TurboCap permite al usuario ejecutar el algoritmo desde cualquier computador disponible con dos puertos de red y capturar paquetes desde la red inalámbrica, el algoritmo también está desarrollado para enviar los paquetes por el mismo puerto si no se requiere leer ninguna información despues de su envío o solamente necesita observar la validación del protocolo.

Para un trabajo futuro se recomienda:

- Realizar un cambio en las librerías de la aplicación para que sea posible la implementación del algoritmo en una tarjeta TurboCap
- Realizar el cambio en la implementación de hilos o multitarea en el código para implementar la captura y envío de paquetes de manera simultanea y así obtener los tiempos reales de cada algoritmo.

BIBLIOGRAFIA

[1] IP RESEARCH & COMUNNITIES. Métodos para controlar First In First Out. [Online]. http://www.freepatentsonline.com/6678756.html. Fuente Inglés (Estados Unidos)

[2] Quality of Service. Cisco, Unified Communications. FIFO queuing [Online] http://globalknowledgeblog.com/technology/unified-communications/quality-of-service-part-9-%E2%80%93-fifo-queuing/ . Fuente Inglés (Estados Unidos)

[3] Quality of Service. Cisco, Unified Communications. Weighted Fair Queueing. [Online] http://globalknowledgeblog.com/technology/unified-communications/quality-of-service-part-10-%E2%80%93-weighted-fair-queuing/ . Fuente Inglés (Estados Unidos)

[4] IP RESEARCH & COMUNNITIES. Metodos para controlar Weighted Fair Queueing [Online] http://www.freepatentsonline.com/7194741.html Fuente Inglés (Estados Unidos)

[5] WinpCap Documents. Sitio oficial [Online] http://www.winpcap.org/ Fuente Ingles (Estados Unidos)

[6] Manual básico de Wireshark. Slide share. [Online] http://www.slideshare.net/DIANYSS2012/manual-bsico-de-wireshark

[7] Redes de computador. PROTOCOLO TCP [Online] http://www.hpca.ual.es/~leo/redes/Rel3-01-02.pdf Fuente Inglés (Estados Unidos)

[8] Analysis of Active Queue Management. Computer Science Department. Worcester Polytechnic Institute. [Online] http://web.cs.wpi.edu/~claypool/papers/queue-law/aqm-analysis.pdf Fuente Inglés (Estados Unidos)

[9] S.Athuraliya, V. H.Li, S. H.Low, and Q. Yin. REM: Active Queue Management. IEEENetwork, May 2001

[10] Network Routing. Algorithms, Protocols and Architectures; DEEPANKAR MEDHI, KARTHIKEYAN RAMASAMY; Morgan Kaufmann.

[11] Algoritmos AQM basados en teoría de control en Opnet; Virginia Alvárez Cuesta; Teresa Álvarez Álvarez; Universidad Valladolid

[12] Algoritmos AQM basados en teoría de control en Opnet; Virginia Alvárez Cuesta; Teresa Álvarez Álvarez; Universidad Valladolid. Pag 38.

[13] Control de congestión con OPNET; Lourdes Nicolás Fenrandez; María Teresa Álvarez Álvarez. Pag 25

[14] Caracterización de retardos en redes Ethernet para aplicaciones de control distribuido; Mario Modesti, Andres Silva; Universidad Tecnológica Nacional. [Online]http://www.investigacion.frc.utn.edu.ar/sensores/Aplicaciones/Publicaciones/024.pdf

[15]Queueing Theory; Harry Perros; Service Management [Online]. http://www4.ncsu.edu/~hp/SSME_QueueingTheory.pdf

[16] Queueing Theory; Harry Perros; Service Management [Online]. http://www4.ncsu.edu/~hp/SSME_QueueingTheory.pdf. . Pag 4

[17] Tipos de Red y redes. Katherine Paredes . [Online] http://katherynnparedes.blogspot.com/

[18] Wirahark y su uso en las redes. Rarst.net [Online] Fuente Ingles (Estados Unidos) http://www.rarst.net/software/wireshark-ports/

FIGURAS

(1) (2) (3) Aprendiendo a programar en LibPcap. Documentation TCPDUMP. [Online] http://www.e-ghost.deusto.es/docs/2005/conferencias/pcap.pdf

(4) (5) TurboCap**;**Gigabit Ethernet Packet Capture; User´s Guide

(6) (7) Manual básico de Wireshark. Slide share. [Online] http://www.slideshare.net/DIANYSS2012/manual-bsico-de-wireshark

(8) Datasheet RTL8931D