Documentación Proyecto 1- Analizador de red

Adrian Garcia
Departamento de Ingeriería de Sistemas
Pontificia Universidad Javeriana
Bogotá, Colombia
adriangarcia@javeriana.edu.co

Nicolás Miranda
Departamento de Ingeriería de Sistemas
Pontificia Universidad Javeriana
Bogotá, Colombia
nicolasmiranda@javeriana.edu.co

Miguel Romero
Departamento de Ingeriería de Sistemas
Pontificia Universidad Javeriana
Bogotá, Colombia
miguel-romero@javeriana.edu.co

I. Introducción

En este proyecto se desarrolló un analizador de protocolos (sniffer) y un medidor de throughput de red.

Dependiendo del protocolo que sea recibido, la aplicación muestra una tabla de datos como los mostraría Wireshark [1], el cual es uno de los analizadores de protocolos de red más conocido y utilizado del mundo, permitiendo ver qué está sucediendo en la red con gran detalle, esto para para poner en práctica los conceptos vistos en clase sobre protocolos de capa de enlace y capa de red.

Para la medición del throughput se implementó un odómetro para mostrar en tiempo real el throughput de la red, tanto para los datos recibidos como para los datos enviados por la máquina.

II. Desarrollo

Este proyecto fue desarrollado en el lenguaje Jjava con el IDE NetBeans 8.2. Para el analizador de protocolos se utilizó la librería Jpcap. La librería Jpcap sirve para hacer la captura de paquetes de red, la cual está basada en libpcap/winpcap, por lo cual puede trabajar en cualquier sistema operativo que soporte libpcap/winpcap. Entre los protocolos que se analizaron están ICMP, IPv4, Ethernet, TCP, UDP, ARP, los cuales son soportados por Jpcap [2].

TCP (Transmission Control Protocol) es un protocolo host-to-host para redes con conmutación de paquetes [3]. UDP (User Datagram Protocol) es un protocolo que hace disponible la comunicación por medio de datagramas entre ordenadores, pero no garantiza la entrega ni la protección ante duplicados [4]. ARP (Adress Resolution Protocol) es un protocolo para la conversión de direcciones, como por ejemplo direcciones IP a direcciones de red [5]. ICMP (Internet Control Message Protocol) es un protocolo que envía mensajes de error, ya sea que el router o host no pueda ser localizado [6].

En NetBeans se utilizaron las clases PacketContents y AnalizadorPaquetes. En PacketContents se crearon las variables de los tipos de los protocolos que se analizan y una lista de objetos. En el método receivePacket, se recibe un paquete, se hace un cálculo para el tiempo de recepción desde el inicio de la captura y se evalúa qué tipo de paquete es. Para cada paquete se obtienen diferentes datos como el número, el tiempo que se calculó, la dirección origen, la dirección destino, la longitud, entre otros específicos a cada tipo de paquete. Estos datos son los que después se muestran en la tabla de paquetes de AnalizadorPaquetes.

En la clase AnalizadorPaquetes se obtienen las interfaces de red que hay en la máquina, por medio del método JpcapCaptor.getDeviceList. Para cada dispositivo se obtienen la descripción, el nombre del enlace de datos y la descripción del enlace de datos. Estas interfaces se listan en un combobox para que el usuario seleccione una. En el método de CapturePackets, se realiza la captura de los paquetes para el dispositivo seleccionado y estos son procesados por medio del método processPacket el cual recibe una instancia de la clase PacketContents.

La librería Jacob fue utilizada para la medición del throughtput. Está librería es un puente para la automatización de componente COM desde Java, permitiendo que las clases se combinen con cualquier DLL o ActiveX, ofreciendo un marco completo para la comunicación con componentes COM / DLL a través de código fuente [2]. Se usó además la librería trident para la configuración del aspecto gráfico de cada odómetro.

Para la medición del throughput de la red se crearon dos clases, la clase Throughput y MedicionThroughput. En la clase Throughput se crearon las siguientes variables: bytesRecibidos, bytesEnviados, anchoBanda, listInterfaces, nInterfaces; junto con sus métodos set y get. En esta, se hizo uso de dos funciones: la función listarInterfaces, que se encarga de hacer un listado de las interfaces para listarlas en un combobox en MedicionThrougput y así que el usuario pueda seleccionar una; la otra función es la función run, la cual obtiene la cantidad de bytes recibidos, bytes enviados y ancho de banda para la interface de red seleccionada.

En la clase MedicionThroughput hay una pestaña en la cual se listan las interfaces y un botón, el cual a la interfaz seleccionada hace el llamado a la función run de la clase throughput para calcular la cantidad de bytes recibidos, la cantidad de bytes enviados, el ancho de banda, ejecutándose cada segundo, para actualizar el odómetro en tiempo real y permitir el cálculo de bps tanto para datos recibidos como enviados. Se realiza el cálculo del throughput aplicando la ecuación (1):

$$throughput = \frac{diferenciaBytes*8}{10^6} \qquad (1)$$

Donde diferenciaBytes es la diferencia en bytes (recibidos o enviados) entre el valor obtenido en el tiempo t+1 s y el tiempo t, por lo que se multiplica por 8 y se divide por 10^6 para conseguir el valor en Mbps. Una vez se calculan el throughput de bytes recibidos y enviados, se utiliza el método setValue de cada odómetro para establecer los valores y así generar los gráficos respectivos, también durante cada segundo. En el caso de valores de throughput menores a 1 Mbps y superiores a 1 Kbps se muestran estos en Kbps en la parte inferior de la ventana y en el caso de valores inferiores a 1 Kbps se muestran los mismos en bps. Esto se realizó de esta manera ya que al ser los valores tan pequeños en comparación al ancho de banda, no es posible visualizar los cambios correctamente en el odómetro.

III. Clases y funciones

Se muestra el diagrama de clases para el proyecto (Fig. 2) y a continuación se habla brevemente de cada clase.

- PacketContents: las variables definidas en esta clase son: tcp, udp, icmp, arp, ip, listaPaquetes y listaEthernet
  - receivePacket: recibe un paquete y se dependiendo del tipo de paquete que sea se le extraen los datos necesarios.
- AnalizadorPaquetes: las variables en esta clase son una lista de interfaces de red, un captor, un hilo, un índice, una bandera, un contador, el estado de la captura, un número y el tiempo de inicio
  - CapturePackets: en un hilo (thread) hace la captura de los paquetes y se pasan al processPacket.
  - processPacket: esta función es propia del JpcapCaptor
- Throughput: esta clase tiene las siguientes variables: bytesRecibidos, bytesEnviados, anchoBanda, listaInterfaces, nInterfaces.
  - Run: por cada interfaz que hay se consiguen los datos de bytes enviados, bytes recibidos y ancho de banda.
  - ListarInterfaces: hace listado de las interfaces que tiene el dispositivo.
- MedicionThroughput: es la ventana en donde se mostrarán los odómetros, por medio de las variables gaugeRecibido, gaugeEnviado, también hay una lista de dispositivos.
  - buttonMedirActionPerformed: al momento de hacer clic será la encargada de hacer los cálculos de los throughputs.
  - MedicionThroughput: en el constructor se inicia el frame y se inicializa la lista de interfaces
- Jpcap_thread: clase auxiliar para el analizadorPaquetes. Sus variables son: value, thread, threadVar.
  - Interrupt: interrumpe al hilo si es null.
  - Start: si el valor de threadVar no es null, se sigue llamando.
- Radial: en esta clase las principales variables son: BASE, minMeasuredImage, maxMeasuredImage, unitStringWidth, angle.
  - Round: para este método, el número de lugares tiene que ser mayor a cero. Luego eleva 10 al número de lugares y lo guarda en una variable factor, posteriormente
  - multiplica el valor que le llega de valor por el factor y redondea ese valor.

IV. Escenario y variables de entorno

Los escenarios en los que este proyecto fue probado son: conexión alámbrica e inalámbrica. Para el caso de conexión alámbrica las pruebas fueron hechas en la red LAN del laboratorio de redes de la Facultad de Ingeniería de la Pontificia Universidad Javeriana, utilizando los computadores de la misma, junto con los switch que se encuentran allí. Para el escenario inalámbrico se utilizaron diferentes redes WiFi incluyendo la de diferentes zonas del campus de la Pontificia Universidad Javeriana. Las pruebas se realizaron sobre equipos con sistema operativo Windows 10, verificándose en todos los casos la correcta recepción de paquetes Ethernet, IPv4, ICMP, TCP, UDP, ARP, además del correcto funcionamiento del medidor de throughput.

V. Resultados

Para mostrar los resultados se muestran a continuación las capturas de pantalla que ilustran el resultado de la aplicación, en donde se puede ver además el funcionamiento de la misma.

- Ventana de inicio

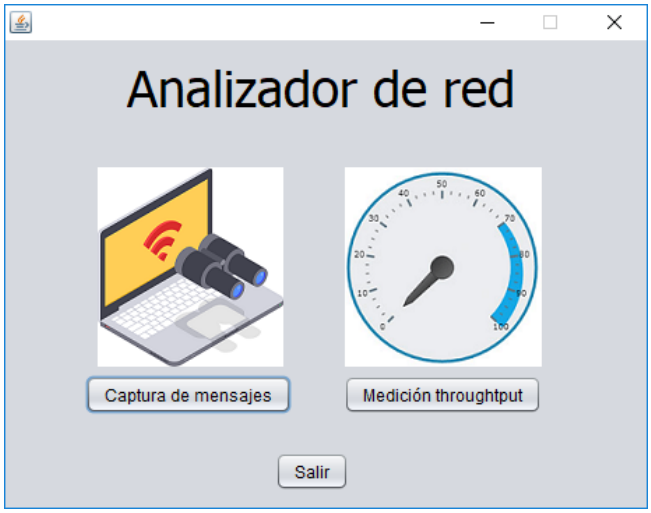

Figura 1. Pantalla de inicio.

- Medición throughput

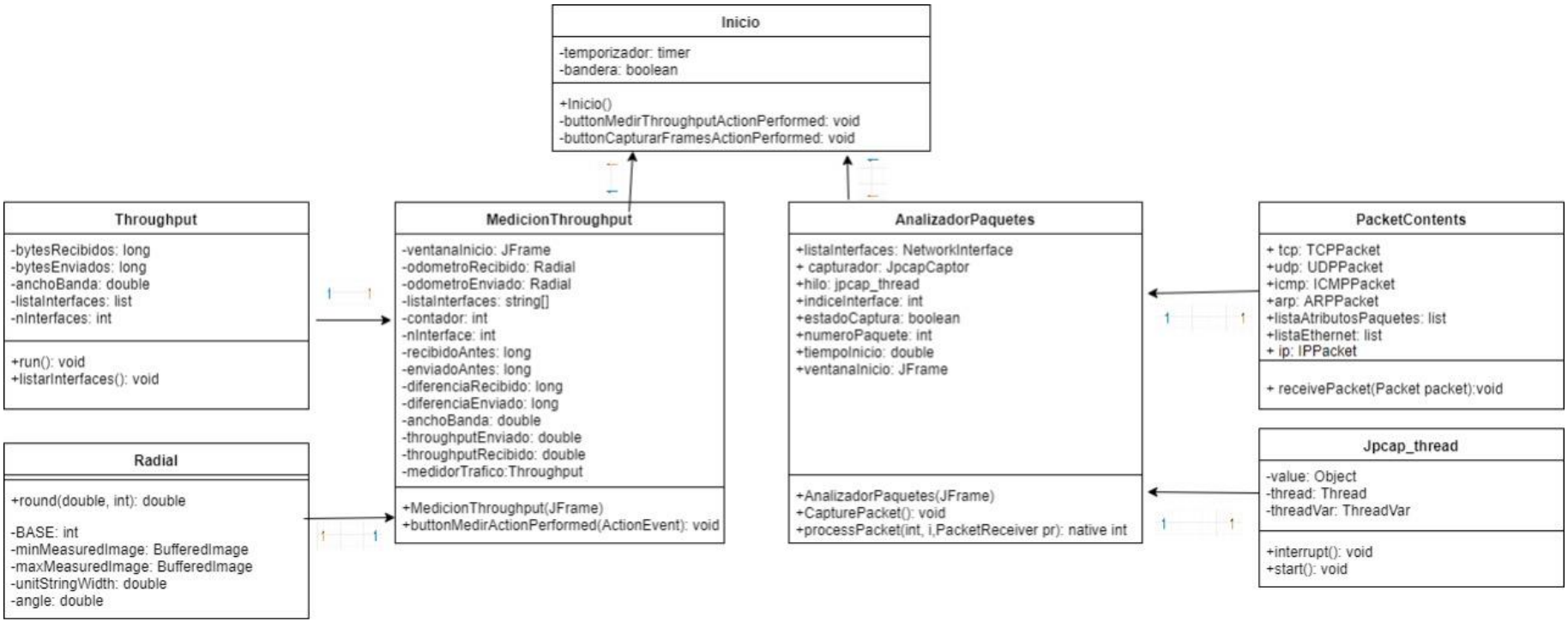

Figura 2. Diagrama de clases del proyecto.

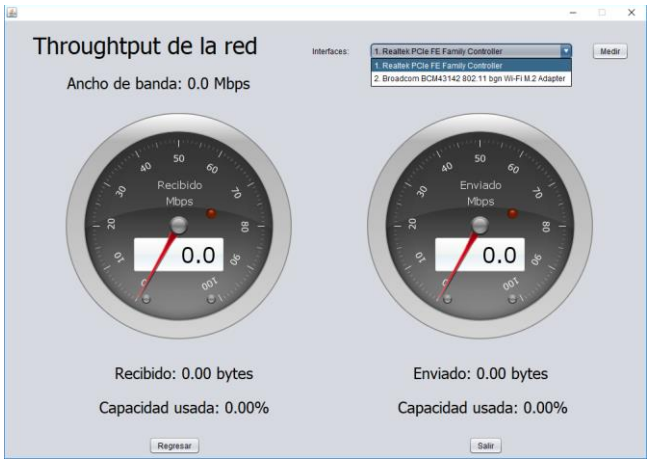

Figura 3. Medición de throughput.

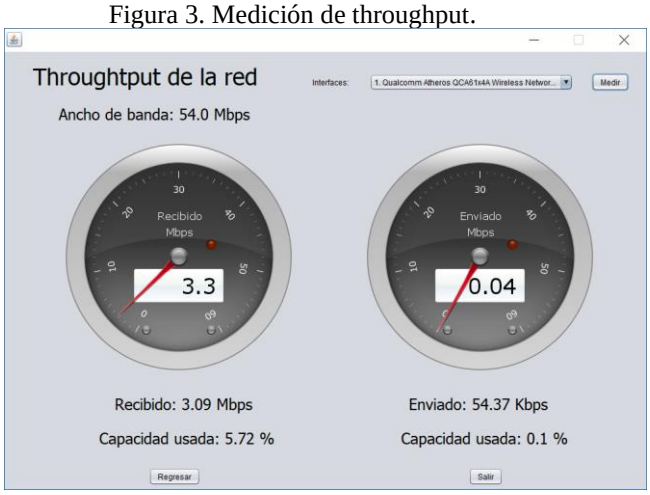

Figura 4. Medición durante reproducción online de vídeo.

- Captura de paquetes

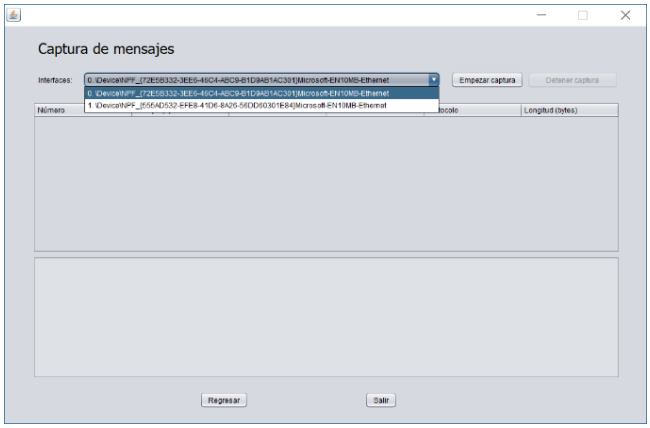

Figura 2. Ejemplo de captura.

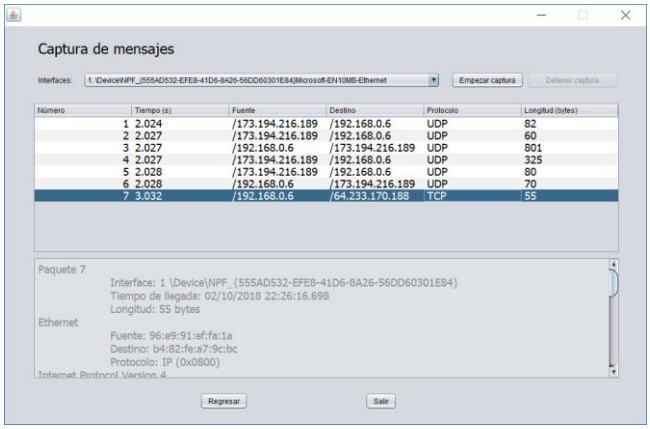

Figura 3. Selección paquete TCP, información general.

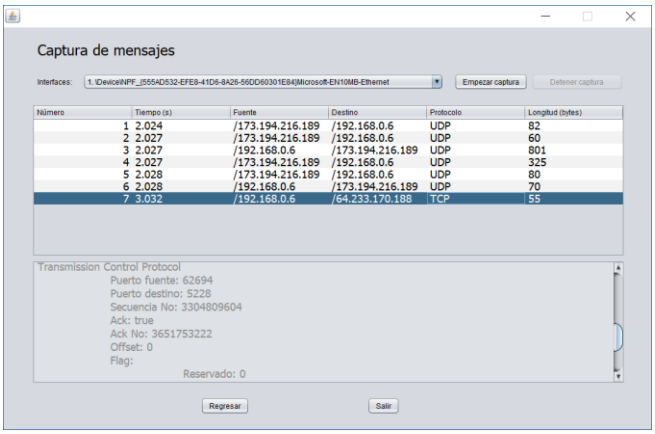

Figura 7. Selección paquete TCP, información específica.

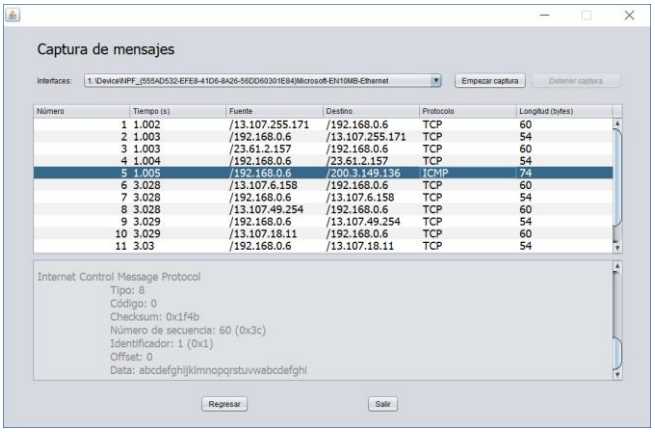

Figura 7. Selección paquete ICMP.

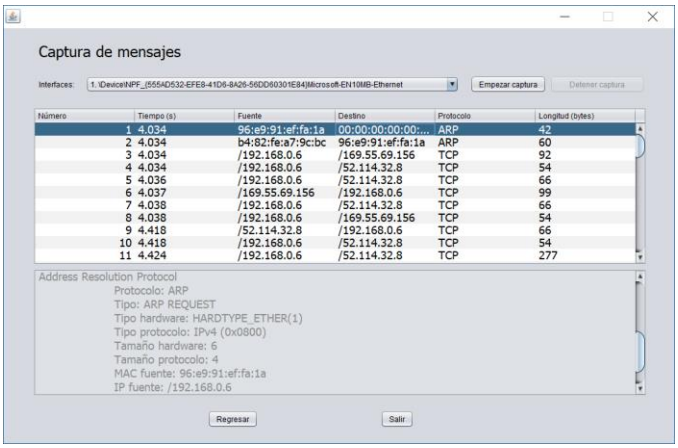

Figura 8. Selección paquete ARP.

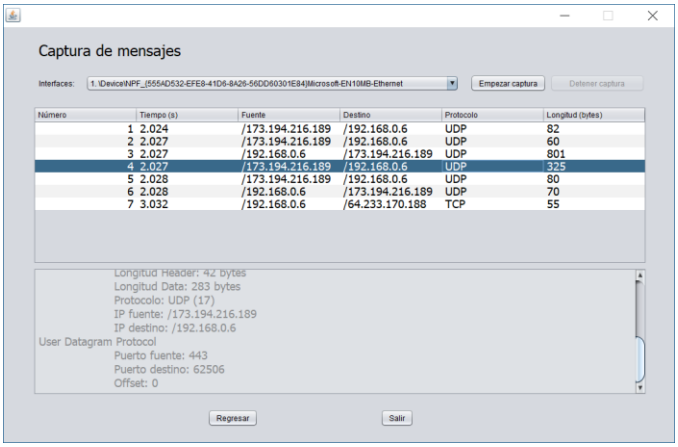

Figura 9. Selección paquete UDP.

VI. CONCLUSIONES

Con la realización de este proyecto se comprendió a profundidad el comportamiento del protocolo Ethernet, sus componentes y su parte lógica. Además, mediante la elaboración del medidor throughput, se observaron las diferentes métricas que se utilizan para cuando sucedan incidentes se puedan tomar medidas de prevención en caso de congestión o alto tráfico. También se puso en práctica los conceptos teóricos vistos en clase relacionados con la capa de enlace y capa de red.

VII. REFERENCIAS

[1] WireShark, “Descripcion WireShark.” [Online]. Available: https://www.wireshark.org/. [Accessed: 20-Aug-2009].

[2] SourceForge. "JACOB - Java COM Bridge." [Online] Available https://sourceforge.net/projects/jacob-project/

[2] D. Rusek, “Documentacion jpcap.” [Online]. Available: https://github.com/mgodave/Jpcap.

[3] I. E. T. Force, “TRANSMISSION CONTROL PROTOCOL.” [Online]. Available: https://www.rfc-es.org/rfc/rfc0826-es.txt.

[4] Rfc.-es.org, “PROTOCOLO DE DATAGRAMAS DE USUARIO.” [Online]. Available: https://www.rfc-es.org/rfc/rfc0768-es.txt.

[5] Rfc.-es.org, “Adress Resolution Protocol.” .

[6] Rfc.-es.org, “INTERNET CONTROL MESSAGE PROTOCOL.” [Online]. Available: https://tools.ietf.org/html/rfc792.