Revista Ciencia & Tecnología

No. 20, 31 de octubre de 2018
ISSN impreso: 1390 - 6321

Visualizador de tráfico de red de comunicación basadas en la arquitectura TCP/IP

Communication network traffic viewer based on TCP/IP architecture

Byron Oviedo[1]
boviedo@uteq.edu.ec
Luís Suarez Litardo[1]
luis.suarez2015@uteq.edu.
Emilio Zhuma Mera[1]
ezhuma@uteq.edu.ec
Amilkar Puris Raúl Hernández[1]
apuris@uteq.edu.ec

Recibido: 1/07/2018, Aceptado: 1/09/2018

RESUMEN

El presente trabajo muestra el desarrollo de una aplicación telemática que permite la captura y visualización de paquetes de red enfocados en la arquitectura TCP/IP, esta aplicación se compone de 2 partes esenciales, la primera es la del servidor la cual cumple la función de capturar los datos de la red y enviarlos a través del protocolo WebSocket y la segunda es por parte del cliente el cual consume los datos que son proporcionados por el servidor. Para el desarrollo de esta aplicación se toma el modelo referencial TCP/IP el cual agrupa capas las cuales son: Acceso a red, Internet, Transporte, Aplicación, existen más de cien protocolos que engloban esta arquitectura pero por cuestiones investigativas se ha tomado un protocolo de cada capa los cuales son respectivamente: Trama Ethernet, IPV4, TCP, UDP, HTTP con la finalidad de entender su funcionamiento y aprender de los procesos que se realizan y la forma de acoplarse con sus protocolos de capa superior. La captura de tráfico de red se la realizo utilizando la librería de google llamada Gopacket esta librería ofrece métodos e interfaces cómodas para el desarrollo de software de red. En el lado del cliente se escribió una pequeña aplicación en JavaScript para el consumo de los datos desde el servidor a través del protocolo webSocket y para que la interfaz gráfica sea agradable al usuario se utilizó un marco de trabajo para interfaces web modernas, responsivas y basado en material design llamado Materialize.

[1] Docentes de la Universidad Técnica Estatal de Quevedo. Ecuador

No. 20, 31 de octubre de 2018
ISSN impreso: 1390 - 6321

Palabras Clave: websocket, javascript, gopacket, TCP/IP

ABSTRACT
The present work shows the development of a telematic application that allows the capture and visualization of network packets focused on the TCP/IP architecture, this application is composed of 2 essential parts, the first is that of the server which fulfills the function of capturing the network data and send it through the WebSocket protocol and the second is by the client which consumes the data that is provided by the server. For the development of this application, the TCP/IP referential model is taken which groups layers which are: Network access, Internet, Transport, Application, there are more than one hundred protocols that encompass this architecture but for research reasons a protocol has been taken of each layer which are respectively: Ethernet, IPV4, TCP, UDP, HTTP frame in order to understand its operation and learn from the processes that are carried out and how to connect with its upper layer protocols. The capture of network traffic was done using the google library called Gopacket. This library offers convenient methods and interfaces for the development of network software. On the client side, a small JavaScript application was written for the consumption of data from the server through the webSocket protocol and to make the graphical user-friendly interface a framework for modern, responsive and based web interfaces was used in material design called Materialize.

Keywords: websocket, javascript, gopacket, TCP/IP

Introducción

La cantidad de información que se generan en una red de datos tiene una manera sistemática de ordenarse y esto se hace a partir de una arquitectura de red la cual es TCP/IP pero no es la única existente, otra muy famosa conocida como OSI (Interconexión de sistemas abiertos), lo esencial es entender el funcionamiento de cada una de las capas y la manera de comunicarse una con la otra y las interfaces que proporciona cada una de ellas.

Cada capa tiene una cierta cantidad de protocolos que realizan ciertas operaciones desde la más baja hasta la más alta, en esta investigación se han estudiado 5 protocolos de comunicación los cuales son: TCP, UDP, ETHERNET TYPE 2, IPV4, IPV6, cada uno de estos protocolos son el resultado de varias investigaciones, la información acerca de cada uno de ellos se encuentran en una serie de artículos publicado en internet por el IETF (Grupo de Trabajo de Ingeniería de Internet) llamados RFC (Request For Coment) estos artículos son muy interesantes ya que muestran de manera minuciosa el funcionamiento, la arquitectura, el formato de cabecera de cada uno de los protocolos entonces esta ha sido la bibliográfica esencial de esta investigación.

Luego de conocer íntimamente el funcionamiento de cada protocolo empezó lo más interesante, el desarrollo de una aplicación que puede captar de alguna forma todos esos paquetes que atraviesan la red, pero ¿cómo hacerlo? El proceso para capturar esos paquetes realmente es muy complejo pero existen herramientas que minimizan ese trabajo y ayuda al desarrollador de software a lograr su objetivo sin invertir una cantidad excesiva de tiempo, es por ello que utilizo el lenguaje de programación golang el cual es una tecnología moderna en el ámbito de desarrollo de software actual ofrece varias características como: tipado estático y dinámico, retorno de

varios valores de diferentes tipos de datos en las funciones, programación concurrente, varios paradigmas de programación como el orientado a objetos, imperativo, compilado y un recolector de basuras muy eficiente.

Google, en 2016, liberó una librería muy interesante llamada Gopacket desarrollada en golang y es esta librería la cual se usó en esta aplicación. Junto a ella gopacket se comunica directamente con otra librería muy conocida en este ámbito llamada libpcap desarrollado en C/C++, aunque se pudo trabajar directamente con libpcap, realmente gopacket es más sencilla de utilizar. Es muy cómoda y se pueden obtener resultados rápidamente mientras que libpcap tomará un poco más de tiempo usarla, pero se podrán entender otros conceptos acerca de cómo está formada y como se realiza la captura de los paquetes en un nivel más bajo de abstracción.

Una vez obtenido los datos es momento de visualizarlos. Para ello se desarrolló un cliente front-end en JavaScript que permita una comunicación full dúplex con el servidor de paquetes de red y otro para el intercambio de datos bidireccional y al mismo instante. Una de estas tecnologías es el protocolo webSocket el cual permite este tipo de intercambio de información entre el cliente y el servidor a través del formato de serialización JSON y se aplicó un poco de diseño para que interfaz gráfica sea más amigable con el usuario usando el framework Vue.js que tiene varias características interesantes.

Métodos

Modelo de Referencia OSI

Este modelo se basa en una propuesta desarrollada por la Organización Internacional de Normas (ISO) como el primer paso hacia la estandarización internacional de los protocolos utilizados en las diversas capas (Day y Zimmerman, 1983). Este modelo se revisó en 1995 (Day, 1995) y se llama Modelo de referencia OSI (Interconexión de Sistemas Abiertos, del inglés Open Systems Interconnection) de la iso puesto que se ocupa de la conexión de sistemas abiertos; esto es, sistemas que están abiertos a la comunicación con otros sistemas. Para abreviar, lo llamaremos modelo OSI (Day, 1995).

El modelo OSI tiene siete capas. Los principios que se aplicaron para llegar a las siete capas se pueden resumir de la siguiente manera:

- Se debe crear una capa en donde se requiera un nivel diferente de abstracción.
- Cada capa debe realizar una función bien definida.
- La función de cada capa se debe elegir teniendo en cuenta la definición de protocolos estandarizados internacionalmente.
- Es necesario elegir los límites de las capas de modo que se minimice el flujo de información a través de las interfaces.
- La cantidad de capas debe ser suficiente como para no tener que agrupar funciones distintas en la misma capa; además, debe ser lo bastante pequeña como para que la arquitectura no se vuelva inmanejable.

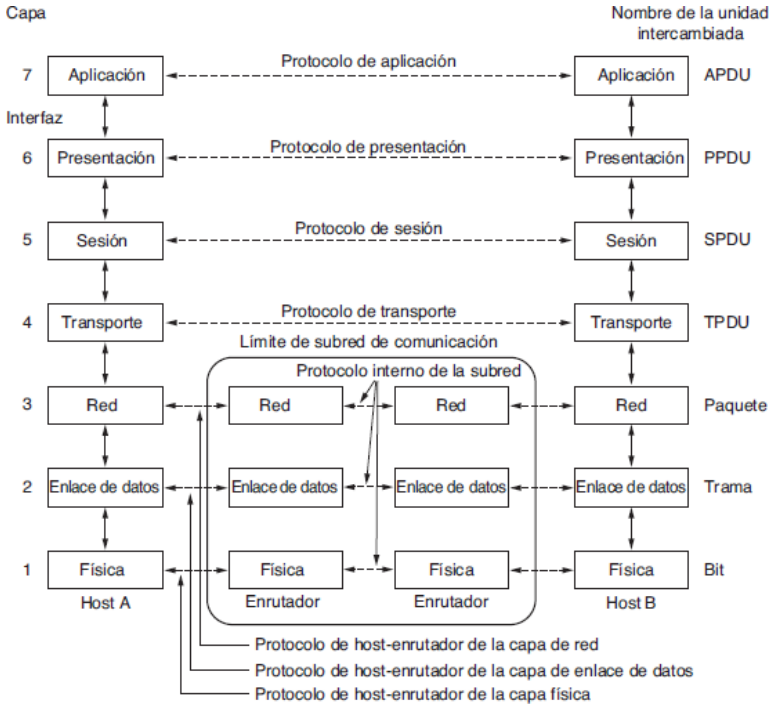

Gráfico 1. Capas, Referencias e interfaces por Tanenbaum
Fuente: Elaboración propia

El modelo de referencia TCP/IP
El término genérico "TCP/IP" usualmente significa cualquier cosa y todo con referencia a los protocolos específicos TCP e IP. Pueden incluir otros protocolos, aplicaciones e incluso los medios de red. Un ejemplo de estos protocolos son: UDP, ARP e ICMP. Un ejemplo de estas aplicaciones son: TELNET, FTP y RPC. Un término más preciso es "tecnología internet". A una red que usa una tecnología internet se le llama una "internet" (Deering y Hinden, 1998).

Esta es la estructura lógica de los protocolos divididos en capas dentro de un computador en internet. Cada computador que se puede comunicar usando una tecnología internet tiene también una estructura.

Es esta estructura lógica la que determina el comportamiento del computador en internet. Las cajas representan el procesado de los datos conforme van atravesando el computador y las líneas que conectan las cajas muestran el camino de los datos. La línea horizontal de abajo representa el cable de Ethernet, el "o" es el transceptor. El "*" es la dirección IP y la "@" es la dirección Ethernet. Entender esta estructura lógica es esencial para entender la tecnología internet; hay referencias a ella a lo largo de todo este tutorial (Socolofsky y Kale, 1991).

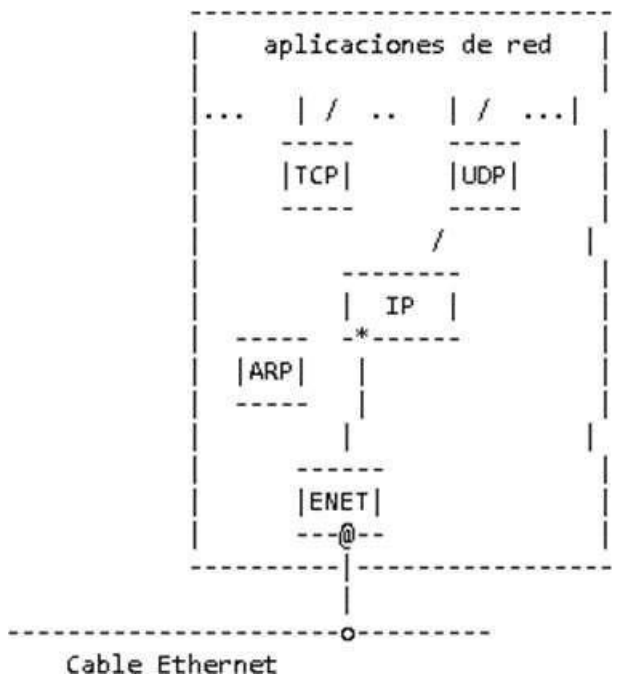

Gráfico 2. Nodo de Red Básico TCP/IP - RFC-1180
Fuente: Elaboración propia

Protocolos de Comunicaciones TCP
El "protocolo de control de transmisión" ('Transmission Control Protocol', TCP) está pensado para ser utilizado como un protocolo 'host' a 'host' muy fiable entre miembros de redes de comunicación de computadoras por intercambio de paquetes y en un sistema interconectado de tales redes (University of Southern California, 1981).

TCP es un protocolo orientado a la conexión, fiable y entre dos extremos, diseñado para encajar en una jerarquía en capas de protocolos que soportan aplicaciones sobre múltiples redes. TCP proporciona mecanismos para la comunicación fiable entre pares de procesos en computadoras 'host' ancladas en redes de comunicación de computadoras distintas, pero interconectadas.

Se hacen muy pocas suposiciones sobre la fiabilidad de los protocolos de comunicación por debajo de la capa de TCP. TCP sólo supone que puede acceder a un servicio de transmisión de datagramas simple, aunque en principio poco fiable, de los protocolos del nivel inferior. En principio, TCP debería ser capaz de operar encima de un amplio espectro de sistemas de comunicaciones que incluye desde conexiones por cables fijos ('hard-wired conections') hasta redes de intercambio de paquetes o redes de circuitos conmutados.

TCP se basa en los conceptos descritos primeramente por Cerf y Kahn. TCP encaja en una arquitectura de protocolos en capas justo por encima del protocolo de internet, protocolo básico que proporciona un medio para TCP de enviar y recibir

segmentos de longitud variable de información envuelta en "sobres" de datagramas de internet. El datagrama de internet proporciona un medio de direccionar TCPs de origen y de destino situados en redes diferentes. El protocolo de internet también trata con la fragmentación y el reensamble de segmentos de TCP que sean necesarios para conseguir el transporte y la entrega sobre múltiples redes y las puertas de enlace que las interconectan. El protocolo de internet también lleva información sobre la prioridad, clasificación de seguridad y compartimentación de los segmentos de TCP, de tal forma que esta información pueda ser comunicada de extremo a extremo entre múltiples redes (Postel, 1981).

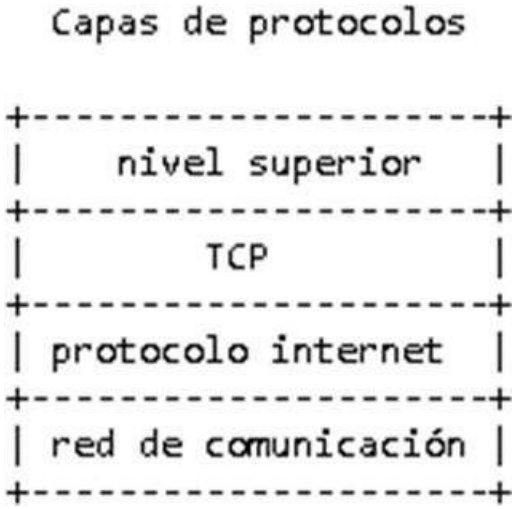

Gráfico 3. Pila TCP/IP incluyendo a tcp - RFC-793
Fuente: Elaboración propia

Trama Ethernet Type II
La trama es lo que se conoce también por el nombre de "frame". El primer campo es el preámbulo que indica el inicio de la trama y tienen el objeto de que el dispositivo que lo recibe detecte una nueva trama y se sincronice. El delimitador de inicio de trama indica que el frame empieza a partir de él. Los campos de MAC (o dirección) de destino y origen indican las direcciones físicas del dispositivo al que van dirigidos los datos y del dispositivo origen de los datos, respectivamente.

La etiqueta es un campo opcional que indica la pertenencia a una VLAN o prioridad en IEEE P802.1p Ethernetype indica con que protocolo están encapsulados los datos que contiene la Payload, en caso de que se usase un protocolo de capa superior (Horning, 1984).

La Payload es donde van todos los datos y, en el caso correspondiente, cabeceras de otros protocolos de capas superiores (Según Modelo OSI, véase Protocolos en informática) que pudieran formatear a los datos que se tramiten (IP, TCP, etc). Tiene un mínimo de 64 Bytes (o 42 si es la versión 802.1Q) hasta un máximo de 1518 Bytes. Los mensajes inferiores a 64 bytes se llaman tramas enanas (runt frames) e indican mensajes dañados y parcialmente transmitidos.

La secuencia de comprobación es un campo de 4 bytes que contiene un valor de verificación CRC (control de redundancia cíclica). El emisor calcula el CRC de toda la trama, desde el campo destino al campo CRC suponiendo que vale 0. El receptor lo recalcula, si el valor calculado es 0 la trama es válida. El gap de final de trama son 12 bytes vacíos con el objetivo de espaciado entre tramas (Horning, 1984).

User Datagram Protocol (UDP)
Este Protocolo de Datagramas de Usuario (UDP: User Datagram Protocol) se define con la intención de hacer disponible un tipo de datagramas para la comunicación por intercambio de paquetes entre ordenadores en el entorno de un conjunto interconectado de redes de computadoras. Este protocolo asume que el Protocolo de Internet (IP: Internet Protocol) se utiliza como protocolo subyacente (Postel, 1980).

Este protocolo aporta un procedimiento para que los programas de aplicación puedan enviar mensajes a otros programas con un mínimo de mecanismo de protocolo. El protocolo se orienta a transacciones, y tanto la entrega como la protección ante duplicados no se garantizan. Las aplicaciones que requieran de una entrega fiable y ordenada de secuencias de datos deberían utilizar el Protocolo de Control de Transmisión (TCP: Transmission Control Protocol).

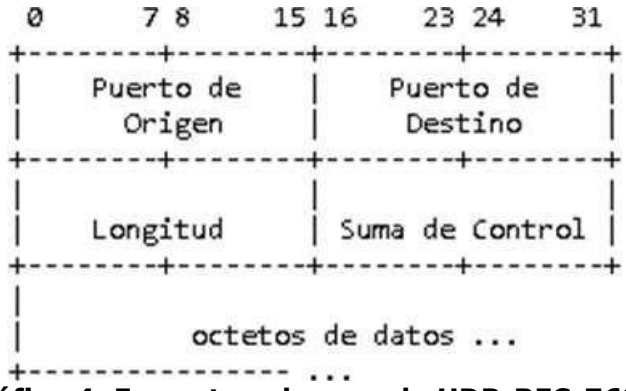

Gráfico 4. Formato cabecera de UDP-RFC-768
Fuente: Elaboración propia

HTTP
El Protocolo de transferencia de hipertexto (en inglés: Hypertext Transfer Protocol o HTTP) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web. HTTP fue desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force, colaboración que culminó en 1999 con la publicación de una serie de RFC, el más importante de ellos es el RFC 2616 que especifica la versión 1.1. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. HTTP es un protocolo sin estado, es decir, no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de sesión y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

Protocolo de Internet Versión 4
El Protocolo Internet está diseñado para su uso en sistemas interconectados de redes de comunicación de ordenadores por intercambio de paquetes. A un sistema de este tipo se le conoce como "catenet". El protocolo internet proporciona los medios necesarios para la transmisión de bloques de datos llamados datagramas desde el origen al destino, donde origen y destino son hosts identificados por direcciones de longitud fija. El protocolo internet tambien se encarga, si es necesario, de la fragmentación y el reensamblaje de grandes datagramas para su transmisión a través de redes de trama pequeña.

El protocolo internet implementa dos funciones básicas: direccionamiento y fragmentación. Los módulos internet usan campos en la cabecera internet para fragmentar y reensamblar los datagramas internet cuando sea necesario para su transmisión a través de redes de "trama pequeña".

El modelo de operación es que un módulo internet reside en cada host involucrado en la comunicación internet y en cada pasarela que interconecta redes. Estos módulos comparten reglas comunes para interpretar los campos de dirección y para fragmentar y ensamblar datagramas internet. Además, estos módulos (especialmente en las pasarelas) tienen procedimientos para tomar decisiones de encaminamiento y otras funciones.

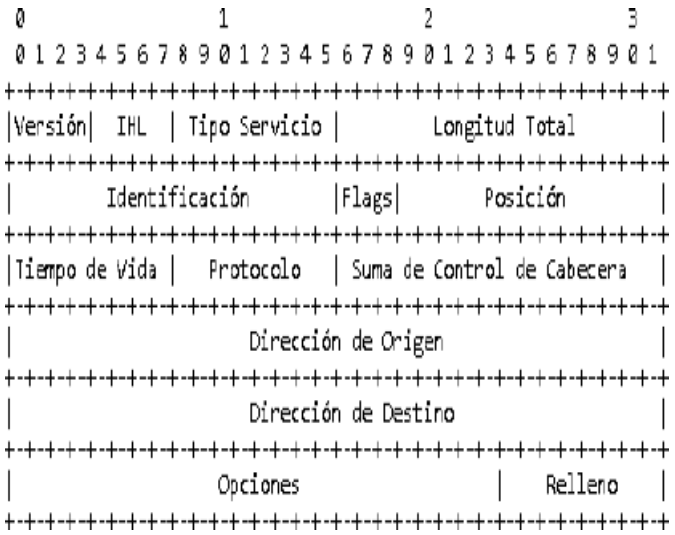

Gráfico 5. Formato de Cabecera de un Datagrama Internet IPV4
Fuente: Elaboración propia

Lenguaje de Programación Go (Golang)
Es un lenguaje de programación creado por la compañía google, La comunidad de Go recomienda llamar a "go" como "golang" para la publicación de información en la red, porque al momento de realizar una búsqueda en internet referente al lenguaje con la palabra "go" es un poco dificultoso encontrar resultados que te lleven al lenguaje de programación go, dicho lo anterior para cualquier tipo de búsqueda usar la palabra "golang" que además se está usando en este documento. El lenguaje de

programación Go es un proyecto de código abierto que hace que los programadores sean más productivos (Tanenbaum y Wetherall, 2012).

Go es expresivo, conciso, limpio y eficiente. Sus mecanismos de concurrencia facilitan la escritura de programas que aprovechan al máximo las máquinas multicore y en red, mientras que su nuevo tipo de sistema permite la construcción de programas flexibles y modulares. Go compila rápidamente al código de la máquina pero tiene la conveniencia de recolección de basura y el poder de la reflexión en tiempo de ejecución. Es un lenguaje rápido, escrito de forma estática y compilada que se siente como un lenguaje interpretado de forma dinámica.

Resultados

Fase 1 Análisis de la Información

Para la realización de este proyecto el cual consiste en una aplicación web para capturar y visualizar los datos que se generan en una red de comunicación, se investigaron varios conceptos fundamentales enfocados en las redes de datos como: redes de área local, arquitectura de red la cual es TCP/IP, los protocolos que involucran esta arquitectura como tcp,udp,icmp,ipv4, ipv6 entre otros existen artículos científicos muy interesantes sobre todo que tienes que ver con las redes de comunicación, estos artículos son llamados RFC (Request for comment) los cuales son una serie de publicaciones del grupo de trabajo de ingeniería de internet que describen diversos aspectos del funcionamiento de internet y otras redes de computadoras, como protocolos, procedimientos, arquitecturas etc.

Cada RFC constituye un memorando que ingenieros o expertos en la materia han hecho llegar al IETF el consorcio de colaboración técnica más importante en internet, para que este sea valorado por el resto de la comunidad. De hecho la traducción literal de RFC al español es "Petición de Comentarios".

En base a la información adquirida de esta seria de publicaciones RFC se pudo obtener los datagramas o formatos de cabeceras de cada uno de los protocolos para poder ser reflejados en la aplicación.

Pero basta con solo obtener la información también se debe de desarrollar el software que permita la obtención de estos campos de los protocolos, para ellos y además es uno de los pilares fundamentales es la librería desarrollada por google en el 2016 llamada Gopacket programada en el lenguaje Go o también llamado Golang que además es el lenguaje con el cual esta aplicación está desarrollada junto con otras herramientas más, entonces básicamente esta librería ha proporcionado una ayuda inmensa en el desarrollo de esta aplicación telemática.

Fase 2 Diseño

Para el diseño de esta aplicación lo primero es saber que es protocolos se va a escanear esto se vio en la fase anterior en esta fase se verá los elementos necesarios para el desarrollo duro y puro de la App:

Primero utilizaremos un sistema operativo gnu/Linux este sistema operativo se utilizara porque es compatible con una librería muy interesante para la captura de trafico de red la cual se llama libpcap esta librería esta diseña en c/c++ pero como anteriormente dije que usaremos la librería gopacket de google, sucede que Gopacket

se comunica directamente con la librería lipcap pero una pregunta interesante seria porque no usar directamente lipcap, es correcto utilizar lipcap no hay ningún problema en cuanto a resultados pero en cuanto a desarrollo gopacketes más sofisticada por el sencillo hecho de estar hecho en golang un lenguaje moderno que acoge las mejores características de todos los lenguajes de programación, pero también gopacket proporciona decodificación de para paquetes para el lenguaje go, además contiene muchos su paquetes con funcionalidades adicionales que pueden resultar muy utilices.

Otro punto importante es tener instalado en lenguaje golang y bien configurado con la estructura recomendada por los desarrolladores de go porque al momento de realizar importaciones de librería estas se organizaran de tal forma como se muestra en la documentación del lenguaje, la versión de go utilizada en esta App es la versión 1.9 una versión estable y recomendada para trabajo duro.

El entorno de desarrollo utilizado será golang por jetbrain es un ide muy completo que ofrece soporte para el lenguaje golang tiene algunas características como: autocompletado de código, sintaxis de color, depuración directa conexión con control de versiones de software entre otras.

La forma de envió de los datos se lo realizo vía JSON(JavaScript Object Notation) es un formato de texto ligero para el intercambio de datos JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando la función eval (), lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.

El desarrollo del cliente front-ent es fundamental para la visualización de los datos en formato json estos se envían atreves de un protocolo de comunicación llamado WEBSOCKET es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor. La API de WebSocket está siendo normalizada por el W3C, mientras que el protocolo WebSocket ya fue normalizado por la IETF como el RFC 6455.

Debido a que las conexiones TCP comunes sobre puertos diferentes al 80 son habitualmente bloqueadas por los administradores de redes, el uso de esta tecnología proporcionaría una solución a este tipo de limitaciones proveyendo una funcionalidad similar a la apertura de varias conexiones en distintos puertos, pero multiplexado diferentes servicios WebSocket sobre un único puerto TCP (a costa de una pequeña sobrecarga del protocolo)

Y por último no menos importante el desarrollo del cliente webSocket para ellos se ha desarrollado un App sencilla que permita obtener los objetos json y mostrarlos

para ello se a utilizado el lenguaje JavaScript junto al framework Vue.Js es un marco de JavaScript de código abierto progresivo para crear interfaces de usuario La integración en proyectos que usan otras bibliotecas de JavaScript se hace más fácil con Vue porque está diseñada para ser adoptable incrementalmente. Vue también puede funcionar como un marco de aplicación web capaz de impulsar aplicaciones avanzadas de una sola página.

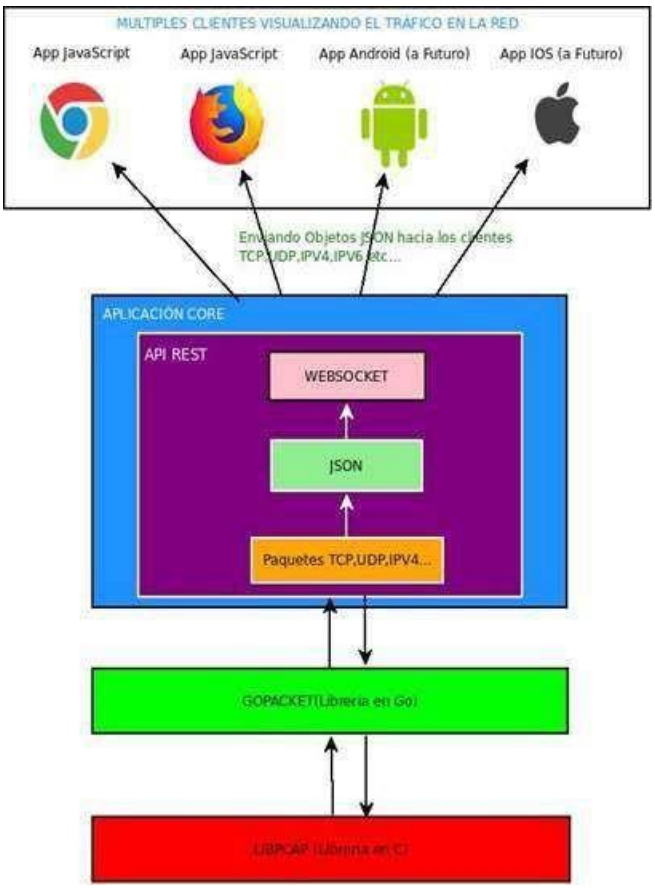

Gráfico 6. Arquitectura del Sistema
Fuente: Elaboración propia

Fase 3 Desarrollo
Para el desarrollo de esta aplicación primero es definir la estructura de los protocolos a analizar en este caso serían 5: TCP, UDP, IPV4, ETHERNET Para ellos se usa en golang una estructura por cada protocolo.

No. 20, 31 de octubre de 2018
ISSN impreso: 1390 - 6321

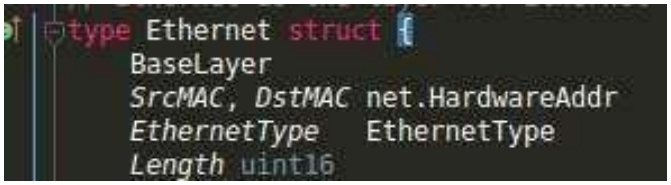

Gráfico 7. Estructura en golang de acuerdo a la trama Ethernet
Fuente: Elaboración propia

Gráfico 8. Estructura en golang de acuerdo al datagrama udp. TYPE 2.
Fuente: Elaboración propia

Gráfico 9. Estructura en golang de acuerdo al datagrama IPV4
Fuente: Elaboración propia

```
type TCP struct {
    BaseLayer
    SrcPort, DstPort                          TCPPort
    Seq                                       uint32
    Ack                                       uint32
    DataOffset                                uint8
    FIN, SYN, RST, PSH, ACK, URG, ECE, CWR, NS bool
    Window                                    uint16
    Checksum                                  uint16
    Urgent                                    uint16
    sPort, dPort                              []byte
    Options                                   []TCPOption
    Padding                                   []byte
    opts                                      [4]TCPOption
    tcpipchecksum
}
```

Gráfico 10. Estructura en golang de acuerdo al datagrama TCP
Fuente: Elaboración propia

Definidas las estructuras se puede empezar el escaneo de los datos, la librería gopacket proporciona métodos para realizar esta operación en la siguiente imagen se podrá observar el escaneo básico de la red de datos.

```
// Abrimos la lectura
handle, err = pcap.OpenLive(dispositivo, longitudCaptura, modoPromiscuo, tiempoSalida)
if err != nil { //En el caso de existir algun error mostrarlo
        return err
}
//Cerrar cuando se termine
defer handle.Close() //Luego de terminar de usar la función handle esta se cerrara

//Utiliza handle para procesar todos los paquetes
packetSource := gopacket.NewPacketSource(handle, handle.LinkType())
for packet := range packetSource.Packets() {
        SendPacket(packet, ws)
}
return nil
```

Gráfico 11. Iniciar Captura usando el método openLive
Fuente: Elaboración propia

Front-end
Para el desarrollo del cliente front-end se utilizó javascript para consumir el webSocket y poder obtener los paquetes de red que son enviados por el servidor.

Inicialización de la Aplicación

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
buffermint-Vm BufferNet # go run main.go  -d ens33
Dispositivo:  ens33
localhost:8000
```

Gráfico 12. Iniciar la Aplicación con el comando go run
Fuente: Elaboración propia

Esta captura indica la inicialización de aplicación la cual ya está corriendo en localhost a través del puerto 8000.

Front-Web Test de Comunicación

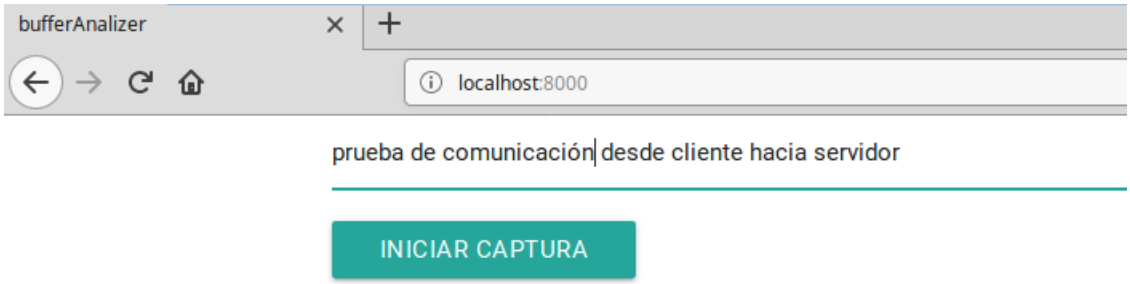

Gráfico 13. Cliente Web para la captura de Datos
Fuente: Elaboración propia

En esta captura se puede visualizar el cliente web que recibirá los datos proporcionado por el servidor, para ello se probara la comunicación entre cliente y servidor enviando este mensaje que dice "prueba de comunicación desde cliente hacia servidor" en el momento de presionar el botón "Iniciar Captura" el mensaje será enviado al servidor.

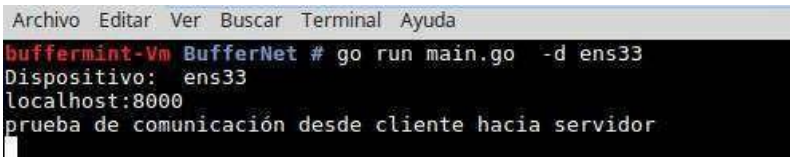

Gráfico 14. Recepción de mensaje de cliente
Fuente: Elaboración propia

Gráfico 15. Generación de Paquetes
Fuente: Elaboración propia

Ventana Modal Con Información del Paquete

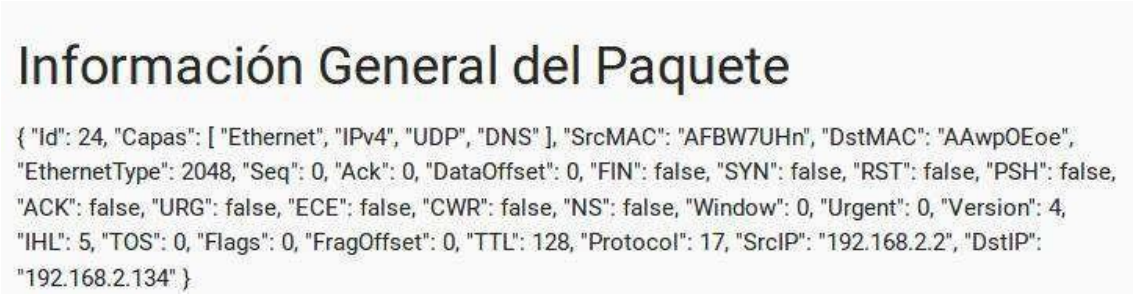

Gráfico 16. Ventana Modal Con Información del Paquete
Fuente: Elaboración propia

Basándose directamente en la arquitectura TCP/IP y los datagramas de cada protocolo se pudo obtener los campos o atributos de cada uno de ellos y mostrarlos a l usuario en tiempo real para su posterior uso.

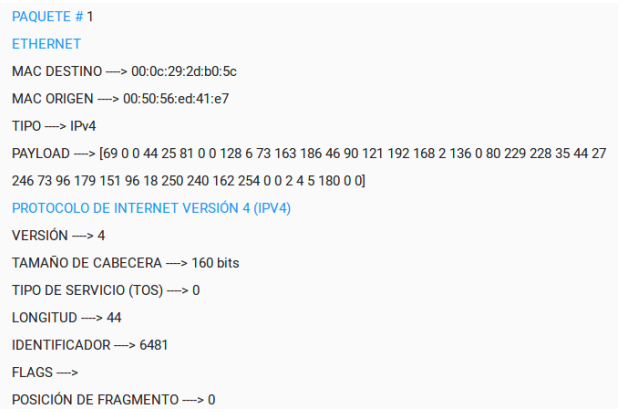

Gráfico 17. Ventana Modal con Información del Paquete de red capturado
Fuente: Elaboración propia

Conclusiones

La bibliografía utilizada permitió obtener los campos de los protocolos tcp, udp, ipv4, Ethernet, Http lo interesante de las Request for comments es el nivel de especificaciones que estas publicaciones tienen además de una explicación excelente de cada uno de los campos de los protocolos analizados existen ciertas analogías humanas descritas en las publicaciones RFC que ayuda el entendimiento de los protocolos de comunicación.

Una vez obtenidos los campos tanto nombre como tipo de datos, se realizó la implementación de las estructuras de los diferentes protocolos en el lenguaje de programación golang para luego ser enviados atreves del protocolo webSocket en formato json al navegador web.

Se realizó el desarrollo de un básico servidor webSocket utilizando los paquetes de gorrilla mux en el lenguaje de programación para él envió en tiempo real de los datos al navegador web esto permite que el usuario no actualizar el navegador web a cada instante para visualizar los datos sino que automáticamente los datos serán visualizados atreves de este protocolo full duplex.

Referencias bibliográficas

Day, J. D., & Zimmerman, H. (1983). The OSI Reference Model. In *Proceedings of IEEE*, Volume-71, No.12, p.1334-13340.

Day, J. (1995). The (Un) Revised OSI Reference Model. Recuperado de: http://ccr.sigcomm.org/archive/1995/oct95/ccr-9510-day.pdf

Deering, S. y Hinden, R. (1998). *Protocolo Internet*, Versión 6 (IPv6), RFC- 2460. Recuperado de: https://tools.ietf.org/html/rfc2460

Horning, A. (1984). *Standard for the Transmission of IP Datagrams over Ethernet Networks*, Cambridge, UK: Ed. Cambridge Press RFC- 894.

Postel, J. (1980). *Protocolo de datagramas de usuario*, RFC- 768. Recuperado de: https://www.rfc-es.org/rfc/rfc0791-es.txt

Postel, J. (1981) *Protocolo de mensajes de control internet*, RFC- 792. Recuperado de: https://www.rfc-es.org/rfc/rfc0792-es.txt

Socolofsky, T. y Kale, C. (1991). Un tutorial de TCP/IP, RFC-1180

Tanenbaum, A. y Wetherall, D. (2012). *Redes de Computadoras*, Quinta ed., USA: Pearson Ed.

University of Southern California (1981). *Protocolo De Control De Transmisión*, Marina del Rey, USA: Ed. USC RFC-793.