Protocolo HTTP 1.1

REDES

Calles Palacios, Osvaldo
UNIVERSIDAD AUTONOMA DE GUADALAJARA | WWW.UAG.MX

Fecha	Revisión	Autor	Documento
14/Agosto/2014	0.1	Osvaldo Calles Palacios	Creación de una plantilla como documento base con las correcciones de las observaciones del profesor.
18/Agosto/2014	0.5	Osvaldo Calles Palacios	Se agregó introducción al protocolo HTTP y funcionamiento a grandes rasgos.
19/Agosto/2014	0.7	Osvaldo Calles Palacios	Se agregó los dibujos de las capturas con Wireshark.
20/Agosto/2014	1.0	Osvaldo Calles Palacios	Se agregó los explicación de three-way-handshake, circuito virtual y la descripción del protocolo http.

Tabla de Contenido

Índice de Tablas

No table of figures entries found.

Índice de Figuras

PROTOCOLO HTTP 1.1

El objetivo de este documento es entender el funcionamiento del protocolo HTTP a través del todo el proceso de comunicación usando la herramienta Wireshark.

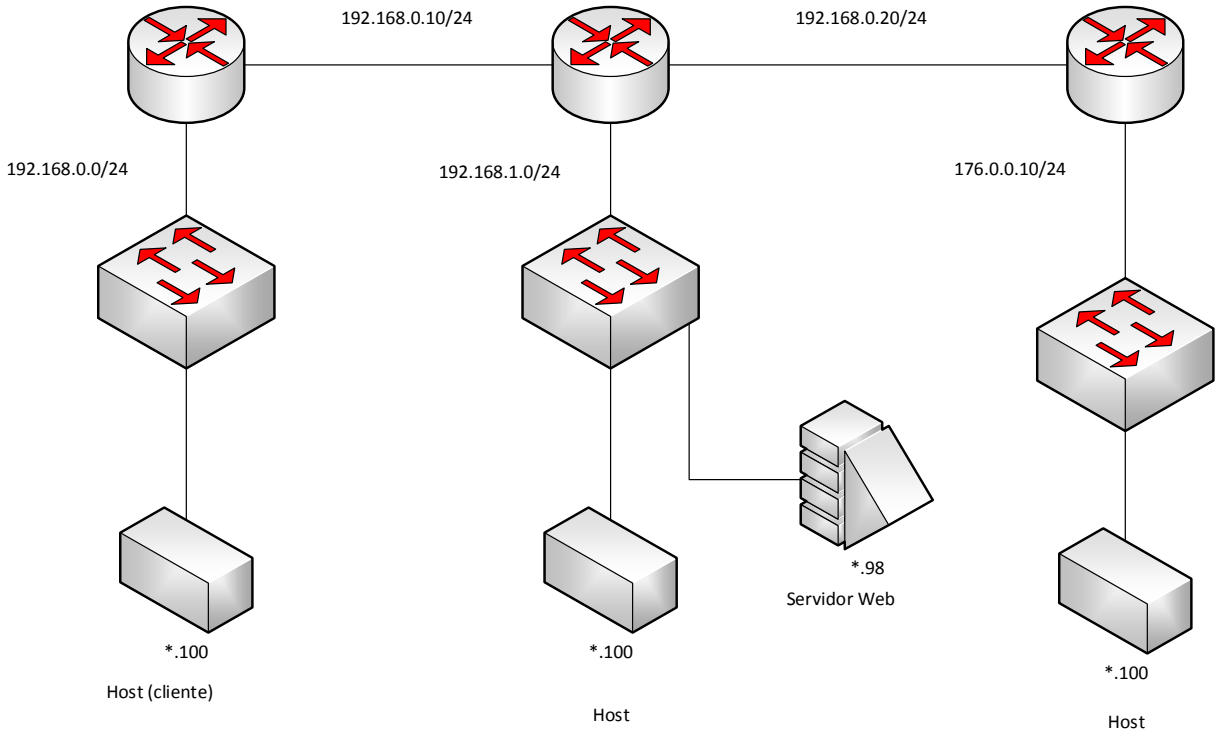


FIGURA 1 - TOPOLOGÍA DE LA RED PARA PROBAR EL PROTOCOLO HTTP

El **cliente** es una computadora de escritorio con sistema operativo Windows 8 que está representado en la figura 1 por el icono de Host con dirección **IP 192.168.0.100.**

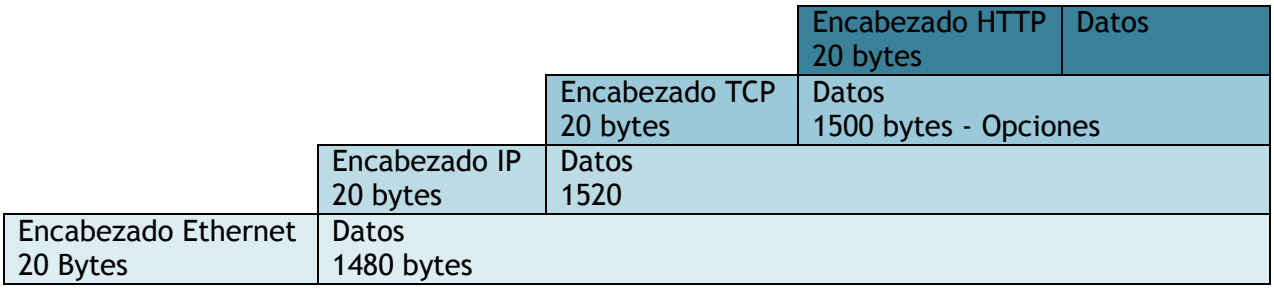
El **servidor web** es una laptop con sistema operativo Linux (Ubuntu) y como servidor web TomCat teniendo la dirección **IP 192.168.1.98**.

1 Introducción

HTTP (**Hypertext Transfer Protocol**) es un protocolo de solicitud y respuesta implementado en la capa de aplicación que usa semántica extendida y mensajes payloads para tener una flexibilidad en la interacción con las redes basadas en sistemas de información de hipertexto.

La especificación de HTTP/1.1 viene dado en los **rfc** (**Request For Comments**):

1. rfc7230 – describe el enrutamiento y sintaxis de los mensajes
2. rfc7231 – Contenido y semántica
3. rfc7232 – Solicitudes Condicionales
4. rfc7233 – Rango de las solicitudes
5. rfc7234 – Cache
6. rfc7235 – Autenticación.



1.1 Mensajes entre Cliente/Servidor

HTTP opera por intercambio de mensajes a través de un transporte confiable o capa de sesión "Conexión".

El término "**cliente**" y "**servidor**" se refiere únicamente a los roles que estos programas realizan para una conexión en particular. El mismo programa podría actuar como un cliente sobre la misma conexión y como servidor en otras. El término "**user agent**" hace referencia a cualquiera de los varios clientes que inicien la petición, incluyendo (pero no limitado a) navegadores web, línea de comandos, aplicaciones creadas y aplicaciones de celulares. El término "**origin server**" hace referencia al programa que puede originar respuestas para una fuente destino. El término "**sender**" y "**recipient**" hace referencia a la implementación que envía o recibe un mensaje dado, respectivamente.

HTTP se basa sobre el **Uniform Resource Identifier** (URI) descrito en el rfc3986 para indicar el destino final y la relación entre los recursos. Los mensajes son pasados in un formato similar al usado por Internet mail (rfc5322).

Mucha de la comunicación consiste de la solicitud de petición (GET) dado por una representación URI. In el caso mas simple, esto se logra a través de una conexión bidireccional entre el "user agent" (UA) y el "origin server" (O).

Petición >

UA (cliente) ============================== O (servidor)

< Respuesta

Un cliente envía una solicitud HTTP a un servidor in la forma de un mensaje de solicitud, empezando por una línea de petición que incluye un método, URI y visión de protocolo, seguido por los campos del encabezado que contiene modificadores, información del cliente, y metadatos, una línea vacía indica el final de la sección del encabezado y finalmente el cuerpo que contiene el payload.

El servidor responde a la solicitud del cliente enviando uno o más mensajes de respuesta HTTP, cada una iniciando con una línea de estado que incluye la versión del protocolo, un código de éxito o error, una frase textual, posiblemente seguido por los campos del encabezado que contiene información del servidor, metadatos, y una línea vacía para indicar el final de la sección del encabezado, y finalmente el mensaje del cuerpo que contiene el payload.

Una conexión puede ser usada para múltiples intercambios de solicitudes/respuestas.

Un ejemplo de un típico intercambio de mensajes para http://www.ejemplo.com/hola.txt para una solicitud GET.

Solicitud del Cliente:

```
GET /hola.txt HTTP/1.1
User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.8.71 zlib/1.2.3
Host: www.ejemplo.com
```

```
Accept-Language: en, mi
```

Respuesta del Servidor:

```
HTTP/1.1 200 OK
Date: Mon, 20 Aug 2014 12:00:00 GMT
Server: Apache
Last-Modified: Wed, 18 Aug 2014 09:23:59 GMT
Etag: “34aa387-d-1568eb00”
Accepted-Length: 51
Vary: Accept-Enconding
Content-Type: text/plain

Hola Mundo!! Mi payload incluye a CRLF.
```

2 Conexión TCP

El servicio confiable que TCP ofrece provee a las aplicaciones cinco propiedades:

- Envio de datos orientado (Stream Orientation)
- Conexión con circuitos virtuales
- Transferencia buffereada
- Comunicación Full Duplex
- Transferencia sin estructura

Antes de una comunicación inicie, ambos aplicaciones, el que envía y recibe deben ponerse de acuerdo para establecer una conexión TCP.

Un circuito virtual está formado por el par IP maquina 1 : Puerto y IP maquina 2 : Puerto. Dado que el circuito no depende únicamente de la IP, podemos formar varios circuitos virtuales usando el mismo puerto, lo cual nos permite tener varias conexiones en un servidor web hacia el mismo puerto 80.

2.1 Estableciendo la Conexión

Para establecer una conexión, TCP usa un three-way-handshake. Eso es, tres mensajes son intercambiados para permitir cliente-servidor llegar a un acuerdo para formar una conexión y saber qué es lo que del otro lado está de acuerdo. El primera parte del handshake puede ser identificado porque tiene el bit SYN del campo código. El segundo mensaje tiene ambos bits SYN y ACK puestos en uno para indicar que es reconocido el primer SYN y continúa el handshake. El último mensaje del handshake es solamente un reconocimiento ACK y es meramente para informar al destino que ambos lados están de acuerdo que la conexión se ha establecido.

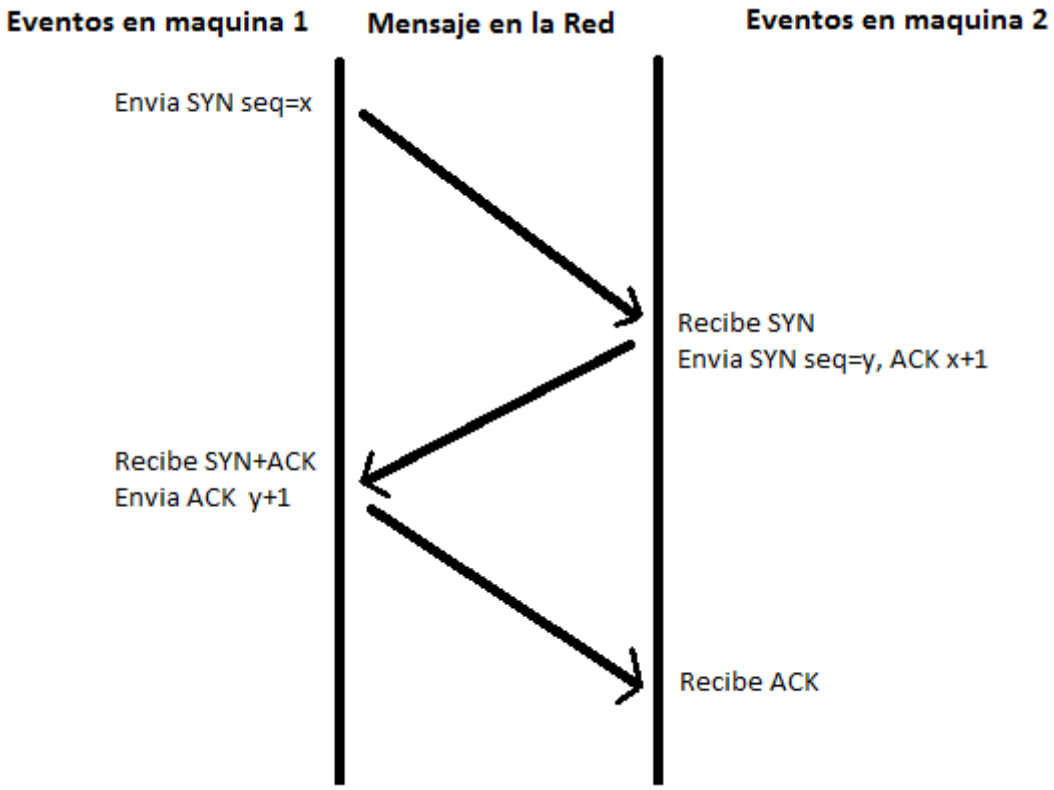


FIGURA 2 - THREE-WAY-HANDSHAKE

Usualmente el software TCP en una maquina espera pasivamente por el handshake, y el software TCP de la otra máquina la inicia. Sin embargo, el handshake está diseñado para funcionar incluso si ambas maquinas inician la conexión simultáneamente.

Una vez que la conexión se ha establecido, los datos pueden fluir en ambas direcciones igualmente. No hay maestro, no hay esclavo y el lado que inicia la conexión no tiene ningún privilegio especial.

2.2 Cerrando la Conexión

Para cerrar la conexión de dos máquinas que se comunican usando TCP y terminar de manera exitosa, es importante que las dos partes estén de acuerdo a cerrar la conexión y finalmente ambos saben que la conexión se cerró. Para entender el handshake usado para cerrar la conexión, hay que recordar que las conexiones TCP son full-duplex y se maneja como si fueran dos fuentes independientes de transferencia, cada una llendo en cada dirección. Cuando la aplicación le dice a TCP que no hay mas datos para enviar, TCP cerrar la conexión in una dirección. Para cerrar su mitad de la conexión, una maquina termina de enviar lo datos que faltan y espera a que el receptor los reconozca y en via una paquete con el bit FIN. Una

vez que se recibe FIN, TCP envía un reconocimiento e informa a la otra parte que se ha finalizado la transmisión de los datos.

Normalmente una aplicación utiliza la operación “close” para apagar una conexión cuando finaliza la transmisión de datos. Cerrar la conexión se considera parte del uso normal, decimos que la conexión termino exitosamente. Sin embargo, algunas veces condiciones anormales pueden forzar una aplicación o el programa de la red a romper la conexión sin un apagado exitoso. TCP provee un reset para facilitar y manejar desconexiones anormales. Cuando un reset ocurre, TCP informa a cualquier aplicación local que estaba usando la conexión.

2.3 Laboratorio - Estableciendo la Conexión

Se muestra a la maquina 1 cliente (**192.168.0.100)** enviando la paquete 1 para de establecer una conexion con la maquina 2 servidor (**192.168.1.98**).

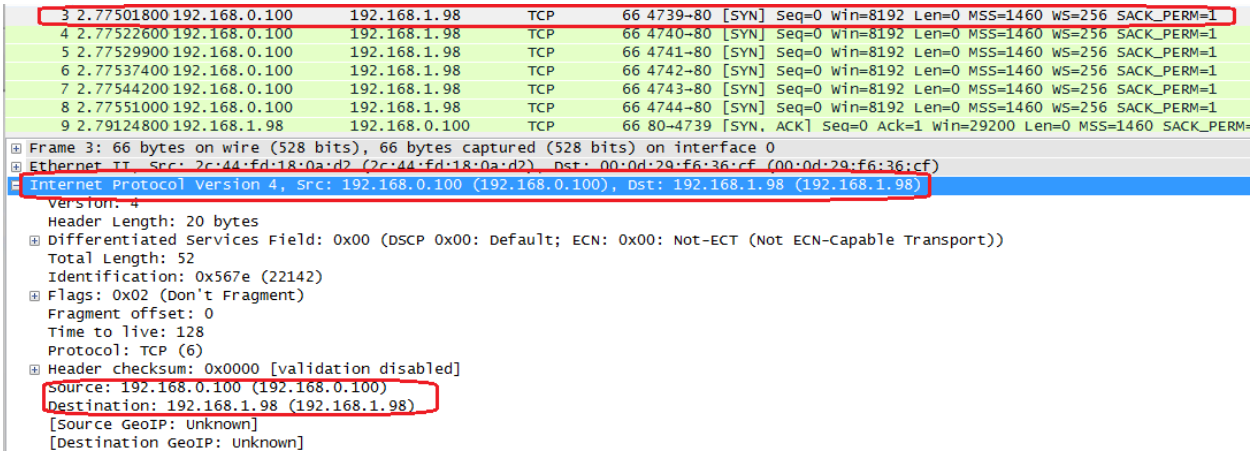


FIGURA 3 - PROTOCOLO TCP MOSTRANDO LA DIRECCIÓN IP FUENTE Y DESTINO

En la paquete 1, el cliente con puerto **4739** envía **SYNC** para iniciar una conexión al servidor en el puerto **80**. Se muestra que “sequence number” = 0 (seq = x = 0).

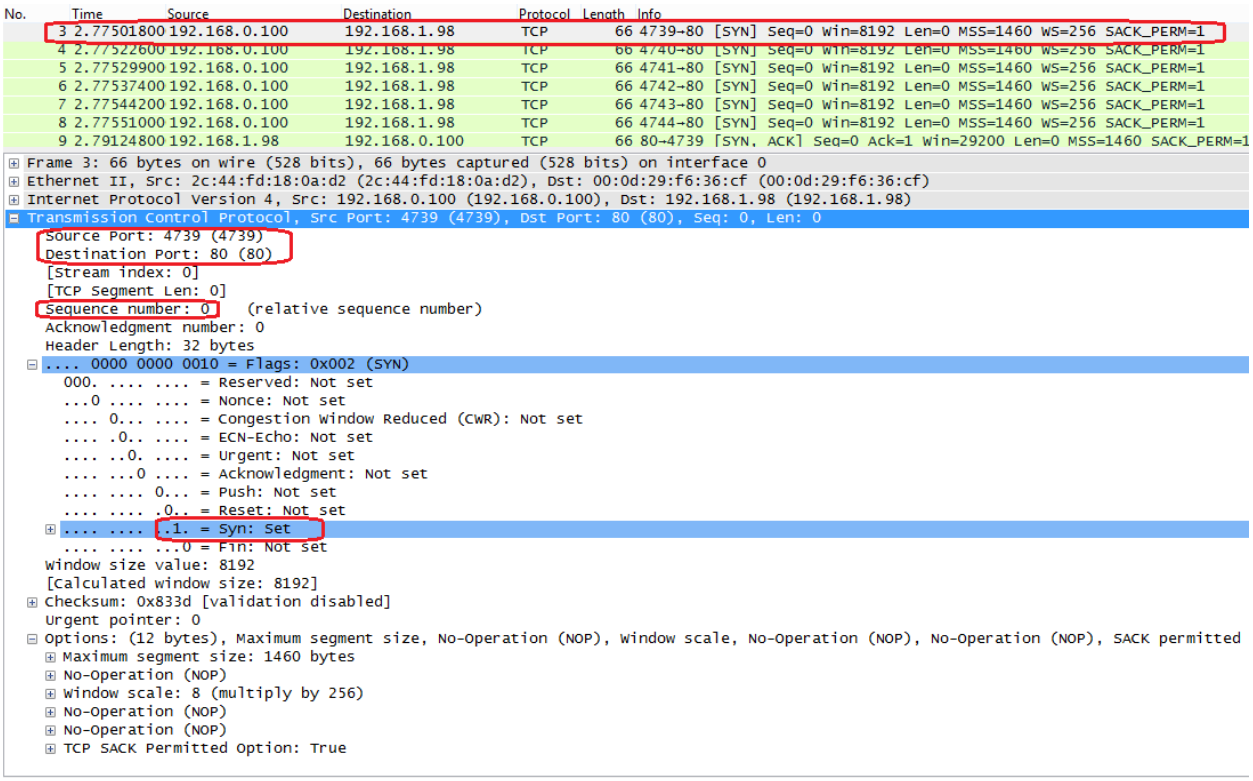


FIGURA 4 - PROTOCOLO TCP MOSTRANDO EL INICIO LA PETICIÓN DEL CIRCUITO VIRTUAL

Aquí se muestra más paquetes que fueron son enviados para establecer otros circuitos virtuales.

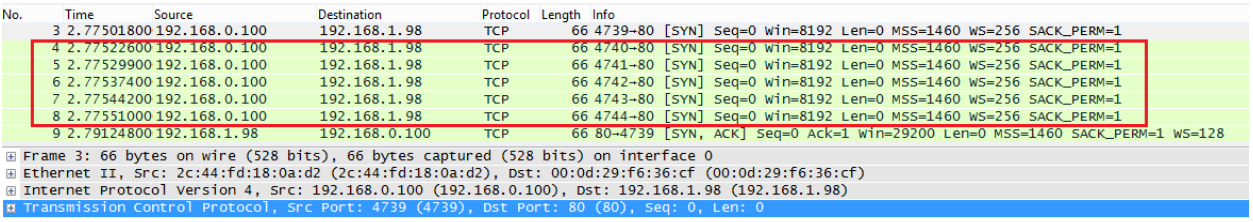


FIGURA 5 - PROTOCOLO TCP MOSTRANDO SOLICITANDO MÁS CIRCUITOS VIRTUALES

2.4 Laboratorio - Reconocimiento (ACK)

La máquina 2 servidor (192.168.1.98) responde con la paquete 2 que tiene envía el bit ACK y SYNC. Se muestra que “sequence number” = 0 (seq = y = 0) y “Acknowledgment number” = 1 (ACK x+1 = 0+1 = 1).

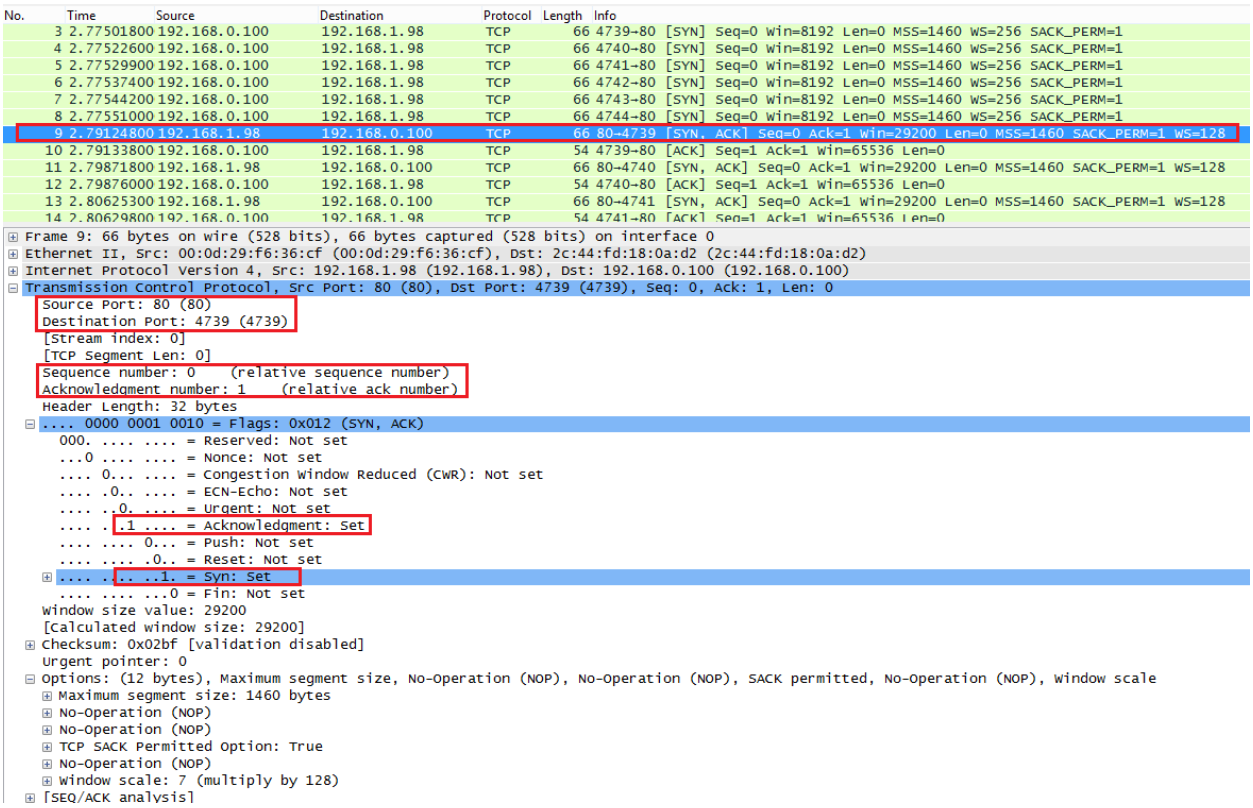


FIGURA 6 - PROTOCOLO TCP MOSTRANDO EL ACK POR PARTE DEL SERVIDOR

2.5 Laboratorio -Creación del Circuito Virtual

La máquina 1 cliente (192.168.0.100), finalmente envía la paquete 3 con el bit ACK y se observa “sequence number” = 1 (ACK y+1 = 0+1 = 1).

Hasta este punto se crea el circuito virtual **192.168.0.100:4739 - 192.168.1.98:80**

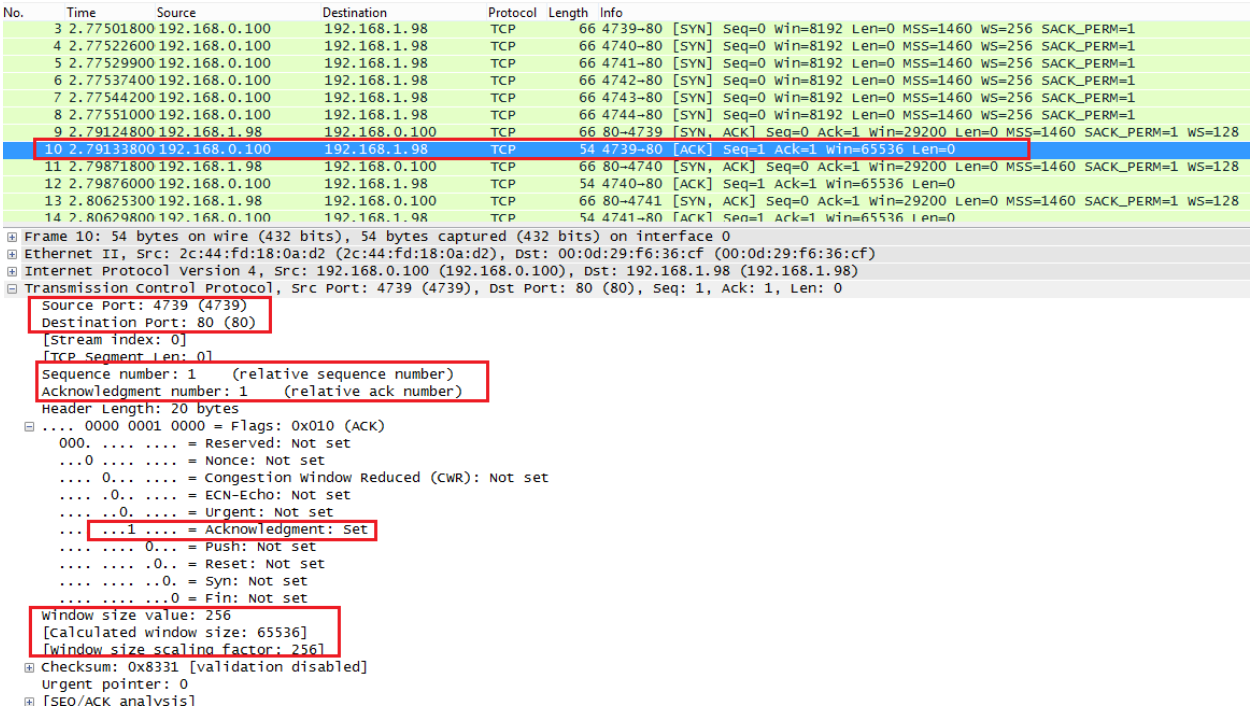


FIGURA 7 - PROTOCOLO TCP MOSTRANDO EL ACK FINAL

Se puede observar que se están creando más circuitos virtuales entre el mismo cliente y el mismo servidor utilizando el mismo puerto 80.

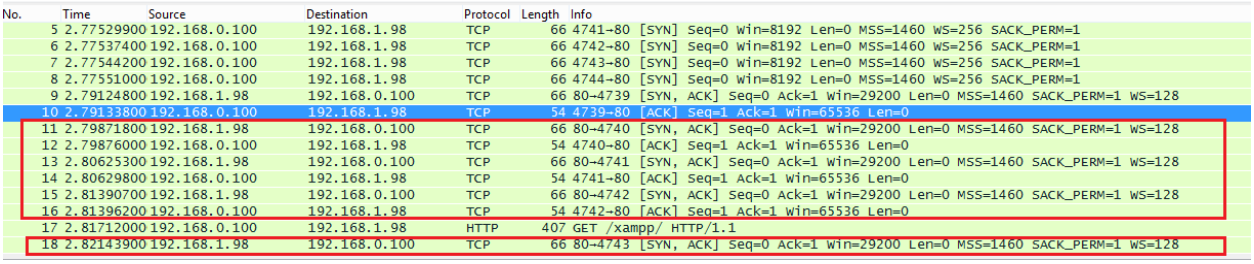


FIGURA 8 - PROTOCOLO TCP MOSTRANDO LA CREACIÓN DE OTROS CIRCUITOS VIRTUALES

3 Acceso al Contenido de las Páginas web con HTTP

3.1 Parámetros del Protocolo

- HTTP Version
  Usa un esquema de numeracion <major><minor> para indicar la versión de protocolo. El <minor> es incrementado cuando se hacen cambios al protocolo para agregar más propiedades. El <major> es incrementado cuando el formato de un mensaje en el protocolo es cambiado.

  Ejemplo: HTTP/1.1

- Uniform Resource Identifiers
  Es conocido por varios nombres tales como direcciones WWW, Identificadores De Documento Universales, Uniform Resource Locator (URL)
  Sintaxis universal de un http URL es
  http_URL = “http:” “//” host [“:” puerto] [ruta_absoluta [“?” query]]

  Ejemplo: http://192.168.1.98/xamp/

- Date/Time Formats
  Aplicaciones HTTP utilizan tres tipos de formato para reprsentar fecha/hora. El preferido es el estándar de Internet definido en el rfc1123
  HTTP_date = rfc1123-date | rfc850-date | asctime-date

  Ejemplo: Sat, 09 Aug 2014 18:13:49 GMT\r\n

- Character Sets
  Es usado para hacer referencia al método usado con uno o más tablas a convertir de octetos a caracteres.

- Content Codings
  Indican el tipo de codificación de la información que puede ser o puede ser aplicado a una entidad. Es usado principalmente para a un documento ser comprimido o ser transformado sin perder su identidad del tipo sin perder información.

  Ejemplo: content-codign = gzip

- Transfer Codings
  Son usados para indicar la codificación de la información que ha sido usado, puede, o necesita aplicarse a un cuerpo de la entidad para asegurarse una transformación

segura a través de la red. Difiere del “content coding” en que el “transfer coding” in una propiedad del mensaje, no de la entidad original.

Ejemplo: transfer-coding = “chunked”

- Media Types
  HTTP usa tipos de media de internet en los parámetros del encabezado “content-type” y “accept” para proveer tipos de datos abiertos y extensibles y tipos de negociación.

  Ejemplo: Cliente -> Accept = text/html. Application/xhtml+xml, application/xml;q=0.9, image/webp, */*;q=0.8\r\n

  Servidor -> content-type = text/html \r\n

- Product Tokens
  Son usados para permitir comunicar aplicaciones para identificar a ellos mismos por nombre de software y versión.
  Sintaxis es
  Product = token [“/” product-version]
  Product-version = token

  Ejemplo: User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebkit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 safari/537.36\r\n

- Quality Values
  Negociacion del contenido HTTP, usa números de punto flotante corto para indicar la importancia relativa (“weights”) de varios parámetros negociables.
  Sintaxis:
  Qvalue = (“0” [“.” 0*3DIGIT])
  | (“1” [“.” 0*3(“0”)])

- Language Tags
  Identifica el lenguaje natural hablado, escrito y de alguna manera por conveniencia para comunicar información a otras personas.

  Ejemplo: Accept-Language = es-Es, es; q=0.8,en;q=0.6\r\n
  Content-Language =

- Entity Tags
  Son usados para comparar dos o mas entidades provenientes de la misma solicitud. HTTP utiliza etiquetas en el Etag, If-Match, If-None-Match y If-Range del encabezado. Est información es comparada como validador de la cache.

- Range Units
  Permite a los cliente solicitar partes (un rango de) del contenido de la respuesta.
  Sintaxis
  Range-unit = bytes-unit | other-range-unit
  Bytes-unit = “bytes”
  Other-range-units= token

3.2 Método GET

El método GET solicita transferencia de la presentación del actual recurso del destino. GET es mecanismo primario de adquisición de información y el foco de casi todas las optimizaciones de rendimiento.

Se podría imaginar a los identificadores de los recursos como rutas de un sistema de archivos remoto y de representación como una copia del contenido de tales archivos. De hecho así es como se implementan varios recursos.

Un cliente puede alterar la semántica de GET para ser un rango solicitado, de tal manera que solo se solicita que se transfiera algunas partes de la representación seleccionada. Esto se modifica a través del parámetro “range” del encabezado en la solicitud.

3.3 Códigos de Estados

Después de recibir e interpretar una solicitud por parte del cliente, un servidor responde con un mensaje HTTP el cual contiene un Código de estado y una frase.

“Status-Code” es un elemento de 3 dígitos que intenta entender y satisfacer una solicitud. La “Reason-Phrase” es una intención de describir textualmente es “Status-Code”.

Estos códigos están definidos en el rfc7231.

- 1xx: Informativo – Solicitud recibida, continua proceso
- 2xx: Éxito – La acción fue recibida exitosamente, entendida y aceptada
- 3xx: Redirección – Se necesita tomar una acción para completar la solicitud
- 4xx: Error en el Cliente – La solicitud contiene mala sintaxis y no puede ser completada
- 4xx: Error en el Servidor- El servidor fallo para completar una aparente solicitud valida.

3.4 Laboratorio - Petición del Cliente (GET)

La máquina 1 cliente (192.168.0.100) hace una solicitud de presentación mediante el método GET. Se puede observar:

sequence number = 1

Next sequence number = 354

Acknoledgment number = 1

Window Size = 256

Calculated Window Size = 65536 <- 256*256 debido a la escala

Windows Size Scaling Factor = 256

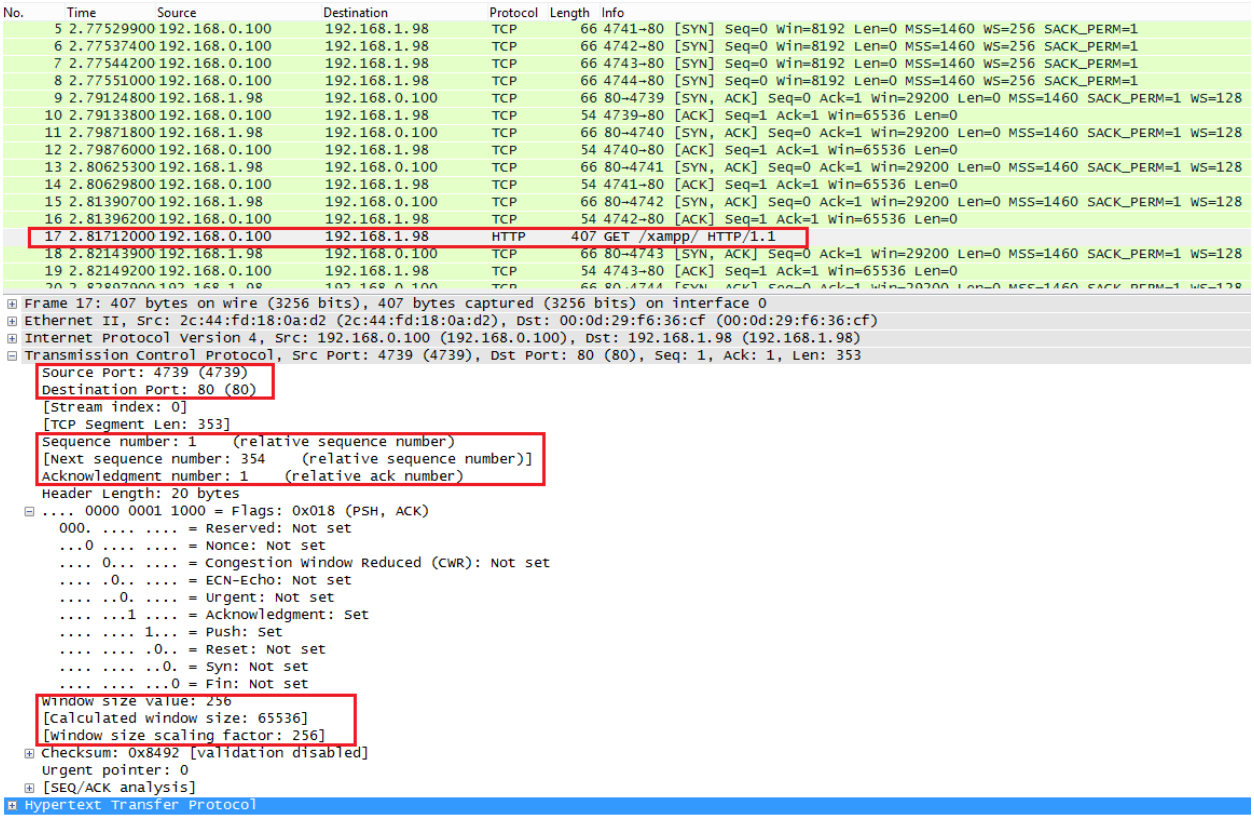


FIGURA 9 - PROTOCOLO HTTP MOSTRANDO LA SOLICITUD DE UNA PÁGINA WEB

Se puede observar que se hace una solicitud GET al host con IP 192.168.1.68 y el recurso /xamp/ usando la versión HTTP/1.1. hacia el puerto 80.

Nos muestra los tipos de datos que puede recibir el cliente y también las versiones del navegador web y tecnologías soportadas por el cliente.

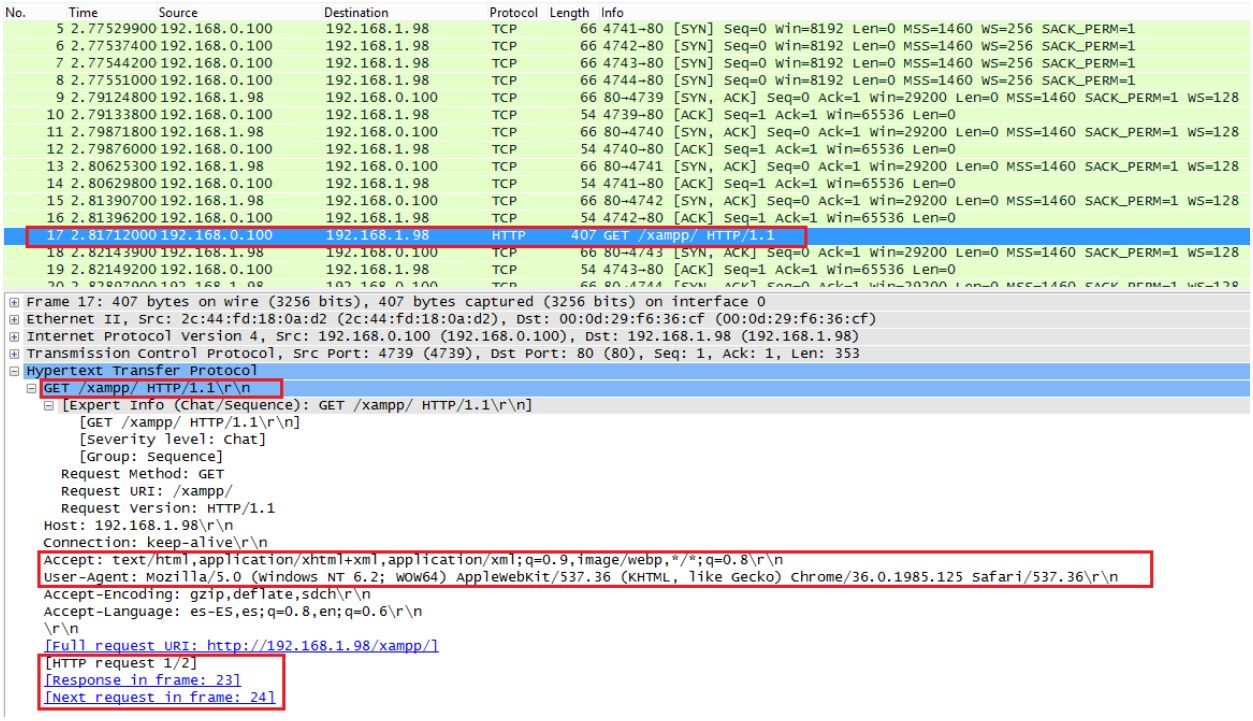


FIGURA 10 - PROTOCOLO HTTP MOSTRANDO LOS PARÁMETROS

3.5 Laboratorio - Respuesta del Servidor

La máquina 2 Servidor (192.168.1.98) responde hacia el puerto 4739 como destino.

Sequence number = 1

Next Sequence number – 862

Acknoledgment number = 354

Calculated Window Size = 30336

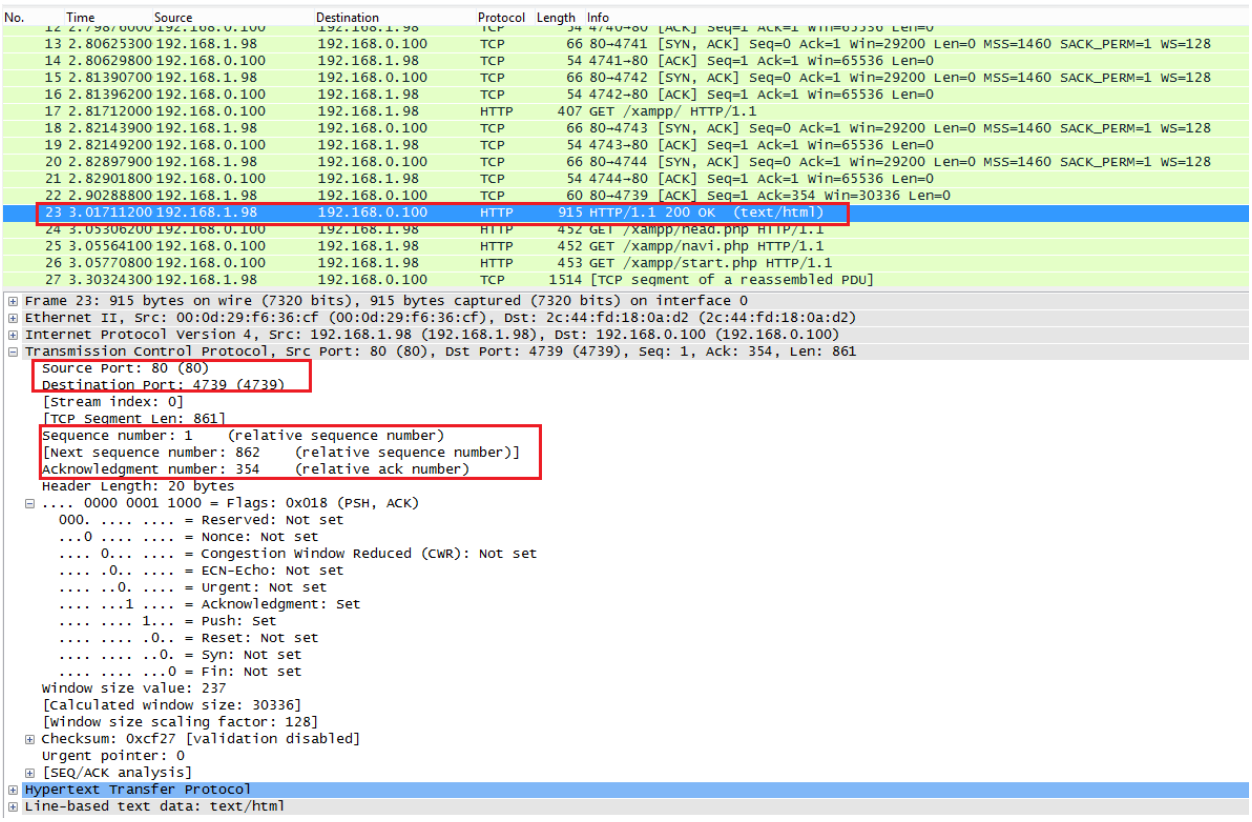


FIGURA 11 - PROTOCOLO HTTP MOSTRANDO LA RESPUESTA DEL SERVIDOR

El estado de la respuesta está dado con el código 200 (OK) y la indica que la solicitud fue exitosa y el contenido es de tipo texto html.

También se puede observar los datos del servidor HTTMP tal como el programa y versión.

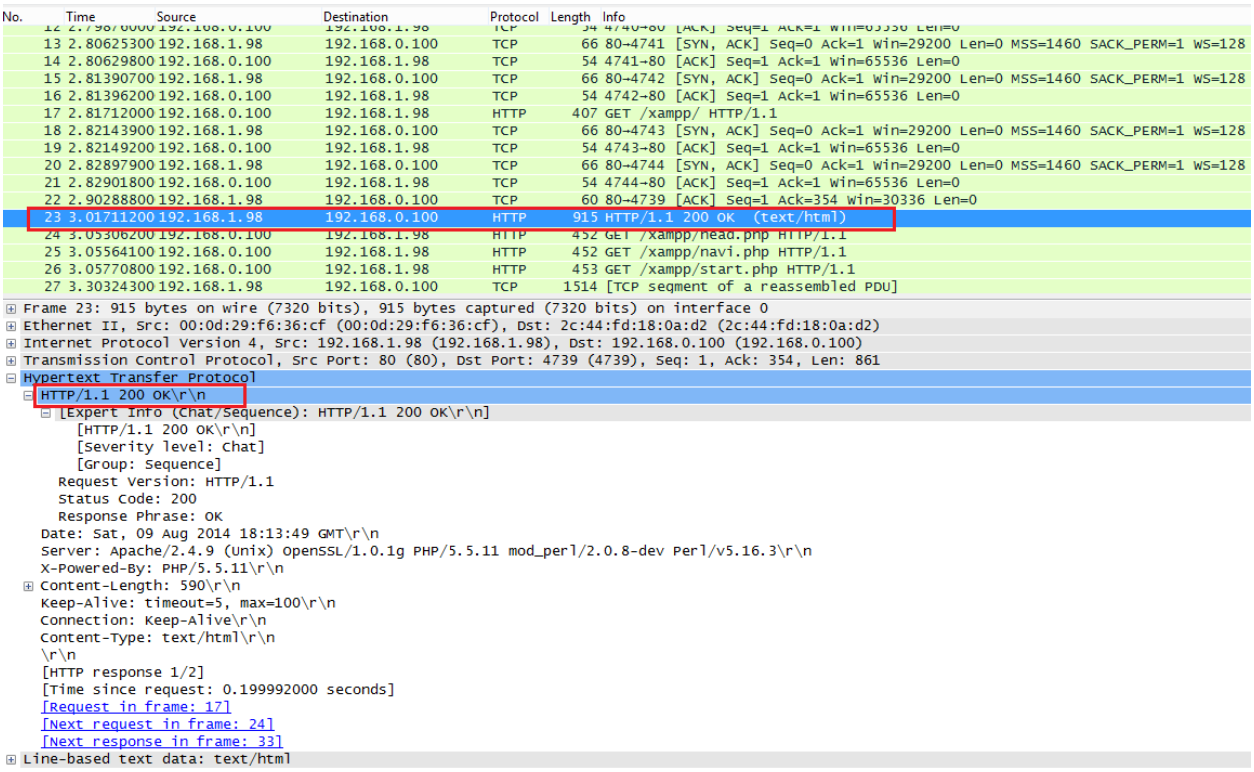


FIGURA 12 - PROTOCOLO HTTP MOSTRANDO EL ESTADO DE LA RESPUESTA DEL SERVIDOR

Finalmente en el mismo paquete se muestra los datos en texto plano que se solicitaron del servidor.

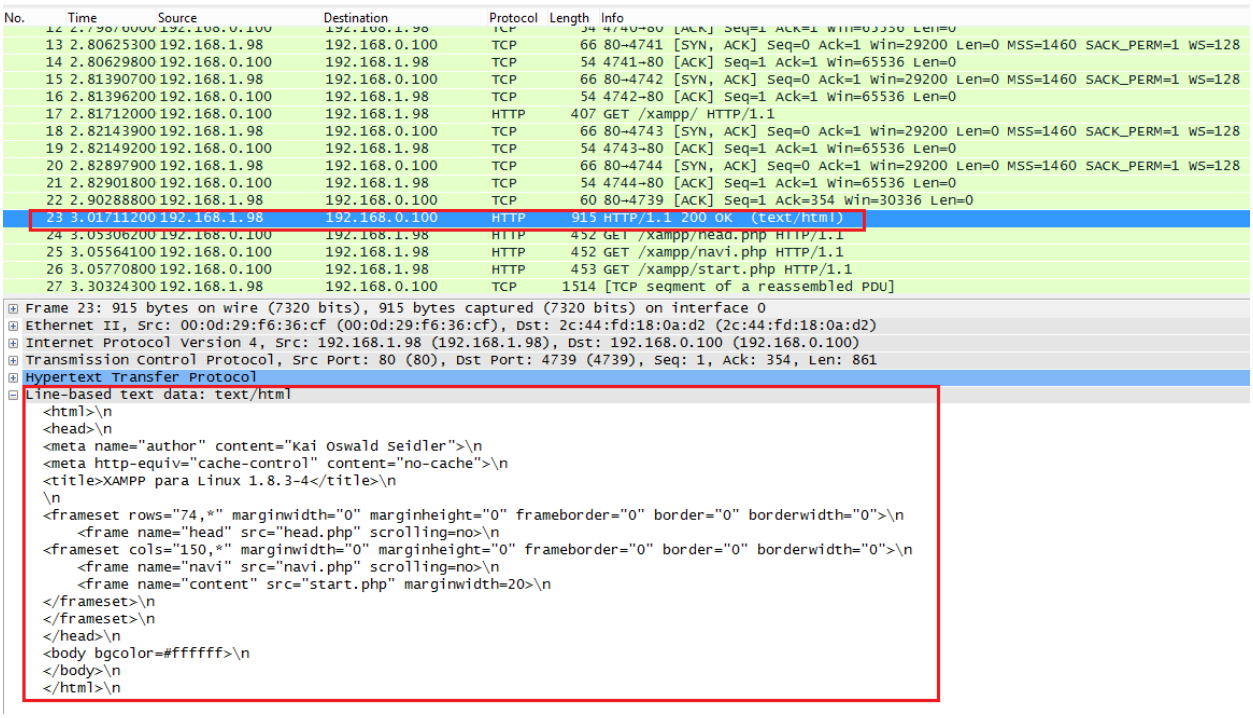


FIGURA 13 - PROTOCOLO HTTP MOSTRANDO LOS DATOS ENVIADOS POR EL SERVIDOR

Otras peticiones desde los otros circuitos virtuales

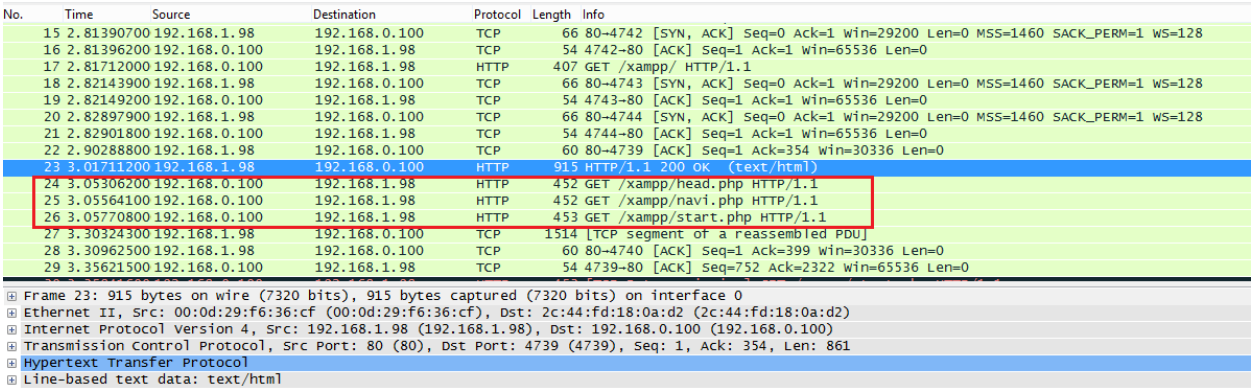


FIGURA 14 - PROTOCOLO HTTP MOSTRANDO OTRAS PETICIONES AL SERVIDOR

TCP mostrando un “Segment of a reassembled PDU”, investigar más sobre el tema

http://serverfault.com/questions/516401/why-does-wireshark-think-this-frame-is-a-tcp-segment-of-a-reassembled-pdu

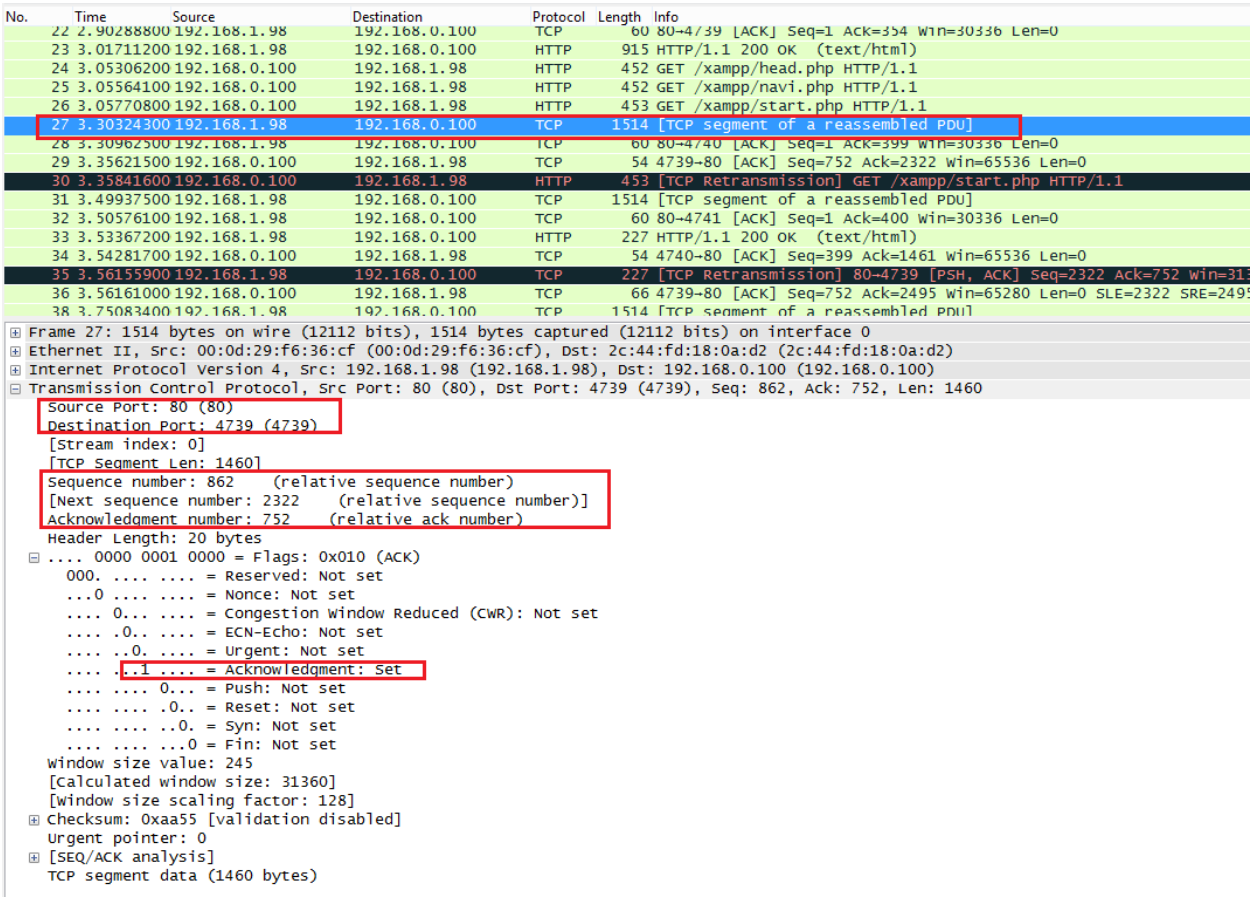


FIGURE 15 - PROTOCOLO HTTP MOSTRANDO UN REASSEMBLED PDU

4 Cerrar Conexión TCP

4.1 Laboratorio – Servidor finaliza transmisión de datos

La máquina 2 Servidor (192.168.1.98) termina de enviar los últimos datos hacia la maquina 2 cliente (192.168.0.100) y envía un el paquete con el bit FIN

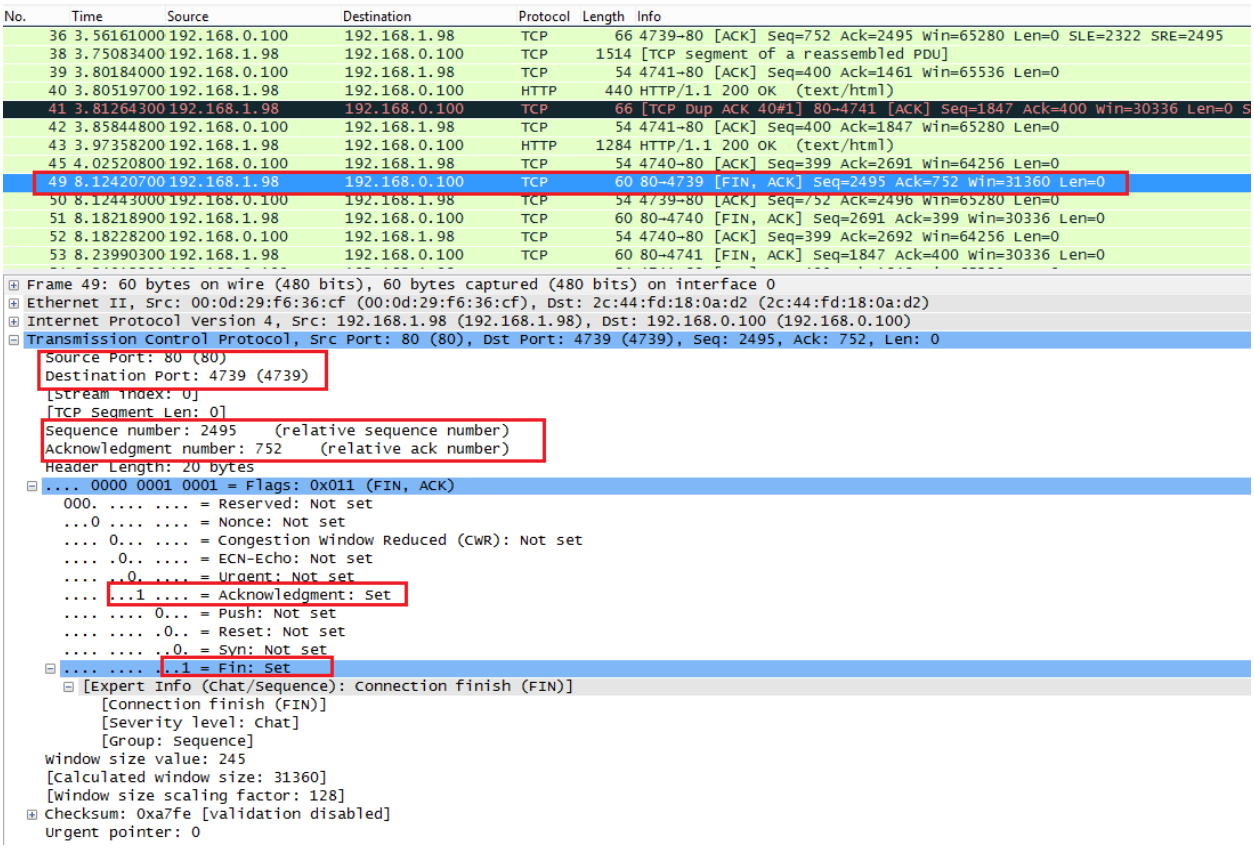


FIGURA 16 - ETIQUETA TCP MOSTRANDO EL CIERRE DEL CIRCUITO VIRTUAL

4.2 Laboratorio – Destrucción del Circuito Virtual

La máquina 1 Cliente (192.168.0.100) hace un reconocimiento de la solicitud del servidor para terminar la conexión y envía un paquete con ACK

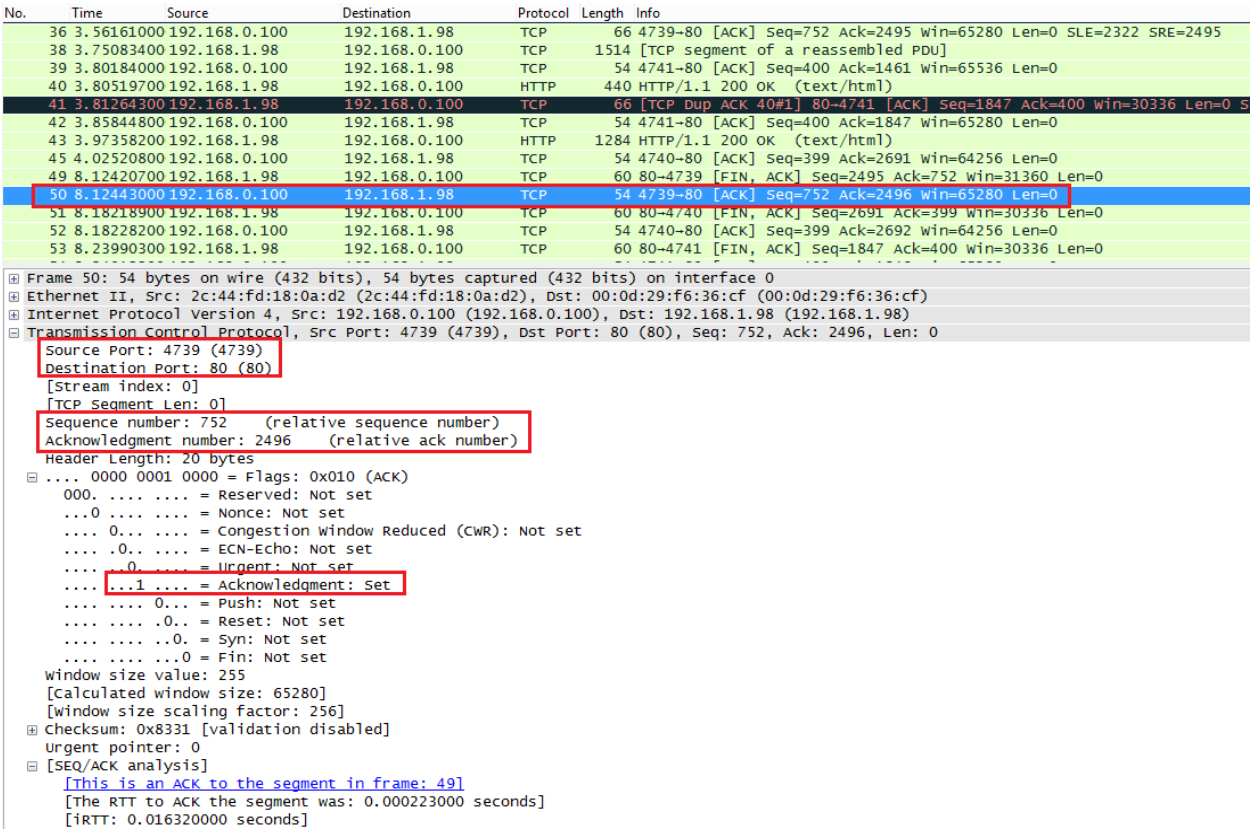
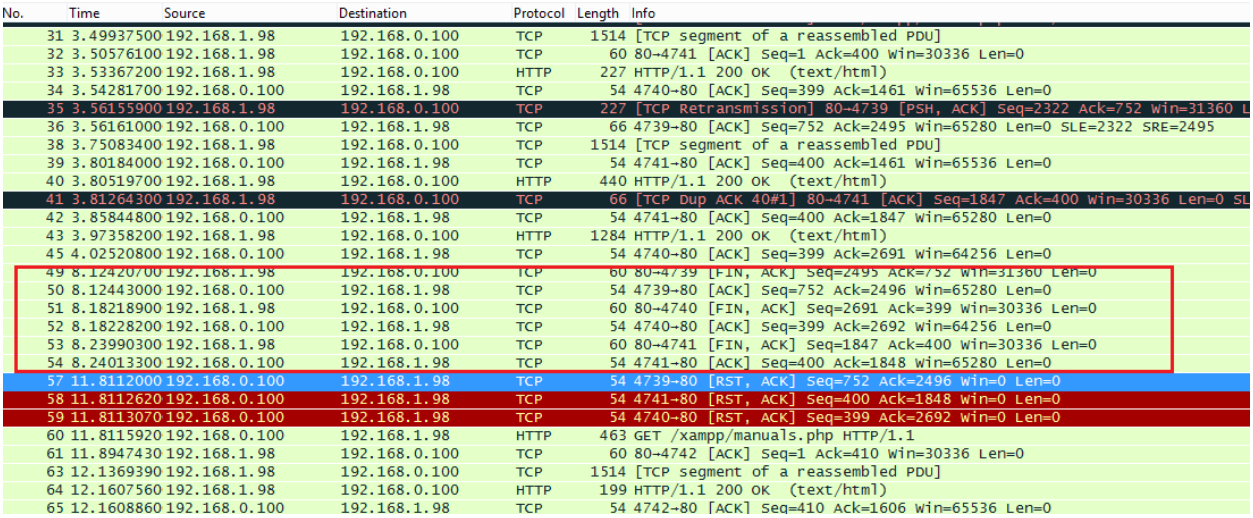


FIGURA 167 - PROTOCOLO TCP MOSTRANDO LA FINALIZACIÓN DEL CIRCUITO VIRTUAL

Se puede observar que genero una retransmisión de la maquina 2 servidor (192.168.1.98) a la maquina 1 cliente (192.168.0.100) debido a que se traslapo una sección nueva con una vieja en el área de datos.

También se puede observar otros paquetes entre el cliente y servidor para finalizar los otros circuitos virtuales.



No.	Time	Source	Destination	Protocol	Length	Info
31	3.49937500	192.168.1.98	192.168.0.100	TCP	1514	[TCP segment of a reassembled PDU]
32	3.50576100	192.168.1.98	192.168.0.100	TCP	60	80→4741 [ACK] Seq=1 Ack=400 Win=30336 Len=0
33	3.53367200	192.168.1.98	192.168.0.100	HTTP	227	HTTP/1.1 200 OK (text/html)
34	3.54281700	192.168.0.100	192.168.1.98	TCP	54	4740→80 [ACK] Seq=399 Ack=1461 Win=65536 Len=0
35	3.56155900	192.168.1.98	192.168.0.100	TCP	227	[TCP Retransmission] 80→4739 [PSH, ACK] Seq=2322 Ack=752 Win=31360 L
36	3.56161000	192.168.0.100	192.168.1.98	TCP	66	4739→80 [ACK] Seq=752 Ack=2495 Win=65280 Len=0 SLE=2322 SRE=2495
38	3.75083400	192.168.1.98	192.168.0.100	TCP	1514	[TCP segment of a reassembled PDU]
39	3.80184000	192.168.0.100	192.168.1.98	TCP	54	4741→80 [ACK] Seq=400 Ack=1461 Win=65536 Len=0
40	3.80519700	192.168.1.98	192.168.0.100	HTTP	440	HTTP/1.1 200 OK (text/html)
41	3.81264300	192.168.1.98	192.168.0.100	TCP	66	[TCP Dup ACK 40#1] 80→4741 [ACK] Seq=1847 Ack=400 Win=30336 Len=0 SL
42	3.85844800	192.168.0.100	192.168.1.98	TCP	54	4741→80 [ACK] Seq=400 Ack=1847 Win=65280 Len=0
43	3.97358200	192.168.1.98	192.168.0.100	HTTP	1284	HTTP/1.1 200 OK (text/html)
45	4.02520800	192.168.0.100	192.168.1.98	TCP	54	4740→80 [ACK] Seq=399 Ack=2691 Win=64256 Len=0
49	8.12420700	192.168.1.98	192.168.0.100	TCP	60	80→4739 [FIN, ACK] Seq=2495 Ack=752 Win=31360 Len=0
50	8.12443000	192.168.0.100	192.168.1.98	TCP	54	4739→80 [ACK] Seq=752 Ack=2496 Win=65280 Len=0
51	8.18218900	192.168.1.98	192.168.0.100	TCP	60	80→4740 [FIN, ACK] Seq=2691 Ack=399 Win=30336 Len=0
52	8.18228200	192.168.0.100	192.168.1.98	TCP	54	4740→80 [ACK] Seq=399 Ack=2692 Win=64256 Len=0
53	8.23990300	192.168.1.98	192.168.0.100	TCP	60	80→4741 [FIN, ACK] Seq=1847 Ack=400 Win=30336 Len=0
54	8.24013300	192.168.0.100	192.168.1.98	TCP	54	4741→80 [ACK] Seq=400 Ack=1848 Win=65280 Len=0
57	11.8112000	192.168.0.100	192.168.1.98	TCP	54	4739→80 [RST, ACK] Seq=752 Ack=2496 Win=0 Len=0
58	11.8112620	192.168.0.100	192.168.1.98	TCP	54	4741→80 [RST, ACK] Seq=400 Ack=1848 Win=0 Len=0
59	11.8113070	192.168.0.100	192.168.1.98	TCP	54	4740→80 [RST, ACK] Seq=399 Ack=2692 Win=0 Len=0
60	11.8115920	192.168.0.100	192.168.1.98	HTTP	463	GET /xampp/manuals.php HTTP/1.1
61	11.8947430	192.168.1.98	192.168.0.100	TCP	60	80→4742 [ACK] Seq=1 Ack=410 Win=30336 Len=0
63	12.1369390	192.168.1.98	192.168.0.100	TCP	1514	[TCP segment of a reassembled PDU]
64	12.1607560	192.168.1.98	192.168.0.100	HTTP	199	HTTP/1.1 200 OK (text/html)
65	12.1608860	192.168.0.100	192.168.1.98	TCP	54	4742→80 [ACK] Seq=410 Ack=1606 Win=65536 Len=0

FIGURA 18 - PROTOCOLO TCP MOSTRANDO LA FINALIZACIÓN DE LOS OTROS CIRCUITOS VIRTUALES

Se envió una retransmisión debido a un traslape en la sección de datos.

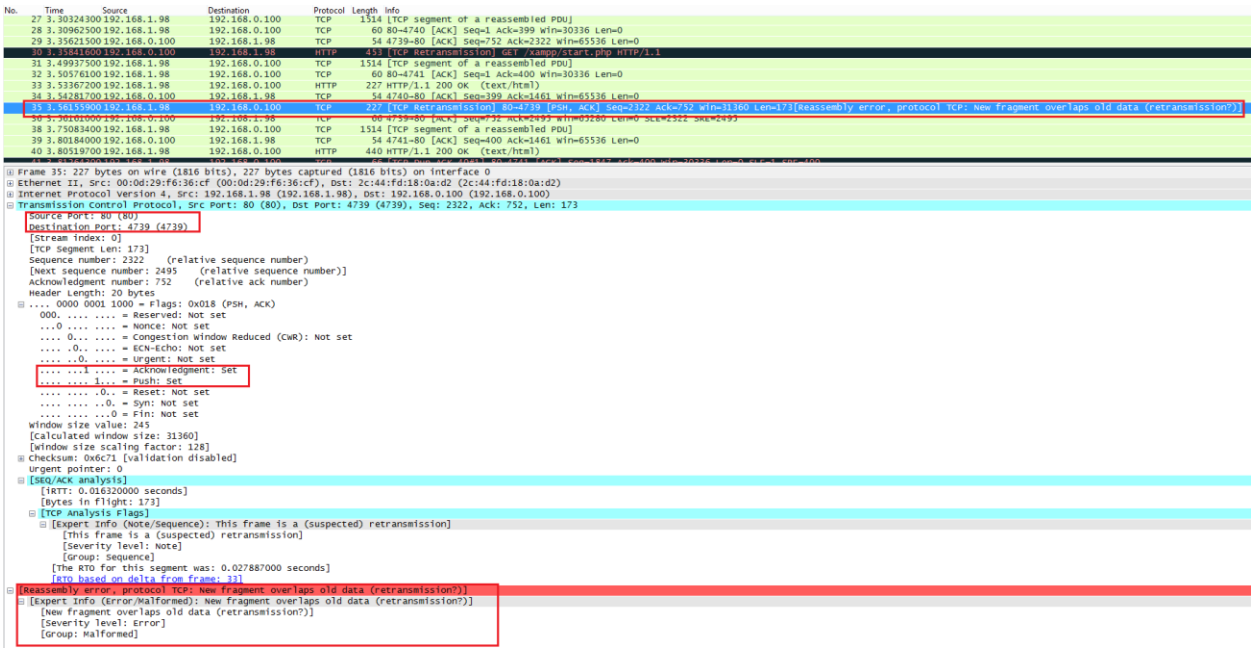


FIGURA 19 - PROTOCOLO TCP MOSTRANDO UNA RETRANSMISIÓN POR TRASLAPE

Un puerto cerrado enviara de regreso un RST/ACK a una solicitud TCP.

Debido a que en el paquete anterior se envió una retransmisión TCP del servidor al cliente, se puede observar como la maquina 1 cliente (192.168.0.100) envía de regreso un SR/ACK a la maquina 2 servidor (192.168.1.98) cuando se hace una solicitud a un puerto cerrado 4739.

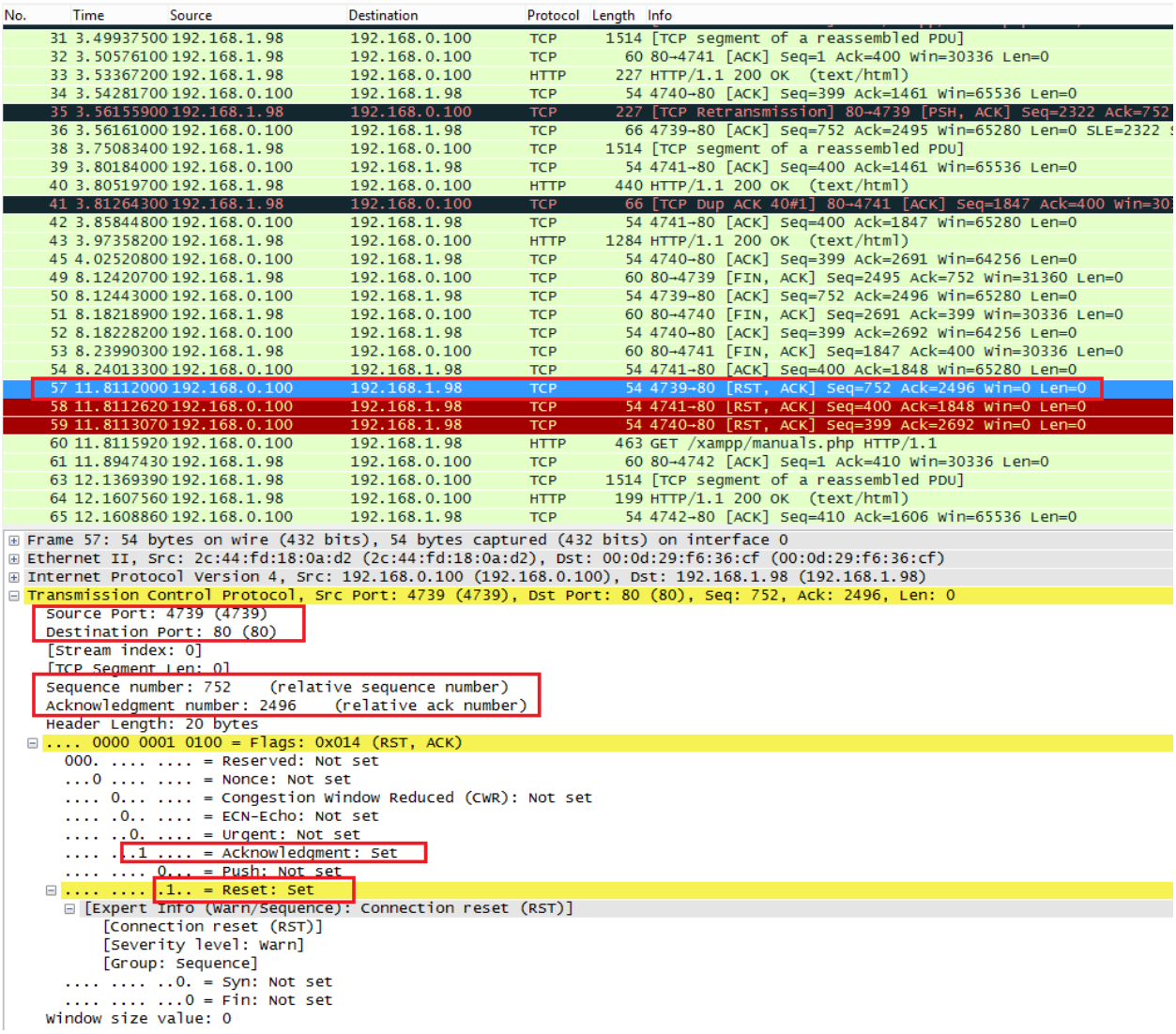


FIGURA 20 - PROTOCOLO TCP MOSTRANDO ACK/RST DEL PUERTO CERRADO

5 Conclusiones

HTTP es un protocolo es que da vida a Internet, y gracias al cual, los clientes y servidores se pueden comunicar. Siendo un protocolo de la capa de aplicación que está diseñado para comunicar un Web Browser con un servidor web.

El cliente envía una petición al servidor. Dicha petición está compuesta por un método a invocar en el servidor (URI) y una versión del protocolo, seguida por un mensaje compatible con MIME con los parámetros de la petición, información del cliente, y un cuerpo opcional con más datos para el servidor. Un ejemplo es:

```
GET  /index.html HTTP/1.0
Accept: text/plain
Accept: text/html
Accept: */*
User-Agent:  Un Agente de Usuario Cualquiera
```

El servidor responde con una línea de estado, incluyendo la versión del protocolo del mensaje y si la petición tuvo éxito o fracaso, con un código de resultado, seguido de un mensaje compatible con MIME con información del servidor, metainformación (datos acerca de la información) de la entidad solicitada y un cuerpo opcional con la entidad solicitada. Un ejemplo es:

```
HTTP/1.0 200 OK
Server: MDMA/0.1
MIME-version: 1.0
Content-type: text/html
Last-Modified: Thu Jul  7 00:25:33 1994
Content-Length: 2003
<title>Página de web del IEEE de Madrid<title>
<hr>
....
<hr>
<h2> Proyectos desarrollados en Internet <h2>
<hr>
```

Los datos se envían en texto plano, lo cual lo hace inseguro si se llegase a capturar los paquetes maliciosamente en medio de la comunicación, por lo cual existe su versión mas segura HTTPS.

6 Glosario

- **Bit** - es el acrónimo Binary digit (‘dígito binario’). Un bit es un dígito del sistema de numeración binario. Las unidades de almacenamiento tienen por símbolo bit.

- **Byte** - es una unidad de información utilizada como un múltiplo del bit. Generalmente equivale a 8 bits,3 4 5 6 7 8 9 10 por lo que en español se le denomina octeto.

- **Cliente** - es una aplicación informática o un ordenador que consume un servicio remoto en otro ordenador conocido como servidor, normalmente a través de una red de telecomunicaciones.

- **Dirección MAC -** la dirección MAC (siglas en inglés de media access control; en español "control de acceso al medio") es un identificador de 48 bits (6 bloques hexadecimales) que corresponde de forma única a una tarjeta o dispositivo de red. Se conoce también como dirección física, y es única para cada dispositivo. Está determinada y configurada por el IEEE (los últimos 24 bits) y el fabricante (los primeros 24 bits) utilizando el organizationally unique identifier. La mayoría de los protocolos que trabajan en la capa 2 del modelo OSI usan una de las tres numeraciones manejadas por el IEEE: MAC-48, EUI-48, y EUI-64, las cuales han sido diseñadas para ser identificadores globalmente únicos. No todos los protocolos de comunicación usan direcciones MAC, y no todos los protocolos requieren identificadores globalmente únicos.

- **Hipertexto** - es una herramienta de software con estructura no secuencial que permite crear, agregar, enlazar y compartir información de diversas fuentes por medio de enlaces asociativos.

- **LAN (Local Área Network)** – Red de Área Local. La topología de red define la estructura de una red. Una parte de la definición topológica es la topología *física*, que es la disposición real de los cables o medios.

- **Modelo OSI** - también llamado **OSI** (en inglés, ***O**pen **S**ystem **I**nterconnection* 'sistemas de interconexión abiertos') es el modelo de red descriptivo, que fue creado por la Organización Internacional para la Estandarización (ISO) en el año 1980. Es un marco de referencia para la definición de arquitecturas en la interconexión de los sistemas de comunicaciones.

- **Navegador Web** - o browser, es un software que permite el acceso a Internet, interpretando la información de archivos y sitios web para que éstos puedan ser leídos. La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados. Además, permite visitar páginas web y hacer actividades en ella, es decir, podemos enlazar un sitio con otro, imprimir, enviar y recibir correo, entre otras funcionalidades más.

- **Paquete de Red -** cada uno de los bloques en que se divide, en el nivel de Red, la información que enviar.

- **RFC (Request For Comments)** – son una serie de publicaciones del Internet Engineering Task Force (IETF) que describen diversos aspectos del funcionamiento de Internet y otras redes de computadoras, como protocolos, procedimientos, etc. y comentarios e ideas sobre estos.1 2 Cada RFC constituye un monográfico o memorando que ingenieros o expertos en la materia han hecho llegar al IETF, el consorcio de colaboración técnica más importante en Internet, para que éste sea valorado por el resto de la comunidad. De hecho, la traducción literal de RFC al español es "Petición de comentarios".

- **Router** – es un dispositivo que proporciona conectividad a nivel de red o nivel tres en el modelo OSI. Su función principal consiste en enviar o encaminar paquetes de datos de una red a otra, es decir, interconectar subredes.

- **Servidor** - Una aplicación informática o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes. Algunos servicios habituales son los servicios de archivos, que permiten a los usuarios almacenar y acceder a los archivos de una computadora y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final. Este es el significado original del término. Es posible que un ordenador cumpla simultáneamente las funciones de cliente y de servidor.

- **Servidor Web** - es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI. El término también se emplea para referirse al ordenador que ejecuta el programa.

- **Switch -** es un dispositivo digital lógico de interconexión de equipos que opera en la capa de enlace de datos del modelo OSI. Su función es interconectar dos o más segmentos de red, de manera similar a los puentes de red, pasando datos de un segmento a otro de acuerdo con la dirección MAC de destino de las paquetes en la red.

- **URI (Uniform Resource Identifier) -** del inglés Uniform Resource Identifier— es una cadena de caracteres que identifica los recursos de una red de forma unívoca.1 La diferencia respecto a un localizador de recursos uniforme (URL) es que estos últimos hacen referencia a recursos que, de forma general, pueden variar en el tiempo.

- **URL (Universal Resource Allocator)** - es un identificador de recursos uniforme (URI) cuyos recursos referidos pueden cambiar, esto es, la dirección puede apuntar a recursos variables en el tiempo.1 Están formados por una secuencia de caracteres, de

acuerdo a un formato modélico y estándar, que designa recursos en una red, como Internet.

- **WAN (Wide Área Network)** – Red de Área Amplia en Español. Lo que no es parte de la LAN. Es una red de computadoras que abarca varias ubicaciones físicas, proveyendo servicio a una zona, un país, incluso varios continentes. Es cualquier red que une varias redes locales, llamadas LAN, por lo que sus miembros no están todos en una misma ubicación física.
- **Wireshark** – antes conocido como Ethereal, es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicación.

7 Bibliografía

7.1 Libros

Comer, D. E. (2006). *Internetworking with TCP/IP: principles, protocol, and architecture*. Nueva Jersey: Pearson.

7.2 Videos

Safari books: https://www.safaribooksonline.com/

Video Entrenamiento: Networking and IP Addressing Fundamentals LiveLessons

7.3 RFCs

- RFC 7230 - HTTP/1.1: Message Syntax and Routing
- RFC 7231 - HTTP/1.1: Semantics and Content
- RFC 7232 - HTTP/1.1: Conditional Requests
- RFC 7233 - HTTP/1.1: Range Requests
- RFC 7234 - HTTP/1.1: Caching
- RFC 7235 - HTTP/1.1: Authentication
- RFC 3886 – Uniform Resource Identifier (URI)