**Capa de Transporte - Parte II Redes y comunicaciones - 2022**

**Práctica 6**

**Capa de Transporte - Parte II**

**9 y 12 (ver de que capturas habla),**

**13 y 14 (no los vamos a hacer)**

**1. ¿Cual es el puerto por defecto que se utiliza en los siguientes servicios?**

**Web / SSH / DNS / Web Seguro / POP3 / IMAP / SMTP**

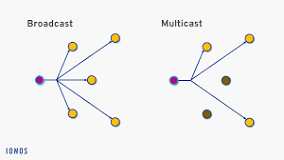
|  |  |
| --- | --- |
| **Protocolo** | **Puerto** |
| HTTP | 80 |
| SSH | 22 |
| DNS | 53 |
| HTTPS | 443 |
| POP3 | 110 |
| IMAP | 143 |
| SMTP | 25 |

**Investigue en qué lugar en Linux y en Windows está descrita la asociación utilizada por defecto para cada servicio.**

En Linux está en el archivo /etc/services.

En Windows está en el archivo C:\WINDOWS\system32\drivers\etc\services.

**2. Investigue qué es multicast.**



* Multicast: Entrega de datos simultánea (Sin enviarlo uno por uno a cada receptor) a un grupo de nodos destino, desde un emisor como origen.
* Unicast: Un emisor se comunica con un único nodo destino.

**¿Sobre cuál de los protocolos de capa de transporte funciona?**

Funciona sobre UDP ya que no necesita confirmación de llegada de la información, TCP necesitaría establecer la conexión y controlar que cada segmento llegue, además de que cada segmento transporta más información que en UDP, eso para cada receptor.

**¿Se podría adaptar para que funcione sobre el otro protocolo de capa de transporte? ¿Por qué?**

No podría adaptarse, ya que éste sólo sirve para establecer una conexión lógica directa, entre dos extremos.

No puede hacer control de sincronización con todos los receptores a la vez.

**3. Investigue cómo funciona el protocolo de aplicación FTP teniendo en cuenta las diferencias en su funcionamiento cuando se utiliza el modo activo de cuando se utiliza el modo pasivo**

FTP (File Transfer Protocol) es un protocolo para transferir archivos de un sistema local a uno remoto, a diferencia de otros protocolos que brindan acceso a archivos: NFS, CIFS.

Modelo cliente/servidor, command/response.

El protocolo corre sobre TCP (requiere protocolo de transporte confiable).

**Comandos FTP**

* RETR: obtener un archivo desde el servidor. A nivel de interfaz de usuario el comando que lo inicia es el get.
* STOR: envía un archivo al servidor. A nivel de interfaz de usuario el comando que lo inicia es el put.
* LIST: listar los archivos del directorio actual en el servidor. A nivel de interfaz de usuario el comando que lo inicia es el ls o dir.
* DELE: comando para borrar un archivo en el servidor.
* SIZE, STAT: obtiene información de un archivo en el servidor.
* CD, PWD, RMD, MKD: cambia de directorio, obtiene el dir. actual, borra y crea dir.

**Funcionamiento de FTP**

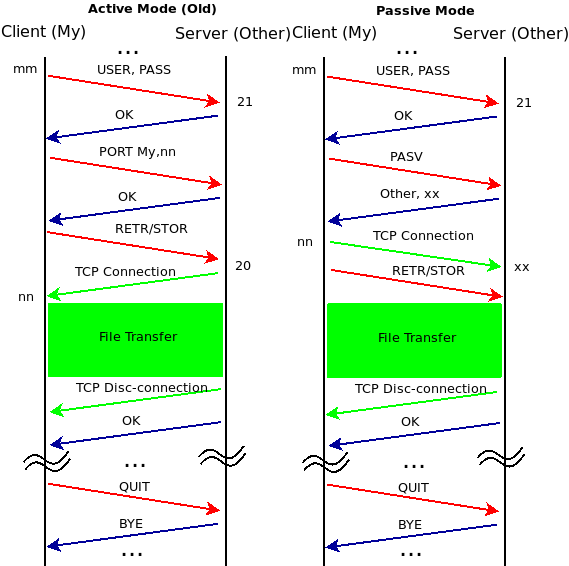
Usa 2 (dos) conexiones TCP desde un puerto no privilegiado, se mantiene permanentemente:

* Conexión de Control (Out-Of-Band Control) **port 21**. Cliente servidor, controla si queres mandar o traer. Similar a http, comando get put. Requiere mínimo delay.
* Conexión para la transferencia de datos que se crea y cierra bajo demanda. Al contrario de http que es la misma se reutiliza. Requiere máximo throughput.

El cliente escoge cualquier puerto no privilegiado, (n > 1023) y genera conexión de control contra el puerto 21 del servidor.

El servidor recibe los comandos por dicha conexión y responde/recibe por la conexión de datos aquellos que lo requieran.

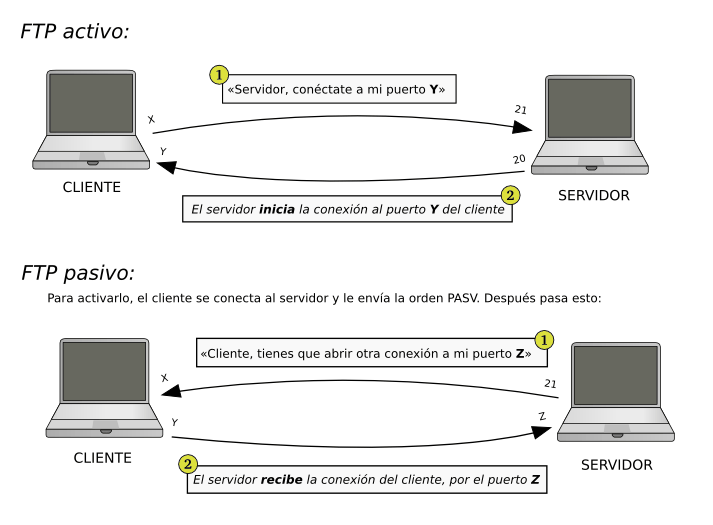
El estado de cada operación se transmite por el canal de control.

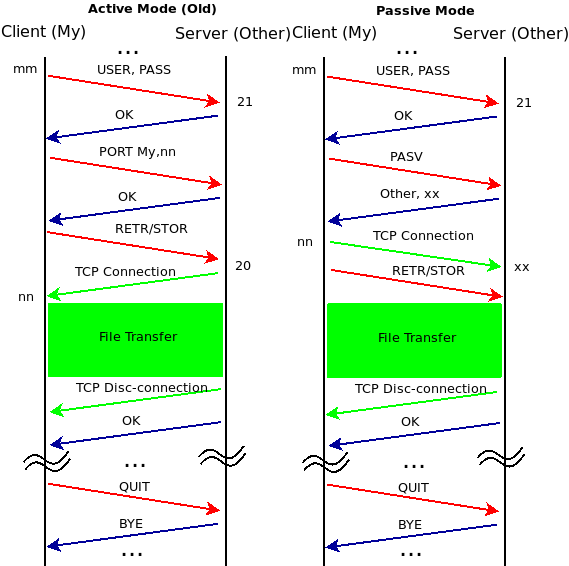


Modalidades de FTP (Activo o pasivo: punto de vista del servidor)

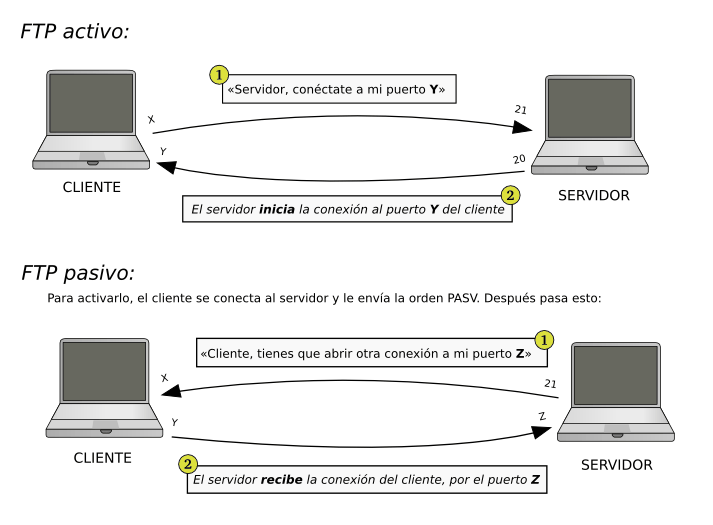
**FTP Activo (modalidad vieja):**

* Conexión de control: port 21.
* Conexión de datos: port 20.
* Se diferencia cómo maneja la conexión de datos.
* El servidor de forma activa se conecta al cliente para generar la conexión de datos.
* A través del canal de control, el cliente le indica al servidor a qué puerto debe conectarse para establecer la conexión de datos

Dos conexiones: Una desde el cliente, hacia el puerto 21 del servidor, mediante la cual se indica al servidor con el comando **PORT** a cuál será el puerto del cliente que se colocará a la escucha de datos. Así, es el servidor quien inicia la transferencia de los datos, desde su puerto 20 al puerto que se le ha sido indicado.

 FTP Pasivo:

* Conexión de control: port 21.
* Conexión de datos: port no privilegiado.
* A través del canal de control, el servidor le indica al cliente a qué puerto debe conectarse para establecer la conexión de datos

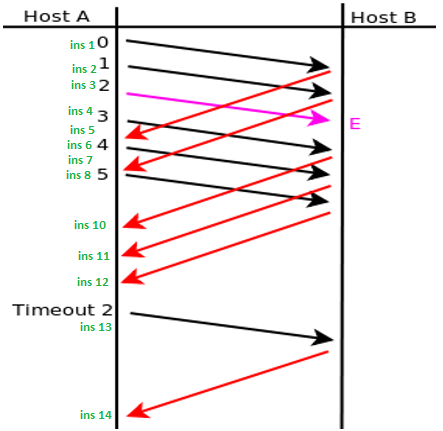
El cliente inicia la conexión con el servidor. Se abre primeramente una conexión de control desde un puerto superior a 1023 del cliente, al puerto 21 del servidor. El cliente pasa a modo pasivo enviando el comando **PASV** y pidiendo un puerto abierto al servidor. Tras recibir la respuesta, el cliente establece la conexión de datos con el servidor a través del puerto que le ha sido enviado.

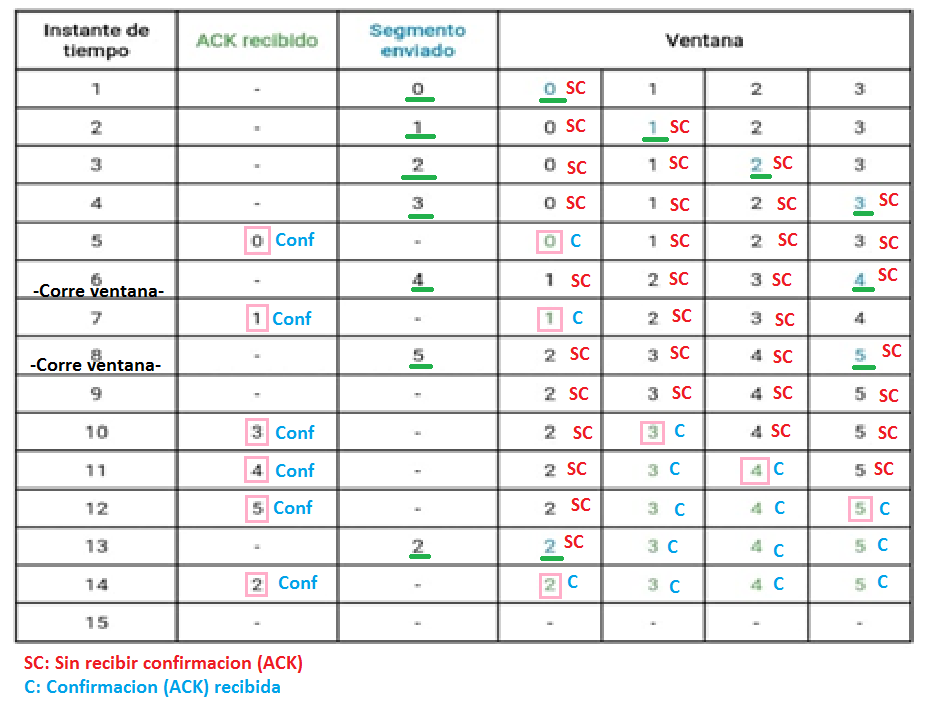
**¿En qué se diferencian estos tipos de comunicaciones del resto de los protocolos de aplicación vistos?**

Usan dos conexiones, una de control y una de datos.

**4. Suponiendo Selective Repeat; tamaño de ventana 4 y sabiendo que E indica que el mensaje llegó con**

**errores. Indique en el siguiente gráfico, la numeración de los ACK que el host B envía al Host A.**

****

****

Después de recibir la confirmación del segmento enviado que esta primero en la ventana, puede correr esta, dejándole lugar a un nuevo segmento para que sea enviado.

**5. ¿Qué restricción existe sobre el tamaño de ventanas en el protocolo Selective Repeat?**

El tamaño de la ventana debe ser menor o igual a 2^(w-1), siendo w los bits de números de secuencia. Esto es porque si no puede pasar que, con un tamaño de ventana 3, mandanado 3 segmentos, si falla un ack, el emisor envía de nuevo, pero el receptor va a pensar que son segmentos nuevos.

[initially] (B's window = [0,1,2])

A -> 0 -> B (B's window = [1,2,3])

A -> 1 -> B (B's window = [2,3,0])

A -> 2 -> B (B's window = [3,0,1])

[lost] ACK0

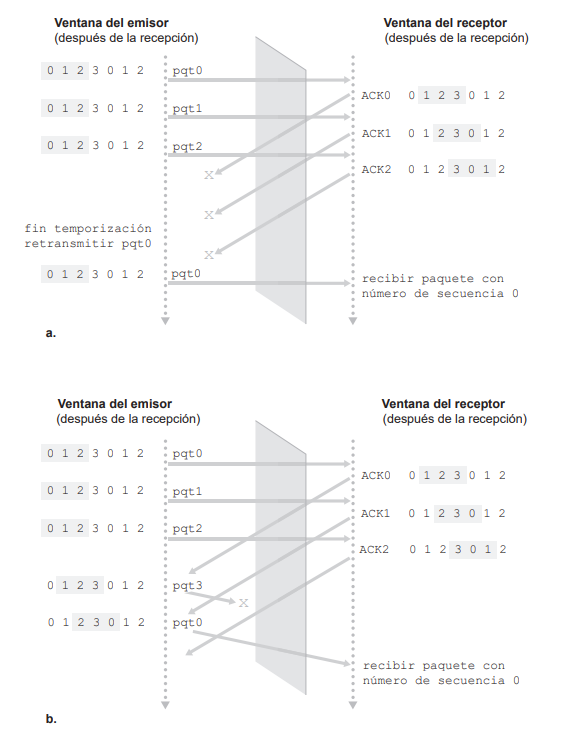
[lost] ACK1

A <- ACK2 <- B

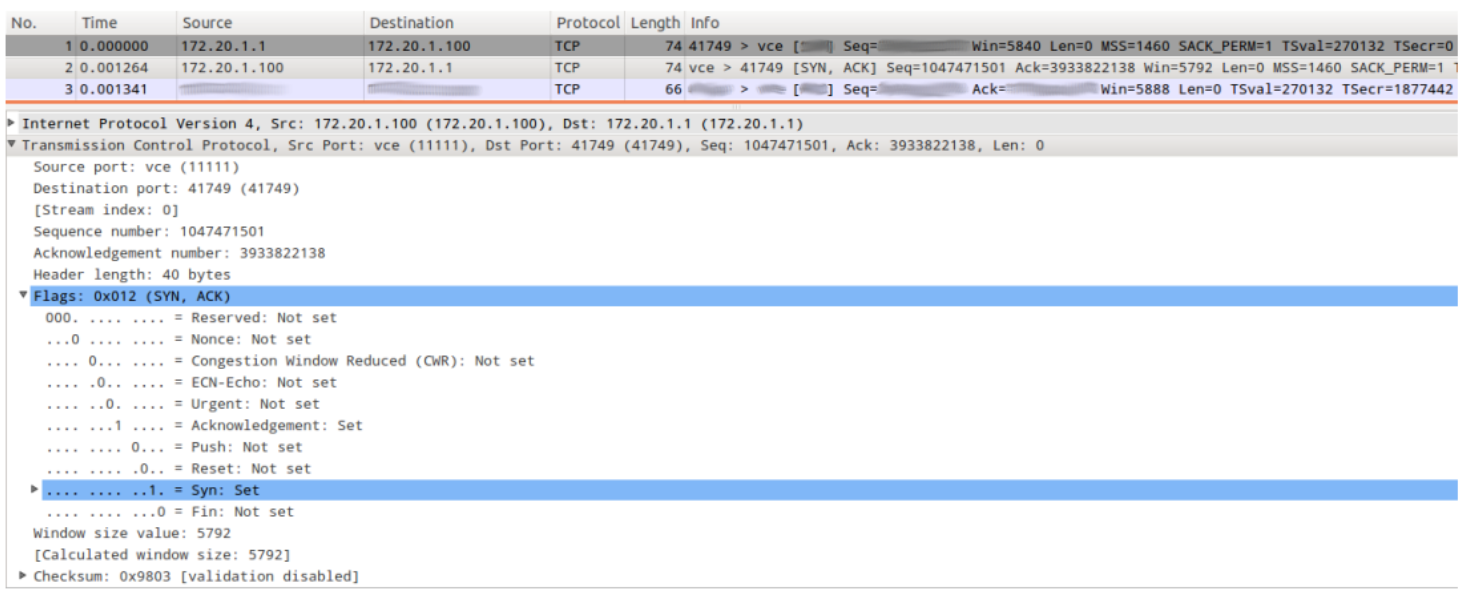
A -> 3 -> B

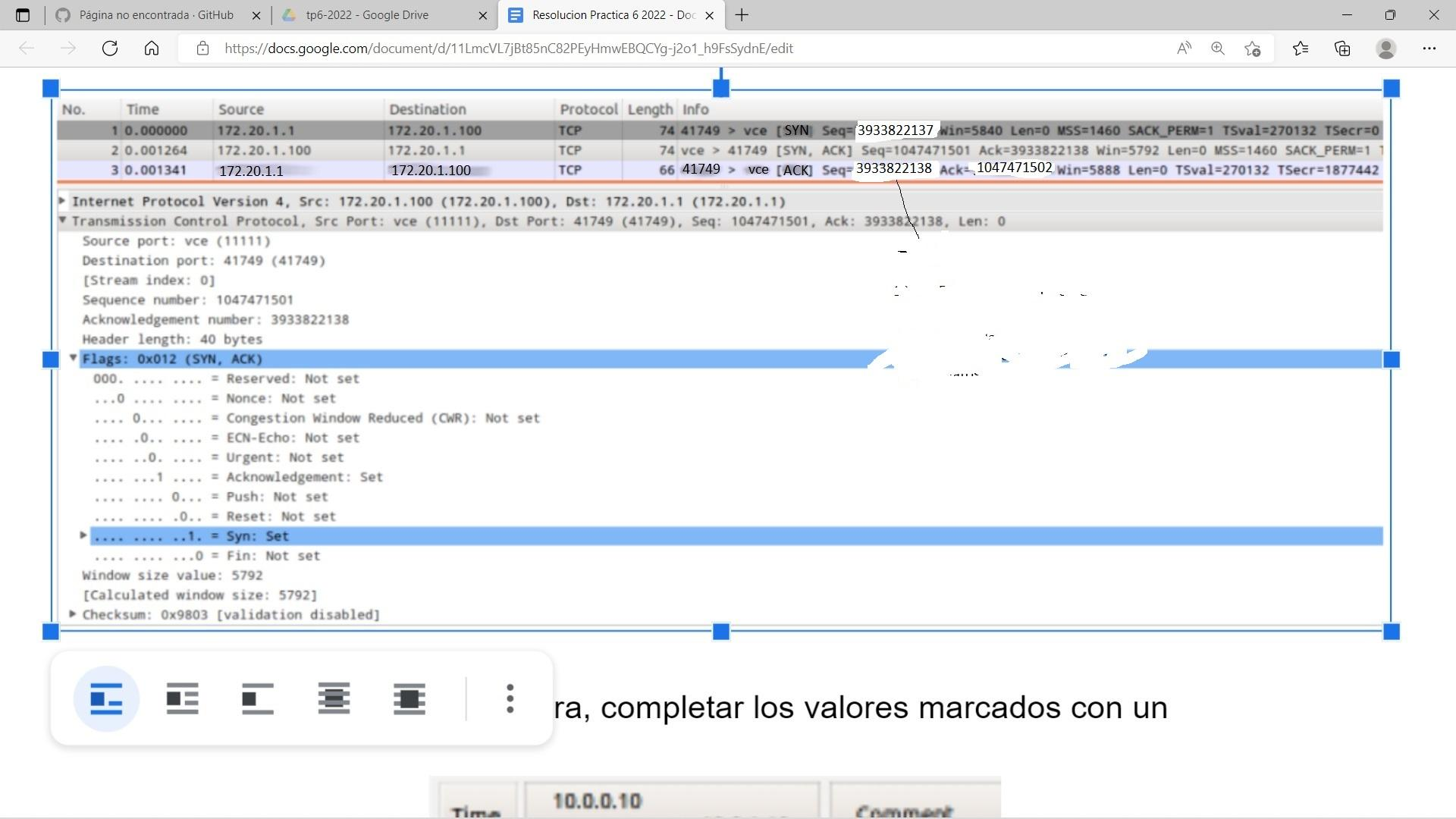
A -> 0 -> B [retransmission]

A -> 1 -> B [retransmission]



**6. De acuerdo a la captura TCP de la siguiente figura, indique los valores de los campos borroneados.**

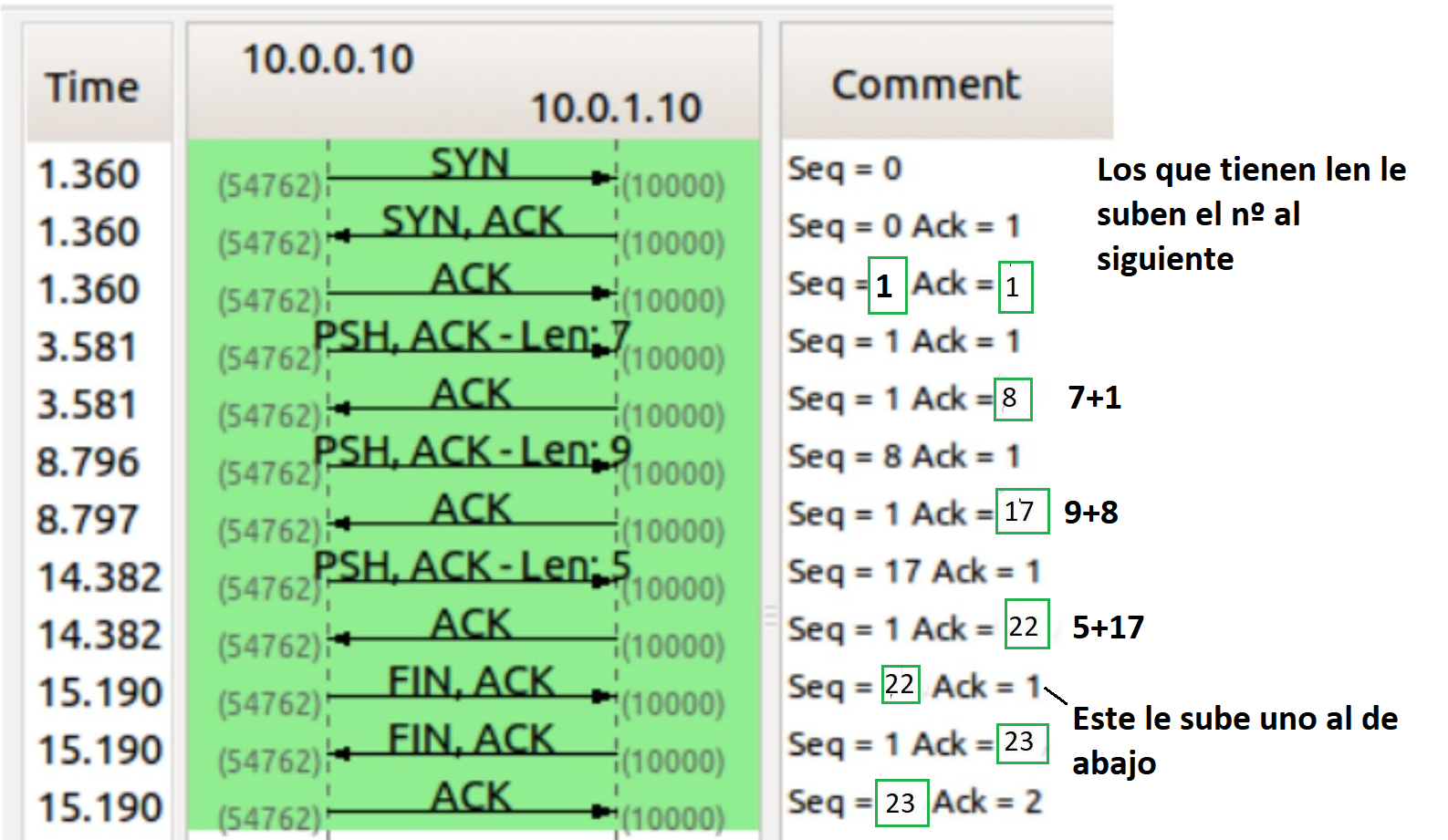
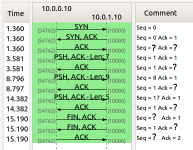




3933822138: el ACK recibido

1047471502: el número de secuencia recibido + 1 (por el flag SYN)

**7. Dada la sesión TCP de la figura, completar los valores marcados con un signo de interrogación.**



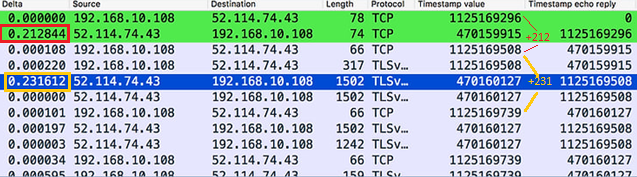
ACK 2 al final porque fin y syn suman 1.

**8. ¿Qué es el RTT y cómo se calcula? Investigue la opción TCP timestamp y los campos TSval y TSecr.**

Las marcas de tiempo (Timestamp) en TCP permiten que los puntos finales mantengan una medición actual del tiempo de ida y vuelta o tiempo de envio + confirmación de su recepción **(RTT)** de la red entre ellos. Ayuda a cada pila TCP a establecer y ajustar su temporizador de retransmisión.

Cada extremo de la conexión deriva un valor único creciente de 4 bytes que se incrementará a medida que avance la conexión. Al extremo opuesto no le importa cuál es el valor, simplemente lo devolverá al remitente original y este puede medir el tiempo entre los paquetes que se enviaron y recibieron con dicho valor único.

En la captura de pantalla a continuación, ambos extremos de la conexión TCP usan marcas de tiempo.



El primer paquete tiene un valor de marca de tiempo de 1125169296. Al comienzo de la conexión, el remitente aún no ha visto un valor de marca de tiempo desde el extremo opuesto de la conexión, por lo que aún no tiene un número para recuperar y se muestra un cero en la columna de respuesta de eco de marca de tiempo.

El segundo paquete muestra al receptor repitiendo el valor de la marca de tiempo, mientras envía un valor único propio. Estos dos números son completamente diferentes y no están relacionados. Sin embargo, en SYN/ACK, el valor de eco de la marca de tiempo debe ser exactamente el mismo que el valor enviado en SYN, o la conexión fallará.

En el tercer paquete, podemos ver que el remitente original aumenta su marca de tiempo en 212, enviando un nuevo valor al otro extremo. En la columna de tiempo delta, podemos ver que el tiempo de ida y vuelta entre las dos estaciones es de 212 milisegundos (capturamos en el extremo 192.168.10.108). Este valor se agrega a la marca de tiempo original y se envía al siguiente segmento.

A medida que avanza la conexión, estos valores aumentarán. Solo se repiten desde el lado opuesto, lo que permite que el remitente marque y mida el tiempo de ida y vuelta. El valor en sí no tiene un significado específico.

* El Round-Trip Time (RTT) es el tiempo entre el envío de un segmento TCP, y la confirmación de su recepción. Al enviar se inicia un timer, al recibirlo de detiene, y el valor medido será el sRTT (RTT sampleado o medido)
* El campo TSval (Timestamp Value), se usa para distinguir segmentos de mismo número de secuencia. Se utiliza junto con el número de secuencia para identificar segmentos de forma única (ya que los números de secuencia pueden ajustarse).
* El campo TSecr (Timestamp Echo Reply) almacena una copia del último valor recibido de ese mismo campo. Se usa para calcular el RTT

**9. Para la captura dada, responder las siguientes preguntas. CUAL CAPTURA, MI CIELA? xD**

**a. ¿Cuántos intentos de conexiones TCP hay?**

**b. ¿Cuáles son la fuente y el destino (IP:port) para c/u?**

**c. ¿Cuántas conexiones TCP exitosas hay en la captura? Cómo diferencia las exitosas de las que no**

**lo son? ¿Cuáles flags encuentra en cada una?**

**d. Dada la primera conexión exitosa responder:**

**i. ¿Quién inicia la conexión?**

**ii. ¿Quién es el servidor y quién el cliente?**

**iii. ¿En qué segmentos se ve el 3-way handshake?**

**iv. ¿Cuáles ISNs se intercambian?**

**v. ¿Cuál MSS se negoció?**

**vi. ¿Cuál de los dos hosts enva la mayor cantidad de datos (IP:port)?**

**e. Identificar primer segmento de datos (origen, destino, tiempo, número de fila y número de secuencia TCP).**

**i. ¿Cuántos datos lleva?**

**ii. ¿Cuándo es confirmado (tiempo, número de fila y número de secuencia TCP)?**

**iii. La confirmación, ¿qué cantidad de bytes confirma?**

**f. ¿Quién inicia el cierre de la conexión? ¿Qué flags se utilizan? ¿En cuáles segmentos se ve (tiempo,**

**número de fila y número de secuencia TCP)?**

**10. Responda las siguientes preguntas respecto del mecanismo de control de flujo.**

**a. ¿Quién lo activa?**

Lo activa cada extremo, el emisor se ajusta al receptor, en cada sentido

**¿De qué forma lo hace?**

En cada segmento, el receptor le indica al emisor cuál es el tamaño máximo de ventana actual, y el emisor se adapta a este valor. Con esto el emisor puede regular según su buffer de recepción.

**b. ¿Qué problema resuelve?**

Hace que un extremo no sature al otro, enviándole más datos de los que puede procesar (o al menos, recibir). Si el emisor sobrecarga al receptor, se descartarían datos que habría que retransmitir, lo que generaría un mal uso de la red, por lo que es mejor que el receptor avise cuanto puede recibir.

**c. ¿Cuánto tiempo dura activo y qué situación lo desactiva?**

Dura toda la sesión TCP

Lo desactiva el cierre de sesión

**11. Responda las siguientes preguntas respecto del mecanismo de control de congestión.**

**a. ¿Quién activa el mecanismo de control de congestión? ¿Cuáles son los posibles disparadores?**

Se puede implementar extremo a extremo, o tomando como partida la red. Cada extremo supone lo que pasa en la red.

**Lo activa el emisor.**

Disparadores para determinar congestión:

* Timeout: Se vencen los timers para ACKs
* Triple ACK: Llegan ACKs duplicados (El primero, y 3 duplicados). El emisor envía, y el receptor le envía ACKs de otros paquetes anteriores. Esto puede pasar porque, si llegan segmentos desordenados, el receptor pedirá el segmento que le falta

El emisor detecta este problema. Sin embargo, no se puede determinar con exactitud si hay congestión, ya que estos eventos pueden darse aun sin que esta exista.

**b. ¿Qué problema resuelve?**

Permite que aplicaciones no saturen o colapsen la red, generando que se descarte tráfico y tenga que retransmitirse. Para esto tiene en cuenta el estado de la red a diferencia del control de flujo que solo ve el receptor.

Problemas de delay en los routers, problemas de overflow y descarte.

**c. Diferencie slow start de congestion-avoidance.**

**Control de Congestion TCP (Old Tahoe/Tahoe)**

• Utiliza Slow Start (SS), ventana crece exponencialmente, inicio cwnd = IW = 1 ∗ MSS (Tamaño Máximo de Segmento), o similar, IW (Init Window) puede ser 2 o 3 segmentos.

• Una vez que se alcanza sstresh se trabaja con Congestion Avoidance (CA).

• Valor inicial ssthresh (Slow Start Threshold o Umbral de inicio lento). = ∞ (un valor alto). También puede arrancar con un histórico que de un umbral estimado. Crece de forma exponencial y después hay un umbral estimado.

• Congestión detectada por RTO o **3DUP ACKs** \*, derivaba en ambos casos en: Slow Start: ssthresh = Min(cwnd/2, 2) ∗ MSS , cwnd = LW = 1 ∗ MSS (Loss Window).

• Puede suceder que MSS sea diferente entre emisor y receptor, para este caso se considera SMSS y RMSS. Los cálculos se hacen en base a SMSS.

**\*3DUP ACKs**: ACK duplicados, muchos para el mismo segmento, hueco en el buffer de recepción. A partir de 3 no espera el timeout, hace retransmisión. Esto genera que se envie, sature y vuelva a arrancar.

\*-\*-\*-\*-\*-\*-\*

Control de Congestion TCP (SS). **Slow Start** (Etapa de crecimiento exponencial):

• Crece exponencialmente, de forma rápida, no es slow (lento).

• Se le llama Slow Start porque comienza a probar con pocos paquetes, menos agresivo que el enfoque de enviar tanto como la ventana de recepción permita. ´

• Inicia cwnd = IW = 1 ∗ MSS (a veces se usa 2 o 3). Transmite y espera ACK.

• ACK recibido, cwnd + +: cwnd = 2 ∗ MSS, nuevos ACKs: cwnd = 4 ∗ MSS ...

• Si destino retarda ACK no se cumple.

• Incrementa cwnd + + (en MSS) por cada ACK (varios por RTT, rafaga). ´

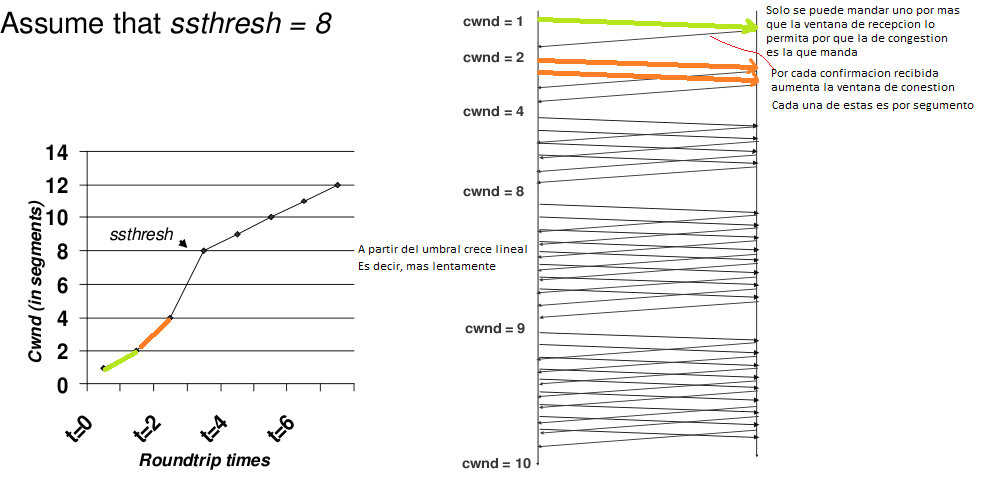
Control de Congestion TCP (CA). **Congestion Avoidance** (Etapa de crecimiento lineal):

• Ante el primer evento de congestión se calcula el ssthresh, primer rafaga SS puro. ´

• Una vez que cwnd >= ssthresh crece de forma lineal.

• Incrementa cwnd + + por cada RTT.

Fast Retransmit: objetivo, recuperarse más rápido que un timeout. En Tahoe, FRT seguido por Slow Start. Vuelve al inicio cwnd = 1 ∗ MSS.

****

Crece exponencialmente hasta que hay congestión. Esto se sabe cuándo hay un segmento que hace un timeout, algo no llego a tiempo. Cuando pasa esto se establece el umbral en la mitad. Ej: Si se llegó a 16, quedaba en 8 y se vuelve a arrancar con la ventana inicial hasta el umbral.

**12. Para la captura dada, responder las siguientes preguntas. CUAL CAPTURA, MI CIELA? xD**

**a. ¿Cuántas comunicaciones (srcIP,srcPort,dstIP,dstPort) UDP hay en la captura?**

**b. ¿Cómo se podrían identificar las exitosas de las que no lo son? mati estuvo aquí**

**c. ¿UDP sigue el modelo cliente/servidor?**

**d. ¿Qué servicios o aplicaciones suelen utilizar este protocolo?**

**e. ¿Qué hace el protocolo UDP en relación al control de errores?**

**f. Con respecto a los puertos vistos en las capturas, ¿observa algo particular que lo diferencie de**

**TCP?**

**g. Dada la primera comunicación en la cual se ven datos en ambos sentidos (identificar el primer**

**datagrama):**

**i. ¿Quién envía el primer datagrama (srcIP,srcPort)?**

**ii. ¿Cuantos datos se envían en un sentido y en el otro?**

**h. ¿Se puede calcular un RTT?**

En UDP no se puede garantizar que udp te responda y menos al instante, por lo que no podes calcular un RTT.

**Programación de sockets**

**Resuelva los siguientes ejercicios utilizando el lenguaje de programación que prefiera (por simpleza, se**

**recomiendan Python o Ruby).**

**13. Desarrolle un cliente y un servidor, donde el cliente envíe un mensaje al servidor y este último imprima en pantalla el contenido del mismo.**

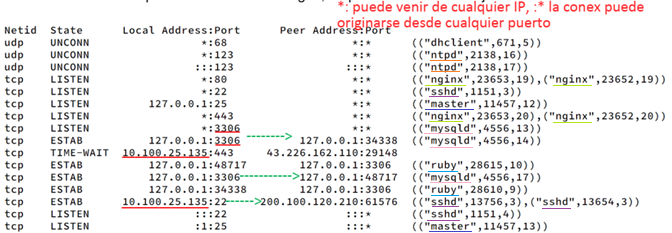
**a. Utilizando UDP.**

**b. Utilizando TCP.**

**14. Compare ambas implementaciones. ¿Qué diferencia nota entre la implementación de cada una? ¿Cuál le parece más simple?**

**Ejercicios de parcial**

**15. Dada la salida que se muestra en la imagen, responda los ítems debajo.**

**Suponga que ejecuta los siguientes comandos desde un host con la IP 10.100.25.90. Responda qué devuelve la ejecución de los siguientes comandos y, en caso que corresponda, especifique los flags.**

El \* dice que el servicio está habilitado a todas las interfaces de la máquina

Si es 127.0.0.1 (localhost), solo los procesos locales pueden acceder

Las tuplas (IPorigen, PUERTOorigen, IPdestino, PUERTOdestino, IDprotocolo) deben ser unívocas, entonces puedo tener múltiples conexiones a un mismo puerto destino de una misma IP, porque algún valor cambia entre las tuplas

:1:25 equivale al localhost, de IPv6

Los que están corriendo de tcp son los que están en estado listen, excepto uno de ipv6 que es el mismo, escuchando en otra dirección (Local host ipv4 e ipv6(?, y lo del puerto 22)

-S ó –syn -> Manda SYN

-p -> puerto

**a. hping3 -p 3306 –udp 10.100.25.135**

No se puede establecer la conexión. El puerto 3306 trabaja con TCP. Udp responde con un ICMP Port Unreachable (Error “puerto inalcanzable”).

**b. hping3 -S -p 25 10.100.25.135**

No se puede establecer la conexión. El puerto trabaja con localhost por lo que a ese servicio se puede conectar solo desde esa máquina

Devuelve RA, es decir TCP (RST = true; ACK = true) (reset), es decir, un segmento de rechazo.

**c. hping3 -S -p 22 10.100.25.135**

Se puede establecer la conexión. Responde con un segmento de respuesta TCP (SYN = true; ACK = true)

Hay una establecida en ese puerto, pero se puede multiplexar. El servidor atiende en el mismo puerto, lo que cambia es el puerto origen del cliente.

**d. hping3 -S -p 110 10.100.25.135**

El puerto 110 no está escuchando. Responde con un segmento de rechazo TCP (RA: RST = true; ACK = true). Siempre que no tiene un puerto al cual pasarle la sesión, TCP hace esto para indicarle al emisor que algo anda mal.

**¿Cuántas conexiones distintas hay establecidas? Justifique.**

e) Hay 3 conexiones establecidas (2 de las 5 que se ven en estado ESTAB representan a una misma, localhost en puerto 3306, la muestra desde ambos extremos):

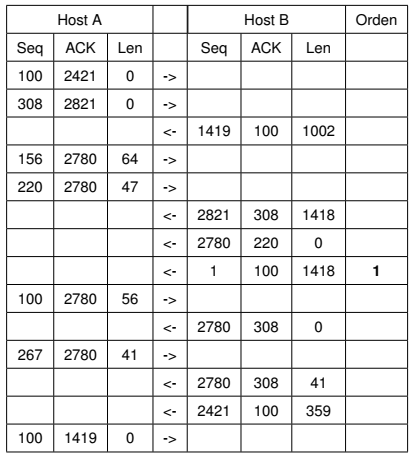
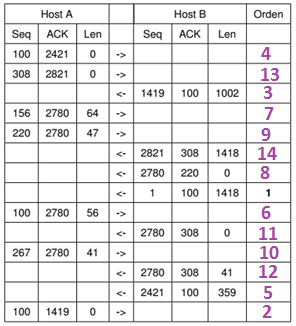
i) Entre 127.0.0.1:3306 y 127.0.0.1:34338

ii) Entre 127.0.0.1:3306 y 127.0.0.1:48717

iii) Entre 127.0.0.1:22 y 200.100.120.210:61576

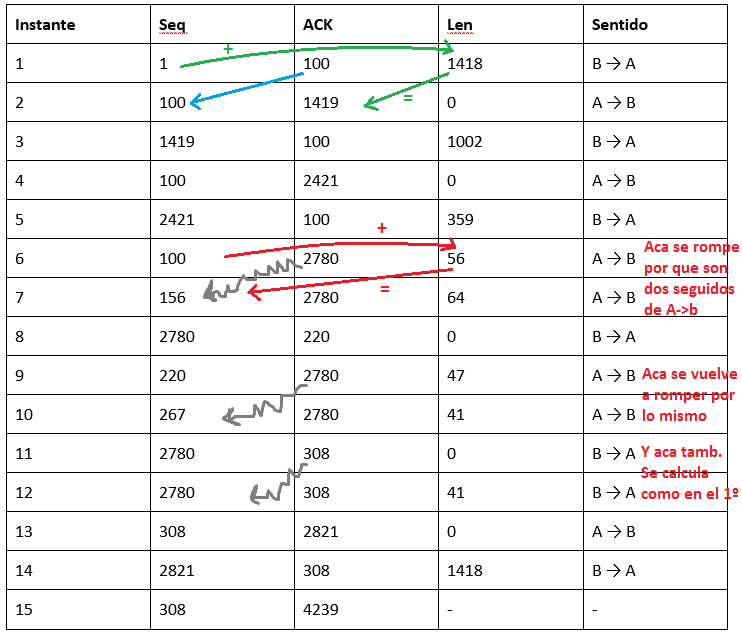
* time-wait es un estado de cierre de sesión, por lo que no se cuenta. Estuvo establecida, pero algún extremo la está cerrando.
* No siempre el que inicia la sesión la cierra, por más que avance correctamente el proceso. Por ejemplo, en http 1.0 el cliente pide un objeto, el servidor se lo manda y cerraba la sesión. El servidor no le decía “te voy a enviar varias cosas” así el cliente esperaba.
* **el cliente siempre se conecta de un puerto >= 1024**
* SOCKET: punto en donde se conecta la aplicación con tcp.

**16. Complete en la columna Orden, el orden de aparición de los paquetes representados en cada fila.**

****

Normalmente seq + len = ACK y el ACK anterior es el siguiente Sec

Cuando son dos seguidos desde el mismo hacia el otro, es seq + len= sec de abajo, y el ACK se mantiene

****