



Departamento de Ingeniería  
Telemática y Electrónica

# Computer Networks (Non-Presential)

## Internet Transport Layer

Spring semester, 2022/2023



## **Content**

1. TCP1-T(NP): The UDP Pseudo-Header.....	3
2. TCP2-T(NP): The SYN Flood Attack .....	3
3. TCP3-T(NP): Silly Window Syndrome .....	3
4. TCP4-T(NP): Questions.....	3
5. TCP5-T(NP): Theory Exercises .....	3
6. TCP5-T(NP): Laboratory Exercises .....	3

## **1. TCP1-T(NP): The UDP Pseudo-Header**

In order to complete the study of the UDP segment format, look for information about “UDP Pseudo-Header” used during UDP checksum computation.

*Reference book: Internetworking with TCP/IP. Vol. 1, Douglas E. Comer*

## **2. TCP2-T(NP): The SYN Flood Attack**

In order to complete the study of the TCP security aspects, look for information of “The SYN flood attack”.

*Reference book: Computer Networking, Kurose & Ross.*

## **3. TCP3-T(NP): Silly Window Syndrome**

Look for information about the “Silly Window Syndrome” and the different techniques to avoid it.

*Reference book: Internetworking with TCP/IP. Vol. 1, Douglas E. Comer*

## **4. TCP4-T(NP): Questions**

See attachment.

## **5. TCP5-T(NP): Theory Exercises**

See attachment.

## **6. TCP5-T(NP): Laboratory Exercises**

See attachment.

# Computer Networks

## Internet Transport Layer

### TCP4-T(NP): Questions



Responder Verdadero (V) o Falso (F).

- V 1. El campo “Acknowledgement number” del segmento TCP indica el número del byte siguiente que se espera recibir.
- F 2. Suponga que el host A envía dos segmentos TCP seguidos al host B a través de una conexión TCP y que todos los segmentos enviados anteriormente están confirmados. El primer segmento tiene el número de secuencia 90 y el segundo tiene el número de secuencia 110. Si el primer segmento se pierde pero el segundo llega a B, el número de reconocimiento del paquete que el host B envía al host A es 110.
- F 3. Un sistema final A ha iniciado una conexión TCP con otro sistema B y ambos se encuentran ya en la fase de transferencia de datos (estado ESTABLISHED). En cierto instante de ese estado, B recibe un segmento SYN desde el mismo sistema A, con los mismos valores de puertos origen y destino. El sistema B, al recibir este segundo SYN, aceptará una nueva conexión para el mismo par de puertos origen-destino, siempre que el número de secuencia inicial sea distinto.
- V 4. Una conexión TCP en un cliente pasará del estado CLOSED al estado SYN\_SENT tras enviar un segmento con los siguientes valores: SYN=1, ACK=0, puerto origen=1300, puerto destino= 80.
- V 5. Los mecanismos principales del algoritmo de control de congestión de TCP son el arranque lento (slow start), la evitación de congestión (congestion avoidance) y la recuperación rápida (fast recovery).
- F 6. Un cliente ha mantenido una conexión TCP sobre los puertos origen y destino 2.222 y 80, respectivamente; los números de secuencia empleados durante la conexión han sido desde 15.000 al 18.000 en el sentido cliente a servidor, y desde 200.000 a 400.000 en el sentido servidor al cliente. Después de que la conexión ha sido terminada por ambos extremos, el cliente recibe un segmento SYNACK con los valores de puerto origen y destino 80 y 2.222 respectivamente y con números de secuencia SEQ=200.000 y ACK=15.001. La acción del cliente será la de transmitir un segmento SYN con número SEQ=15.000.
- V 7. Considere una conexión TCP entre los sistemas A y B donde se han fijado como números iniciales de secuencia los valores  $x$  (de A a B) e  $y$  (de B a A). Si el primer segmento de datos recibido por B contiene 1.000 bytes y tiene número de secuencia  $z$ , con  $z > x+1$ , B reaccionará transmitiendo ACK  $x+1$  y guardando los datos en el buffer de recepción.
- F 8. Existe un rango de 65535 puertos disponibles para el nivel de transporte: si un número de puerto es utilizado por TCP, no podrá ser utilizado dicho número por UDP hasta que se libere la conexión.
- F 9. Cuando una entidad IP recibe un datagrama, decide a qué protocolo de transporte entregar su campo de información, analizando para ello si los datos transportados incluyen una cabecera TCP o bien UDP

10. El servicio no fiable que proporciona el nivel de transporte en Internet, puede perder, duplicar, o desordenar los mensajes enviados.
11. Si A y B son dos hosts diferentes e inician una sesión Telnet con el servidor S, es posible que el número de puerto origen que va en los segmentos de A a S sea el mismo que el que va en los segmentos de B a S.
12. En el estado de arranque lento el valor de la ventana de congestión se establece en 1 MSS y se incrementa 1 MSS cada vez que se produce el primer reconocimiento de un segmento transmitido.
- F 13. En el algoritmo de “fast retransmit” se reenvía el paquete TCP perdido tras vencer el temporizador.
14. TCP permite el envío de los últimos bytes de información de la secuencia en el segmento de liberación ordenada (indicador FIN activo).
- F 15. Durante la fase de transferencia en TCP, una entidad TCP sólo mantiene en el buffer de envío aquellos bytes que se encuentren dentro de su ventana ofrecida.
16. En TCP, la velocidad media de transmisión de información es un valor independiente para cada sentido de la conexión.
- F 17. En un instante determinado de la operación del protocolo TCP, uno de los dos sistemas finales se encuentra en el estado que determinan las siguientes variables, donde todos los valores están medidos en bytes:  
 $RcvWindow = 3.000$ ;  $CongWin = 3.000$ ;  $LastByteSent = 2.000$ ;  $LastByteAcked = 2.000$ ;  $MSS = 1.000$   
En ese mismo instante, el buffer de transmisión tiene 5.000 bytes pendientes de transmitir. Entonces, el sistema en cuestión transmitirá 5 segmentos de datos, de 1.000 bytes cada uno.
- F 18. Cuando vence un temporizador en un emisor TCP Reno durante la fase de evitación de congestión (congestion avoidance), el emisor divide por 2 el valor del umbral.
19. Cuando se reciben tres ACK duplicados en un emisor TCP Tahoe durante la fase de evitación de congestión (congestion avoidance), el emisor disminuye a 1 MSS el valor de la ventana de congestión.
- F 20. La entidad TCP que establece una conexión de forma pasiva no puede realizar el cierre activo de la misma.
21. Los segmentos TCP con indicador SYN activo pueden ser utilizados como ataque de denegación de servicio al destinatario
22. El protocolo UDP añade, al servicio ofrecido por IP, sólo la capacidad de multiplexación de puertos.

- 23.** Suponga que la ruta que siguen los segmentos entre dos sistemas finales TCP/IP no provoca pérdidas por congestión. Si en un punto de la ruta un segmento UDP sufre errores que afectan exclusivamente al campo de checksum, el segmento será descartado por el primer router de la ruta.
- 24.** Una de las posibles políticas que podría tener establecida un router para gestionar la congestión es, analizar el estado de sus colas y cuando detecte que están próximas a llenarse tirar uno o más paquetes de tráfico TCP de cada cola que vaya a llenarse.
- 25.** El rango de puertos del protocolo TCP es el mismo en un sistema operativo de 64 bits que en un sistema operativo de 32 bits.
- 26.** El complemento a 1 de la suma de los siguientes bytes: 01010011, 01010100 y 01110100 es 00011100
- 27.** La ventana de congestión impone una restricción sobre la velocidad a la que el emisor TCP puede enviar tráfico a la red.
- 28.** Se dice que TCP es "auto-temporizado" porque usa los paquetes de reconocimiento para temporizar sus incrementos del tamaño de la ventana de recepción.
- 29.** Sólo puede ocurrir un cierre simultáneo si ambas entidades TCP han establecido la conexión de forma simultánea (establecimiento activo en ambos extremos de la conexión).
- 30.** En TCP, el indicador RST permite solicitar una liberación abrupta de la conexión.
- F** **31.** En un instante de la operación del protocolo TCP, uno de los dos sistemas finales se encuentra en el estado que determinan las siguientes variables, donde todos los valores están medidos en bytes:  
LastByteAcked=9.999, LastByteSent= 17.999, CongWin = 8.000, Threshold=32.000, MSS=1.000  
Al recibir dos segmentos con números de ACK 11.000 y 12.000, el nuevo valor de CongWin será 32.000
- F** **32.** En un emisor TCP la velocidad de transmisión media es (suponer el tamaño expresado en bytes y el tiempo en segundos): (Tamaño de la ventana de recepción anunciada)/RTT Bytes/s
- 33.** Los ataques por inundación SYN se producen en implementaciones TCP que, en el lado del servidor, reservan recursos antes de haber completado el tercer y último paso del proceso de acuerdo de conexión. Se propone la solución llamada cookies SYN que la generación del número de secuencia inicial se hace en el servidor tanto en el momento de envío del SYNACK como en momento de recibir el ACK asociado.

- 34.** La entidad TCP de un cliente inicia la liberación de una conexión enviando al servidor un segmento con el bit FIN=1. Como respuesta a este segmento, la entidad TCP del servidor debe enviar un segmento con el bit ACK=1
- 35.** Cuando una entidad TCP recibe un ACK con número de secuencia menor que el borde inferior de la ventana, retransmite los datos entre el número de secuencia que llega y el borde inferior de la ventana.
- 36.** Si se ha establecido una conexión TCP, se puede deducir que necesariamente una de las dos entidades TCP involucradas ha realizado un establecimiento pasivo.
- 37.** El valor del temporizador usado en TCP debe ser dos o tres órdenes de magnitud mayor que el RTT para conseguir una rápida reacción ante la pérdida de segmentos.
- 38.** En el contexto de una sesión de comunicación entre una pareja de procesos, el proceso que inicia la comunicación es el cliente y el que espera a ser contactado para comenzar la sesión es el servidor.
- 39.** Los componentes que integran el núcleo de la red Internet permiten la comunicación entre dos sistemas terminales, y que a nivel de transporte resultan transparentes para estos últimos.
- 40.** En el arranque lento de TCP, la velocidad de transmisión se va incrementando de manera exponencial hasta que se produce la primera perdida de segmento.



Departamento de Ingeniería  
Telemática y Electrónica

# Computer Networks

## Internet Transport Layer

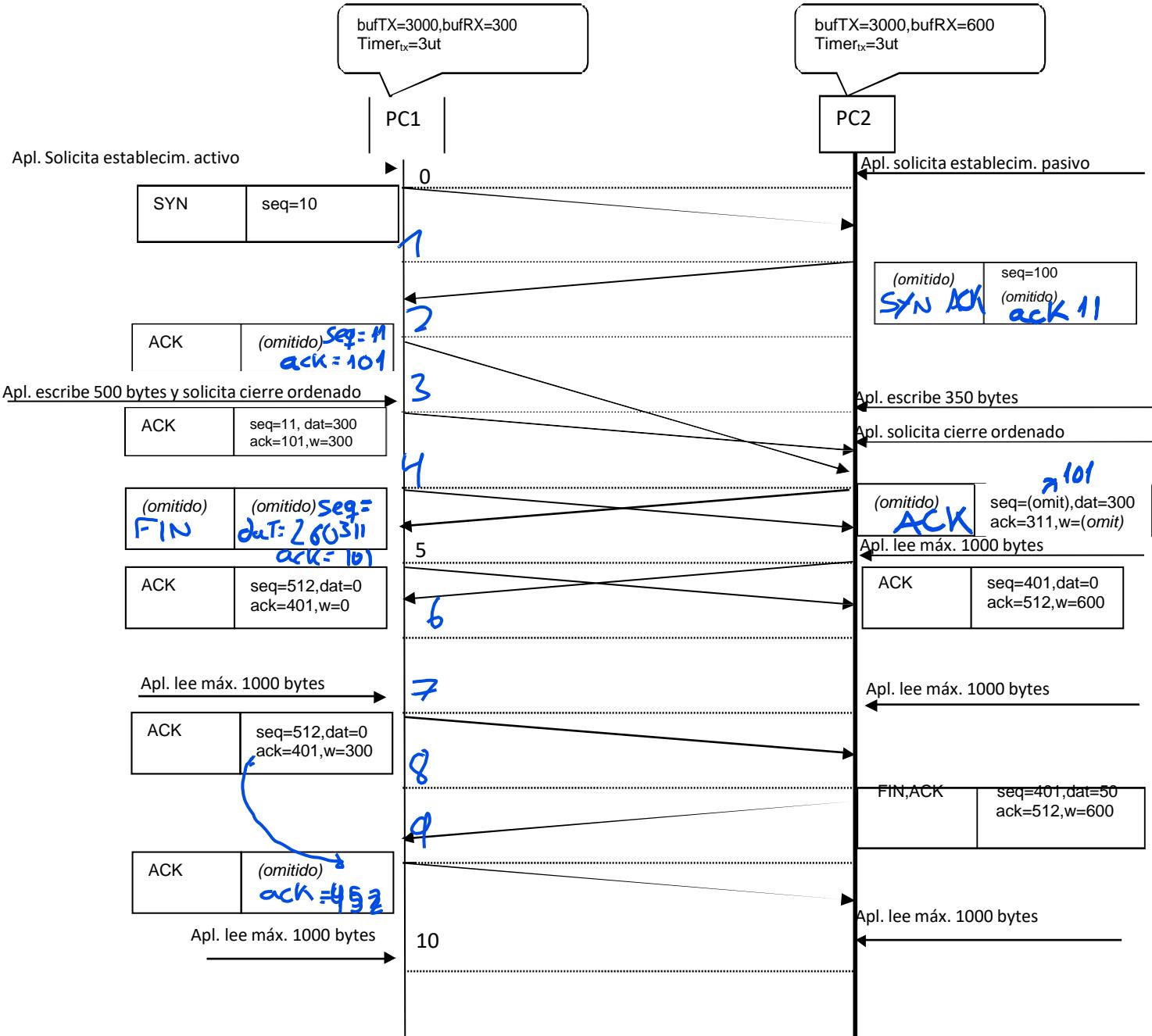
### TCP5-T(NP): Theory Exercises

Spring semester, 2022/2023



**Ejercicio 1**

Se dispone del siguiente intercambio de segmentos en una conexión TCP:



Responda a las siguientes cuestiones, siempre de acuerdo al funcionamiento definido para el protocolo TCP.

1. Rellene los campos indicados del segmento que PC2 transmite en la u.t. 1, sabiendo PC2 oferta a PC1 el máximo posible de bytes:

Flags=	seq=100 d= 0 <b>SYN ACK</b>
	ack= 11 w= 600

2. Rellene los campos indicados del segmento que PC1 transmite en la u.t. 2:

Flags=ACK	seq= 11 d= 0 ack= 10 w= 300
-----------	--------------------------------

3. Rellene los campos indicados del segmento que PC1 transmite en la u.t. 4:

Flags= FIN <b>ACK</b>	seq= 311 d= 200 ack= 101 w= 300
--------------------------	------------------------------------

4. Rellene los campos indicados del segmento que PC2 transmite en la u.t. 4:

Flags=	seq= 101 d=300 <b>ACK</b>
	ack=311 w= 300

5. Rellene los campos indicados del segmento que PC1 transmite en la u.t. 9:

Flags=ACK	seq= 512 d= 0 ack= 457 w= 250
-----------	----------------------------------

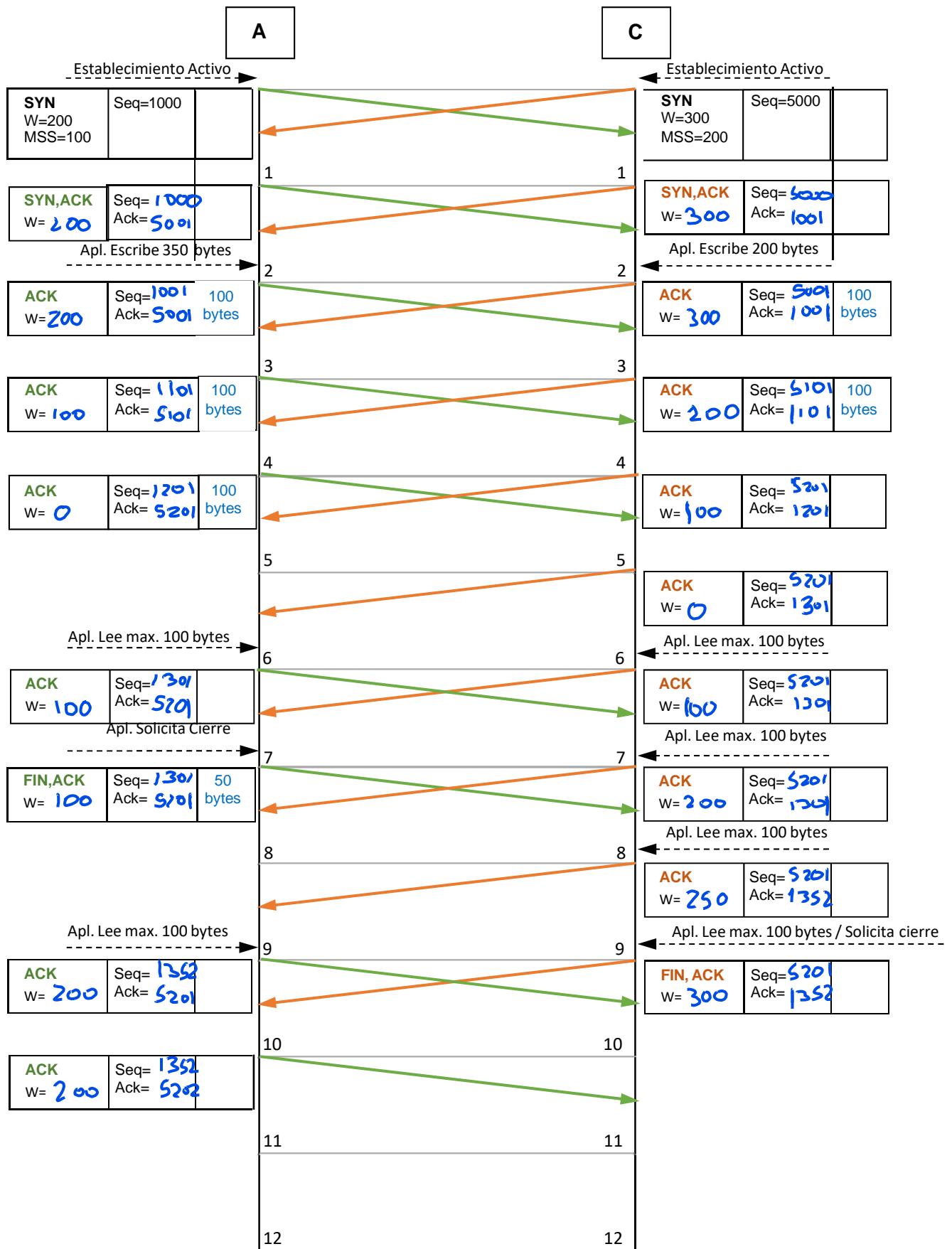
**Ejercicio 2**

Completar, en la plantilla anexa, los datos que faltan en el intercambio de segmentos en una conexión TCP.

$$MSS = \min\{100, 200\} = 100$$

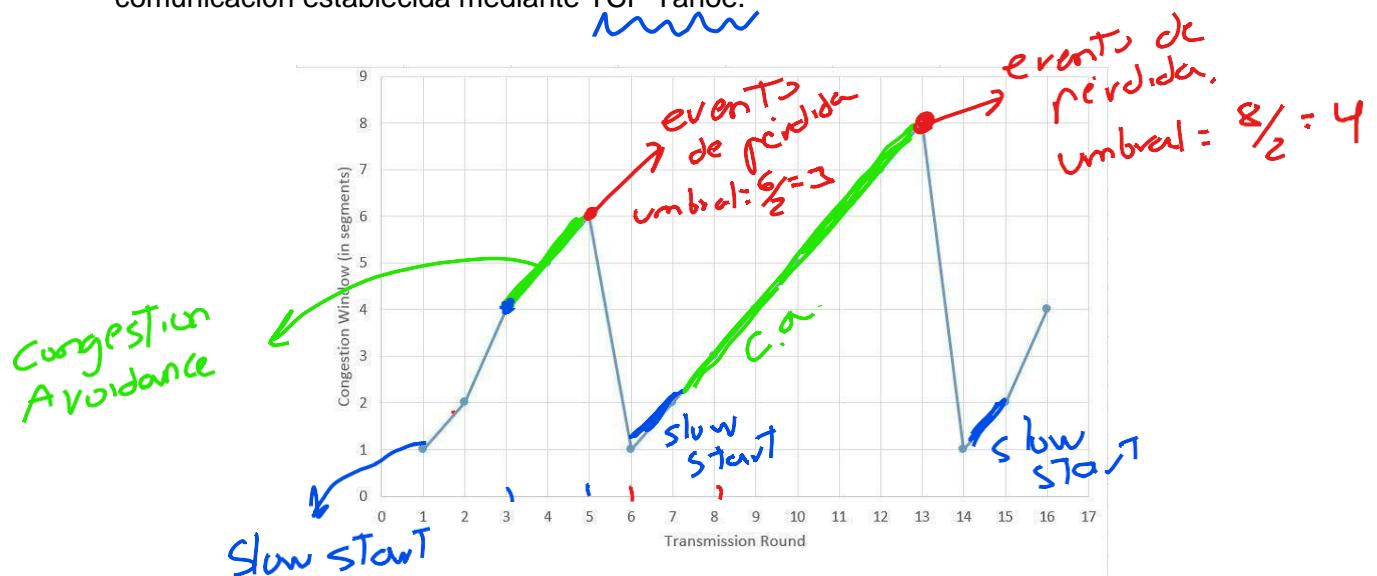
Internet Transport Layer

TCP5-T(NP): Theory Exercises



**Ejercicio 3**

La siguiente gráfica muestra la evolución de la ventana de congestión en una comunicación establecida mediante TCP Tahoe.



- A) ¿Qué tipo de eventos de pérdida se producen?

No se puede saber porque TCP Tahoe no distingue entre pérdidas leves y graves.

- B) ¿Qué mecanismo de control de congestión se está utilizando entre las rondas de transmisión 3 y 5? ¿Y entre la 6 y la 8?

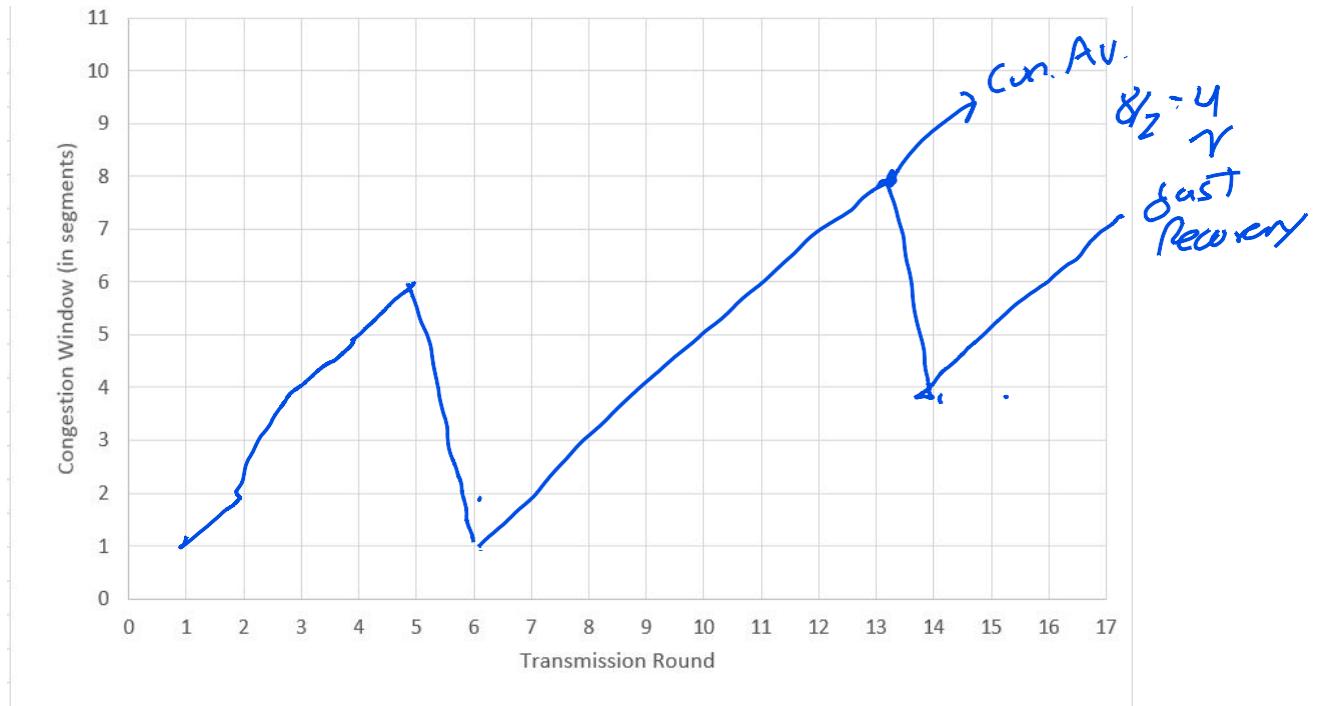
3-5 → Congestion Avoidance  
6-8 → Slow Start

- C) ¿Cuál es el valor inicial, en MSS, de la variable ssthresh? ¿Y después del primer evento de pérdida?

Umbral inicial: 4 MSS  
Umbral 1º evento:  $cwnd/2 = 3 MSS$

D) Independientemente de lo contestado en los apartados anteriores, dibuje en la siguiente plantilla la evolución de la ventana de congestión de la comunicación mostrada en la gráfica anterior, suponiendo que se usa **TCP Reno** y considerando que:

- El valor inicial de ssthresh es el mismo.
- Los eventos de pérdida se producen con los mismos valores de ventana de congestión.
- El primer evento de pérdida se produce por timeout y el segundo por ACKs duplicados.



## **Ejercicio 4**

Do the exercises P1, P2 and P25 from Chapter 3, proposed in Computer Networking, Kurose & Ross (Fifth Edition).

## **Ejercicio 5**

Show all the TCP segments sent and received between the host bsdi, port number 1025, where the Telnet client is installed and the host srv4, port number 23, where the Telnet server is installed, in the following situation:

- the Telnet client (host bsdi) sends a date request command: "date"
- the Telnet server (host srv4) replies with the date: "Sat Feb 6 07:52:17 MST 1993"
- the Telnet server (host srv4) sends the system prompt: "srv4%"

For answering this exercise, we have to suppose that:

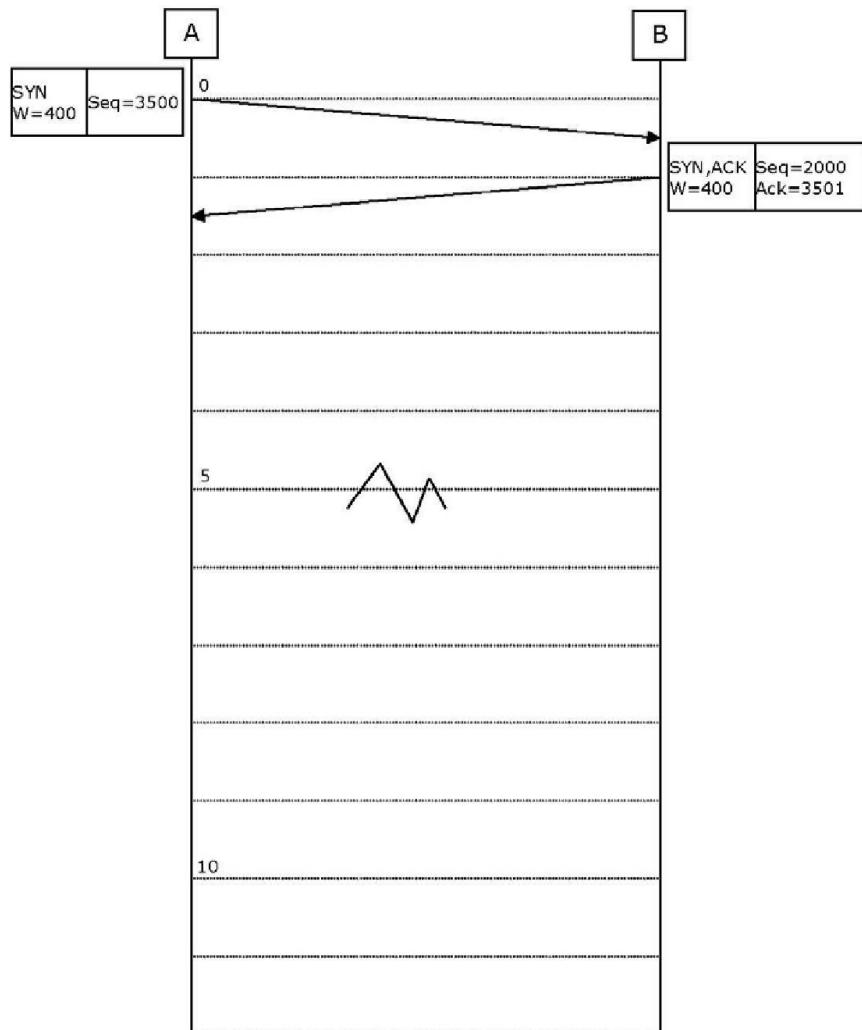
- A TCP connection is established between the telnet application in the host bsdi (telnet client) and the host srv4 (telnet server)
- The telnet application generates a TCP data segment for every character typed by the client. The server replies to the client with an echo of every character previously sent by the client. To summarize, for every character typed by the client, the following TCP segments are created:
  1. A segment created by the client with the character typed.
  2. A segment created by the server that acknowledge the reception of the character. Its content is an echo of the character.
  3. A segment created by the client that acknowledge the reception of the echo.

## **Ejercicio 6**

The host A wants to send 400 bytes of data to host B and the host B wants to send 200 bytes of data to the host A.

You have to finish the following diagram that shows the segment interchange that takes place at the TCP layer.

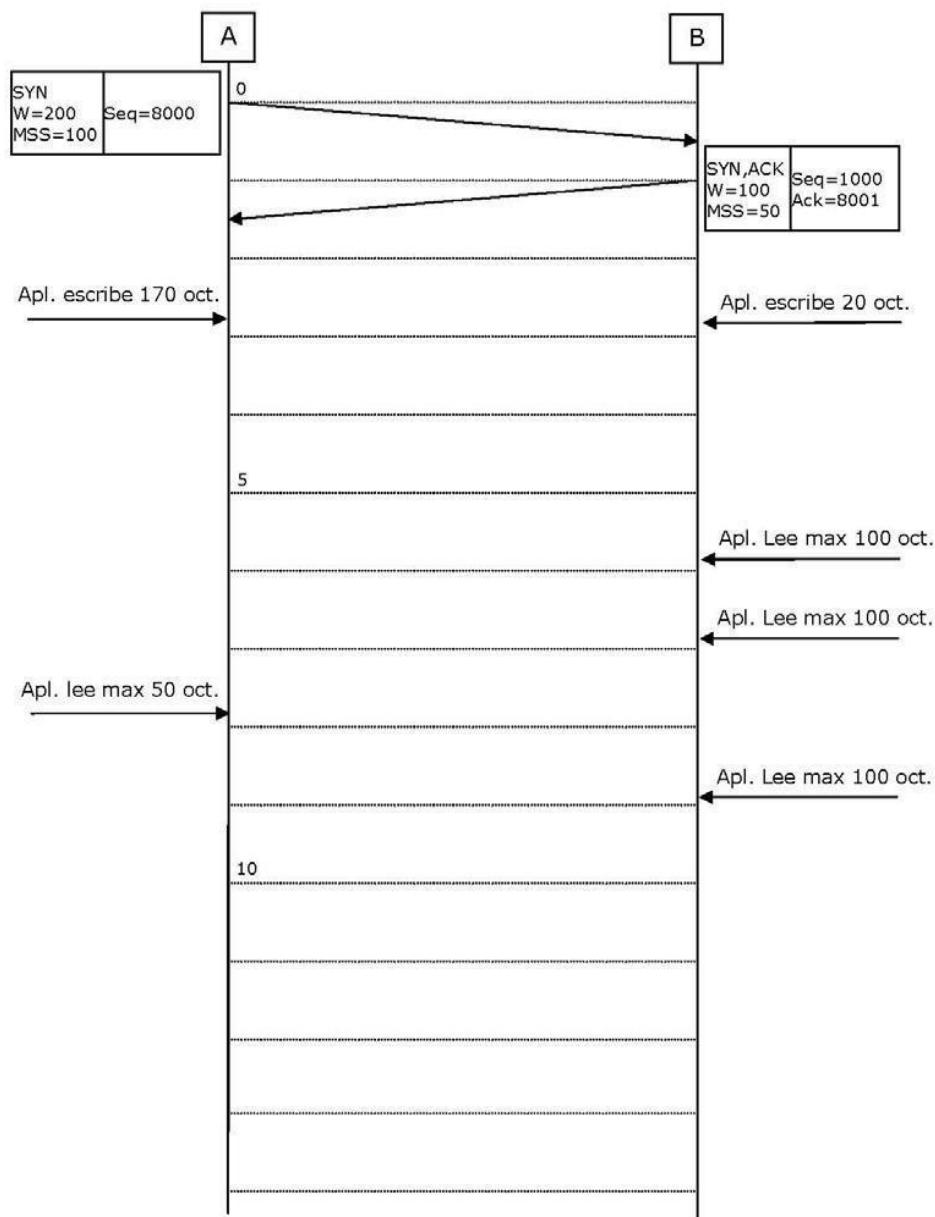
- All packets are always sent at the beginning of a time unit. ("dashed line" means a time unit).
- The transmission delay of all packets (from the sender to the receiver) is always constant equal to 0.5 units of time.
- Both TCP layers send each other segments with 200 bytes of data as soon as they are ready.
- An acknowledge packet is always sent after receiving a data segment.
- When the host A has sent all the data bytes and has received all acknowledges, it finishes the connection.
- Any host modify the window size established at the beginning of the connection.
- All packets sent at the time unit number 5 are lost due to network problems.
- The retransmission delay for both hosts is 3 time units.



## **Ejercicio 7**

Once the connection is established, the host A wants to send 170 bytes of data to the host B and the host B wants to send 20 bytes of data to the host A. You have to finish the following diagram that shows the segment interchange that takes place at the TCP layer.

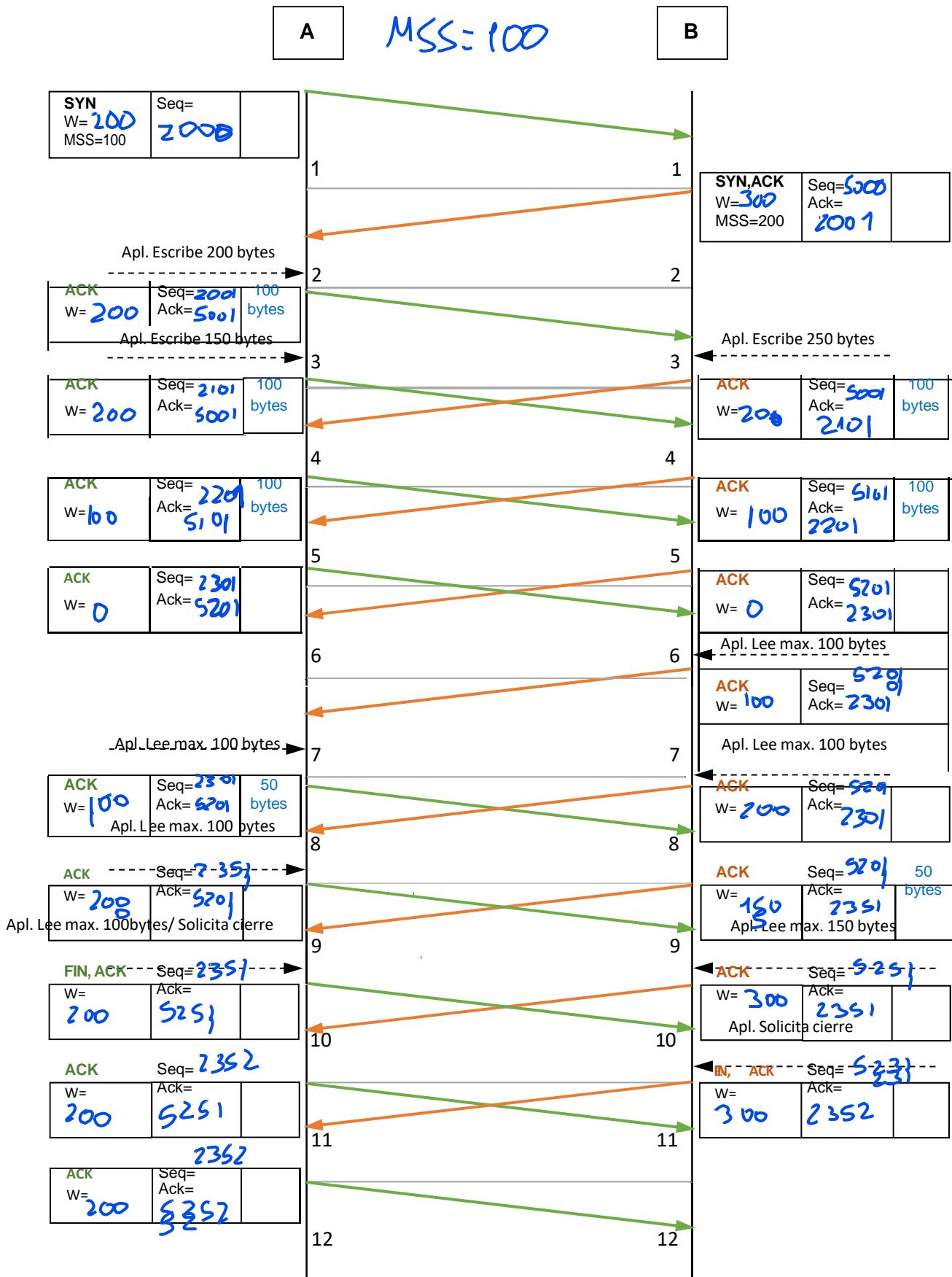
- All packets are always sent at the beginning of a time unit. ("dashed line" means a time unit).
- The transmission delay of all packets (from the sender to the receiver) is always constant equal to 0.5 units of time.
- An acknowledge packet is always sent after receiving a data segment.
- When the host A has received acknowledge of its last byte sent, it finishes the connection.
- The last data byte received at the host B comes with the finished mark, so the host B starts the closing sequence.
- The size of host B reception buffer is 100 bytes for each connection.
- The TCP applications in both hosts only read (if they are available) or write data bytes at the times marked with arrows.



### **Ejercicio 8**

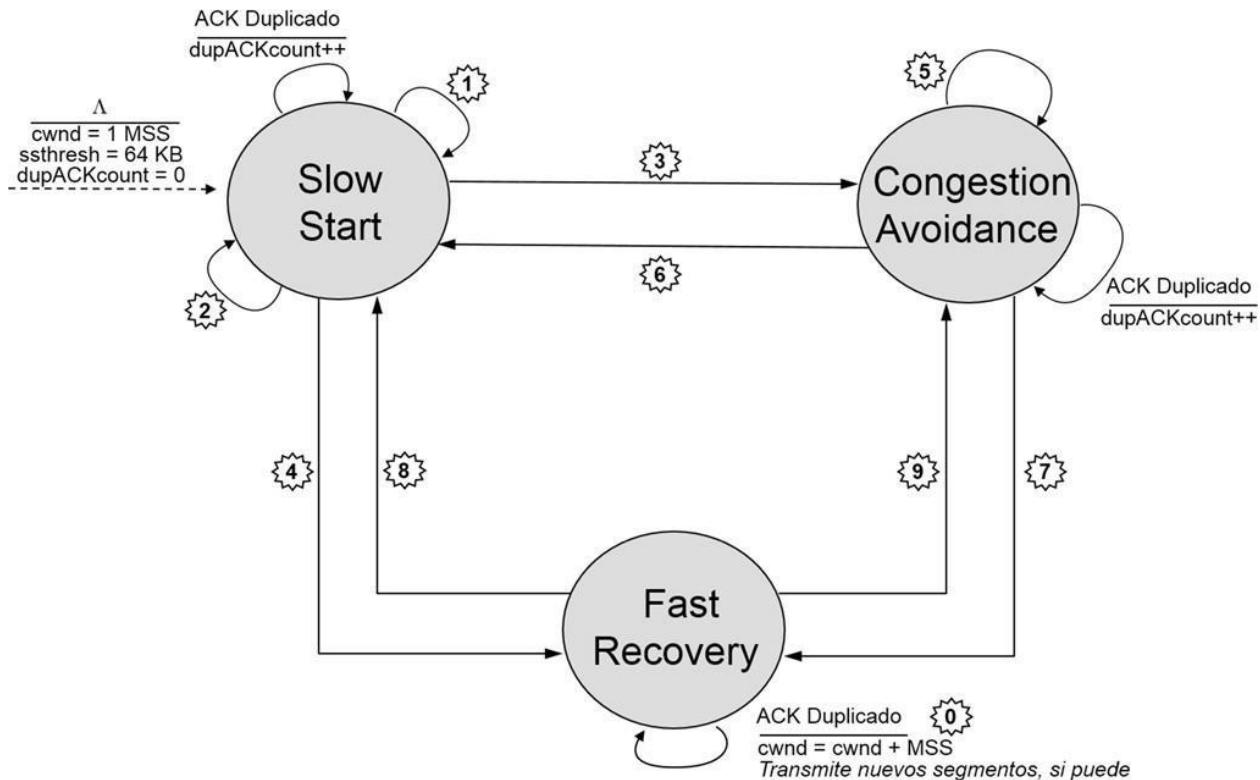
Completar los datos que faltan en el siguiente intercambio de segmentos en una conexión TCP, teniendo en cuenta que:

- Los números de secuencia inicial de cada máquina son  $ISN_A=2000$  y  $ISN_B=5000$ .
- Las máquinas A y C tienen 200 bytes y 300 bytes de buffer de recepción respectivamente, para cada conexión TCP.
- Los ACK reconocen todos los datos pendientes de confirmar.



**Ejercicio 9**

La siguiente figura muestra el diagrama de estados correspondiente al control de congestión en TCP. Complete, en la tabla adjunta, la información correspondiente a los datos que faltan.



Evento	Descripción	Variables	Acción a tomar
0	ACK Duplicado	cwnd=cwnd+MSS	Transmite nuevos segmentos, si puede
1			
2	Timeout		
3		----	----
4			
5			Transmite nuevos segmentos, si puede
6			Retransmite el segmento con menor número de secuencia
7			Retransmite el segmento perdido
8			
9	ACK Nuevo		----

**Ejercicio 10**

- a) Considera una conexión TCP entre los sistemas A y B donde se han fijado como números iniciales de secuencia los valores x (de A a B) e y (de B a A). Si el primer segmento de datos recibido por A contiene 1.000 bytes y tiene número de secuencia z, con  $z > y+1$ , ¿cómo reaccionará A ante la recepción de este segmento?.

**Transmitiendo un segmento con ACK  $y+1$  y guardando los datos en el buffer de recepción.**

- b) En un instante determinado de la operación del protocolo TCP, uno de los dos sistemas finales se encuentra en el estado que determinan las siguientes variables, donde todos los valores están medidos en bytes:

Caso 1: RcvWindow = 5.000; CongWin = 3.000; LastByteSent = 2.000; LastByteAcked = 2.000; MSS = 1.000

Caso 2: RcvWindow = 3.000; CongWin = 5.000; LastByteSent = 2.000; LastByteAcked = 2.000; MSS = 1.000

Caso 3: RcvWindow = 5.000; CongWin = 5.000; LastByteSent = 2.000; LastByteAcked = 2.000; MSS = 2.000

Caso 4: RcvWindow = 5.000; CongWin = 5.000; LastByteSent = 2.000; LastByteAcked = 1.000; MSS = 2.000

En ese mismo instante, el buffer de transmisión tiene 5.000 bytes pendientes de transmitir. Indicar en cada uno de los casos, cuántos segmentos y de cuántos bytes cada uno, transmitirá este sistema

*Caso 1:  $\text{LastByteSent} - \text{LastByteAcked} = 0$  ( $\times$ )  
 $\min\{\text{Cwnd}, \text{rwnd}\} = 3000$  ( $y$ )*

$$y - x = 3000 / \text{MSS} = 3 \text{ segmentos de 1000 cada uno.}$$

Caso 1: 3 segmentos de 1000 bytes cada uno.  
Caso 2: 3 segmentos de 1000 bytes cada uno.  
Caso 3: 2 segmentos de 2000 bytes cada uno y un segmento de 1000 bytes.  
Caso 4: 2 segmentos de 2000 bytes cada uno.

- c) En un instante determinado de la operación del protocolo TCP, uno de los dos sistemas finales se encuentra en el estado que determinan las siguientes variables, donde todos los valores están medidos en bytes:

LastByteAcked=9.999, LastByteSent= 17.999, CongWin = 8.000, Threshold=32.000, MSS=1.000

Al recibir dos segmentos con números de ACK 11.000 y 12.000, ¿cuál será el nuevo valor de la ventana de congestión?

$$\text{Cwnd} = \text{Cwnd} + \text{Nº ACK (No Duplicados)} \cdot \text{MSS} = 8000 + 2 \cdot 1000 = 10000 \text{ bytes}$$

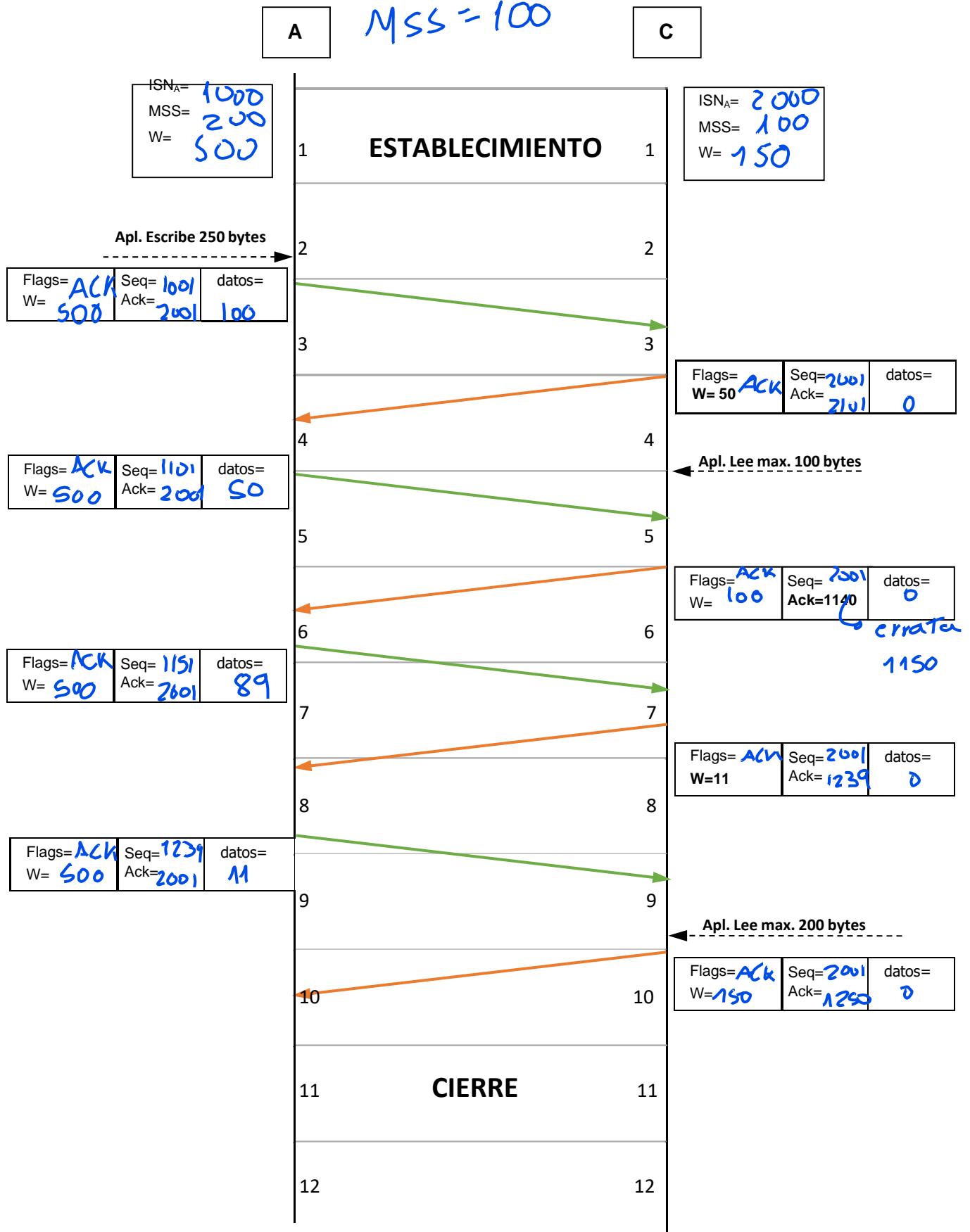
- d) Si después de un evento de pérdida el valor de la ventana de congestión fuese de 1MSS, ¿qué tipo de TCP se estaría usando?

*No se puede saber ya que tanto TCP Reno como TCP Tahoe manejan igual los errores graves (Timeout), por lo que nos tendrían que decir de qué tipo de error se trata.*

## Ejercicio 11

Completar los datos que faltan en el siguiente intercambio de segmentos en una conexión TCP, teniendo en cuenta que:

- Los segmentos se transmiten siempre al inicio de una unidad de tiempo (u.t.) y tardan 0,5 u.t. en alcanzar su destino. Considerar 1 u.t = 1 seg.
- Los números de secuencia inicial usados en esta conexión son 1000 en la máquina A y 2000 en la máquina C.
- La cantidad máxima de datos que se pueden enviar en un segmento para que no se produzca fragmentación en los niveles inferiores son: 200 bytes en la máquina A y 100 bytes en la máquina C.  $MSS=100$
- La máquina A dispone de un búffer de recepción para cada conexión de 500 bytes.  $W_A=500$
- Si no se indica lo contrario, los ACK reconocen todos los datos pendientes de confirmar.
- Las aplicaciones de ambas máquinas sólo leen (si están disponibles) o escriben bytes en los momentos indicados en el diagrama.

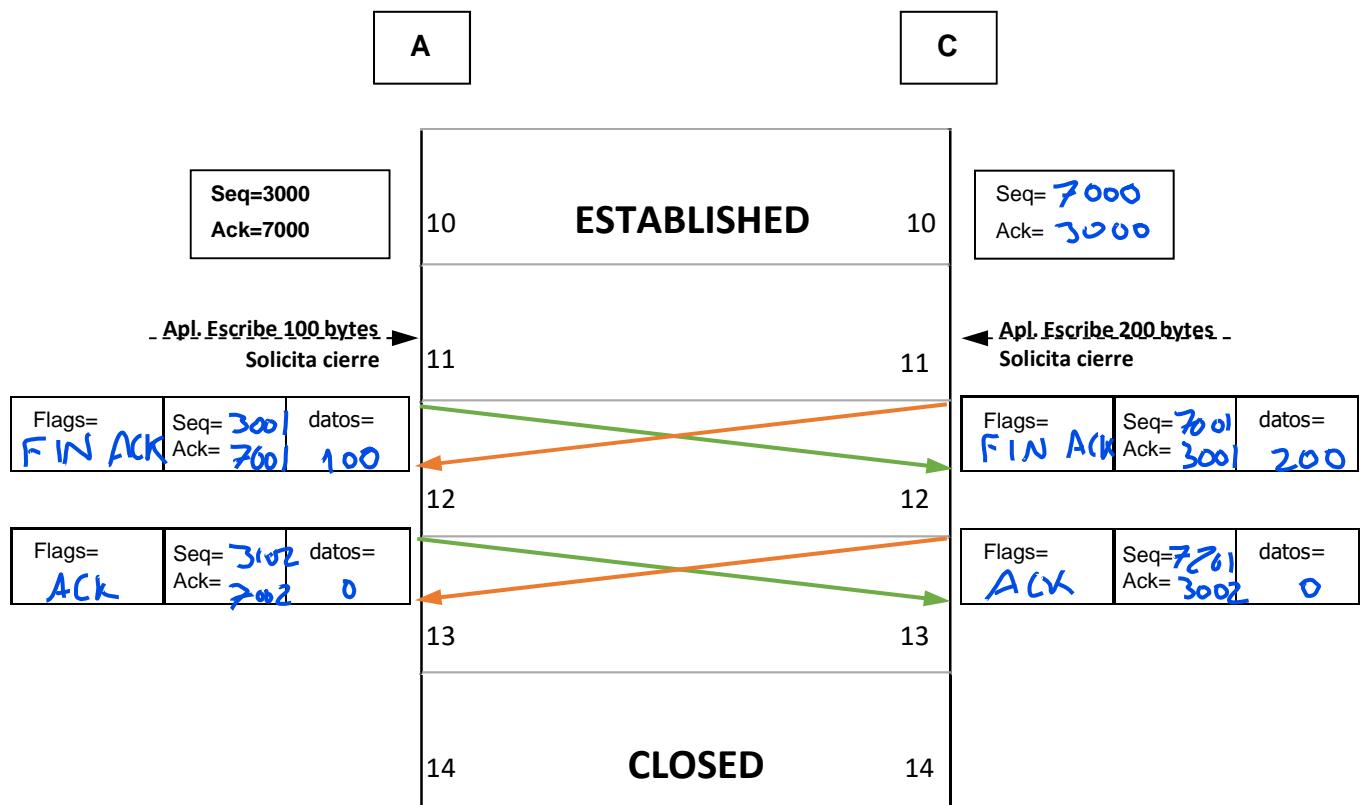


**Ejercicio 12**

Completar los datos que faltan en el siguiente intercambio de segmentos en el cierre de una conexión TCP, teniendo en cuenta que:

- El MSS negociado en la fase de establecimiento fue de 300 bytes.
- Todos los datos enviados hasta la unidad de tiempo 10 se encuentran confirmados.
- Los buffer de recepción de ambas máquinas son mayores que 1MSS.

$$MSS = 300$$



**Ejercicio 13**

- A) ¿Qué relación existe entre los valores de MTU y MSS? ¿Qué objetivo se persigue con esa relación? ¿En qué tipo de segmentos TCP se incluye el valor MSS y en qué campo de la cabecera se hace?

MSS=MTU-Long Cabecera IP – Long Cabecera TCP, que en la mayoría de los casos será MSS-MTU-40 (medido en bytes).  
 El objetivo es dimensionar el MSS de tal forma que se evite la fragmentación en el nivel de acceso a red.  
 El MSS se negocia durante la fase de establecimiento de la conexión TCP, por lo que su valor se incluye en los segmentos TCP con el flag SYN activado.  
 En estos segmentos, el valor del MSS se incluye en el campo “Opciones” de la cabecera TCP

- B) ¿Bajo qué condiciones se puede producir un cierre abrupto (Reset)?

Cuando la entidad TPC receptora no puede establecer la conexión (por ejemplo, porque no hay ningún proceso escuchando en el puerto solicitado, es decir, no se ha realizado un “passive open” sobre ese puerto), o bien cuando el segmento SYN recibido tiene un error.

- C) ¿Qué implicaciones tendría poner el valor del temporizador de retransmisión TCP demasiado corto o demasiado largo?.

Si fuera demasiado corto podría vencer prematuramente dando lugar a retransmisiones innecesarias.  
 Si fuera demasiado largo se tardaría demasiado en reaccionar ante la pérdida real de un segmento.

- D) El algoritmo AIMD usado en el control de congestión en TCP se dice que es justo o que hace justicia (TCP Fairness). ¿Por qué?

Porque asegura el mismo ancho de banda para todas las conexiones TCP que comparten un mismo enlace, es decir, si K conexiones comparten un enlace de capacidad R, la velocidad media de transmisión de cada una de ellas será R/K.

- E) ¿Qué relación existe entre los parámetros Cwnd, Rwnd, LastByteSent y LastByteAcked?

LastByteSent-LastByteAcked<=min(Cwnd, Rwnd)

### Ejercicio 14

- A) ¿Qué protocolo de transporte suelen usar las aplicaciones multimedia?. Razonar la respuesta.

Las aplicaciones multimedia suelen usar UDP como protocolo de transporte debido a que este tipo de aplicaciones son sensibles al retardo pero no lo son tanto a la pérdida de datagramas.  
Además, necesitan una velocidad de transmisión constante, cosa que TCP nos les puede asegurar debido al control de congestión que implementa.

- B) ¿Qué mecanismos de control de congestión implementa TCP Reno?.

Arranque lento (Slow Start)  
Evitación de congestión (Congestion Avoidance)  
Recuperación rápida (Fast Recovery)

- C) ¿En qué momento la ventana de congestión pasa de crecer exponencialmente a crecer linealmente?.

Cuando alcanza el valor de la variable ssthresh (slow start threshold), cuyo valor es la mitad de lo que valía la ventana de congestión cuando se produjo el último evento de pérdida.

- D) ¿Cuánto se incrementa la ventana de congestión en la fase de crecimiento exponencial cada vez que se recibe un ACK duplicado?.

No se incrementa, simplemente se contabiliza el ACK duplicado recibido

- E) ¿Cuánto se incrementa la ventana de congestión en la fase de crecimiento lineal cada vez que se recibe un ACK nuevo?.

Se incrementa según la fórmula  
 $cwnd = cwnd + MSS * (MSS/cwnd)$

**Ejercicio 15**

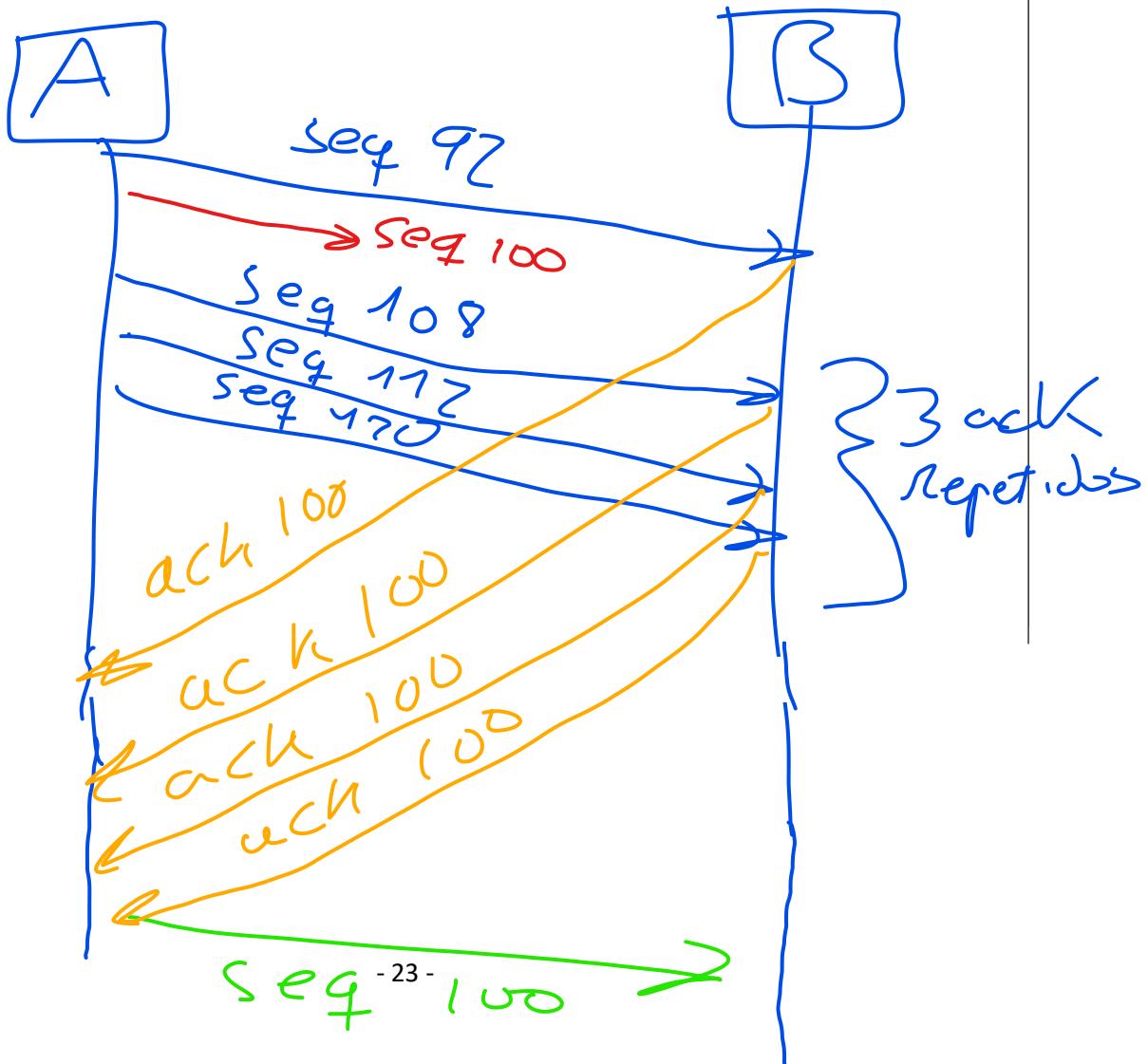
- A) Indicar bajo qué condiciones se arranca el temporizador de retransmisión de TCP.

El temporizador de retransmisión de TCP se arranca cuando...

- 1.-...se envía un nuevo segmento TCP con datos y el temporizador no se encuentra arrancado previamente.
- 2.-...después de haber expirado el temporizador (evento de timeout) se retransmite el segmento TCP con datos con número de secuencia más bajo.
- 3.-...después de recibir un ACK que confirma datos nuevos (previamente sin confirmar) y todavía quedan datos enviados sin confirmar.

- B) Explicar, mediante un ejemplo que lo ilustre, en qué consiste la retransmisión rápida (Fast Retransmit) en TCP

En el caso de que se reciban 3 ACKs duplicados, el emisor TCP realiza una retransmisión rápida, reenviando el segmento que falta antes de que expire el temporizador.



C) Completar en la siguiente tabla las acciones del receptor TCP para la generación de mensajes ACK según la RFC 1122 y la RFC 2581.

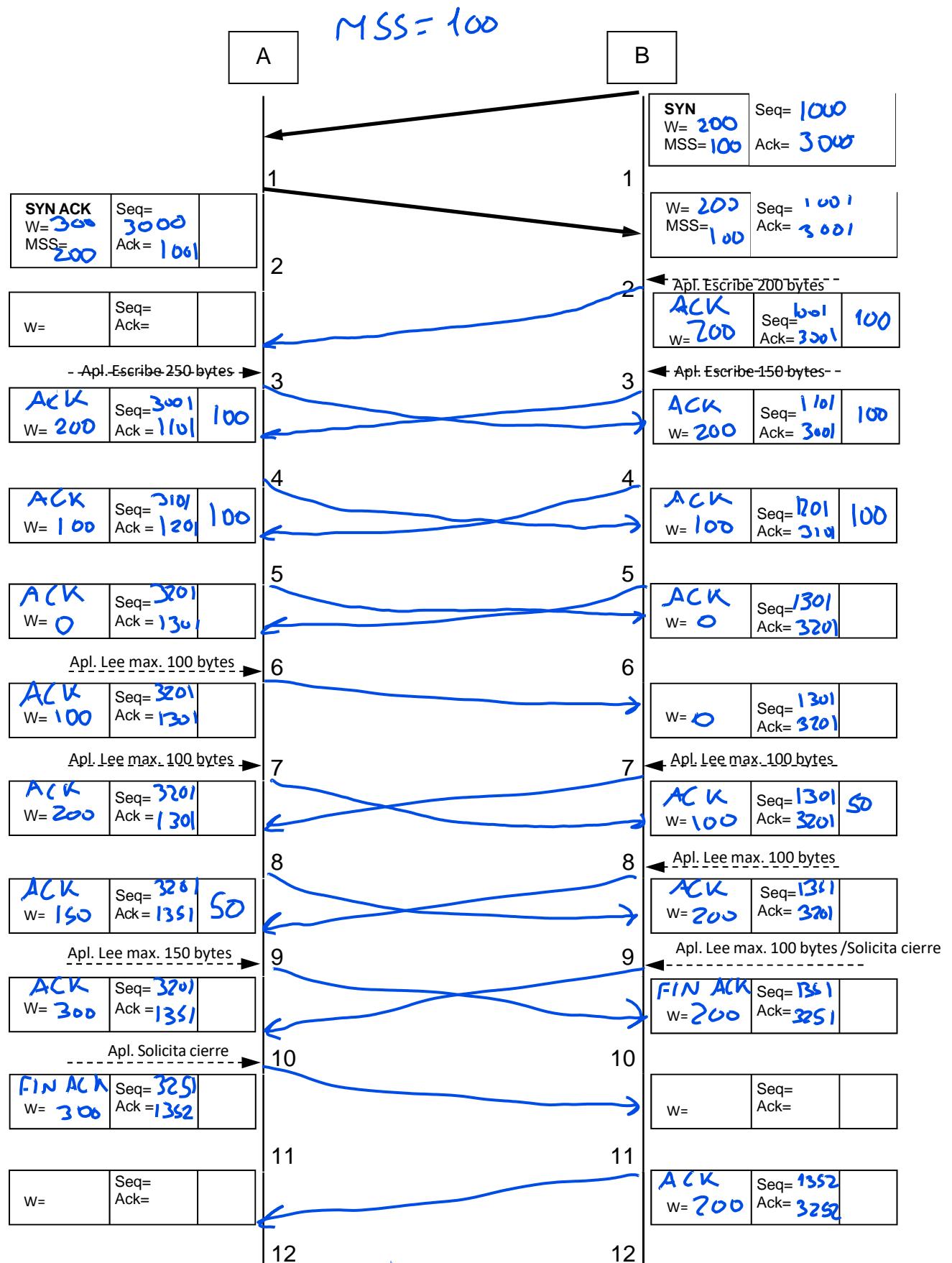
Suceso	Acción del receptor TCP
Llegada de un segmento en orden con el número de secuencia esperado. Todos los datos hasta el número de secuencia esperado ya han sido reconocidos.	<b>ACK retardado. Esperar hasta 500 milisegundos la llegada de otro segmento en orden. Si el siguiente segmento en orden no llega en ese intervalo, enviar un ACK.</b>
Llegada de un segmento en orden con el número de secuencia esperado. Hay otro segmento en orden esperando la transmisión de un ACK.	<b>Enviar inmediatamente un ACK acumulativo, reconociendo ambos segmentos ordenados.</b>
Llegada de un segmento desordenado con un número de secuencia más alto que el esperado. Se detecta un hueco.	<b>Enviar inmediatamente un ACK duplicado, indicando el número de secuencia del siguiente byte esperado (que es el límite inferior del hueco).</b>
Llegada de un segmento que completa parcial o completamente el hueco existente en los datos recibidos.	<b>Enviar inmediatamente un ACK, suponiendo que el segmento comienza en el límite inferior del hueco.</b>

### Ejercicio 16

Dos aplicaciones, que se ejecutan en máquinas distintas, desean enviarse información entre ellas usando TCP como protocolo de nivel de transporte. La aplicación que se ejecuta en la máquina **B** desea enviar 350 bytes de información a la máquina **A** y la aplicación que se ejecuta en la máquina **A** desea enviar 250 bytes de información a la aplicación que se ejecuta en la máquina **B**.

Utilice la plantilla para representar el intercambio de segmentos que se produce en el nivel TCP, teniendo en cuenta que:

- Los segmentos se transmiten siempre al inicio de una unidad de tiempo (u.t.) y tardan 0,5 u.t. en alcanzar su destino.
- Los números de secuencia inicial de cada máquina son  $ISN_A=3000$  y  $ISN_B=1000$ .
- Las máquinas A y B tienen 300 bytes y 200 bytes de buffer de recepción respectivamente, para cada conexión TCP.
- La cantidad máxima de datos que se pueden enviar en un segmento para que no se produzca fragmentación en los niveles inferiores son: 200 bytes en la máquina A y 100 bytes en la máquina B.
- Los niveles TCP de ambas máquinas envían un reconocimiento inmediatamente después de recibir un segmento con datos (reconociendo todos los datos que estén pendientes de confirmar), o bien cuando se produce algún cambio en su ventana de recepción.
- Ante un cierre de ventana, no se envían segmentos sonda.
- Las aplicaciones que se ejecutan en las máquinas A y B inicián el cierre de la conexión inmediatamente después de que le sea confirmado su último byte de datos.
- Las aplicaciones de ambas máquinas sólo leen (si están disponibles) o escriben bytes en los momentos indicados en la plantilla.



**Ejercicio 17**

- A) En un instante determinado de la operación del protocolo TCP, uno de los dos sistemas finales se encuentra en el estado que determinan las siguientes variables, donde todos los valores están medidos en bytes:

Caso 1: rwnd = 4000; cwnd = 2000; LastByteSent = 1000; LastByteAcked = 1000; MSS = 500  
 Caso 2: rwnd = 2000; cwnd = 4000; LastByteSent = 1000; LastByteAcked = 1000; MSS = 500  
 Caso 3: rwnd = 4000; cwnd = 4000; LastByteSent = 1000; LastByteAcked = 1000; MSS = 500  
 Caso 4: rwnd = 2000; cwnd = 2000; LastByteSent = 1000; LastByteAcked = 500 ; MSS = 500  
 Caso 5: rwnd = 4000; cwnd = 4000; LastByteSent = 1000; LastByteAcked = 0 ; MSS = 800

En ese mismo instante, el buffer de transmisión tiene 3000 bytes pendientes de transmitir. Indicar en cada uno de los casos, cuántos segmentos y de cuántos bytes cada uno, transmitirá este sistema.

*Caso 1:*

1. DATOS  $TX = \min\{4000, 2000\} = 2000$
2.  $1000 - 1000 = 0$
3.  $2000 / 500 = 4$  segmentos de 500.

*Caso 2: 4 segmentos de 500 bytes cada uno.*

*Caso 3: 6 segmentos de 500 bytes cada uno.*

*Caso 4: 3 segmentos de 500 bytes cada uno.*

*Caso 5: 3 segmentos de 800 bytes cada uno y uno de 600 bytes.*

- B) Indicar los valores que toma la ventana de congestión en función del evento de pérdida producido y del tipo de TCP usado.

*Evento de Pérdida: Timeout  
 cwnd = 1MSS, independientemente del tipo de TCP usado  
 Evento de Pérdida: 3 ACKs duplicados  
 cwnd = 1MSS, si se usa TCP Tahoe  
 cwnd = ssthresh, si se usa TCP Reno*

- C) ¿En qué consiste el “síndrome de la ventana tonta” (Silly Window Syndrome)?

Después de un cierre de ventana en el receptor, los reconocimientos enviados por éste sólo informan de una pequeña cantidad de espacio disponible en su buffer, lo que originará que la parte transmisora envíe pequeñas cantidades de datos, que seguramente volverán a cerrar la ventana de nuevo

- D) ¿En qué consiste el “ataque por inundación de SYN” (The SYN flood attack)?

El atacante envía un gran número de segmentos SYN, pero al recibir los correspondientes segmentos SYN-ACK por parte del servidor, no termina de completar el tercer paso del proceso de conexión (Three-way handshake). Esto hace que los recursos del servidor se agoten, al haberlos asignado a conexiones no se terminarán por completar. Cuando esto ocurre, el servicio se denegará a los clientes legítimos.

En relación a la Notificación Explícita de Congestión (ECN),

- E) ¿Qué debería hacer el nivel de transporte TCP de un sistema final si recibiera datagramas IP con los bits ECN=11?

Mandaría un segmento TCP con el flag ECE=1, con el fin de avisar al nivel de transporte TCP del sistema final remoto que se ha detectado congestión en la red.

- F) ¿Qué debería hacer el nivel de transporte TCP de un sistema final cuando se le indica que la red está congestionada?

Debería disminuir el valor de su ventana de congestión e indicar este hecho al nivel de transporte TCP del sistema final remoto, activando para ello el flag CWR.

### **Ejercicio 18**

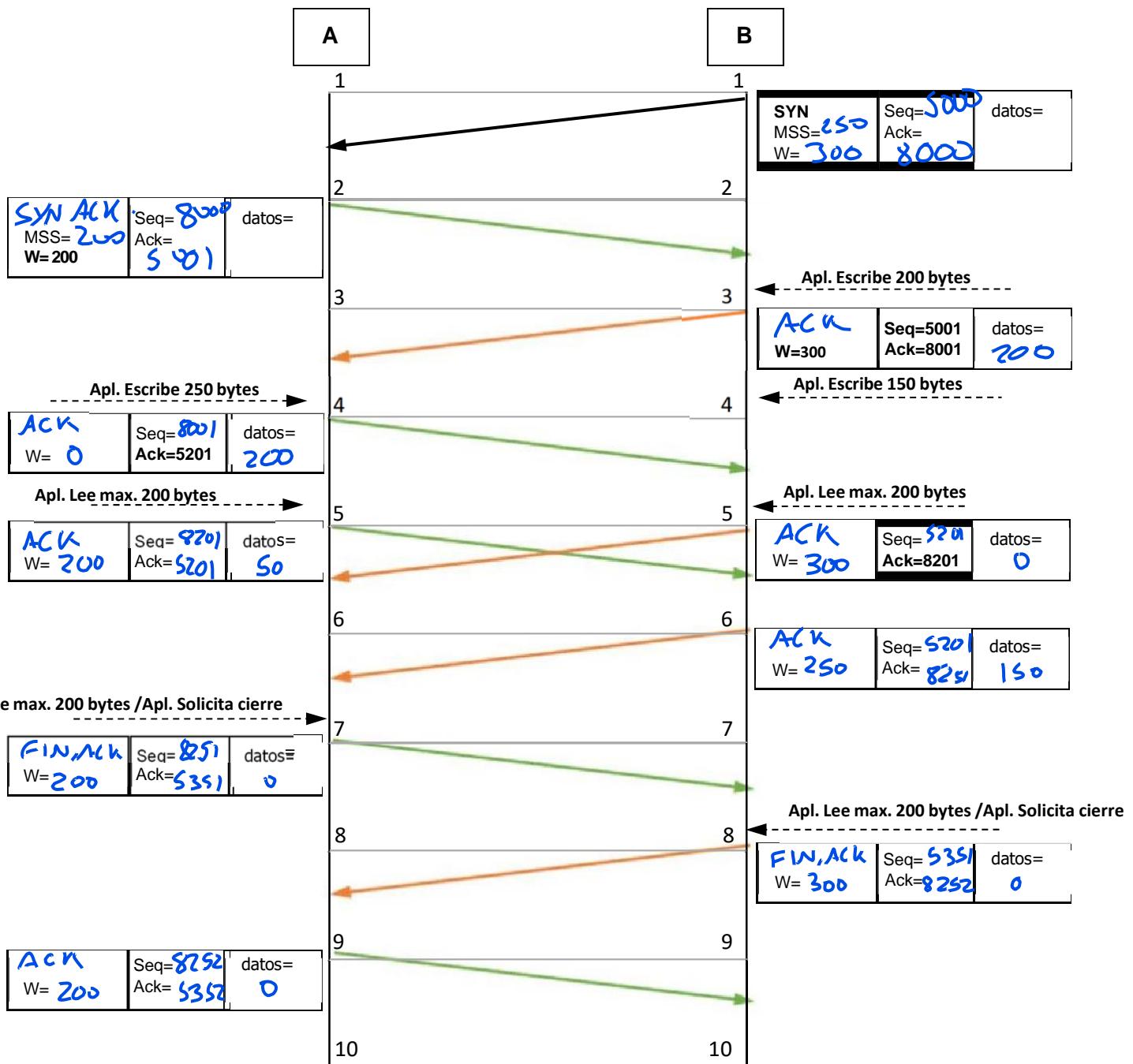
Dos aplicaciones, que se ejecutan en máquinas distintas, desean enviarse información entre ellas usando TCP como protocolo de nivel de transporte.

La plantilla adjunta muestra el intercambio de tráfico producido entre ellas cuando la aplicación que se ejecuta en la máquina B envía 350 bytes de información a la máquina A y la aplicación que se ejecuta en la máquina A envía 250 bytes de información a la aplicación que se ejecuta en la máquina B.

Teniendo en cuenta los siguientes datos:

- Los segmentos se transmiten siempre al inicio de una unidad de tiempo (u.t.) y tardan 0,5 u.t. en alcanzar su destino.
- La cantidad máxima de datos que se pueden enviar en un segmento para que no se produzca fragmentación en los niveles inferiores son: 200 bytes en la máquina A y 250 bytes en la máquina B. **MSS = 200**
- Los niveles TCP de ambas máquinas envían un reconocimiento inmediatamente después de recibir un segmento con datos (reconociendo todos los datos que estén pendientes de confirmar), o bien cuando se produce algún cambio en su ventana de recepción.
- Ante un cierre de ventana, no se envían segmentos sonda.
- Las aplicaciones que se ejecutan en las máquinas A y B inician el cierre de la conexión inmediatamente después de que le sea confirmado su último byte de datos.
- Las aplicaciones de ambas máquinas sólo leen (si están disponibles) o escriben bytes en los momentos indicados en la plantilla.

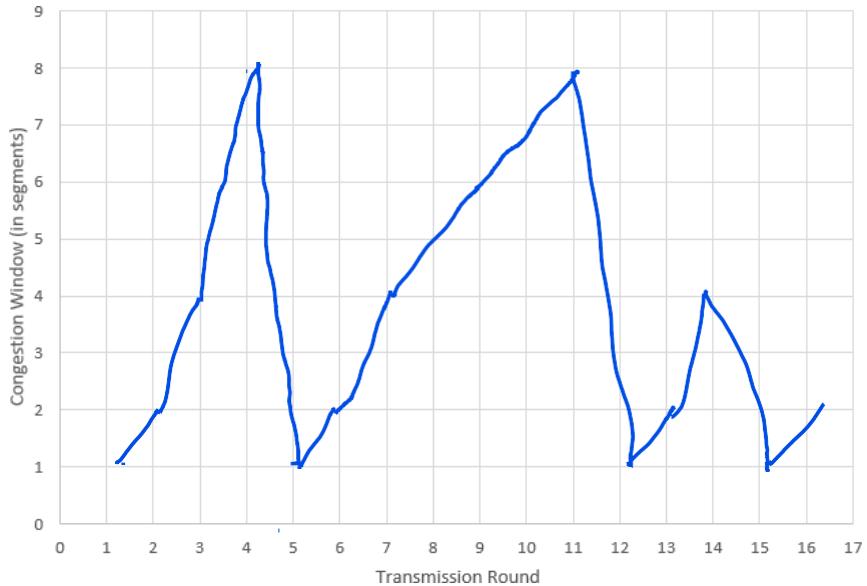
Se pide, completar sobre la plantilla los datos que faltan.





- B) Dibujar cómo habría evolucionado la ventana de congestión si en tipo de TCP hubiera sido Tahoe.

Nota: Suponer que los eventos de pérdida se producen en los mismos instantes y son del mismo tipo que los del apartado anterior.



- C) Indicar, para TCP Reno y TCP Tahoe, cuánto se incrementa la ventana de congestión en la fase de "Congestion Avoidance" cada vez que se recibe un ACK nuevo.

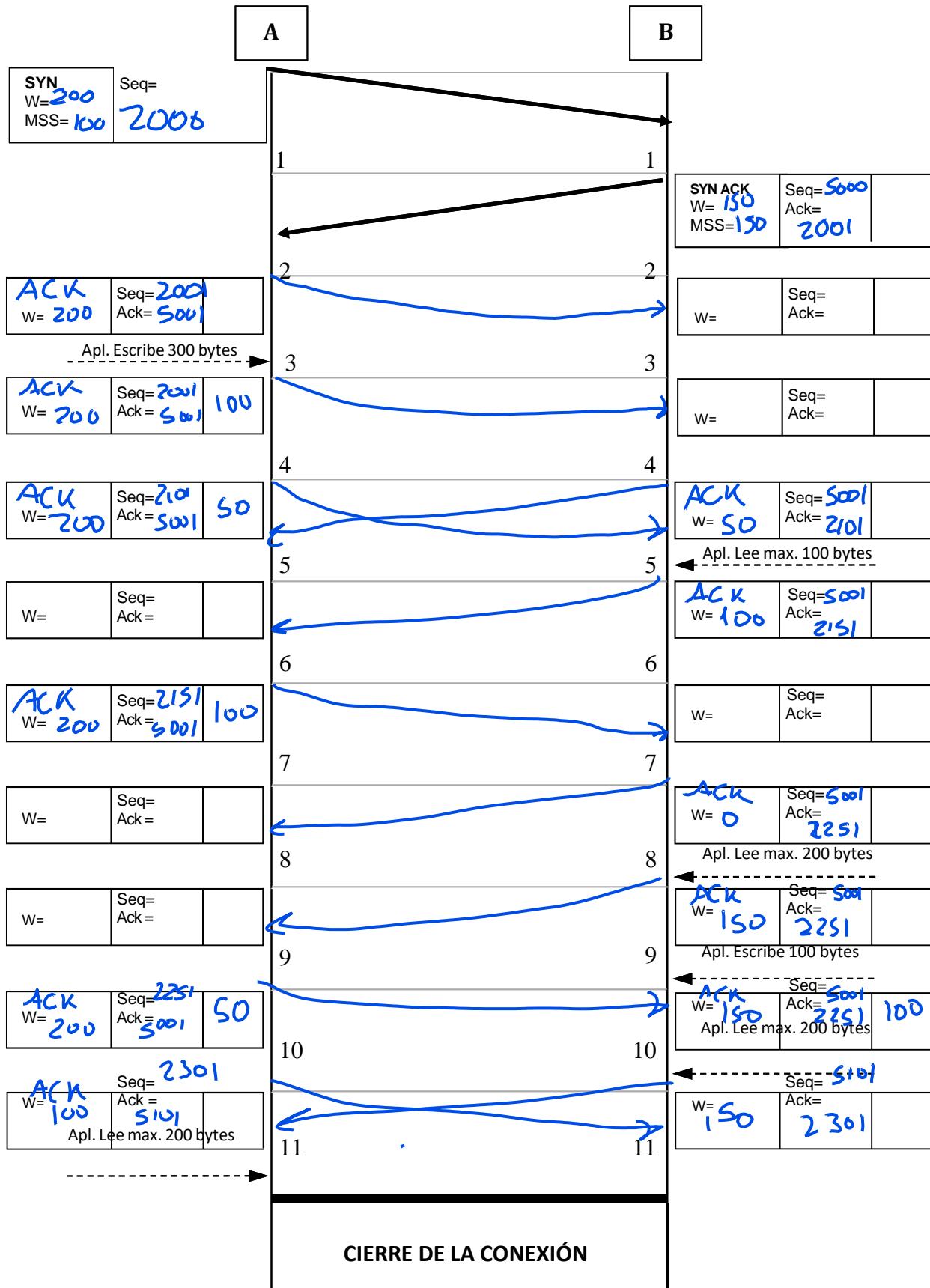
se incrementa en ambos casos IMSS, solo se diferencia en las acciones que se toman cuando hay pérdidas Ivers

### Ejercicio 20

Dos aplicaciones, que se ejecutan en máquinas distintas, desean enviarse información entre ellas usando TCP como protocolo de nivel de transporte. La aplicación que se ejecuta en la máquina **A** desea enviar 300 bytes de información a la máquina **B** y la aplicación que se ejecuta en la máquina **B** desea enviar 100 bytes de información a la aplicación que se ejecuta en la máquina **A**.

Utilice la plantilla para representar el intercambio de segmentos que se produce en el nivel TCP, teniendo en cuenta que:

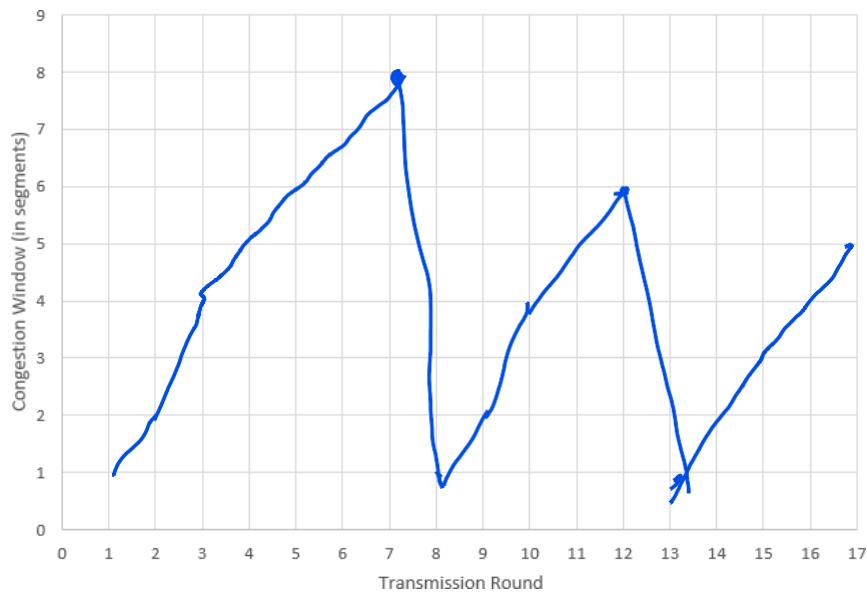
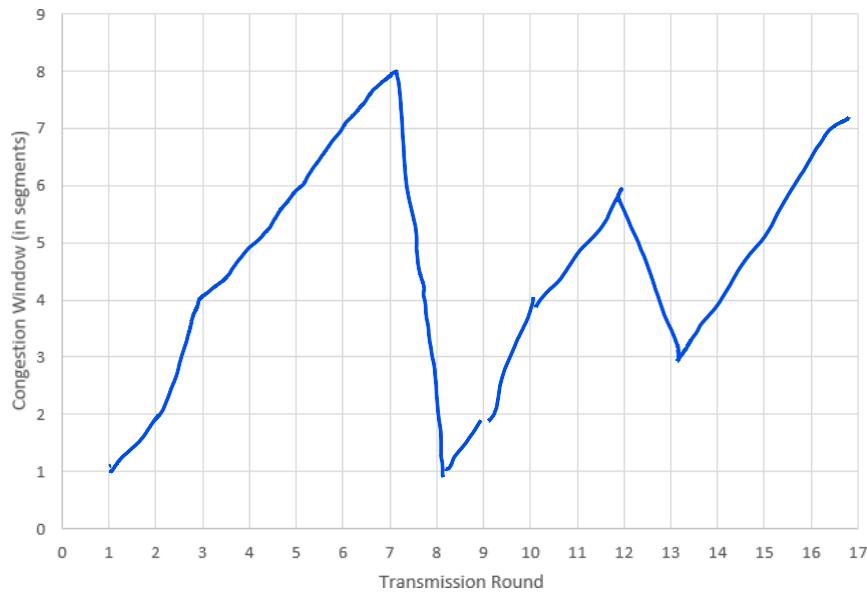
- Los segmentos se transmiten siempre al inicio de una unidad de tiempo (u.t.) y tardan 0,5 u.t. en alcanzar su destino.
- Los números de secuencia inicial de cada máquina son  $ISN_A=2000$  y  $ISN_B=5000$ .
- Las máquinas A y B tienen 200 bytes y 150 bytes de buffer de recepción respectivamente, para cada conexión TCP.
- La cantidad máxima de datos que se pueden enviar en un segmento para que no se produzca fragmentación en los niveles inferiores son: 100 bytes en la máquina A y 150 bytes en la máquina B. MSS = 100
- Los niveles TCP de ambas máquinas envían un reconocimiento inmediatamente después de recibir un segmento con datos (reconociendo todos los datos que estén pendientes de confirmar), o bien cuando se produce algún cambio en su ventana de recepción.
- Ante un cierre de ventana, no se envían segmentos sonda.
- Las aplicaciones que se ejecutan en las máquinas A y B inicián el cierre de la conexión inmediatamente después de que le sea confirmado su último byte de datos. No es necesario indicar la fase de cierre de conexión en este ejercicio.
- Las aplicaciones de ambas máquinas sólo leen (si están disponibles) o escriben bytes en los momentos indicados en la plantilla.



**Ejercicio 21**

- A) Dibujar la evolución de la ventana de congestión, tanto en TCP Tahoe como en TCP Reno, sabiendo que se producen dos eventos de pérdida: el primero debido a un timeout cuando la ventana de congestión alcanza el valor de 8MSS y el segundo por la recepción de 3 ACK duplicados cuando la ventana de congestión alcanza el valor de 6MSS. (6 puntos)

Nota: Considerar que el último evento de pérdida, anterior a los dos comentados anteriormente, se produjo con un valor de ventana de congestión de 10MSS.

TCP TahoeTCP Reno

- B) En un instante determinado de la operación del protocolo TCP, uno de los dos sistemas finales se encuentra en el estado que determinan las siguientes variables, donde todos los valores están medidos en bytes:

Caso 1: rwnd = 2800; cwnd = 2100; LastByteSent = 1500; LastByteAcked = 1500; MSS = 700

Caso 2: rwnd = 2100; cwnd = 2800; LastByteSent = 2000; LastByteAcked = 2000; MSS = 700

Caso 3: rwnd = 5000; cwnd = 5000; LastByteSent = 4000; LastByteAcked = 1800; MSS = 700

Caso 4: rwnd = 5000; cwnd = 5000; LastByteSent = 2500; LastByteAcked = 2500 ; MSS = 1000

Caso 5: rwnd = 3600; cwnd = 3600; LastByteSent = 3000; LastByteAcked = 2400; MSS = 1000

En ese mismo instante, el buffer de transmisión tiene 3000 bytes pendientes de transmitir. Indicar en cada uno de los casos, cuántos segmentos y de cuántos bytes cada uno, transmitirá este sistema.

$$\text{caso 1: } 0(x) \min\{2100, 2800\} = 2100 / 700 = 3 \text{ seg.}$$

$$\text{caso 2: } 0(x) \min\{2100, 2800\} \rightarrow 3 \text{ seg.}$$

$$\text{caso 3: } 4000 - 1800 = 2200 \min\{5000, 5000\} \rightarrow 5000 - 2200 = 2800 \rightarrow 2800 / 1000 = 2 \text{ seg.}$$

$$\text{caso 4: } \min\{5000, 5000\} = 5000 \rightarrow 3000 / 1000 = 3 \text{ seg.}$$

$$\text{caso 5: } 3000 - 2400 = 600 \min\{3600, 3600\} = 3600 - 600 = 3000 / 1000 = 3 \text{ seg.}$$

4 seg.

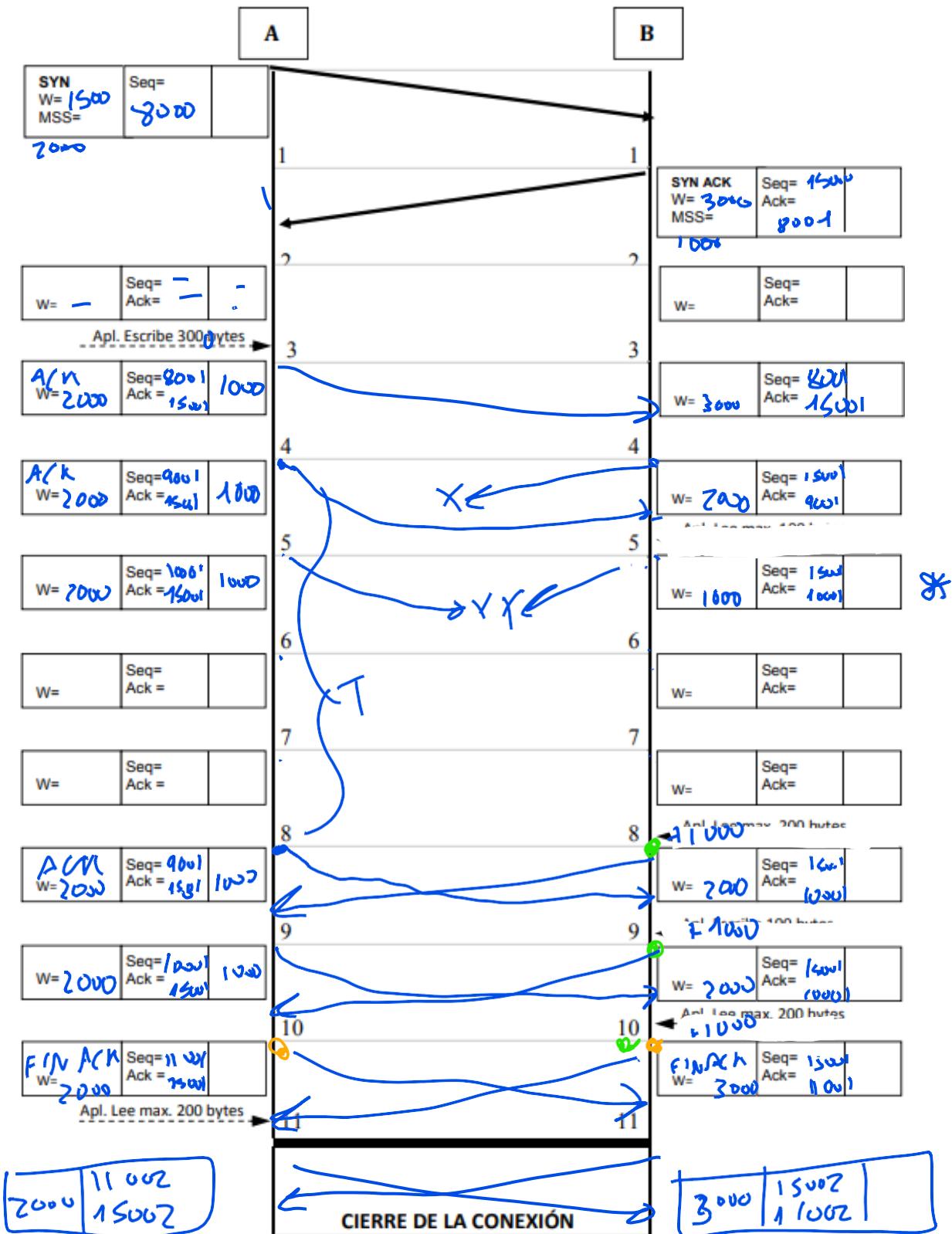
- C) En relación a la Notificación Explícita de Congestión (ECN), ¿para qué usa el nivel de transporte de un sistema final el flag CWR?

El nivel de transporte TCP de un sistema final usa el flag CWR para indicar al nivel de transporte TCP del sistema final remoto, que ha disminuido el valor de su ventana de congestión como consecuencia de haber recibido un segmento con el flag ECE activado.

## Ejercicio 22

Dos aplicaciones, que se ejecutan en máquinas distintas, desean enviarse información entre ellas usando TCP como protocolo de nivel de transporte. Únicamente la aplicación que se ejecuta en la máquina **A** desea enviar 3000 bytes de información a la máquina **B**, y se sabe que:

- Los segmentos se transmiten siempre al inicio de una unidad de tiempo (u.t.) y tardan 0,5 u.t. en alcanzar su destino.
- Ambas máquinas envían un segmento de solicitud de conexión en la u.t.= 0
- El temporizador de retransmisión es de 4 u.t.
- Los números de secuencia inicial de cada máquina son  $ISN_A=8000$  y  $ISN_B=15000$ .
- Las máquinas A y B tienen 1500 bytes y 3000 bytes de buffer de recepción respectivamente, para cada conexión TCP.
- La cantidad máxima de datos que se pueden enviar en un segmento para que no se produzca fragmentación en los niveles inferiores son: 2000 bytes en la máquina A y 1000 bytes en la máquina B.
- Los niveles TCP de ambas máquinas envían un reconocimiento inmediatamente después de recibir un segmento con datos (reconociendo todos los datos que estén pendientes de confirmar), o bien cuando se produce algún cambio en su ventana de recepción.
- Ante un cierre de ventana, no se envían segmentos sonda.
- En u.t.= 4 los segmentos que envía la máquina B se pierden.
- En u.t.= 5 los segmentos que envían ambas máquinas se pierden.
- La aplicación de la máquina A escribe 3000 bytes en el buffer de transmisión en u.t.=2,9
- La aplicación de la máquina B lee 1000 bytes de su buffer de recepción en u.t.=7,9 en u.t.=8,9 y en u.t.= 9,9
- Las aplicaciones que se ejecutan en las máquinas A y B inician el cierre de la conexión en u.t.= 10

Internet Transport LayerTCP5-T(NP):Theory Exercises

**Se pide:**

- A) Rellenar los campos indicados del segmento que la máquina A transmite en la u.t. 8

Flags=	seq= 9001 data= 1000
ACK	ack= 15001 w= 2000

- B) Rellenar los campos indicados del segmento que la máquina B transmite en la u.t. 5

Flags=	seq= 15001 data= 0
ACK	ack= 9001 w= 1000

- C) Rellenar los campos indicados del segmento que la máquina B transmite en la u.t. 8

Flags=	seq= 15001 data= 0
ACK	ack= 10001 w= 2000

- D) Rellenar los campos indicados del segmento que la máquina B transmite en la u.t. 11

Flags=	seq= 15002 data= 0
ACK	ack= 11002 w= 3000

**Ejercicio 5 (8 puntos) (10 min)**

Estudiando la evolución de la ventana de congestión en una conexión TCP, se observa que se producen los siguientes eventos de pérdida:

- En  $t=5\text{RTT}$  se produce un evento de pérdida por 3 ACK duplicados.
- En  $t=13\text{RTT}$  se produce un evento de pérdida por 3 ACK duplicados.
- En  $t=16\text{RTT}$  se produce un evento de pérdida por Timeout.

Se pide:

Completar la siguiente tabla, indicando el valor (en MSS) que tendría la ventana de congestión (cwnd) en los instantes indicados en función del algoritmo de control de congestión usado.

Nota: El valor inicial del ssthresh es de 4MSS

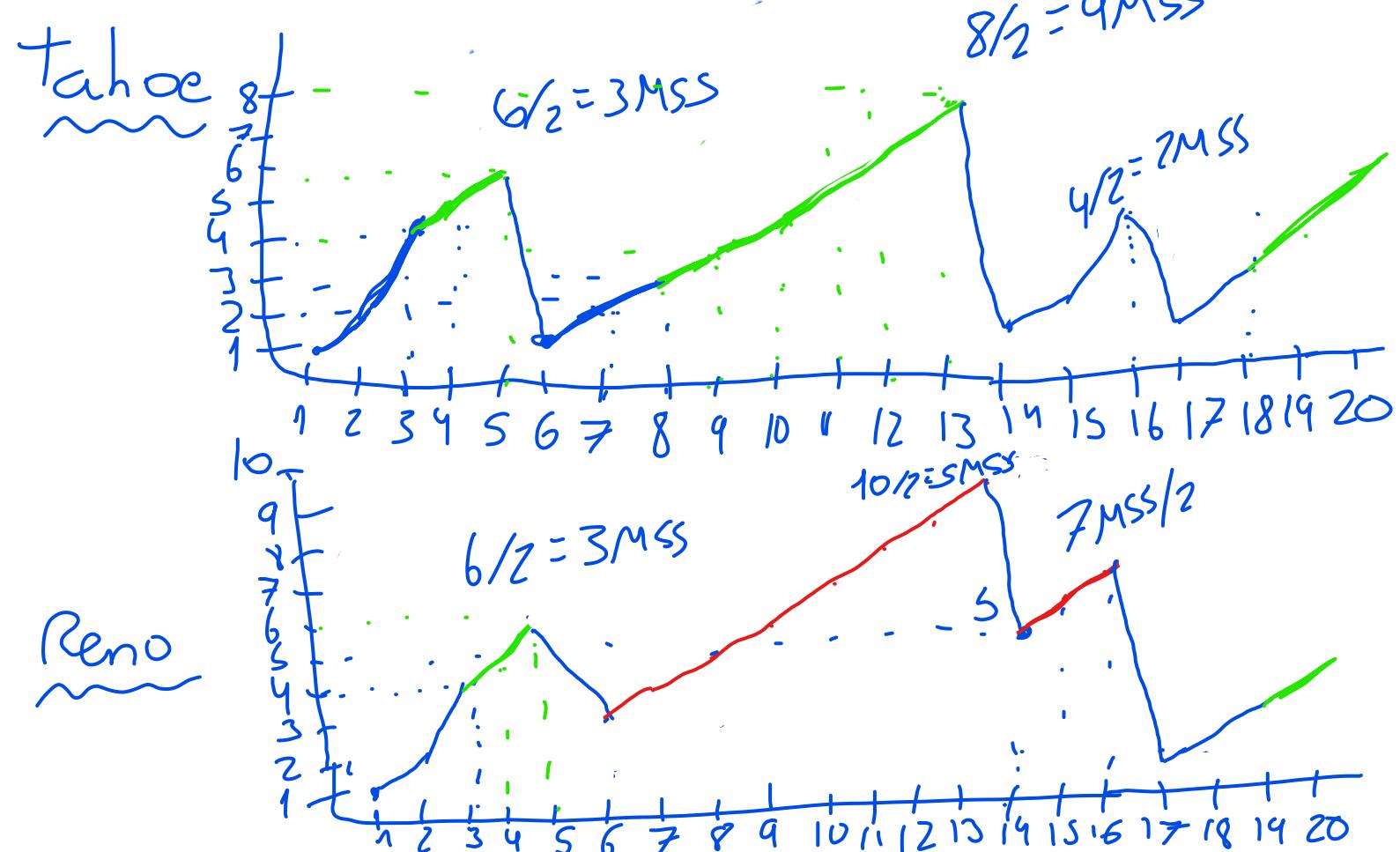
✓ C.A.

• Slow Start

• C. A.

• Fast R.

Instante	cwnd (TCP Tahoe)	cwnd (TCP Reno)
$t=5\text{RTT}$	6	6
$t=13\text{RTT}$	8	10
$t=16\text{RTT}$	4	7
$t=20\text{RTT}$	4	4



**Ejercicio 24**

En un instante determinado de la operación del protocolo TCP, uno de los dos sistemas finales se encuentra en el estado que determinan las siguientes variables, donde todos los valores están medidos en bytes:

Caso 1: BufferTx=1700, MSS = 340, cwnd = 1822; rwnd = 2153; LastByteAcked = 3682; LastByteSent = 3682

Caso 2: BufferTx=2100, MSS = 525, cwnd = 4041; rwnd = 4041; LastByteAcked = 7331; LastByteSent = 7331

Caso 3: BufferTx=2200, MSS = 440, cwnd = 1974; rwnd = 1974; LastByteAcked = 1347; LastByteSent = 2441

Caso 4: BufferTx=1700, MSS = 425, cwnd = 2068; rwnd = 2068; LastByteAcked = 3983; LastByteSent = 4351

En ese mismo instante, el buffer de transmisión tiene pendientes de transmitir los bytes que se indican en BufferTx.

**Se pide:**

Indicar en cada uno de los casos, cuántos segmentos y de cuántos bytes cada uno, transmitirá este sistema.

	Nº de Segmentos	Tamaño del segmento (bytes)
Caso 1	5	340
Caso 2	4	525
Caso 3	2	440, 413
Caso 4	4	425

$$\text{Caso 1: } 0, \min\{1822, 2153\} = 1822 \rightarrow 1700 / 340 = 5 \text{ seg.}$$

$$\text{Caso 2: } 0, \min\{4041, 4041\} = 4041 \rightarrow 2100 / 525 = 4 \text{ seg.}$$

$$\text{Caso 3: } \begin{array}{r} 2441 \\ - 1347 \\ \hline 1094 \end{array} \quad \min\{1947, 1947\} = 1947 - 1694 = 853 / 440 \rightarrow \begin{array}{r} 1947 \\ - 1094 \\ \hline 853 \end{array} \quad 2 \text{ segmentos}$$

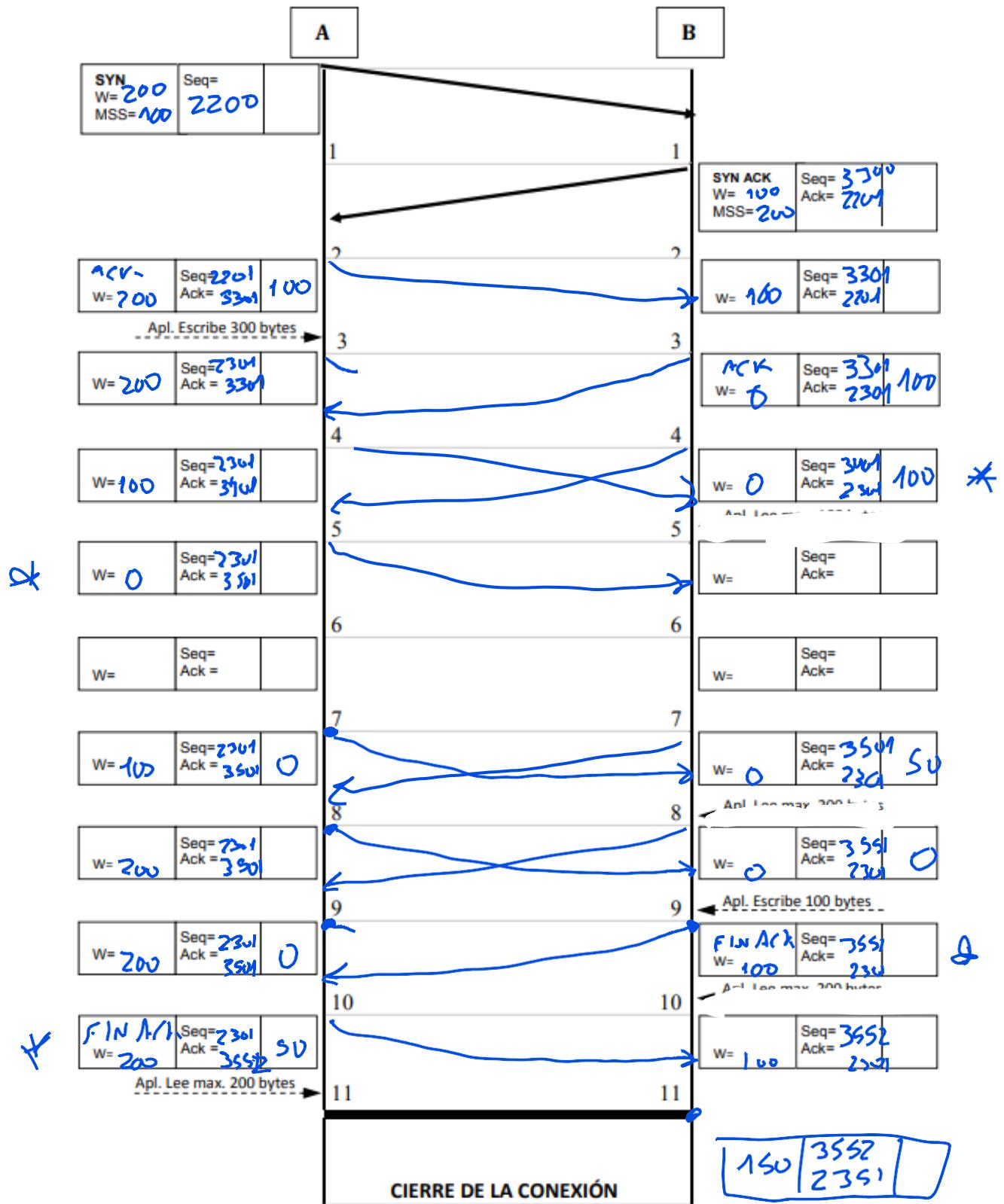
$$\text{Caso 4: } \begin{array}{r} 4351 \\ - 3983 \\ \hline 368 \end{array} \quad \min\{2068, 2068\} = 2068 - 368 = 1700 / 425 = 4 \text{ seg}$$

## Ejercicio 25

Dos aplicaciones, que se ejecutan en máquinas distintas, desean enviarse información entre ellas usando TCP como protocolo de nivel de traspporte. La aplicación que se ejecuta en la máquina **B** desea enviar 250 bytes de información a la máquina **A** y la aplicación que se ejecuta en la máquina **A** desea enviar 150 bytes de información a la aplicación que se ejecuta en la máquina **B**, y se sabe que:

- Los segmentos se transmiten siempre al inicio de una unidad de tiempo (u.t.) y tardan 0,5 u.t. en alcanzar su destino
- La máquina A solicita establecer una conexión TCP con la máquina B en u.t.=0
- Los números de secuencia inicial de cada máquina son  $I_A = 0$        $I_B = 0$   
*seq.*      *scq.*
- El tamaño de los buffer de recepción de las máquinas A y B son de 200 bytes y 100 bytes respectivamente, para cada conexión TCP 
- La cantidad máxima de datos que se pueden enviar en un segmento para que no se produzca fragmentación en los niveles inferiores son: 100 bytes en la máquina A y 200 bytes en la máquina B **MSS**
- Los niveles TCP de ambas máquinas envían un reconocimiento inmediatamente después de recibir un segmento con datos (reconociendo todos los datos que estén pendientes de confirmar), o bien cuando se produce algún cambio en su ventana de recepción
- Ante un cierre de ventana, no se envían segmentos sonda
- La aplicación de la máquina A escribe 150 bytes en el buffer de transmisión en u.t.=1,9
- La aplicación de la máquina B escribe 250 bytes en el buffer de transmisión en u.t.=2,9
- La aplicación de la máquina A lee un máximo de 100 bytes de su buffer de recepción en u.t.=6,9 en u.t.=7,9 en u.t.=8,9 y en u.t.=9,9
- La aplicación de la máquina B lee un máximo de 100 bytes de su buffer de recepción en u.t.=8,9 y en u.t.=10,9
- La aplicación de la máquina A solicita el cierre de la conexión en u.t.=9,9
- La aplicación de la máquina B solicita el cierre de la conexión en u.t.=8,9

Nota: Usar la plantilla adjunta para la resolver el ejercicio.

Internet Transport LayerTCP5-T(NP):Theory Exercises

**Se pide:**

- E) Rellenar los campos indicados del segmento que la máquina **B** transmite en la u.t. 4

Flags=	seq= 3401	data= 100
ACK	ack= 2301	w= 0

- F) Rellenar los campos indicados del segmento que la máquina **A** transmite en la u.t. 5

Flags=	seq= 2301	data= 0
ACK	ack= 3501	w= 0

- G) Rellenar los campos indicados del segmento que la máquina **B** transmite en la u.t. 9

Flags=	seq= 3551	data= 0
FIN ACK	ack= 2301	w= 100

- H) Rellenar los campos indicados del segmento que la máquina **A** transmite en la u.t. 10

Flags=	seq= 2301	data= 50
FIN ACK	ack= 3552	w= 200

**Ejercicio 26**

Dos sistemas finales (equipo A y equipo B), que usan Ethernet como tecnología de acceso al medio, han establecido una conexión TCP en la que el MSS se ha negociado para que no se produzca fragmentación en ninguno de ellos.

En un momento determinado se sabe que:

- el valor de la ventana de recepción que el equipo B ha anunciado al equipo A es **mayor** que el valor de la ventana de congestión del equipo A en ese instante. **cwnd**
- el equipo A ha enviado **1840 bytes** de los que aún no ha recibido reconocimiento por parte del equipo B
- el valor del campo "acknowledgment number" presente en la cabecera del último segmento que el equipo B envió al equipo A fue de **160 ACK 160**

$$\rightarrow MTU = 1500$$

$$MSS = MTU - 40 = 1460$$

En esta situación, el equipo A puede enviar (y envía) un máximo de **4 segmentos** al equipo B, **con la máxima cantidad de datos** que puede contener cada uno de ellos.

**Se pide:**

- A) Indicar la cantidad mínima de datos (en bytes) que se encuentran en el buffer de transmisión del equipo A

$$1840 + (4 \cdot 1460) = 7680 \text{ bytes}$$

- B) Indicar, razonando la respuesta, por qué tipo de control (de flujo o de congestión) está limitada la transmisión de datos del equipo A

**Cwnd < Rwnd → Control de congestión**

- C) Indicar cuál fue el número de secuencia del último byte que envió el equipo A

La cantidad de datos enviados y todavía no reconocidos (sin tener en cuenta los últimos 4 segmentos que envía al equipo B) será:  
 $LastByteSent - LastByteAcked = 1840$ , donde  $LastByteAcked=160$ ,  
 luego el número de secuencia del último byte enviado será 2000

- D) Indicar cuál es el valor mínimo (en bytes) de la ventana de congestión del equipo A

Puesto que ha podido enviar 5840 bytes teniendo 1840 bytes enviados sin confirmar, el valor de la ventana de congestión debe ser al menos de 7680 bytes

- E) Teniendo en cuenta el valor de la ventana de congestión del apartado anterior, indicar cuál es el mecanismo de control de congestión en el que se encuentra el equipo A, si el valor de su variable  $ssthresh$  es de 10000 bytes.

$cwnd < ssthresh \rightarrow \text{slow start.}$

8 de julio de 2022

TEORÍA

**Ejercicio 4 (7 puntos) (15 min)**

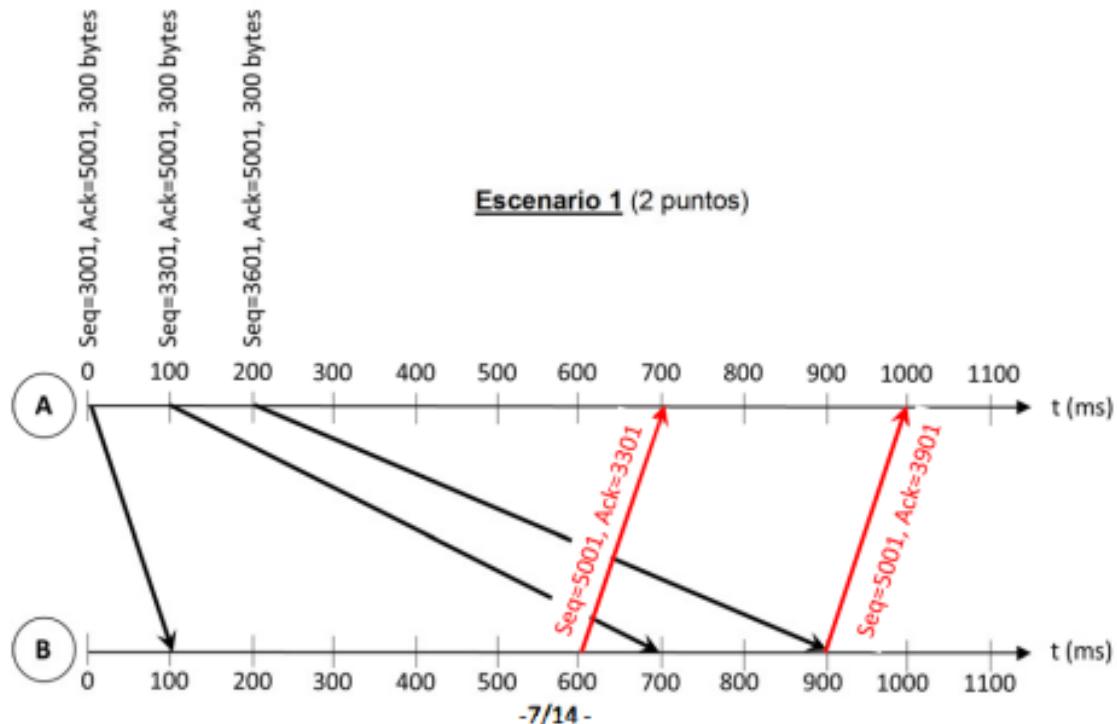
Una aplicación que se ejecuta en la máquina A desea enviar información a otra aplicación que se ejecuta en la máquina B, y se sabe que:

- Los números de secuencia inicial de cada máquina son  $ISN_A=3000$  y  $ISN_B=5000$ .
- La conexión ya se ha establecido con éxito, en la que el MSS negociado ha sido de 300 bytes.
- El tiempo de propagación de los datagramas que envía la máquina B a la máquina A es fijo y su valor es de 100 ms.
- La implementación del protocolo TCP que se ejecuta en la máquina B no guarda los segmentos recibidos fuera de orden.

En esta situación, se plantean tres escenarios distintos en los que:

- El nivel TCP de la máquina A inicia el envío de segmentos hacia la máquina B en los instantes indicados.
- El nivel TCP de la máquina B recibe los segmentos procedentes de la máquina A en los instantes indicados y genera segmentos ACK según las RFC-1122 y RFC-5681

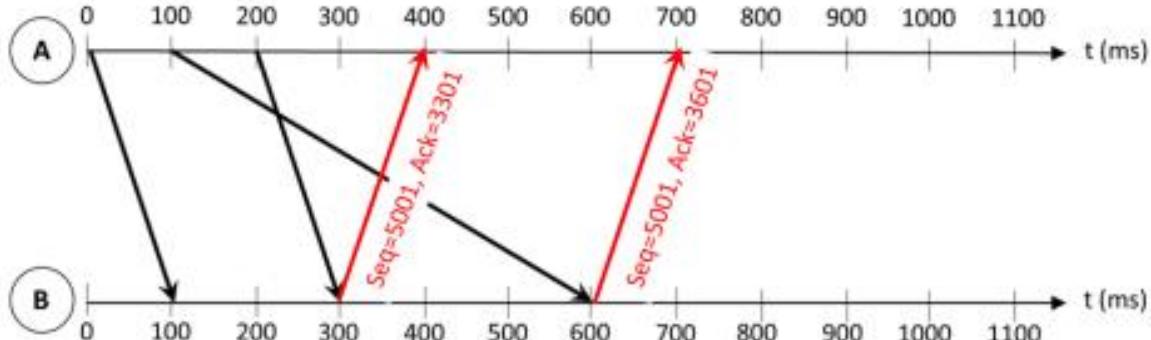
**Se pide:** Dibujar sobre las siguientes plantillas los segmentos TCP que la máquina B enviaría a la máquina A, indicando los números de secuencia y reconocimiento de cada uno de ellos



Seq=3001, Ack=5001, 300 bytes

Seq=3301, Ack=5001, 300 bytes

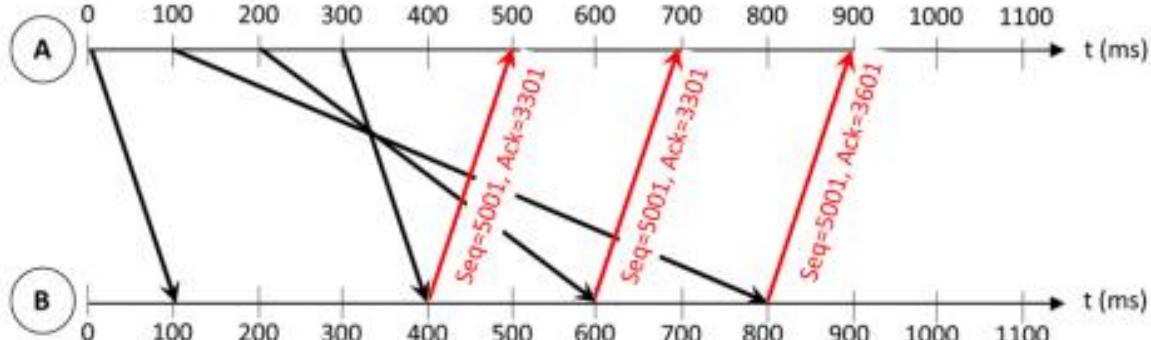
Seq=3601, Ack=5001, 300 bytes

Escenario 2 (2 puntos)

Seq=3001, Ack=5001, 300 bytes

Seq=3301, Ack=5001, 300 bytes

Seq=3601, Ack=5001, 300 bytes

Escenario 3 (2 puntos)

Indicar qué cambiaría en cada uno de los escenarios anteriores si la implementación del protocolo TCP que se ejecuta en la máquina B hubiese guardado los segmentos recibidos fuera de orden. (1 punto)

**Escenario 1:** No cambiaría

**Escenario 2:** El número de secuencia del segmento que envía la máquina B en  $t=600$  ms sería 3901

**Escenario 3:** El número de secuencia del segmento que envía la máquina B en  $t=800$  ms sería 4201

Telemática y Electrónica	1 de junio de 2023	TEORÍA
--------------------------	--------------------	--------

### Ejercicio 4 (15 puntos) (20 min)

Una aplicación que se ejecuta en la máquina A desea enviar información a otra aplicación que se ejecuta en la máquina B, y se sabe que:

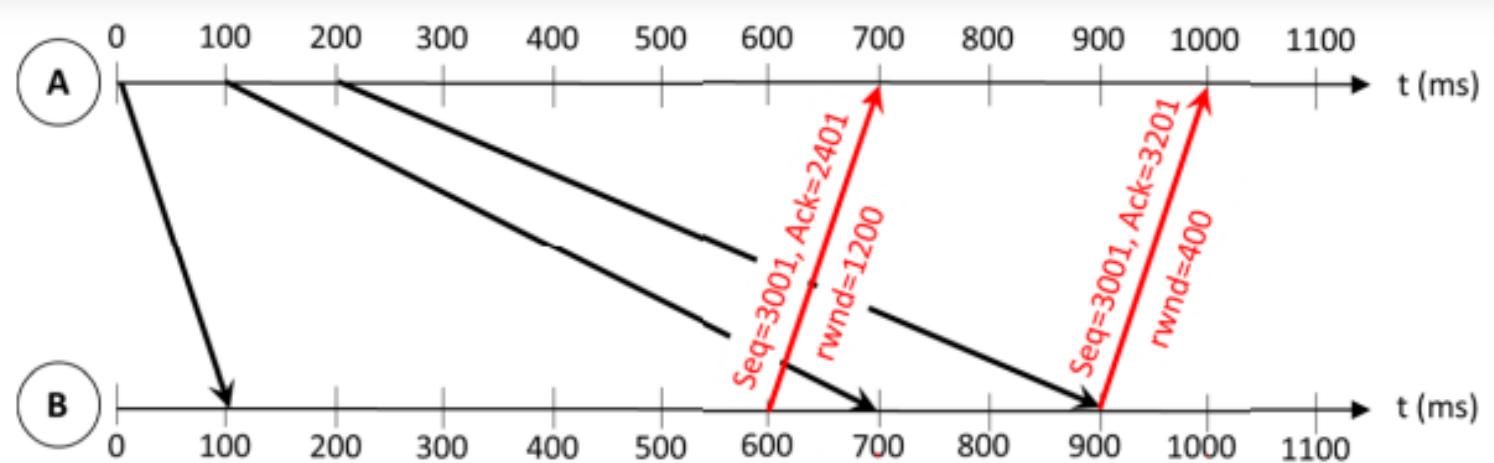
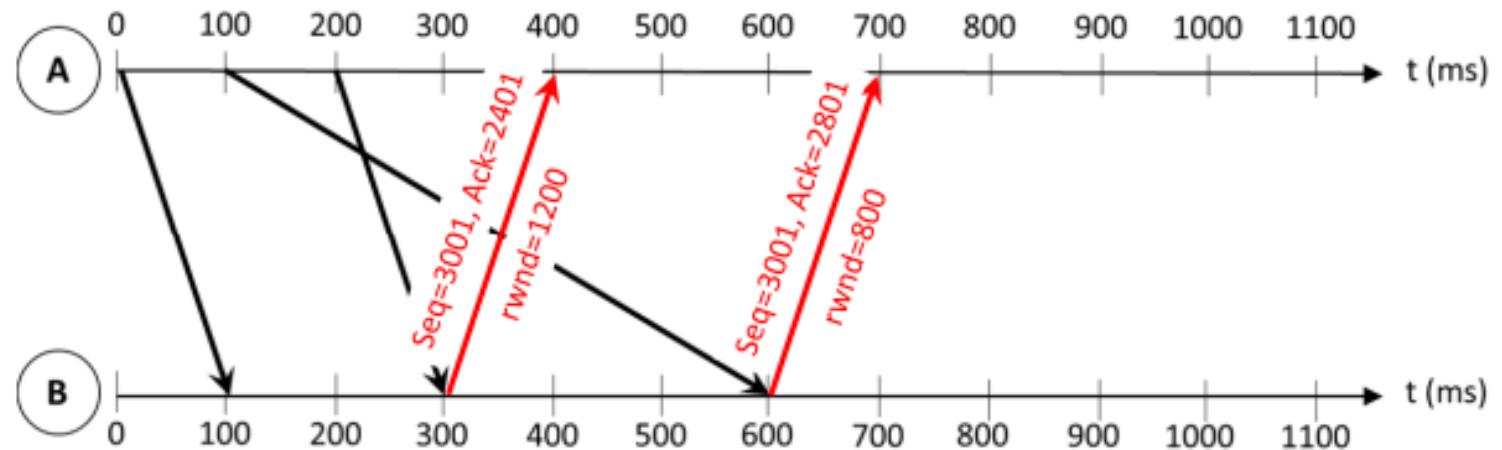
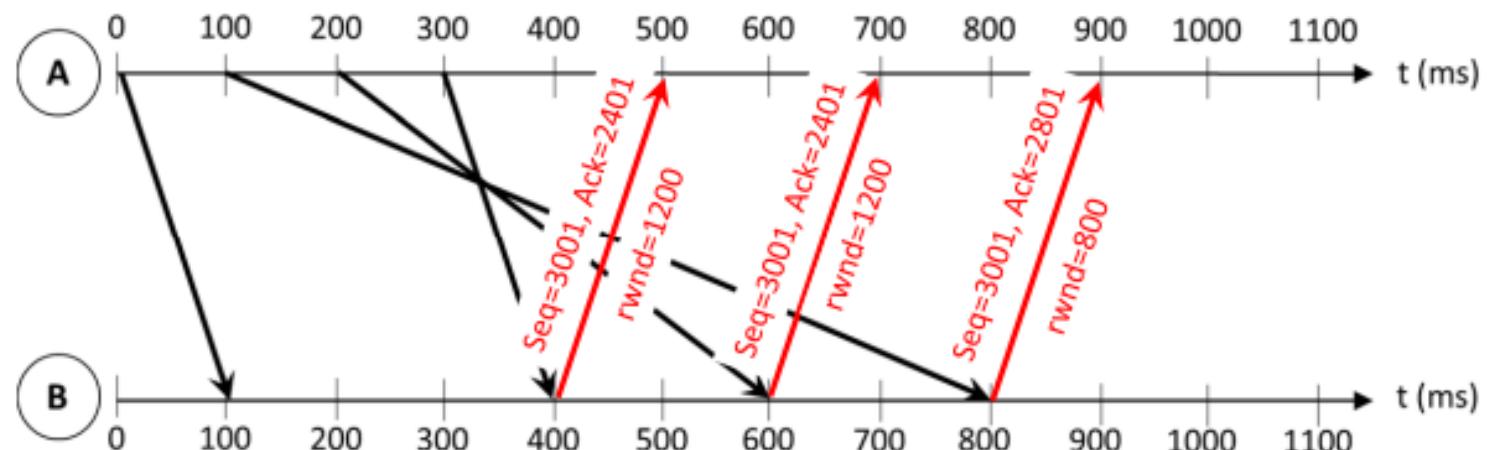
- Los números de secuencia inicial de cada máquina son  $ISN_A=2000$  y  $ISN_B=3000$ .
- El tamaño de los buffer de recepción de las máquinas A y B son de 1000 bytes y 1600 bytes respectivamente, para cada conexión TCP
- La cantidad máxima de datos que se pueden enviar en un segmento para que no se produzca fragmentación en los niveles inferiores son: 500 bytes en la máquina A y 400 bytes en la máquina B
- El primer segmento TCP con datos que envía la máquina A lo hace en  $t=0$
- Todos los segmentos TCP se envían con la cantidad máxima de datos que se pueden enviar
- La aplicación de la máquina B nunca lee su buffer de recepción
- El tiempo de propagación de los datagramas que envía la máquina B a la máquina A es fijo y su valor es de 100 ms
- La implementación del protocolo TCP que se ejecuta en la máquina B no guarda los segmentos recibidos fuera de orden

Después de haber establecido la conexión, se plantean tres escenarios distintos en los que:

- El nivel TCP de la máquina A inicia el envío de segmentos hacia la máquina B en los instantes indicados.
- El nivel TCP de la máquina B recibe los segmentos procedentes de la máquina A en los instantes indicados y genera segmentos ACK según las RFC-1122 y RFC-5681

Se pide:

Dibujar sobre las siguientes plantillas los segmentos TCP que la máquina B enviaría a la máquina A, indicando en cada uno de ellos los números de secuencia y reconocimiento, así como el valor de la ventana de recepción.

Escenario 2 (4 puntos)Escenario 3 (4 puntos)

Indicar qué cambiaría en cada uno de los escenarios anteriores si la implementación del protocolo TCP que se ejecuta en la máquina B hubiese guardado los segmentos recibidos fuera de orden. (3 puntos)

Escenario 1: No cambiaría

Escenario 2: Los segmentos que se enviarían serían:

En t=300 ms: Seq=3001 Ack=2401 rwnd=800

En t=600 ms: Seq=3001 Ack=3201 rwnd=400

Escenario 3: Los segmentos que se enviarían serían:

En t=400 ms: Seq=3001 Ack=2401 rwnd=800

En t=600 ms: Seq=3001 Ack=2401 rwnd=400

En t=800 ms: Seq=3001 Ack=3601 rwnd=0

$MSS = 1460$

Ethernet  $\rightarrow MTU = 1500$

$cwnd \rightarrow \text{Las 16 bytes send}$

En un momento determinado se sabe que:

- el valor de la ventana de congestión del equipo A en ese momento es de 4440 bytes y que el valor de la ventana de recepción que el equipo B ha anunciado al equipo A es de 17760 bytes
- el valor de la ventana de congestión del equipo A a partir del cual empieza a crecer de forma lineal es de 16280 bytes
- en el equipo A, el valor de la variable `SendBase` = a

A partir de este instante, el equipo B envía 13 segmentos al equipo A cuyo valor del campo "acknowledgment number" presente en la cabecera de cada uno de ellos se muestra en la siguiente tabla:

$cwnd < umbral \rightarrow \text{ow start}$        $cwnd = cwnd + \underbrace{\text{No Ack} \cdot MSS}_{\text{NO Duplicados}}$

	Segmento	Acknowledgment Number	
	N	a	→ ACK Duplicado
1	N+1	b ( $b > a$ )	→ ACK No Dup.
2	N+2	b	→ ACK Dup.
3	N+3	c ( $c > b$ )	→ ACK No Dup.
4	N+4	d ( $d > c$ )	→ ACK No Dup.
5	N+5	e ( $e > d$ )	→ ACK No Dup.
6	N+6	e	→ ACK Dup.
7	N+7	e	→ ACK Dup.
8	N+8	f ( $f > e$ )	→ ACK No Dup.
9	N+9	f	→ ACK Dup.
10	N+10	g ( $g > f$ )	→ ACK No Dup.
11	N+11	h ( $h > g$ )	→ ACK No Dup.
12	N+12	h	→ ACK Dup.

$\rightarrow \text{SendBase} \rightarrow h$ .

En esta situación, después de que el equipo A haya recibido estos 13 segmentos,

Se pide:

- A) Indicar, razonando la respuesta, qué segmentos corresponden a ACK duplicados (2 puntos)

$$N, N+2, N+6, N+7, N+9, N+12.$$

- B) Indicar cuál será el valor de la variable Sendbase después de recibir el último segmento (1 punto)

$$\text{Sendbase} = h$$

- C) Indicar, razonando la respuesta, qué mecanismo de control está limitando esta comunicación (1 punto)

$cwnd < rwnd \rightarrow$  control de congestión

$cwnd > rwnd \rightarrow$  control de flujo

- D) Indicar, razonando la respuesta, el valor final de la ventana de congestión del equipo A (4 puntos)

$$cwnd = 4440 \text{ bytes}$$

$$ssthresh = 16280 \text{ bytes.}$$

$$MSS = 1460$$

$\Rightarrow$  ACK nuevos (sin duplicados)

$$cwnd_{final} = 4440 + 7 \cdot MSS = 4440 + 10220 = >$$

$$cwnd_{final} = 14660 \text{ bytes.}$$

Si inicialmente, el valor de la ventana de congestión del equipo A a partir del cual empieza a crecer de forma lineal, hubiera sido de 2960 bytes:

$\checkmark ssthresh$

- E) Indicar, razonando la respuesta, cuánto se incrementaría la ventana de congestión del equipo A al recibir el segmento N+6 (2 puntos)

$cwnd > ssthresh \rightarrow$  congestión Avoidarse

$N+6 \rightarrow$  Mismo número de frecuencia que  $N+5 \rightarrow$  No se incrementa.

**Ejercicio 5 (7 puntos) (15 min)**

Dos sistemas finales (equipo A y equipo B), que usan una tecnología de acceso al medio cuya MTU=1040 bytes, han establecido una conexión TCP en la que el MSS se ha negociado para que no se produzca fragmentación en ninguno de ellos.

$$MSS = MTU - 40 = 1000 \text{ bytes}$$

En un momento determinado se sabe que:

- el valor de la ventana de congestión del equipo A en ese instante es de 25000 bytes y que el valor de la ventana de recepción que el equipo B ha anunciado al equipo A es de 36000 bytes.
- el valor de la ventana de congestión del equipo A a partir del cual empezaría a crecer de forma lineal es de 35000 bytes.
- en el equipo A, el valor de la variable SendBase=x1

A partir de este instante, el equipo B envía 13 segmentos al equipo A cuyo valor del campo "acknowledgment number" presente en la cabecera de cada uno de ellos se muestra en la siguiente tabla:

	Segmento	Acknowledgment Number	
	N	x1	→ ACK duplicado
1	N+1	x1	→ ACK Dup.
2	N+2	x2 (> x1)	→ ACK No Dup.
3	N+3	x3 (> x2)	→ ACK No Dup.
4	N+4	x4 (> x3)	→ ACK No Dup.
5	N+5	x4	→ ACK Dup.
6	N+6	x5 (> x4)	→ ACK No Dup.
7	N+7	x6 (> x5)	→ ACK No Dup.
8	N+8	x7 (> x6)	→ ACK No Dup.
	N+9	x8 (> x7)	→ ACK No Dup.
	N+10	x9 (> x8)	→ ACK No Dup.
	N+11	x9	→ ACK Dup.
	N+12	x9	→ ACK Dup.

En esta situación, después de que el equipo A haya recibido estos 13 segmentos,

Se pide:

- A) Indicar, razonando la respuesta, el nuevo valor de la ventana de congestión del equipo A (3 puntos)

$cwnd < ssthresh \rightarrow \text{slow start}$ .

$cwnd = 25000 \text{ bytes}$

$MSS = 1000 \text{ bytes}$

$$cwnd_{\text{final}} = cwnd + N^{\circ} \text{ ACKs} \cdot MSS = \\ \downarrow \\ \text{No Dyn.}$$

$$25000 + 8000 = 33000 \text{ bytes}$$

Si inicialmente, el valor de la ventana de congestión del equipo A a partir del cual empezaría a crecer de forma lineal, hubiera sido de 20000 bytes:

$\Rightarrow ssthresh$

- B) Indicar, razonando la respuesta, cuánto se incrementaría (en bytes) la ventana de congestión del equipo A al recibir el segmento N+5 (2 puntos)

$cwnd < ssthresh \rightarrow \text{congestion Avoidance}$   
 como el N° de secuencia del segmento enviado en N+5 es el mismo que el enviado en N+4, No se incrementará cwnd

Si el tipo de control de congestión usado en los niveles TCP de ambas máquinas se cambiara a "Notificación Explícita de la Congestión"

- C) Indicar, qué campos habría que añadir y en qué PDU, para contemplar este nuevo tipo de control de congestión (2 puntos)

En la cabecera del datagrama IP habría que añadir 2 bits (ECN) dentro del campo "Type Of Service" (ToS)

En la cabecera del segmento TCP se añadirían 2 flags (ECE y CWR) dentro del campo "Reserved" (Not Used)

**Ejercicio 5 (18 puntos) (20 min)**

- A) Indicar bajo qué condiciones se arranca el temporizador de retransmisión de TCP (6 puntos)

El temporizador de retransmisión de TCP se arranca cuando...

1....se envía un nuevo segmento TCP con datos y el temporizador no se encuentra arrancado previamente.

2....después de haber expirado el temporizador (evento de timeout) se retransmite el segmento TCP con datos con número de secuencia más bajo.

3....después de recibir un ACK que confirma datos nuevos (previamente sin confirmar) y todavía quedan datos enviados sin confirmar.

- B) ¿Qué relación existe entre los valores de MTU y MSS? ¿Qué objetivo se persigue con esa relación? ¿En qué tipo de segmentos TCP se incluye el valor MSS y en qué campo de la cabecera se hace? (4 puntos)

$MSS=MTU-\text{Long Cabecera IP} - \text{Long CabeceraTCP}$ , que en la mayoría de los casos será  $MSS=MTU-40$  (medido en bytes).

El objetivo es dimensionar el MSS de tal forma que se evite la fragmentación en el nivel de acceso a red.

El MSS se negocia durante la fase de establecimiento de la conexión TCP, por lo que su valor se incluye en los segmentos TCP con el flag SYN activado.

En estos segmentos, el valor del MSS se incluye en el campo "Opciones" de la cabecera TCP

- C) ¿Bajo qué condiciones se puede producir un cierre abrupto (Reset)? (2 puntos)

Cuando la entidad TPC receptora no puede establecer la conexión (por ejemplo, porque no hay ningún proceso escuchando en el puerto solicitado, es decir, no se ha realizado un "passive open" sobre ese puerto), o bien cuando el segmento SYN recibido tiene un error.

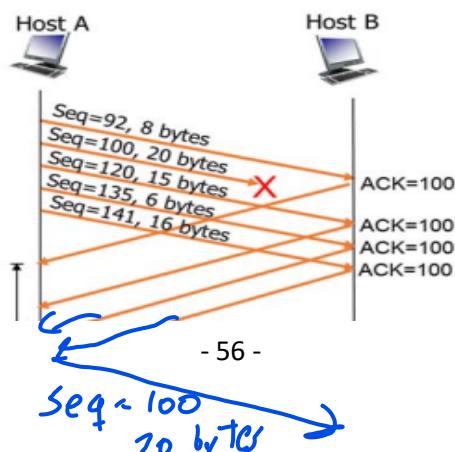
- D) ¿Qué implicaciones tendría poner el valor del temporizador de retransmisión TCP demasiado corto o demasiado largo? (2 puntos)

Si fuera demasiado corto podría vencer prematuramente dando lugar a retransmisiones innecesarias.

Si fuera demasiado largo se tardaría demasiado en reaccionar ante la pérdida real de un segmento.

- E) Explicar, mediante un ejemplo que lo ilustre, en qué consiste la retransmisión rápida (Fast Retransmit) en TCP (4 puntos)

En el caso de que se reciban 3 ACKs duplicados, el emisor TCP realiza una retransmisión rápida, reenviando el segmento que falta antes de que expire el temporizador.



**Ejercicio 6 (12 puntos) (15 min)**

- A) El algoritmo AIMD usado en el control de congestión en TCP se dice que es justo o que hace justicia (TCP Fairness). ¿Por qué? (2 puntos)

Porque asegura el mismo ancho de banda para todas las conexiones TCP que comparten un mismo enlace, es decir, si K conexiones comparten un enlace de capacidad R, la velocidad media de transmisión de cada una de ellas será  $R/K$ .

- B) En un instante determinado de la operación del protocolo TCP, uno de los dos sistemas finales se encuentra en el estado que determinan las siguientes variables, donde todos los valores están medidos en bytes: (2 puntos)

LastByteAcked=1000, LastByteSent= 5000, cwnd = 6000, ssthresh=8000, MSS=500

Al recibir cuatro segmentos con números de ACK 1501, 1501, 2001 y 2001, ¿cuál será el nuevo valor de la ventana de congestión?

$$1500 + 1500 + 2000 + 2000 = 7000 \rightarrow \text{cwnd}$$

- C) ¿Cuánto se incrementa la ventana de congestión en la fase de crecimiento lineal cada vez que se recibe un ACK nuevo? (2 puntos)

Se incrementa según la fórmula  
 $cwnd = cwnd + MSS * (MSS/cwnd)$

- D) Si después de un evento de pérdida el valor de la ventana de congestión fuese de 1MSS, ¿qué tipo de TCP se estaría usando? (2 puntos)

Si no se sabe de qué tipo es el evento de pérdida (timeout ó 3 ACKs duplicados) no se puede saber, ya que tanto TCP Tahoe como TCP Reno reaccionan de esta forma si el evento de pérdida fuera por timeout.

En relación a la Notificación Explícita de Congestión (ECN),

- E) ¿Qué debería hacer el nivel de transporte TCP de un sistema final si recibiera datagramas IP con los bits ECN=11? (2 puntos)

Mandaría un segmento TCP con el flag ECE=1, con el fin de avisar al nivel de transporte TCP del sistema final remoto que se ha detectado congestión en la red.

- F) ¿Qué debería hacer el nivel de transporte TCP de un sistema final cuando se le indica que la red está congestionada? (2 puntos)

Debería disminuir el valor de su ventana de congestión e indicar este hecho al nivel de transporte TCP del sistema final remoto, activando para ello el flag CWR.



Departamento de Ingeniería  
Telemática y Electrónica

# Computer Networks

## Internet Transport Layer

TCP5-T(NP): Laboratory Exercises

Spring semester, 2022/2023



## Ejercicio 1

En las máquinas RROO-A y RROO-B se ha configurado un servicio de TFTP similar al utilizado en la práctica TCP1-L(P). Las siguientes figuras muestran parte de los ficheros usados en esta configuración:

```
[root@RROO-A /]# cat /etc/services | grep "Trivial File Transfer" | grep udp
tftp          69/udp      #Trivial File Transfer
[root@RROO-A /]#
```

```
[root@RROO-A /]# cat /etc/inetd.conf | grep tftp
tftp    dgram   udp     wait    root    /usr/libexec/tftpd      tftpd -l -s /tftpboot
#tftp    dgram   udp6    wait    root    /usr/libexec/tftpd      tftpd -l -s /tftpboot
[root@RROO-A /]#
```

```
[root@RROO-B ~]# cat /etc/inetd.conf | grep tftp
mitftp  dgram   udp     wait    root    /usr/libexec/tftpd      tftpd -l -s /repositorio
#tftp    dgram   udp6    wait    root    /usr/libexec/tftpd      tftpd -l -s /tftpboot
[root@RROO-B ~]#
```

Con estos servicios configurados y funcionando correctamente, se ejecutan los siguientes comandos antes de proceder a realizar transferencias de ficheros:

```
[root@RROO-A /]# ifconfig em0
em0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=9b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM>
        ether 00:50:56:aa:aa:00
        inet 20.0.0.1 netmask 0xffffffff broadcast 20.0.0.7
            nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
            media: Ethernet autoselect (1000baseT <full-duplex>)
            status: active
[root@RROO-A /]#
```

```
[root@RROO-B ~]# ifconfig em0
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=9b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM>
        ether 00:50:56:bb:bb:00
        inet 20.0.0.6 netmask 0xffffffff broadcast 20.0.0.7
            nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
            media: Ethernet autoselect (1000baseT <full-duplex>)
            status: active
[root@RROO-B ~]#
```

En esta situación, ambas máquinas transfieren un fichero de la una a la otra, dando lugar a las capturas de tráfico mostradas en las siguientes figuras:

Captura 1:

Source	Destination	Protocol	Length	Info
20.0.0.1	20.0.0.6	UDP	81	Source port: 52490 Destination port: 60000
20.0.0.6	20.0.0.1	UDP	66	Source port: 55932 Destination port: 52490
20.0.0.1	20.0.0.6	UDP	46	Source port: 52490 Destination port: 55932
20.0.0.6	20.0.0.1	UDP	558	Source port: 55932 Destination port: 52490
20.0.0.1	20.0.0.6	UDP	46	Source port: 52490 Destination port: 55932
20.0.0.6	20.0.0.1	UDP	558	Source port: 55932 Destination port: 52490
20.0.0.1	20.0.0.6	UDP	46	Source port: 52490 Destination port: 55932
20.0.0.6	20.0.0.1	UDP	146	Source port: 55932 Destination port: 52490
20.0.0.1	20.0.0.6	UDP	46	Source port: 52490 Destination port: 55932

```

▶ Frame 4: 558 bytes on wire (4464 bits), 558 bytes captured (4464 bits) on interface 0
▶ Ethernet II, Src: Vmware_bb:bb:00 (00:50:56:bb:bb:00), Dst: Vmware_aa:aa:00 (00:50:56:aa:aa:00)
▶ Internet Protocol Version 4, Src: 20.0.0.6 (20.0.0.6), Dst: 20.0.0.1 (20.0.0.1)
▶ User Datagram Protocol, Src Port: 55932 (55932), Dst Port: 52490 (52490)
▶ Data (516 bytes)

```

Captura 2:

Source	Destination	Protocol	Length	Info
20.0.0.6	20.0.0.1	TFTP	80	Read Request, File: file100, Transfer type: netas
20.0.0.1	20.0.0.6	TFTP	65	Option Acknowledgement, tsize\000=100\000, rollov
20.0.0.6	20.0.0.1	TFTP	46	Acknowledgement, Block: 0
20.0.0.1	20.0.0.6	TFTP	146	Data Packet, Block: 1 (last)
20.0.0.6	20.0.0.1	TFTP	46	Acknowledgement, Block: 1

```

▶ Frame 1: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0
▶ Ethernet II, Src: Vmware_bb:bb:00 (00:50:56:bb:bb:00), Dst: Vmware_aa:aa:00 (00:50:56:aa:aa:00)
▶ Internet Protocol Version 4, Src: 20.0.0.6 (20.0.0.6), Dst: 20.0.0.1 (20.0.0.1)
▶ User Datagram Protocol, Src Port: 26898 (26898), Dst Port: 69 (69)
▶ Trivial File Transfer Protocol

```

Basándose en esta información.

**Se pide:**

- A) Indicar el nombre de la máquina que actúa como servidor en la transferencia del fichero *file100* (idéntico al usado en la práctica).

20.0.0.1 → RR00-A

- B) Indicar el tamaño (en bytes) del fichero que se transfiere en la Captura 1. (Se deben indicar los cálculos realizados).

$$512 + 512 + 100 = 1124 \text{ bytes}$$

- C) Indicar en qué directorio y en qué máquina (que actúa como servidor) debe estar almacenado el fichero que se transfiere en la Captura 1.

10.0.0.6:52490 /repositorio de RROO-B

- D) Indicar la línea que se ha tenido que incluir en el fichero /etc/services de la máquina RROO-B para que funcione correctamente su servicio TFTP.

/etc/services: tftp 69/tcp

- E) Indicar porqué el campo "Protocol" de la Captura 1 es UDP es vez de ser TFTP.

No identifica el protocolo de aplicación encapsulado en UDP debido a que no usa el "Well-known port" 69, reservado para TFTP.

**Ejercicio 2**

Teniendo en cuenta la información de la siguiente figura, en la que se muestra una captura de tráfico TCP:

Source	Destination	Protocol	Length	Info
20.0.0.1	20.0.0.6	TCP	74	64809 → 7 [SYN] Seq=0 Win=65535 Len=0 MSS=960 WS=64 SACK_
20.0.0.6	20.0.0.1	TCP	74	7 → 64809 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=960
20.0.0.1	20.0.0.6	TCP	66	64809 → 7 [ACK] Seq=1 Ack=1 Win=66304 Len=0 TSval=2752968
20.0.0.1	20.0.0.6	TCP	74	41163 → 7 [SYN] Seq=0 Win=65535 Len=0 MSS=960 WS=64 SACK_
20.0.0.6	20.0.0.1	TCP	74	7 → 41163 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=960
20.0.0.1	20.0.0.6	TCP	66	41163 → 7 [ACK] Seq=1 Ack=1 Win=66304 Len=0 TSval=2753996
20.0.0.1	20.0.0.6	TCP	74	25209 → 7 [SYN] Seq=0 Win=65535 Len=0 MSS=960 WS=64 SACK_
20.0.0.6	20.0.0.1	TCP	74	7 → 25209 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=960
20.0.0.1	20.0.0.6	TCP	66	25209 → 7 [ACK] Seq=1 Ack=1 Win=66304 Len=0 TSval=2754978
20.0.0.1	20.0.0.6	TCP	80	25209 → 7 [PSH, ACK] Seq=1 Ack=1 Win=66304 Len= TSval=2
20.0.0.6	20.0.0.1	TCP	80	7 → 25209 [PSH, ACK] Seq=1 Ack=15 Win=66304 Len= TSval=
20.0.0.1	20.0.0.6	TCP	66	25209 → 7 [ACK] Seq=15 Ack=15 Win=66304 Len=0 TSval=27556
20.0.0.1	20.0.0.6	TCP	84	41163 → 7 [PSH, ACK] Seq=1 Ack=1 Win=66304 Len= TSval=2
20.0.0.6	20.0.0.1	TCP	84	7 → 41163 [PSH, ACK] Seq=1 Ack=19 Win=66304 Len= TSval=
20.0.0.1	20.0.0.6	TCP	66	41163 → 7 [ACK] Seq=19 Ack=19 Win=66304 Len=0 TSval=27566
20.0.0.1	20.0.0.6	TCP	80	64809 → 7 [PSH, ACK] Seq=1 Ack=1 Win=66304 Len= TSval=2
20.0.0.6	20.0.0.1	TCP	80	7 → 64809 [PSH, ACK] Seq=1 Ack=15 Win=66304 Len= TSval=
20.0.0.1	20.0.0.6	TCP	66	64809 → 7 [ACK] Seq=15 Ack=15 Win=66304 Len=0 TSval=27573
20.0.0.1	20.0.0.6	TCP	66	41163 → 7 [FIN, ACK] Seq=19 Ack=19 Win=66304 Len=0 TSval=
20.0.0.6	20.0.0.1	TCP	66	7 → 41163 [ACK] Seq=19 Ack=20 Win=66304 Len=0 TSval=13283
20.0.0.6	20.0.0.1	TCP	66	7 → 41163 [FIN, ACK] Seq=19 Ack=20 Win=66304 Len=0 TSval=
20.0.0.1	20.0.0.6	TCP	66	41163 → 7 [ACK] Seq=20 Ack=20 Win=66304 Len=0 TSval=27586
20.0.0.1	20.0.0.6	TCP	66	64809 → 7 [FIN, ACK] Seq=15 Ack=15 Win=66304 Len=0 TSval=
20.0.0.6	20.0.0.1	TCP	66	7 → 64809 [ACK] Seq=15 Ack=16 Win=66304 Len=0 TSval=18795
20.0.0.6	20.0.0.1	TCP	66	7 → 64809 [FIN, ACK] Seq=15 Ack=16 Win=66304 Len=0 TSval=
20.0.0.1	20.0.0.6	TCP	66	64809 → 7 [ACK] Seq=16 Ack=16 Win=66304 Len=0 TSval=27585
20.0.0.1	20.0.0.6	TCP	66	25209 → 7 [FIN, ACK] Seq=15 Ack=15 Win=66304 Len=0 TSval=
20.0.0.6	20.0.0.1	TCP	66	7 → 25209 [ACK] Seq=15 Ack=16 Win=66304 Len=0 TSval=34386
20.0.0.6	20.0.0.1	TCP	66	7 → 25209 [FIN, ACK] Seq=15 Ack=16 Win=66304 Len=0 TSval=
20.0.0.1	20.0.0.6	TCP	66	25209 → 7 [ACK] Seq=16 Ack=16 Win=66304 Len=0 TSval=27591

**Se pide:**

- A) Indicar la dirección de los socket cliente y servidor de las conexiones que se hayan establecido.

Clients (C)  
 20.0.0.1:64809  
 20.0.0.6:41163  
 20.0.0.1:25209

Server (S)  
 20.0.0.6:7

- B) Indicar la cantidad de datos (en bytes) que se han intercambiado en cada conexión.

①	$C: 15 - 1 = 14 \text{ bytes}$	$S: 14$	Total: 28 bytes
②	$C: 19 - 1 = 18 \text{ bytes}$	$S: 18$	Total: 36 bytes
③	$C: 15 - 1 = 14 \text{ bytes}$	$S: 14$	Total: 78 bytes

- C) Indicar el valor de la MTU del nivel de enlace, sabiendo que el MSS de cada conexión se ha ajustado para evitar fragmentación en los niveles inferiores.

$$MSS = 960$$

$$MTU = MSS + 40 = 1000 \text{ bytes}$$

Antes de realizar la captura anterior se observó la siguiente información:

```
[root@RR00-A ~]# netstat -na | grep tcp4 | grep *.7
tcp4       0      0  *.*                  LISTEN
```

- D) Indicar qué habría ocurrido si al ejecutar el comando anterior no se hubiera obtenido ninguna información.

Sería indicativo de que no habría ninguna aplicación "escuchando" en el puerto 7 y por lo tanto todas las solicitudes de conexión a él serían respondidas con un segmento TCP con el flag RST activado.

### Ejercicio 3

En la máquina RROO-B, con dirección IP 10.2.2.2, se desea configurar un servicio de TFTP alternativo al utilizado en la práctica TCP1-L(P). En esta práctica, el servicio de TFTP se ofrecía en su puerto por defecto (*well-known port*) y su configuración, en el fichero correspondiente, era la siguiente:

```
tftp    dgram   udp     wait    root    /usr/libexec/tftpd      tftpd -l -s /tftpboot
```

En su nueva configuración, el servicio se llamará **tftpalt**, se ofrecerá en el puerto **48500** y servirá los ficheros que se encuentren en el directorio **/documentos**

Basándose en esta información.

**Se pide:**

- A) Indicar, **de forma detallada**, cómo se configuraría este servicio de TFTP alternativo.

En el fichero /etc/services se debe añadir la siguiente línea:  
tftpalt 48500/udp  
En el fichero /etc/inetd.conf se debe añadir la siguiente línea:  
tftpalt dgram udp wait root /usr/libexec/tftpd tftpd -l -s /documentos  
Se debe reiniciar el demonio inetd mediante el comando:  
/etc/rc.d/inetd restart  
Se debe crear el directorio /documentos mediante el comando:  
mkdir /documentos

Después de realizar la configuración anterior, y con el fin de probar su correcto funcionamiento, la máquina RROO-B ejecuta el siguiente comando:

```
printf "Prueba TFTP alternativo" > /documentos/fichprueba
```

- B) Completar la siguiente plantilla en la que se muestra cómo la máquina RROO-A (cliente TFTP), con dirección IP 10.1.1.1, obtiene el fichero anterior.

```
[root@RROO-A ~]# tftp 10.2.2.48500
tftp> get fichprueba
Received 23 bytes during 0.0 seconds in 1 blocks
tftp> quit
[root@RROO-A ~]#
```

Después de comprobar que el servicio de TFTP alternativo funciona correctamente, la máquina RROO-A realiza la descarga de un fichero de 2048 bytes, almacenado previamente en el directorio /documentos de la máquina RROO-B.

Durante esta transferencia, en la máquina RROO-B se realiza una captura del tráfico intercambiado entre ambas máquinas.

- C) Completar sobre la plantilla los datos que faltan en la captura realizada.

Nota: Tener en cuenta que el tamaño de la cabecera TFTP para los paquetes de datos es de 4 bytes.

Source	Destination	Protocol	Length	Info
10.1.1.1	10.2.2.2		81	Source port: <b>37748</b> Destination port: <b>48500</b>
10.2.2.2	10.1.1.1		66	Source port: <b>26206</b> Destination port: <b>37748</b>
10.1.1.1	10.2.2.2		60	Source port: <b>37748</b> Destination port: <b>26206</b>
10.2.2.2	10.1.1.1		558	Source port: <b>26206</b> Destination port: <b>37748</b>
10.1.1.1	10.2.2.2		60	Source port: <b>37748</b> Destination port: <b>26206</b>
10.2.2.2	10.1.1.1		558	Source port: <b>26206</b> Destination port: <b>37748</b>
10.1.1.1	10.2.2.2		60	Source port: <b>37748</b> Destination port: <b>26206</b>
10.2.2.2	10.1.1.1		558	Source port: <b>26206</b> Destination port: <b>37748</b>
10.1.1.1	10.2.2.2		60	Source port: <b>37748</b> Destination port: <b>26206</b>
10.2.2.2	10.1.1.1		558	Source port: <b>26206</b> Destination port: <b>37748</b>
10.1.1.1	10.2.2.2		60	Source port: <b>37748</b> Destination port: <b>26206</b>
10.2.2.2	10.1.1.1		46	Source port: <b>26206</b> Destination port: <b>37748</b>
10.1.1.1	10.2.2.2		60	Source port: <b>37748</b> Destination port: <b>26206</b>

2648 - 512 - 512 - 512 - 512 = 0      Último paquete, 46 bytes.

↓      ↓      ↓      ↓

558      558      558      558

↑      ↑      ↑      ↑

cuando se envían más de 512 bytes, se envía uno más.

## Ejercicio 4

Teniendo en cuenta la información de la siguiente figura, en la que se muestra una captura de tráfico TCP entre un cliente y un servidor:

No.	Source	Destination	Protocol	Length	Info
1	10.1.1.1	10.2.2.2	TCP	74	12558→60000 [SYN] Seq=0
2	10.2.2.2	10.1.1.1	TCP	74	60000→12558 [SYN, ACK] Seq=0 Ack=1
3	10.1.1.1	10.2.2.2	TCP	66	12558→60000 [ACK] Seq=1 Ack=1 Win=65536
4	10.2.2.2	10.1.1.1	TCP	73	60000→12558 [PSH, ACK] Seq=1 Ack=1 Win=65536
5	10.2.2.2	10.1.1.1	TCP	66	60000→12558 [FIN, ACK] Seq=8 Ack=1 Win=65536
6	10.1.1.1	10.2.2.2	TCP	66	12558→60000 [ACK] Seq=1 Ack=9
7	10.1.1.1	10.2.2.2	TCP	66	12558→60000 [FIN, ACK] Seq=1 Ack=9 Win=65536
8	10.2.2.2	10.1.1.1	TCP	66	60000→12558 [ACK] Seq=9 Ack=2 Win=65536

→ E de la conexión → Envío de datos } Fin

Options: (20 bytes), Maximum segment size, No-Operation (NOP), Window scale, SACK permitted, Timestamps

- ▶ Maximum segment size: 464 bytes MSS
- ▶ No-Operation (NOP)
- ▶ Window scale: 6
- ▶ TCP SACK Permitted Option: True
- ▶ Timestamps: TSval 37006525, TSecr 0

**Se pide:**

- A) Indicar la dirección de los socket cliente y servidor de las conexiones que se hayan establecido.

Cliente: 10.11.1.12:558  
Servidor: 10.7.7.2:60000

- B) Indicar la cantidad de datos (en bytes) que se han intercambiado y en qué número de trama se ha hecho.

$9 - 2 = 7$  bytes Trama 4.  
↓  
FIN + SYN

- C) Indicar el valor de la MTU del nivel de enlace, sabiendo que el MSS se ha ajustado para evitar fragmentación en los niveles inferiores.

$$\begin{aligned}MSS &= MTU - 40 \\MSS &= 464 \\MTU &= 504 \text{ bytes}\end{aligned}$$

- D) Indicar el valor, en hexadecimal, del campo “Receive Window” presente en la cabecera TCP de la trama nº 4

$$\begin{aligned} \text{WS} &= 2^6 = 64 \\ \text{RW} &= 65536 / 64 = 1024 \rightarrow 2^{10} \\ \text{RW} &= \underline{\underline{0 \times 0400}} \end{aligned}$$

Teniendo en cuenta la información de la siguiente figura, en la que se muestra el resultado de ejecutar el comando `netstat -na | grep tcp4` en una máquina con dirección IP 10.2.2.2

tcp4	0	49658	10.2.2.2.19	10.3.3.3.17170	ESTABLISHED
tcp4	0	6	10.2.2.2.19	10.1.1.1.19011	ESTABLISHED
tcp4	0	0	10.2.2.2.7	10.1.1.1.45506	ESTABLISHED
tcp4	0	0	10.2.2.2.7	10.3.3.3.11526	ESTABLISHED
tcp4	0	0	*.110	.*.*	LISTEN
tcp4	0	0	*.19	.*.*	LISTEN
tcp4	0	0	*.7	.*.*	LISTEN
tcp4	0	0	*.13	.*.*	LISTEN
tcp4	0	0	*.60000	.*.*	LISTEN
tcp4	0	0	*.23	.*.*	LISTEN

**Se pide:**

- E) Indicar qué comandos se han ejecutado y en qué máquinas para tener establecidas las conexiones que se muestran en la figura anterior.

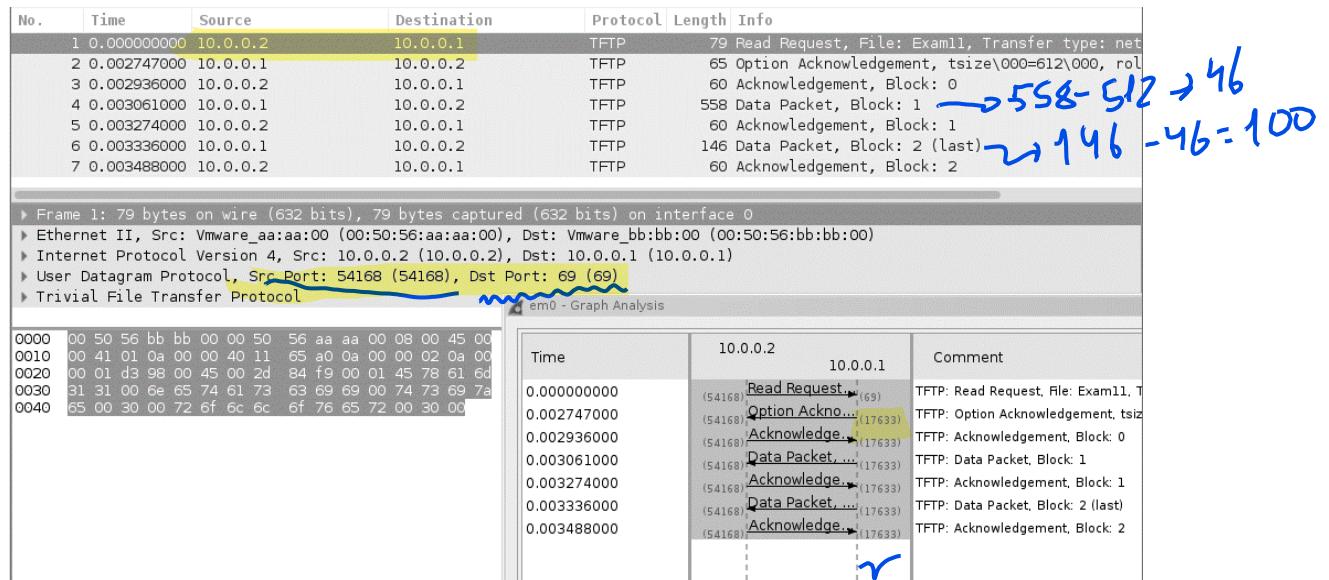
En la máquina 102.2.2  
 Telnet 10.3.3.3 17170  
 Telnet 10.1.1.1 19011  
 Telnet 10.1.1.1 45506  
 Telnet 10.3.3.3 11526

- F) Indicar qué comando ejecutaría y en qué máquina para que la máquina con dirección IP 10.2.2.2 respondiera con un segmento TCP de Reset.

Telnet 10.2.2.2 (cualquier puerto que no esté en listen.)

## Ejercicio 5

La siguiente captura muestra un intercambio de tráfico en el que un cliente obtiene un fichero de un servidor, de forma similar a los obtenidos en la práctica TCP1-L(P): UDP Protocol Scenarios



### Se pide:

- A) Indicar el tamaño, en bytes, del fichero transferido

$$512 + 100 = 612 \text{ bytes}$$

- B) Indicar la dirección del socket del cliente durante la transferencia de datos (dir\_ip:puerto)

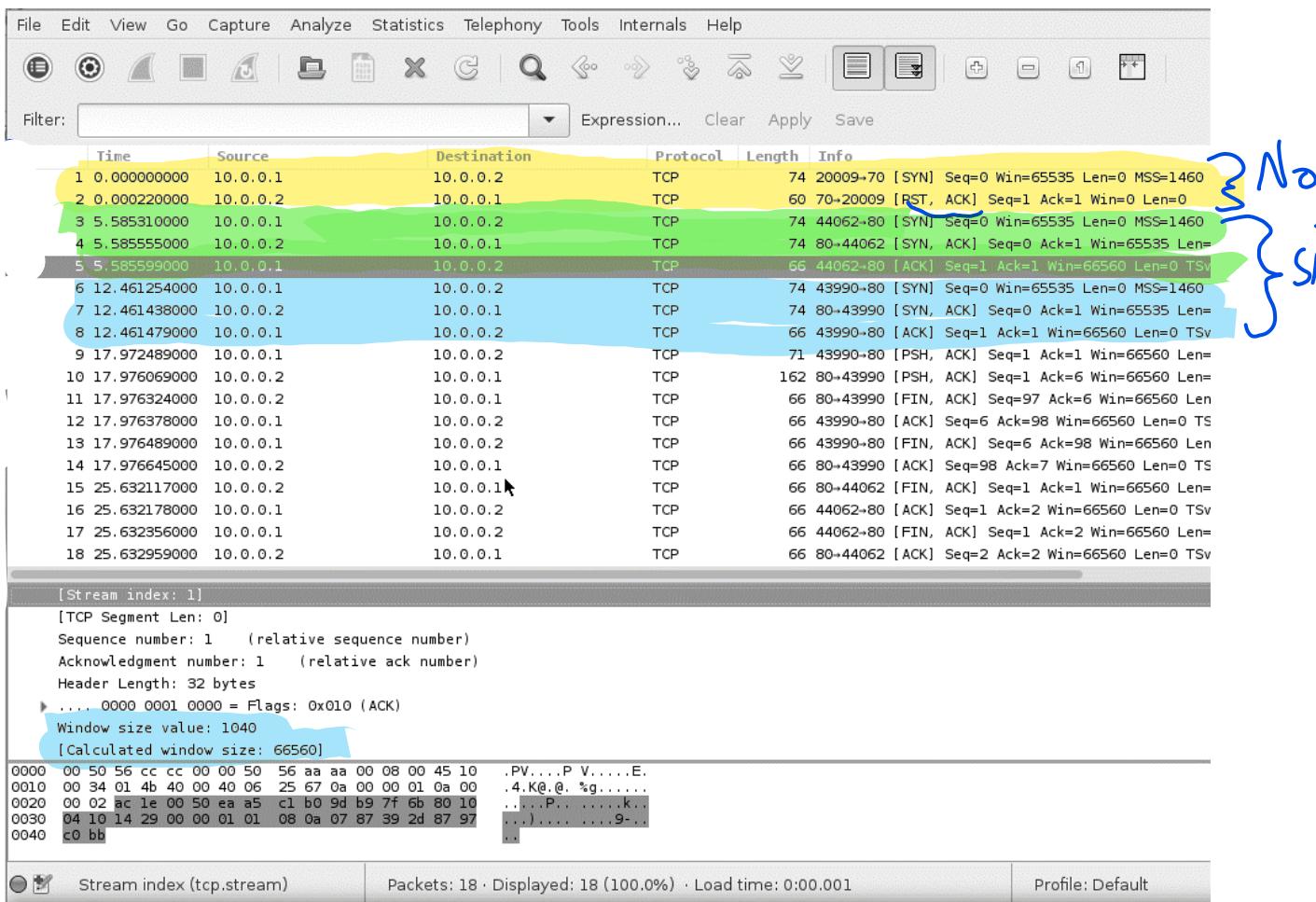
10.0.0.2:54168

- C) Indicar la dirección del socket con el que se contacta al servidor (dir\_ip:puerto)

10.0.0.1:69

10.0.0.1:17633 (transferencia)

La siguiente captura muestra un intercambio de tráfico TCP entre un cliente y un servidor, de forma similar al obtenido durante la práctica TCP2-L(P): TCP Protocol Scenarios



### Se pide:

- A) Indicar cuántos intentos de conexión, completados o no, se han producido y, de ellos, cuántos se corresponden con servicios HTTP → 80

3 intentos, se completan los 2 +HTTP.

- B) Indicar el socket del servidor (dir\_ip:puerto) en el caso de que alguno de los intentos de conexión anteriores no se correspondiera con un servicio HTTP

10.0.0.2:70

- C) Indicar, para la primera conexión **completada**, la cantidad máxima de bytes que se pueden intercambiar el cliente y el servidor en un único segmento TCP.  
¿Dentro de qué campo de la cabecera TCP se intercambia este dato?

MSS: 1460 bytes  
 ↪ campo Opciones

- D) Indicar, para la primera conexión **completada**, el valor del factor de escalado de ventana

$$\frac{66560}{1040} = 64 = 2^6 \rightarrow 6$$

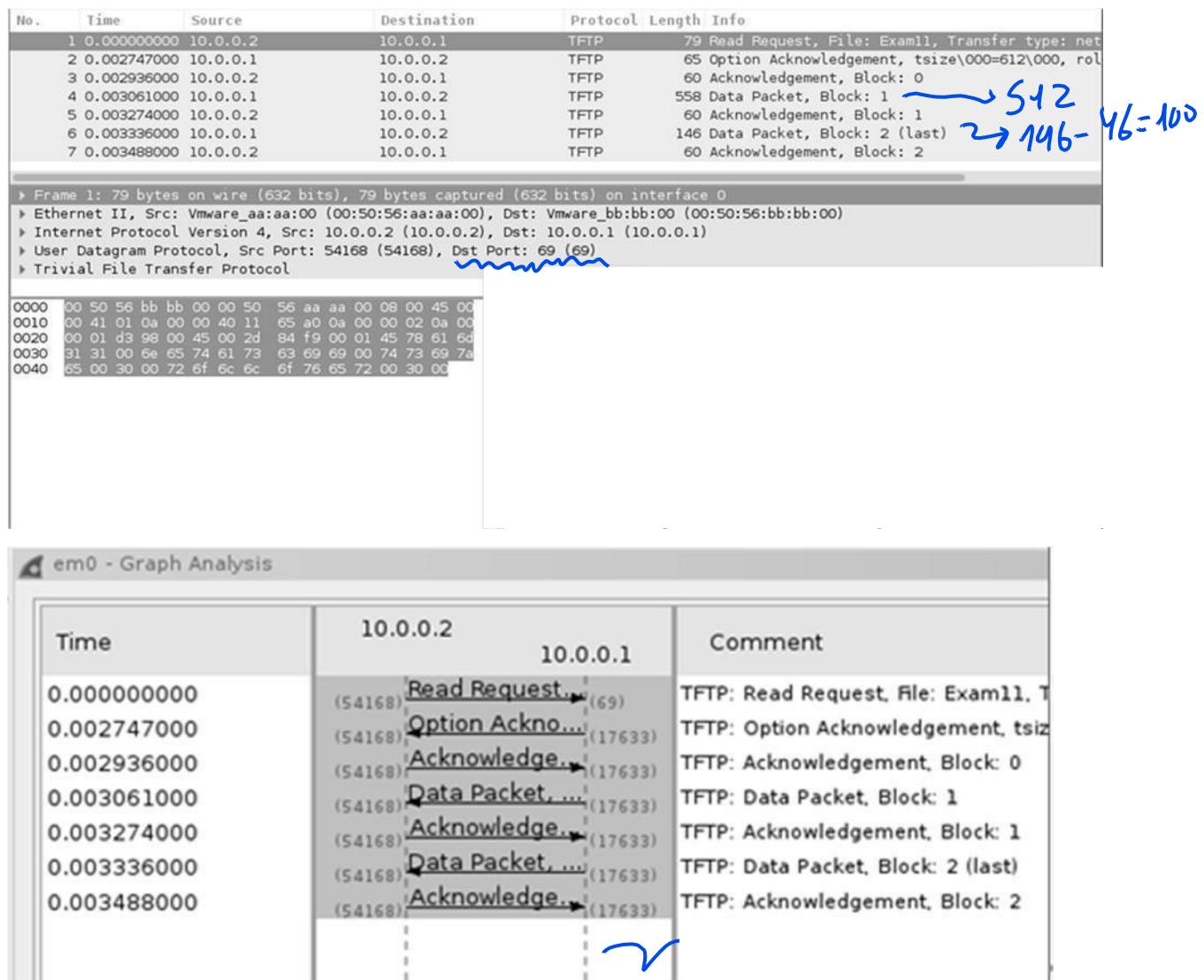
- E) Indicar cuánto tiempo permanecerá el servidor en el estado TIME\_WAIT si se estableciera una nueva conexión después de ejecutar de forma satisfactoria el comando

`sysctl net.inet.tcp.msl=1024`

$$T_{meout}(2 \times MSL) = 2048 \text{ ms}$$

## Ejercicio 6

La siguiente figura muestra una captura, realizada con Wireshark, del intercambio de tráfico que se produce cuando un cliente solicita un fichero a un servidor TFTP, de forma similar al proceso realizado en la práctica TCP1-L(P): UDP Protocol Scenarios.



**Se pide:**

- A) Indicar el tamaño en bytes del fichero transferido (2 puntos)

512 + 100 ~ 612

- B) Indicar los sockets del servidor que utiliza el cliente para comunicarse con él. Usar el formato “dirección ip:puerto”

10.0.0.1.69

10.0.0.1.17633 (Transferencia)

- C) Indicar, **razonando la respuesta**, si el proceso tftpd se encuentra arrancado en el servidor de forma permanente

El proceso o daemon tftpd NO se encuentra arrancado de forma permanente. Cuando se recibe una petición (en este caso la petición del servicio tftp) el proceso o “super servidor” inetd determina (consultando el fichero inetc.conf) que el daemon tftpd es el encargado de atenderla y se arranca en ese momento

- D) Indicar mediante qué variable y en qué fichero se establece que el proceso inetc se arranque al iniciar la máquina virtual

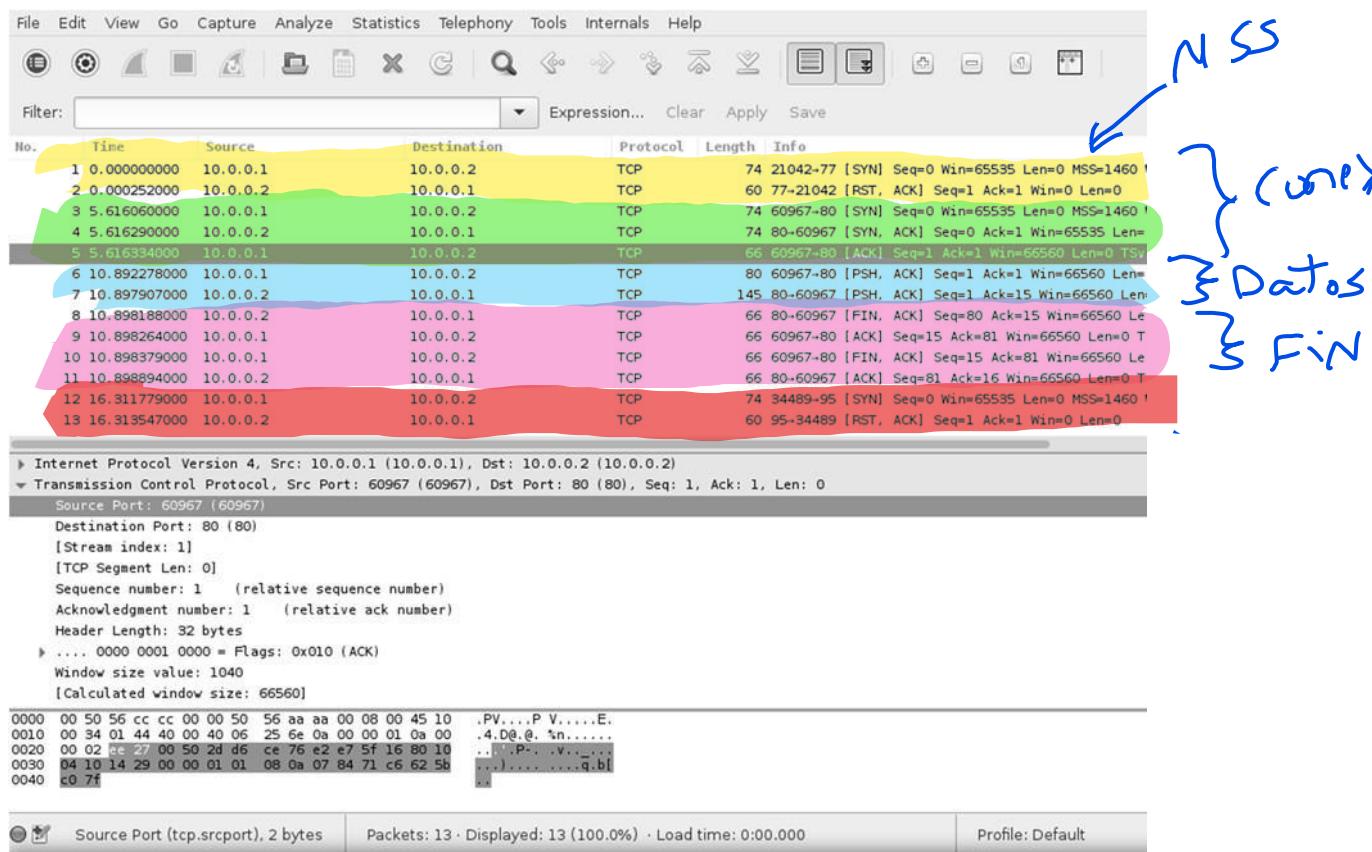
Mediante la variable: inetc\_enable="YES"  
En el fichero: /etc/rc.conf

- E) Rellenar sobre la siguiente plantilla los campos en blanco con el contenido necesario para que la maquina cliente obtenga el fichero anterior.

```
[root@RROO-A ~]# tftp [REDACTED]  
tftp> [REDACTED]  
Received [REDACTED] bytes during 0.1 seconds in [REDACTED] blocks  
[REDACTED]  
tftp> quit  
[root@RROO-A ~]#
```

## Ejercicio 7

La siguiente figura muestra una captura, realizada con Wireshark, de un intercambio de tráfico TCP, similar al proceso realizado en la práctica TCP2-L(P): TCP Protocol Scenarios.



### Se pide:

- A) Indicar cuántos intentos de conexión, completados o no, se han producido. Si alguno de ellos estuviera relacionado con el servicio http, indicar también el socket del cliente en el formato "dirección ip:puerto"

3 intentos de conexión, 1 de ellos es HTTP. 10.0.0.1:60967

- B) En el primer intento de conexión, indicar cuál sería la cantidad máxima de bytes que se hubieran podido intercambiar el cliente y el servidor en un solo segmento TCP. ¿Qué variable indica ese valor y en qué campo de la cabecera TCP se envía?

MSS : 1460 bytes

- C) Indicar, en el siguiente intento de conexión, el valor del factor de escala de la ventana [Windows size scaling factor]

$$\frac{66560}{1040} = 64 \rightarrow 2^6 \rightarrow \boxed{6}$$

- D) Teniendo en cuenta todas las conexiones, completadas o no, indicar cuántos bytes se han transferido en cada sentido de la comunicación

Cliente : 14  
Servidor : 79

- E) En el primer intento de conexión completado, indicar en qué momento (antes o después de qué número de trama) la máquina con dirección IP 10.0.0.2 pasa al estado TIME-WAIT

Pasa a ese estado después de enviar un ACK como respuesta al FINACK recibido por la máquina 10.0.0.1, es decir, después de enviar la PDU Nº 11

- F) Indicar el tiempo que permanecería en ese estado, sabiendo que en esa máquina se ha ejecutado previamente el comando:  
`sysctl.net.inet.tcp.msl=2048`

Permanecería en TIME-WAIT  $2^*MSL$ , esto es:  $2048^*2$  4096 ms

	8 de julio de 2022	LABORATORIO
--	--------------------	-------------

### Ejercicio 2 (10 puntos) (15 min)

Las siguientes figuras muestran información relacionada con la realización de las prácticas de laboratorio TCP1-L(P) y TCP2-L(P) del nivel de transporte.

Las figuras 1 y 2 muestran la salida parcial del contenido de los ficheros `/etc/services` y `/etc/inetd.conf` de la máquina utilizada como servidor TFTP durante la práctica TCP1-L(P) y la figura 3 muestra una captura realizada durante la práctica TCP2-L(P).

tftp	69/tcp	#Trivial File Transfer
tftp	69/udp	#Trivial File Transfer
subntbcst_tftp	247/tcp	#subntbcst_tftp
subntbcst_tftp	247/udp	#subntbcst_tftp

Figura 1. Verificación de la asignación de puertos al servicio TFTP en el fichero `/etc/services` (práctica TCP1-L(P))

tftp	dgram	udp	wait	root	/usr/libexec/tftpd	tftpd -l -s /tftpboot
#tftp	dgram	udp6	wait	root	/usr/libexec/tftpd	tftpd -l -s /tftpboot

Figura 2. Verificación de la asignación del daemon tftp con el servicio TFTP en el fichero `/etc/inetd.conf` (práctica TCP1-L(P))



No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	20.0.0.1	20.0.0.3	TCP	74	12351>80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3291412 TSecr=37996
2	0.0002510000	20.0.0.3	20.0.0.1	TCP	74	80->12351 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3291412 TSecr=37996
3	0.0002990000	20.0.0.1	20.0.0.3	TCP	66	12351>80 [ACK] Seq=1 Ack=1 Win=66560 Len=0 TSval=3291412 TSecr=37996
4	5.1265170000	20.0.0.1	20.0.0.3	TCP	74	28850>80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3299524 TSecr=1931084865
5	5.1267600000	20.0.0.3	20.0.0.1	TCP	74	80->28850 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3299524 TSecr=1931084865
6	5.1268897000	20.0.0.1	20.0.0.3	TCP	66	28850>80 [ACK] Seq=1 Ack=1 Win=66560 Len=0 TSval=3299532 TSecr=1931084865
7	5.1121680000	20.0.0.1	20.0.0.3	TCP	71	28850>80 [PSH, ACK] Seq=1 Ack=1 Win=66560 Len=0 TSval=3299524 TSecr=1931084865
8	8.2157380000	20.0.0.3	20.0.0.1	TCP	66	80->28850 [ACK] Seq=1 Ack=6 Win=66560 Len=0 TSval=1931084863 TSecr=3299524
9	8.2272030000	20.0.0.3	20.0.0.1	TCP	267	80->28850 [PSH, ACK] Seq=1 Ack=6 Win=66560 Len=0 TSval=1931084865 TSecr=3299524
10	8.2285420000	20.0.0.3	20.0.0.1	TCP	66	80->28850 [FIN, ACK] Seq=202 Ack=6 Win=66560 Len=0 TSval=1931084875 TSecr=3299524
11	8.2287000000	20.0.0.1	20.0.0.3	TCP	66	28850>80 [ACK] Seq=6 Ack=203 Win=66560 Len=0 TSval=32995640 TSecr=1931084875
12	8.2289430000	20.0.0.1	20.0.0.3	TCP	66	28850>80 [FIN, ACK] Seq=6 Ack=203 Win=66560 Len=0 TSval=32995642 TSecr=1931084875
13	8.2293430000	20.0.0.3	20.0.0.1	TCP	66	80->28850 [ACK] Seq=203 Ack=7 Win=66560 Len=0 TSval=1931084875 TSecr=3299524
14	10.6707950000	20.0.0.1	20.0.0.3	TCP	71	12351>80 [PSH, ACK] Seq=1 Ack=1 Win=66560 Len=0 TSval=3302083 TSecr=3799927593
15	10.6714370000	20.0.0.3	20.0.0.1	TCP	267	80->12351 [PSH, ACK] Seq=1 Ack=6 Win=66560 Len=0 TSval=3799927593 TSecr=3799927593
16	10.6715550000	20.0.0.3	20.0.0.1	TCP	66	80->12351 [FIN, ACK] Seq=202 Ack=6 Win=66560 Len=0 TSval=3799927593 TSecr=3799927593
17	10.6716060000	20.0.0.1	20.0.0.3	TCP	66	12351>80 [ACK] Seq=6 Ack=203 Win=66368 Len=0 TSval=3302083 TSecr=3799927593
18	10.6718930000	20.0.0.1	20.0.0.3	TCP	66	12351>80 [FIN, ACK] Seq=6 Ack=203 Win=66560 Len=0 TSval=3302083 TSecr=3799927593
19	10.6720280000	20.0.0.3	20.0.0.1	TCP	66	80->12351 [ACK] Seq=203 Ack=7 Win=66560 Len=0 TSval=3799927593 TSecr=3799927593
20	1070.369230X	Vmware_aa:aa:Broadcas		ARP	42	Who has 20.0.0.3? Tell 20.0.0.1
21	1070.369524X	Vmware_cc:cc:Vmware_aa:aa:		ARP	60	20.0.0.3 is at 00:50:56:cc:cc:00
22	1070.369594X	20.0.0.1	20.0.0.3	TCP	74	65362>97 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=3302083 TSecr=3799927593
23	1070.369806X	20.0.0.3	20.0.0.1	TCP	60	97->65362 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Figura 3. Captura de tráfico realizada durante la práctica TCP2-L(P)

**Se pide:**

- A) Indicar el protocolo del nivel de transporte y el puerto utilizado por los clientes para solicitar un fichero al servidor TFTP. (1 punto)

La Tal y como se muestra en los contenidos de los ficheros /etc/services y /etc/inetd.conf  
los clientes enviarán tráfico UDP al puerto 69 del servidor TFTP

- B) Indicar razonadamente el nombre del directorio que contendrá los ficheros que servirá el servidor TFTP. (2 puntos)

Tal como se indica en el dicherio /etc/inetd.conf al invocar el damenon tftp con el argumento "-s /tftpboot" se indica que el directorio "/tftpboot almacenará los ficheros servidos por TFTP.

- C) Indicar las conexiones TCP que hay en el tráfico que se muestra en la figura 3. Para cada conexión, indicar la IP y el puerto cliente y la IP y puerto del servidor. (2 puntos)

2 conexiones:

① Cliente: 20.0.0.1: 12351 y Servidor: 20.0.0.3: 80

② Cliente: 20.0.0.1: 28850 y Servidor 20.0.0.3: 80

- D) Indicar la cantidad de datos (bytes) que contienen los segmentos TCP de las tramas 7, 9, 14 y 15 de la captura de tráfico que se muestra en la figura 3. (2 puntos)

Trama 7: 5 bytes

Trama 9: 201 bytes

Trama 14: 5 bytes

Trama 15: 201 bytes

- E) Indicar el comando telnet completo que generó el tráfico de las tramas 22 y 23 de la figura 3. ¿Qué tipo de segmento TCP contiene la trama 23 y por qué se ha provocado?. (3 puntos)

Telnet 20.0.0.3:97

RST → el puerto 97 no está disponible.

Telemática y Electrónica

9 de julio de 2021

MESA:

**Ejercicio 4 (20 puntos) (15 min)**

La captura que aparece a continuación (similar a las realizadas en el laboratorio de la asignatura) muestra una parte de una comunicación entre una aplicación cliente y una aplicación servidor que han utilizado el protocolo TCP para intercambiar una serie de datos

```

Line Source Destination Protocol Length Info
1 0.000000 130.100.2.1 130.100.2.2 TCP 74 52345>80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERMIT
2 0.000029 130.100.2.2 130.100.2.1 TCP 74 80>52345 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERMIT
3 0.000249 130.100.2.1 130.100.2.2 TCP 66 52345>80 [ACK] Seq=1 Ack=1 Win=66560 Len=0 TStamp=188859509
4 3.133769 130.100.2.1 130.100.2.2 TCP 71 52345>80 [PSH, ACK] Seq=1 Ack=1 Win=66560 Len=5 TStamp=188859509
5 3.137812 130.100.2.2 130.100.2.1 TCP 98 80>52345 [PSH, ACK] Seq=1 Ack=6 Win=66560 Len=32 TStamp=188859509
6 3.138029 130.100.2.2 130.100.2.1 TCP 66 80>52345 [FIN, ACK] Seq=34 Ack=6 Win=66560 Len=0 TStamp=188859509
7 3.138282 130.100.2.1 130.100.2.2 TCP 66 52345>80 [ACK] Seq=34 Ack=34 Win=66560 Len=0 TStamp=188859509
8 3.138394 130.100.2.1 130.100.2.2 TCP 66 52345>80 [FIN, ACK] Seq=34 Ack=34 Win=66560 Len=0 TStamp=188859509
9 3.138913 130.100.2.2 130.100.2.1 TCP 66 80>52345 [ACK] Seq=34 Ack=7 Win=66560 Len=0 TStamp=188859509

> Frame 5: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
> Ethernet II, Src: VMware_cc:cc:00 (00:50:56:cc:cc:00), Dst: VMware_aa:aa:00 (00:50:56:aa:aa:00)
> Internet Protocol Version 4, Src: 130.100.2.2 (130.100.2.2), Dst: 130.100.2.1 (130.100.2.1)
> Transmission Control Protocol, Src Port: 80 (80), Dst Port: 52345 (52345), Seq: 1, Ack: 6, Len: 92
  Source Port: 80 (80)
  Destination Port: 52345 (52345)
  Stream index: 0
  [TCP Segment Len: 92]
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 33 (relative sequence number)]
  Acknowledgment number: 6 (relative ack number)
  Header Length: 32 bytes
  >.... 0000 0001 1000 = Flags: 0x018 (PSH, ACK)
  Window size value: 1040
  [Calculated window size: 66560]
  [Window size scaling factor: 64]
  >.... 0000 0000 0000 = Checksum: 0xa9c9 (Validation disabled)
0020 02 01 00 53 cc 79 01 db 52 f8 c1 b1 9e 05 80 18 .. P.y., R.., ...
0030 04 10 e9 c9 00 00 b1 01 08 0a 48 0c d4 b9 0b 41 .. .... .h...A
0040 64 fg 31 32 33 34 35 36 37 38 39 30 0e 31 32 33 123456 7890.123
0050 34 35 36 37 38 39 30 0e 31 32 33 34 35 36 37 38 4567890. 12345678

```

**Se pide:**

- A) Indicar el socket con el que se contacta al servidor (dir\_ip:puerto) (2 puntos)

130.100.2.1:52345

- B) Indicar el valor del identificador que se ha asignado al puerto en el cliente (2 puntos)

52345

- C) Indicar el valor absoluto (valor real) del número de secuencia (Seq) de la PDU 5, expresado en hexadecimal (4 puntos)

El valor absoluto del número de secuencia en hexadecimal aparece en la cabecera ocupando los 4 bytes anteriores al campo "Acknowledgment number" (este ultimo aparece marcado en la captura), por tanto el valor pedido seria: 01db52f8

- D) Indicar cuántos bytes de datos en total envía el servidor al cliente (3 puntos)

$$34 - 2 = 32 \text{ bytes}$$

- E) Indicar cuántos bytes de control han sido necesarios para enviar estos datos (considere sólo los bytes de control en la, o las, PDU en las que se han enviado los bytes de datos). Especifique la cantidad de bytes de control de cada protocolo de los que intervienen (5 puntos)

$$32 \text{ bytes} + 20 \text{ de IP} + 14 \text{ Ethernet} = 66 \text{ bytes}$$

- F) Indicar el número de la PDU que provocará que el servidor pase al estado TIME-WAIT (4 puntos)

La 9 después de recibir el ACK.

27 de mayo de 2021

MESA:

**Ejercicio 3 (30 puntos) (20 min)**

La siguiente captura muestra un intercambio de tráfico en el que un cliente obtiene un fichero de un servidor, de forma similar a los obtenidos en la práctica TCP1-L(P): UDP Protocol Scenarios.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.2	10.0.0.1	TFTP	79	Read Request, File: Exam1, Transfer type: net
2	0.002747000	10.0.0.1	10.0.0.2	TFTP	65	Option Acknowledgement, tsize\000=612\000, rsize\000=612\000, rel
3	0.002936000	10.0.0.2	10.0.0.1	TFTP	60	Acknowledgement, Block: 0
4	0.003061000	10.0.0.1	10.0.0.2	TFTP	558	Data Packet, Block: 1
5	0.003274000	10.0.0.2	10.0.0.1	TFTP	60	Acknowledgement, Block: 1
6	0.003336000	10.0.0.1	10.0.0.2	TFTP	146	Data Packet, Block: 2 (last)
7	0.003488000	10.0.0.2	10.0.0.1	TFTP	60	Acknowledgement, Block: 2

→ 558 - 46 = 512  
→ 146 - 46 = 100

```

Frame 1: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface 0
Ethernet II, Src: VMware_aa:aa:00 (00:50:56:aa:aa:00), Dst: VMware_bb:bb:00 (00:50:56:bb:bb:00)
Internet Protocol Version 4, Src: 10.0.0.2 (10.0.0.2), Dst: 10.0.0.1 (10.0.0.1)
User Datagram Protocol, Src Port: 54168 (54168), Dst Port: 69 (69)
Trivial File Transfer Protocol

```

Time	Source	Destination	Comment
0000 00:50:56 bb:bb:00 00:00:50:56:aa:aa → 10.0.0.2	VMware_aa:aa:00 (00:50:56:aa:aa:00)	10.0.0.1	TFTP: Read Request, file: Exam1, transfer type: net
0010 00:41:01 aa:00 00:00:40:11 → 10.0.0.1	VMware_bb:bb:00 (00:50:56:bb:bb:00)	10.0.0.2	TFTP: Option Acknowledgement, tsize\000=612\000, rsize\000=612\000, rel
0020 00:01:d3 98:00 00:00:2d → 10.0.0.2	10.0.0.1	VMware_aa:aa:00 (00:50:56:aa:aa:00)	TFTP: Acknowledgement, Block: 0
0030 31:31:00 00:65:74 81:73 → 10.0.0.1	10.0.0.2	VMware_aa:aa:00 (00:50:56:aa:aa:00)	TFTP: Data Packet, Block: 1
0040 25:00:30:00 72:0f 0c:6c → 10.0.0.1	10.0.0.2	VMware_aa:aa:00 (00:50:56:aa:aa:00)	TFTP: Acknowledgement, Block: 1
	10.0.0.2	10.0.0.1	TFTP: Data Packet, Block: 2 (last)
	10.0.0.2	10.0.0.1	TFTP: Acknowledgement, Block: 2

**Se pide:**

- A) Indicar el tamaño, en bytes, del fichero transferido (3 puntos)

$$512 + 100 = 612 \text{ bytes}$$

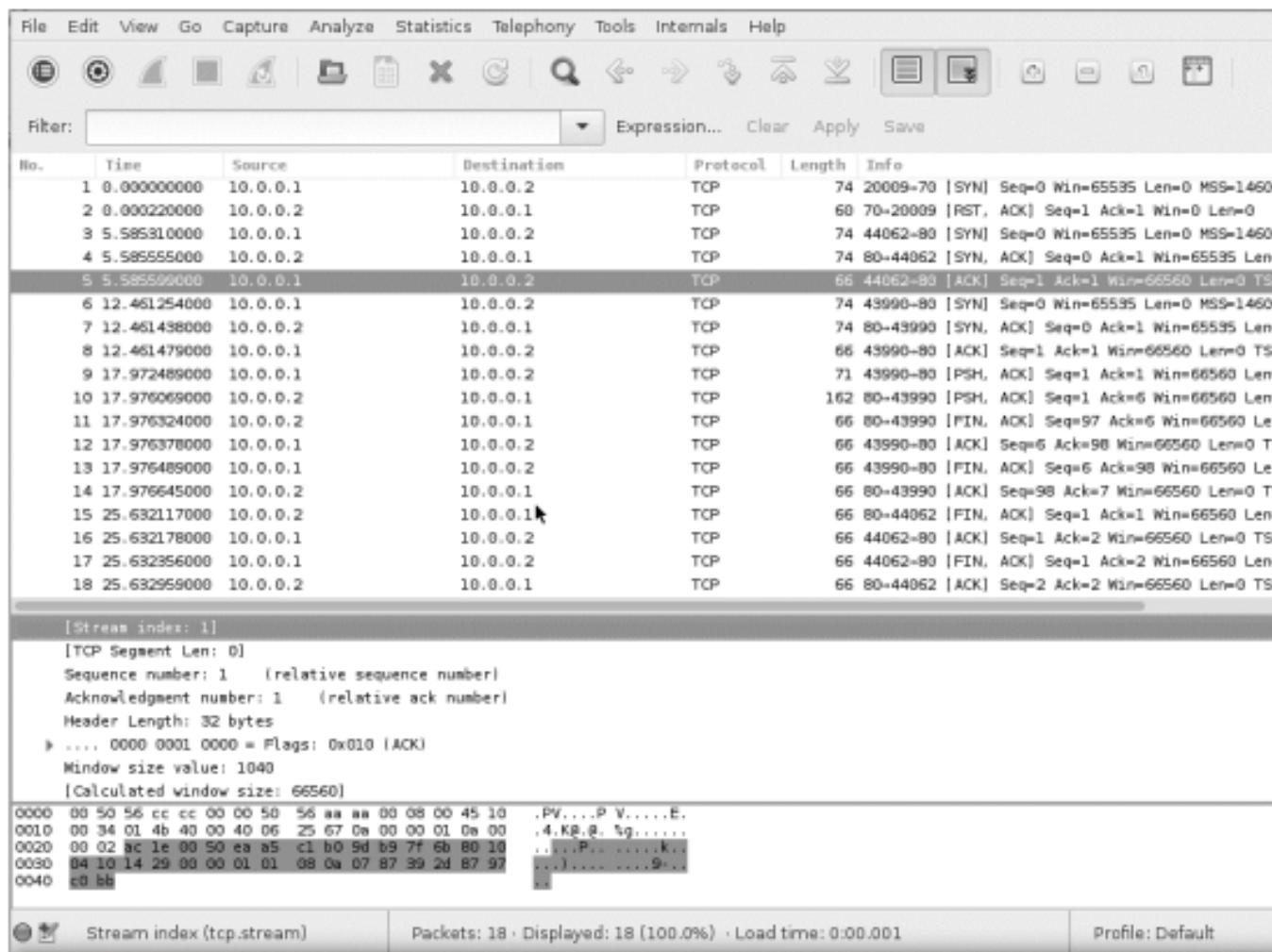
- B) Indicar la dirección del socket del cliente durante la transferencia de datos (dir\_ip:puerto) (3 puntos)

10.0.0.2:54168

- C) Indicar la dirección del socket con el que se contacta al servidor (dir\_ip:puerto) (3 puntos)

10.0.0.1:69

Forma similar al resultado durante la práctica TIC-ETI / TIC - Internet Transport Layer



### Se pide:

- A) Indicar cuántos intentos de conexión, completados o no, se han producido y, de ellos, cuántos se corresponden con servicios HTTP (4 puntos)

Ha habido tres intentos de conexión, 2 de ellos se corresponden con servicios HTTP

- B) Indicar el socket del servidor (dir\_ip:puerto) en el caso de que alguno de los intentos de conexión anteriores no se correspondiera con un servicio HTTP (3 puntos)

10.0.0.2:70

Telemática y Electrónica

27 de mayo de 2021

MESA:

- C) Indicar, para la primera conexión **completada**, la cantidad máxima de bytes que se pueden intercambiar el cliente y el servidor en un único segmento TCP. ¿Dentro de qué campo de la cabecera TCP se intercambia este dato? (6 puntos)

Se pueden intercambiar un máximo de 1460 bytes por segmento.

Este valor viene determinado por el parámetro MSS (Maximun Segmet Size) que se intercambia dentro del campo Opciones de la cabecera TCP.

- D) Indicar, para la primera conexión **completada**, el valor del factor de escalado de ventana (4 puntos)

Según se observa en el detalle de la trama nº 5, el valor de escalado de ventana sería:

$66560/1040 = 64$ , luego  $2^R = 64 \rightarrow \text{Window scale} = 6$

- E) Indicar cuánto tiempo permanecerá el servidor en el estado TIME\_WAIT si se estableciera una nueva conexión después de ejecutar de forma satisfactoria el comando `sysctl net.inet.tcp.msl=1024` (4 puntos)

El tiempo que permanecerá en el estado TIME\_WAIT será de  $2 * MSL = 2028$  ms

2 de julio de 2019

LABORATORIO

**Ejercicio 3 (10 puntos) (15 min)**

La siguiente captura se ha obtenido como resultado de hacer la transferencia de un fichero mediante el protocolo TFTP:

No.	Source	Destination	Protocol	Length	Info
1	10.1.1.1	10.2.2.2	UDP	81	Source port: 48418 Destination port: 50000
2	10.2.2.2	10.1.1.1	UDP	66	Source port: 58922 Destination port: 48418
3	10.1.1.1	10.2.2.2	UDP	60	Source port: 48418 Destination port: 58922
4	10.2.2.2	10.1.1.1	UDP	558	Source port: 58922 Destination port: 48418
5	10.1.1.1	10.2.2.2	UDP	60	Source port: 48418 Destination port: 58922
6	10.2.2.2	10.1.1.1	UDP	558	Source port: 58922 Destination port: 48418
7	10.1.1.1	10.2.2.2	UDP	60	Source port: 48418 Destination port: 58922
8	10.2.2.2	10.1.1.1	UDP	46	Source port: 58922 Destination port: 48418
9	10.1.1.1	10.2.2.2	UDP	60	Source port: 48418 Destination port: 58922

```

> Frame 3: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
> Ethernet II, Src: VMware_aa:aa:00 (00:50:56:aa:aa:00), Dst: VMware_bb:bb:00 (00:50:56:bb:bb:00)
> Internet Protocol Version 4, Src: 10.1.1.1 (10.1.1.1), Dst: 10.2.2.2 (10.2.2.2)
> User Datagram Protocol, Src Port: 48418 (48418), Dst Port: 58922 (58922)
> Data
0000  00 50 56 bb bb 00 00 50 56 aa aa 00 08 00 45 00 .PV....P V....E.
0010  00 20 05 d3 00 00 40 11 5d f5 0a 01 01 01 0a 02 . ....@. I.....
0020  02 02 bd 22 e6 2a 00 0c 45 7f 00 04 00 00 30 00 ...*.*. E.....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

- A) Indicar qué línea se ha tenido que incluir en el fichero /etc/services y en qué máquina se ha hecho, para que el servicio TFTP anterior haya funcionado correctamente. (2 puntos)

En el fichero /etc/services de la máquina 10.2.2.2 se ha tenido que incluir la línea:

tftp 50000/udp

- B) Indicar el tamaño (en bytes) del fichero transmitido. (2 puntos)

$512 + 512 = 1024$

- C) Indicar el tamaño (en bytes) del campo "Data" presente en la captura anterior. (2 puntos)

Los protocolos encapsulados dentro de la trama ethernet marcada tienen un tamaño de 32 bytes, de los cuales 20 bytes corresponden a la cabecera IP y 8 bytes a la cabecera UDP, por lo tanto el tamaño del campo "Data" será de 4 bytes.

Teniendo en cuenta la información mostrada en la siguiente captura:

172.20.0.2	172.20.0.1	TCP	74 43841->97 [SYN] Seq=0 Win=65535 Len=0 MSS=146
172.20.0.1	172.20.0.2	TCP	60 97->43841 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

- D) Indicar el resultado de ejecutar el comando `netstat -na | grep tcp4 | grep 97` en la máquina con dirección IP 172.20.0.1 (2 puntos)

Como la máquina 172.20.0.1 responde con un segmento RST, indica que no hay ninguna aplicación escuchando en el puerto 97, luego la ejecución del comando no mostrará ninguna información.

Una aplicación servidor ha transferido datos hacia una aplicación cliente haciendo uso del protocolo de transporte TCP. La siguiente captura muestra el tráfico generado en esa transferencia:

Source	Destination	Protocol	Info
10.2.2.2	10.1.1.1	TCP	19019->13 [SYN] Seq=3925948648 Win=65535 Len=0 MSS=1460 WS=64
10.1.1.1	10.2.2.2	TCP	13->19019 [SYN, ACK] Seq=2788447056 Ack=3925948649 Win=65535
10.2.2.2	10.1.1.1	TCP	19019->13 [ACK] Seq=3925948649 Ack=2788447057 Win=66560 Len=0
10.1.1.1	10.2.2.2	TCP	13->19019 [FIN, PSH, ACK] Seq=2788447057 Ack=3925948649 Win=6
10.2.2.2	10.1.1.1	TCP	19019->13 [ACK] Seq=3925948649 Ack=2788447084 Win=66560 Len=0
10.2.2.2	10.1.1.1	TCP	19019->13 [FIN, ACK] Seq=3925948649 Ack=2788447084 Win=66560
10.1.1.1	10.2.2.2	TCP	13->19019 [ACK] Seq=2788447084 Ack=3925948650 Win=66560 Len=0

- E) Indicar la cantidad de datos (en bytes) que ha transmitido la aplicación servidor. (2 puntos)

Servidor: 10.1.1.1.

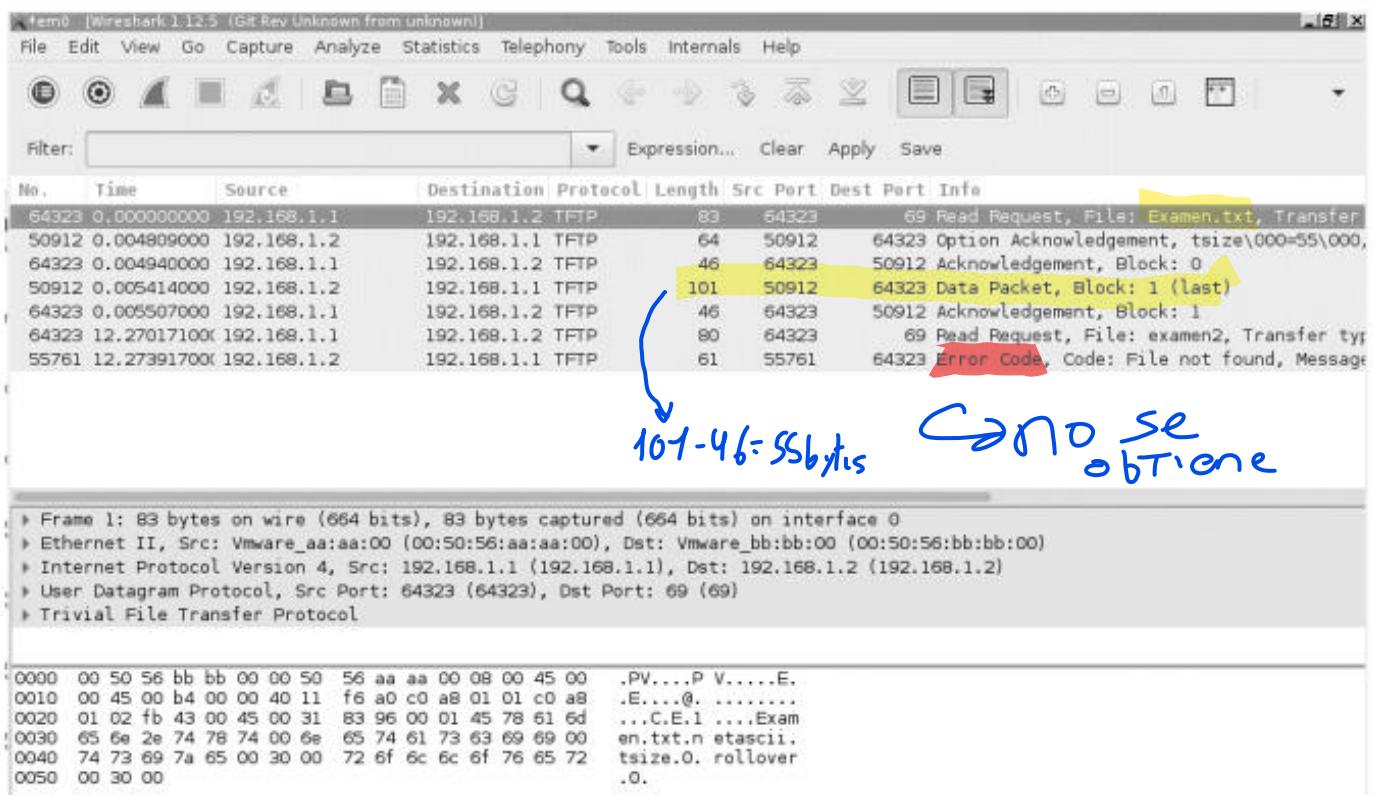
2788447084  
 - 2788447056  
 \_\_\_\_\_  
 - 28  
 2 → SYN + FIN

26 bytes

	1 de junio de 2023	LABORATORIO
--	--------------------	-------------

**Ejercicio 3 (25 puntos) (25 min)**

La siguiente captura muestra un intercambio de tráfico en el que un cliente obtiene un fichero de un servidor, de forma similar a los obtenidos en la práctica TCP1-L(P): UDP Protocol Scenarios

**Se pide:**

- A) Indicar nombre y el tamaño, en bytes, de los ficheros transferidos desde el servidor al cliente (2 puntos)

$$101 - 46 = 55 \text{ bytes}$$

Examen.Txt.

- B) Indicar los sockets (ip:puerto) del cliente y del servidor en las comunicaciones que se pueden observar en la captura. (4 puntos)

1

En la primera comunicación:

El socket inicial del cliente es 192.168.1.1:64323 y del servidor 192.168.1.2:69 para luego cambiar en el servidor a 192.168.1.2:50912

2

En la segunda comunicación:

El socket inicial del cliente es 192.168.1.1:64323 y del servidor 192.168.1.2:69 para luego cambiar en el servidor a 192.168.1.2:55761

- C) Rellenar sobre la siguiente plantilla los campos en blanco con el contenido necesario para que la maquina cliente obtenga el fichero anterior. (4 puntos)

```
[root@RROO-A ~]# tftp   
tftp>   
Received  bytes during 0.1 seconds in  blocks  
tftp> quit  
[root@RROO-A ~]#
```

1 de junio de 2023

**LABORATORIO**

La siguiente captura muestra un intercambio de tráfico TCP entre un cliente y un servidor, de forma similar al obtenido durante la práctica TCP2-L(P): TCP Protocol Scenarios

No.	Time	Source	Destination	Length	Src Port	Dest Port	Info
1	0.000000000	192.168.1.1	192.168.1.2	74	65127	20	65127->20 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64
2	0.001035000	192.168.1.2	192.168.1.1	60	20	65127	20->65127 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	3.849497000	192.168.1.1	192.168.1.2	74	58414	21	58414->21 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64
4	3.850223000	192.168.1.2	192.168.1.1	60	21	58414	21->58414 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5	10.014734000	192.168.1.1	192.168.1.2	74	39013	22	39013->22 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64
6	10.015207000	192.168.1.2	192.168.1.1	74	22	39013	22->39013 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64
7	10.015321000	192.168.1.1	192.168.1.2	66	39013	22	39013->22 [ACK] Seq=1 Ack=1 Win=66560 Len=0 TSval=1460 TSecr=1460
8	10.049533000	192.168.1.2	192.168.1.1	115	22	39013	Server: Protocol (SSH-2.0-OpenSSH_6.6.1_hpnl3v11 Fingerprint: 10:0d:11:3f:1e:4b:5d:2c:3f:4b:5d:2c:3f:4b:5d:2c) [len=115]
9	10.153293000	192.168.1.1	192.168.1.2	66	39013	22	39013->22 [ACK] Seq=1 Ack=50 Win=66560 Len=0 TSval=1460 TSecr=1460
10	17.001555000	192.168.1.1	192.168.1.2	72	39013	22	Client: Encrypted packet (len=6)
11	17.000212000	192.168.1.2	192.168.1.1	85	22	39013	Server: Encrypted packet (len=19)
12	17.005586000	192.168.1.2	192.168.1.1	66	22	39013	22->39013 [FIN, ACK] Seq=69 Ack=7 Win=66560 Len=0 TSval=1460 TSecr=1460
13	17.005721000	192.168.1.1	192.168.1.2	66	39013	22	39013->22 [ACK] Seq=7 Ack=70 Win=66560 Len=0 TSval=1460 TSecr=1460
14	17.005982000	192.168.1.1	192.168.1.2	66	39013	22	39013->22 [FIN, ACK] Seq=7 Ack=70 Win=66560 Len=0 TSval=1460 TSecr=1460
15	17.006566000	192.168.1.2	192.168.1.1	66	22	39013	22->39013 [ACK] Seq=70 Ack=8 Win=66560 Len=0 TSval=1460 TSecr=1460
16	20.556347000	192.168.1.1	192.168.1.2	74	38320	23	38320->23 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64
17	20.556588000	192.168.1.2	192.168.1.1	74	23	38320	23->38320 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64
18	20.556651000	192.168.1.1	192.168.1.2	66	38320	23	38320->23 [ACK] Seq=1 Ack=1 Win=66560 Len=0 TSval=1460 TSecr=1460
19	20.577813000	192.168.1.2	192.168.1.1	69	23	38320	Telnet Data ...

Frame 19: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface 0  
 Ethernet II, Src: Vmware\_bb:bb:00 (00:50:56:bb:bb:00), Dst: Vmware\_aa:aa:00 (00:50:56:aa:aa:00)  
 Internet Protocol Version 4, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.1 (192.168.1.1)  
 Transmission Control Protocol, Src Port: 23 (23), Dst Port: 38320 (38320), Seq: 1, Ack: 1, Len: 3  
 Telnet

```

0000 00 50 56 aa aa 00 00 50 56 bb bb 00 08 00 45 10 .PV....P V.....E.
0010 00 37 00 2c 40 00 40 06 b7 31 c0 a8 01 02 c0 a8 .7.,@. .1.....
0020 01 01 00 17 95 b0 de 2b 79 29 ff 48 38 df 80 18 .....+ y).HB...
0030 04 10 f5 ee 00 00 01 01 08 0a 59 06 a4 d4 00 d9 ..... .Y.....
0040 b0 68 ff fd 25 .h.%.
  
```

em0: <live capture in progress>... | Packets: 60 - Displayed: 60 (100.0%) | Profile: Default

**Se pide:**

- A) Indicar cuántos intentos de conexión se han producido y cuantos se han completado, identificando los puertos de cada uno de ellos. (2 puntos)

4 intentos, 2 completados. Servidores: 192.168.1.2:23  
 192.168.1.2:22  
 clientes: 192.168.1.1:39013  
 19?

- B) Indicar qué servicios están ejecutándose en el servidor para que se produzca el comportamiento mostrado en la captura. (3 puntos)

Los servicios que se están ejecutando en el servidor son el SSH en el puerto 22 y el TELNET en el puerto 23.

- C) Indicar la cantidad máxima de bytes que se pueden intercambiar el cliente y el servidor en un único segmento TCP. ¿Cómo se llama este parámetro y dentro de qué campo de la cabecera TCP se intercambia? (3 puntos)

MSS (Maximum Segment Size)= 1460 bytes  
dentro del campo "opciones".

- D) Indicar, para la primera conexión completada, quién inicia la secuencia de finalización de la conexión y en qué tramas capturadas se realiza.(3 puntos)

Empieza en la Trama 12, va de  
la 12 a la 15 (Rosa).

- E) Indicar, para la primera conexión completada, la cantidad de bytes que las aplicaciones cliente y servidor se han transferido mutuamente. (4 puntos)

$$70 - 2 = 68 \text{ bytes}$$

$$8 - 2 = 6 \text{ bytes}$$

$$68 + 6 = \boxed{74 \text{ bytes}}$$