Ejercicios resueltos de Redes de Computadores – ETSINF – curso 2012-13
Tema 4

1. El computador A desea solicitar un servicio al computador B. Para ello abre una conexión TCP y transmite su petición de 500 bytes. Una vez recibida, B transmite una respuesta de 100 bytes y cierra la conexión. Describe el intercambio total de segmentos entre A y B, teniendo en cuenta que el tamaño máximo de segmento que ambos negocian es de 100 bytes, la ventana de recepción de A (v_recA) es de 200 bytes mientras que la de B (v_recB) es de 300 bytes. Ambos extremos emplean reconocimientos retardados, y el tiempo de transmisión se considera despreciable frente al RTT. El número de secuencia inicial de A es 1000, mientras que el de B es 3000. Los mecanismos de control de congestión actúan de la forma habitual. La respuesta ha de darse en forma de tabla con las columnas: origen, nº de secuencia, nº de reconocimiento y datos.

Solución:

Orden de los segmentos	Origen	Nº Secuencia	Flags	Nº recon.	Datos
1	A	1000	SYN		
2	B	3000	SYN, ACK	1001	
3	A	1001	ACK	3001	
4	A	1001	ACK	3001	Petición bytes 1 a 100
5	A	1101	ACK	3001	Petición bytes 101 a 200
6	B	3001	ACK	1201	
7	A	1201	ACK	3001	Petición bytes 201 a 300
8	A	1301	ACK	3001	Petición bytes 301 a 400
9	A	1401	ACK	3001	Petición bytes 401 a 500
10	B	3001	ACK	1401	
11	B	3001	ACK	1501	Respuesta bytes 1 a 100
12	A	1501	ACK	3101	
13	B	3101	FIN, ACK	1501	
14	A	1501	FIN, ACK	3102	
15	B	3102	ACK	1502	

La tabla se ha elaborado desde el punto de vista de A, por eso primero se envían los segmentos 7, 8 y 9 y, posteriormente se reciben los ACKs de B 10 y 11. Si se hiciese según la actuación de B, sería también correcto y variaría ligeramente. En ese caso, tras recibir los segmentos 7 y 8 se enviaría el ACK de B (actual segmento 10), y después de recibir el último segmento de datos de A (actual segmento 9) se enviaría el segmento que lleva los datos de B (actual segmento 11).

2. El computador A ha establecido una conexión TCP con el computador B. Contesta si son posibles les secuencias siguientes de segmentos TCP. Justifica la respuesta.

Solución:

Supuesto a)

	N. Seq	*N. ACK*	*Ventana*	*Datos*	*Flags*
A	3000	15000	1000	1000	ACK
B	15000	4000	500	1000	ACK
B	16000	4000	500	1000	ACK
A	4000	17000	1000	500	ACK

***No es posible**. En el primer segmento A anuncia una ventana de recepción de 1000 bytes. A continuación B transmite dos segmentos de 1000 bytes, superando por tanto el tamaño de dicha ventana de recepción*

Supuesto b)

	N. Seq	*N. ACK*	*Ventana*	*Datos*	*Flags*
A	3000	0	1000	0	SYN
B	15000	3001	1000	1000	SYN,ACK
A	3001	16000	1000	1000	ACK

***No es posible**. No se permite que segmentos con el bit SYN activado lleven datos. Además, el valor del campo de reconocimiento del tercer segmento es incorrecto, debería ser 16001.*

3. ¿**Por qué** la actualización de la ventana de congestión TCP es diferente en función de que la pérdida de un segmento se haya detectado al vencer un temporizador o no?

Solución:

En el protocolo TCP la pérdida de un segmento se considera un indicio de congestión de la red, y puede detectarse mediante la recepción de tres ACKs duplicados o por el vencimiento de un temporizador de retransmisión. De las dos situaciones, el vencimiento del temporizador se considera un indicio de congestión más grave. En el caso de los ACKs duplicados, alguno de los segmentos enviados no ha llegado al receptor, pero han llegado segmentos enviados posteriormente, lo que significa que la red posiblemente empieza a perder algún paquete pero todavía hay circulación, se siguen entregando paquetes al destino. Sin embargo, el vencimiento de un temporizador puede indicar un estado de congestión mayor, donde ya no llega nada al destino o no llega nada desde el mismo.

4. El computador A tiene una conexión TCP establecida con el servidor S. El último segmento recibido desde S contiene la siguiente información:

   Nº Secuencia = 35200; Nº Reconocimiento = 12100; v_rec = 4000

   Sabiendo que Vcong = 3500 bytes, indica qué rango de bytes (**byte inicial y byte final**) como máximo puede enviar A antes de volver a recibir otro segmento desde S. Justifica brevemente el porqué.

   Solución:

Como vtrans = min(vcong, v_rec) = min(4000,3500) el emisor puede transmitir 3.500 bytes desde el valor indicado en el Nº Reconocimiento.
Byte inicial: 12.100 (byte que está esperando en este momento el receptor)
Byte final: 12.100 +3.500 - 1 = 15.599

5. Un host con una única dirección IP solicita a un servidor una resolución DNS (UDP) y una solicitud de *echo* (UDP). ¿Cómo diferencia el servidor entre los datos correspondientes a ambas solicitudes?¿Cómo distingue el host cliente las respuestas de ambas?

   Solución:

   El servidor recibe las peticiones en diferentes puertos. El servicio de DNS, por defecto tiene asociado el puerto udp 53 como puerto de escucha del servidor, mientras que el servicio de echo tiene asociado el puerto udp 7.

   El host cliente no tiene ningún problema para distinguir entre ambas respuestas. Probablemente, habrán sido generadas por diferentes aplicaciones cliente, con lo que cada una de ellas habrá reservado un puerto udp distinto en el que recibirá los datos procedentes del servidor. Además, aunque hubiesen sido generadas por la misma aplicación cliente vienen de distintos puertos en el servidor con lo que también se tendría un valor para diferenciar los datos recibidos.

6. ¿Es posible emplear un mismo *socket* TCP para intercambiar datos con dos procesos diferentes? ¿Y si el *socket* es UDP? Justifica las respuestas.

   Solución:

   No, ya que un socket TCP, o bien está en modo pasivo (no intercambia datos, sino que espera conexiones), o está conectado a un único socket remoto, y por tanto a un único proceso.

   En el caso de sockets UDP, sí es posible, puesto que estos sockets no conectan con otros, sino que es necesario especificar la dirección destino para cada DatagramPacket. Cada datagrama que se envía puede ir destinado a un socket distinto.

7. a) ¿Puede un cliente abrir dos sockets sobre el mismo puerto local (del mismo protocolo)? Justifica la respuesta.
   b) ¿Pueden existir en un servidor dos *sockets* con el mismo puerto y protocolo? Justifica la respuesta, y pon un ejemplo.

   Solución:

   a) No. La apertura del segundo socket produciría un error, ya que de permitirse la existencia de este segundo socket haría que TCP no supiera a cuál entregar un segmento si se emplean ambos para acceder al mismo servidor.

   b) Si, únicamente en los servidores TCP, ya que en este caso los sockets se diferencian por la conexión, de la cual forma parte la dirección del socket remoto (del cliente), ya que son sockets conectados. Ejemplo: Se crean sockets con el mismo puerto en un servidor concurrente cada vez que se ejecuta la llamada a accept .

8. a)¿Qué parámetros se utilizan para llevar a cabo el control de flujo y el control de la congestión en TCP? b)¿Quién actualiza cada uno? c)¿Sobre qué actúan estos parámetros y cómo?

   Solución:

   a) Control de flujo: campo ventana (v_rec) de la cabecera TCP.

   Control de la congestión: ventana de congestión

   b) v_rec lo actualiza el receptor y la ventana de congestión la actualiza el emisor.

   c) Ambos actúan sobre la ventana de transmisión: Vtrans = min(Vcong, v_rec).

9. Refleja en la siguiente tabla la evolución de las ventanas de congestión y transmisión con respecto al tiempo teniendo en cuenta que en el RTT 7 se produce un error detectado por tres ACK's duplicados, y en el RTT 9 un error detectado por *TimeOut*. Los permisos del receptor (v_rec) se expresan en segmentos y siguen la evolución indicada en la tabla. Se supone que el emisor siempre tiene suficientes datos para transmitir hasta el tamaño máximo de Vtrans y que para cada segmento recibido se genera un ACK de forma inmediata.

Solución:

RTT	1	2	3	4	5	6	7	8	9	10	11	12
v_rec	64	64	64	64	64	64	64	30	30	40	30	20
Umbral	64	64	64	64	64	64	64	32	32	15	15	15
Vcong	2	4	8	16	32	64	65	32	33	1	2	4
Vtrans_max	2	4	8	16	32	64	64	30	30	1	2	4

10. El computador A ha establecido una conexión TCP con el computador B (supuesto *a*) o va a establecerla (supuesto *b*). Indica si son posibles las secuencias siguientes de segmentos TCP. En caso negativo indica por qué.

Supuesto a)

	N. Sec	**N. ACK**	**Ventana**	**Datos**	**Flags**
A	3000	15000	2000	0	FIN, ACK
B	15000	3001	2000	1000	ACK
B	16000	3001	2000	1000	ACK
A	3001	17000	2000	0	ACK
B	17000	3001	2000	0	FIN, ACK
A	3001	17001	2000	0	ACK

Solución:

Es una secuencia correcta

Supuesto b)

	N. Sec	**N. ACK**	**Ventana**	**Datos**	**Flags**
	3000	-------	1000	0	SYN
B	15000	3001	1000	0	SYN,ACK
A	3001	15001	1000	0	ACK
A	3002	15001	1000	1000	ACK

Solución:

El establecimiento es correcto, pero el número de secuencia del último segmento debe ser 3001 , luego es imposible.

11. La conexión entre el cliente **cliente.upv.es** (puertos 1025 y superiores) y el servidor web **www.uji.es** (puerto 80) tiene lugar a través del proxy **proxy.upv.es** (puerto servidor 8080; puerto cliente 5000 y superiores). El navegador instalado en el cliente sólo soporta la versión 1.0 de HTTP, mientras que la parte cliente del proxy y el servidor final utilizan la versión 1.1 de HTTP (sin pipeline y persistente). El cliente inicia la descarga de la página web `index.html` y sus objetos asociados `fig1.jpg y fig2.jpg`. Supondremos que todos los objetos solicitados existen en el servidor final y además, el objeto `fig2.jpg` está en la caché del proxy.

a) Enumera, en orden cronológico, las conexiones TCP que se establecen:

Número conexión	Computador origen	Puerto origen	Computador destino	Puerto destino
1	cliente.upv.es	1025	proxy.upv.es	8080
2	proxy.upv.es	5000	www.uji.es	80
3	cliente.upv.es	1026	proxy.upv.es	8080
4	cliente.upv.es	1027	proxy.upv.es	8080

b) Enumera, en orden cronológico, la **línea inicial** de los mensajes HTTP que se intercambian, indicando el número de conexión TCP por el que tiene lugar dicho intercambio (según la tabla anterior) y el emisor del mensaje:

Número conexión	Origen	Línea inicial del mensaje HTTP
1	cliente.upv.es	GET http://www.uji.es/index.html HTTP/1.0
2	proxy.upv.es	GET /index.html HTTP/1.1
2	www.uji.es	HTTP/1.1 200 OK
1	proxy.upv.es	HTTP/1.1 200 OK
3	cliente.upv.es	GET http://www.uji.es/fig1.jpg HTTP/1.0
2	proxy.upv.es	GET /fig1.jpg HTTP/1.1
2	www.uji.es	HTTP/1.1 200 OK
3	proxy.upv.es	HTTP/1.1 200 OK
4	cliente.upv.es	GET http://www.uji.es/fig2.jpg HTTP/1.0
4	proxy.upv.es	HTTP/1.1 200 OK

c) Enumera los mensajes DNS intercambiados para hacer posible la interacción HTTP anterior. Supondremos que el cliente y el proxy sólo conocen la dirección IP del servidor de nombres local y que el servidor de nombres local tiene en su caché las direcciones IP de los servidores TLD necesarios.

Origen	Destino	R/I*	RegDNS**	P/R***	Objeto
cliente.upv.es	DNS local	R	A	P	Averiguar IP proxy.upv.es

DNS local	cliente.upv.es	R	A	R	Enviar IP proxy.upv.es
proxy.upv.es	DNS local	R	A	P	Averiguar la IP del servidor web
DNS local	TLD es	I	A	P	Averiguar la IP del servidor web
TLD es	DNS local	I	NS, A	R	Envía la IP del servidor DNS del dominio uji.es
DNS local	DNS uji	I	A	P	Averiguar la IP del servidor web
DNS uji	DNS local	I	A	R	Envía la IP del servidor web
DNS local	proxy.upv.es	R	A	R	Envía la IP del servidor web

(*) Recursiva o iterativa. (**) Tipo registro DNS. (***) Pregunta o respuesta.

12. ¿Cuánto tiempo se necesita para transmitir un mensaje de 1,5 x 10^6 bytes sobre un enlace con una velocidad de transmisión de 1 Mbps? Asumid que los paquetes de datos son de 1500B, el tamaño del ACK es de 50B, y el retardo de propagación de 10 ms. **NO** hay que considerar en los paquetes a transmitir las cabeceras de los protocolos empleados en los distintos niveles
    a) Si el control de flujo empleado es de parada y espera.
    b) Si el control de flujo empleado es de ventana deslizante, con tamaño máximo de la ventana de transmisión de 7 paquetes.

Solución:

v_trans = 1 x 10^6 bps
tprop = 10 ms = 10 x 10^{-3} s
Número de paquetes a transmitir = 1,5 x 10^6/1.500=1.500x 10^3/1.500=1.000 paquetes
Long_ACK = 50B = 50 x 8 = 400 bits
Long_paq = 1.500 x 8 = 12.000 bits
t_trans_paq = Long_paq/v_trans = 12.000/ 1 x 10^6 = 12 x 10^{-3} s
t_trans_ACK = Long_ACK/v_trans = 4.000/ 1 x 10^6 = 0,4 x 10^{-3} s

a) Tiempo total con parada y espera

Para cada paquete a transmitir se necesitará un tiempo de transmisión y un tiempo de propagación para que el paquete llegue al destino. Tras la recepción del paquete, el ACK tardará un tiempo de transmisión del ACK y un tiempo de propagación

RTT = (12 + 10 +0,4 + 10) x 10^{-3} seg, = 32,4 x 10^{-3} seg.

Si consideramos que el mensaje ha sido correctamente recibido cuando llega al emisor el último reconocimiento, para ello son necesarios 1000 RTT's...

T_Total= 1000 x 32,4x 10^{-3} = 32,4 seg.

b) Tiempo total con ventana deslizante

Puesto que el RTT es de 32,4 x 10^{-3} segundos y el tiempo de transmisión es 12 x 10^{-3} segundos, puede observarse que antes de finalizar la transmisión del tercer paquete ya ha llegado el reconocimiento del primero, y por tanto la ventana de transmisión nunca supera los tres paquetes, alcanzando el envío continuo. Por tanto, el paquete 1000 partirá del emisor tras un tiempo equivalente a 1000 veces el tiempo de transmisión. A continuación únicamente es necesario esperar un tiempo de propagación (10 ms) para que todo el mensaje haya llegado al destino, y un tiempo de transmisión del ACK y otro tiempo de propagación para que el reconocimiento vuelva al emisor.

$$T\_Total = 1000 \times 12 \times 10^{-3} + 10 \times 10^{-3} + 0{,}4 \times 10^{-3} + 10 \times 10^{-3} = 12{,}0204 \text{ segundos}$$

NOTA: Se consideran también válidas, si se razonan correctamente, las soluciones basadas en el computo del tiempo hasta la llegada al receptor del último bit del mensaje (restando a la solución anterior un t_trans_ACK y un Tprop en cada apartado) o incluso cuando sale el último bit del receptor (restando, además de t_trans_ACK, dos Tprop).

13. La siguiente tabla refleja la evolución de la ventana de transmisión de una conexión TCP. Completa con los valores correspondientes e identifica en qué momentos actúa cada uno de los mecanismos de control de la congestión. Salvo que la evolución de V_{trans} no pueda explicarse de otra forma, v_rec permanece constante.

Solución:

RTT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
V_{trans}	2	4	8	16	16	16	8	9	10	11	12	13	1	2	4
V_{cong}	2	4	8	16	17	18	8	9	10	11	12	13	1	2	4
Umbral	16	16	16	16	16	16	8	8	8	8	8	8	6,5	6,5	6,5

Arranque lento en los RTT's 1 a 6 y de 13 a 15

Evitación de la congestión desde el RTT 7 hasta el 12

Reducción multiplicativa por 3 ACK's en el RTT 7

Reducción multiplicativa por TimeOut en el RTT 13

Durante los RTT's 1 a 4 actúa el mecanismo de arranque lento (Slow Start). Se detiene cuando al ventana de congestión llega al umbral.

Los RTT's 5 y 6 actúa evitación de la congestión (*Congestion Avoidance*).

En el RTT 7 se detecta la pérdida de un segmento por tres ACK's duplicados. Por tanto, el umbral se fija a la mitad de la ventana de transmisión (8) y la ventana de congestión comienza desde ese punto.

Durante los RTT's 8 a 12 se aplica evitación de la congestión.

En el RTT 13 se detecta la pérdida de un segmento por vencimiento de un temporizador. El umbral baja a la mitad de la ventana de transmisión (6,5 segmentos), y la ventana de congestión baja a uno.

Finalmente, durante los RTT's 14 y 15 se aplica nuevamente arranque lento.

14. Un cliente HTTP se conecta a un servidor web para obtener una página HTML de 800 bytes que contiene un gráfico de 4.500 bytes (dos objetos en total). El diálogo HTTP se hará empleando una conexión persistente que permanecerá abierta después de enviar todos los objetos por si el cliente desea realizar nuevas peticiones. Todas las peticiones HTTP tienen un tamaño de 50 bytes. Además:

    1. El tamaño total de las cabeceras añadidas por todos los niveles (transporte, red y enlace) es

de 100 bytes. La conexión emplea los mecanismos de control de la congestión TCP y ACKs retrasados. El MSS establecido al inicio de la conexión es de 1000 bytes. A lo largo de toda la conexión, los tamaños de la ventana del cliente y el servidor permanecerán constantes con los valores siguientes: v_rec(C) = v_rec(S) = 16 segmentos.

2. Suponiendo que no se pierden paquetes ni se producen retransmisiones, describe el intercambio de segmentos entre el cliente y el servidor hasta que todos los datos enviados hayan quedado reconocidos. Se supone que los números de secuencia iniciales son NSI(C) = 6.000 y NSI(S) = 7.500.
3. NOTA: El formato de los segmentos a representar tendrá en cuenta el número de secuencia, los flags de la cabecera TCP, el reconocimiento (si procede) y el campo de datos (si procede). Así como los valores de las ventanas de congestión y de transmisión (expresados en segmentos) tras enviar el segmento. Así por ejemplo:

Origen	Nº secuencia	Flags	Nº ACK	Datos	VCong	VTrans
C	51	ACK	200	51..100	2	1

representa un segmento emitido por el cliente C que lleva 50 bytes de datos, con números de secuencia del 51 al 100, un reconocimiento hasta el octeto 199, con el flag ACK activo. Una ventana de congestión que permite enviar un máximo de dos segmentos y una ventana de transmisión que indica que hay un único segmento de datos enviado y pendiente de reconocimiento.

Solución:

Origen	Nº sec.	Flags	Nº ACK	Datos	VCong	VTrans
C	6000	SYN	-	-	2	0
S	7500	SYN, ACK	6001	-	2	0
C	6001	ACK	7501	-	2	0
C	6001	ACK, PUSH	7501	6001…6050	2	1
S	7501	ACK, PUSH	6051	7501…8300	2	1
C	6051	ACK, PUSH	8301	6051…6100	3	1
S	8301	ACK, PUSH	6101	8301…9300	3	1
S	9301	ACK, PUSH	6101	9301…10300	3	2
S	10301	ACK, PUSH	6101	10301…11300	3	3
C	6101	ACK	10301	-	4	0
C	6101	ACK	11301	-	4	0
S	11301	ACK, PUSH	6101	11301…12300	5	1
S	12301	ACK, PUSH	6101	12301…12800	5	2
C	6101	ACK	12801	-	4	0

15. ¿Por qué no se aplica la técnica de reconocimientos retrasados cuando se envían reconocimientos duplicados?

Solución:

Para que el otro extremo detecte cuanto antes la pérdida del segmento y lo pueda retransmitir.

16. Si arrancamos dos clientes web (navegador) en el mismo computador y enviamos una petición HTTP por cada uno de los clientes hacia el mismo servidor y el mismo puerto (80):

    a) ¿Cómo diferencia el servidor a qué navegador debe entregar cada respuesta?

    b) Si en lugar de la aplicación web se tratase de una aplicación que utilizara el servicio UDP para el transporte de los mensajes, ¿qué diferencias de funcionamiento existirían con el caso anterior?

    Solución:

    a) Para cada segmento recibido se analiza a qué Conexión pertenece. Una conexión se caracteriza, no solo por IP y puerto destino, sino también por origen. Puesto que los puertos origen son distintos, no hay ambigüedad con el socket destino.

    b) En UDP no hay conexiones, y por tanto todos los datagramas se entregan al mismo socket independientemente de su origen.

17. Explica de forma breve cómo funciona la opción SACK de TCP.

    Solución:

Cuando un receptor TCP emplea la técnica de retransmisión selectiva, es decir, almacena los segmentos recibidos fuera de orden, no puede reconocer estos segmentos mediante el mecanismo de ACK convencional como consecuencia de los ACK acumulativos, y por tanto estos segmentos serán retransmitidos cuando venza su *TimeOut*. Mediante la opción SACK el receptor puede indicar al emisor la recepción correcta de estos segmentos, indicando expresamente qué bloques de datos se encuentran ya en la ventana de recepción.

Ejemplo opcional: el emisor envía una serie de cinco segmentos cada uno con 1.000 bytes de datos y números de secuencia 1.000, 2.000, 3.000, 4.000 y 5.000. De los cinco, el receptor recibe correctamente sólo el primero, el tercero y el quinto. Tras recibir el primer segmento con el número de secuencia 1.000, y suponiendo que no hay segmentos anteriores pendientes de ser recibidos podría generar un segmento con ACK=2.000. Si a continuación recibe el tercero, podría generar un segmento con ACK = 2.000 (sigue esperando el segundo segmento) y SACK = 3.000- 4.000. Tras recibir el quinto, podría enviar ACK = 2.000, SACK = 5.000-6.000, 3.000-4.000.

18. Define el concepto de MSS (*Maximum Segment Size)* y especifica su relación con la tecnología de red empleada.

    Solución:

    EL MSS es la mayor cantidad de datos de aplicación que se permiten en un segmento TCP. A dicha cantidad será necesario sumarle el tamaño de la cabecera TCP (mínimo 20 bytes) para obtener el máximo tamaño de segmento.

    En cuanto a la relación con la tecnología, cada red tiene limitado el número máximo de bytes de datos por trama. Este valor es conocido como MTU. Así, para evitar la fragmentación a nivel de IP, debe cumplirse que MTU >= MSS + CabIP + CabTCP

    Por ejemplo, en Ethernet la MTU es 1500, y el valor típico del MSS suele ser 1460, contando con 20 bytes para la cabecera TCP y otros 20 para la cabecera IP.

19. Refleja en la siguiente tabla la evolución del **umbral** y de las **ventanas de congestión y transmisión** con respecto al tiempo (en RTT's), teniendo en cuenta los eventos que se indican (suceden durante el RTT, y se detectan al final del mismo). Todos los valores se expresan en segmentos. El valor de v_rec indicado en cada RTT es válido desde el **principio** del mismo. El emisor siempre tiene nuevos datos que transmitir. No se emplean ACKs retrasados.

(**) = recepción de 3 ACK's duplicados (*) = *TimeOut*

					**					*						**					
RTT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
v_rec	64	62	59	52	47	45	53	43	32	16	16	32	32	16	13	10	8	9	6	5	3
Umbral																					
Vcong																					
Vtrans																					

Indica qué algoritmos para el control de la congestión actúan en cada tramo y explica cómo se calculan los nuevos valores de Umbral, Vcong y Vtrans tras la recepción de 3 ACKs duplicados en el RTT 5 y tras el vencimiento de un temporizador en el RTT 10.

Solución:

					**					*						**					
RTT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
v_rec	64	62	59	52	47	45	53	43	32	16	16	32	32	16	13	10	8	9	6	5	3
Umbral	64	64	64	64	64	16	16	16	16	16	8	8	8	8	8	8	5	5	5	5	5
Vcong	2	4	8	16	32	16	17	18	19	20	1	2	4	8	9	10	5	6	7	8	9
Vtrans	2	4	8	16	32	16	17	18	19	16	1	2	4	8	9	10	5	6	6	5	3

RTTs 1-5 y 11-14: arranque lento (*slow-start*).

RTTs 6-10 y 15-21: evitación de la congestión.

El nuevo valor del Umbral en los tres casos se calcula como Umbral = max(2,Vtrans/2), tomando el valor de Vtrans en los RTTs 5 y 10. Es decir:

a) Nuevo valor de Umbral en el RTT 6, Umbral_RTT6 = 32/2 = 16.

b) Nuevo valor de Umbral en el RTT 11, Umbral_RTT11 = 16/2 = 8.

Tras recibir 3 ACKs duplicados Vcong = Umbral. Por lo tanto, Vcong_RTT6 = 16.

Tras vencer un temporizador Vcong = 1. Por lo tanto, Vcong_RTT11 = 1.

Por último, Vtrans = min(v_rec, Vcong).

Vtrans_RTT6 = min(45,16) = 16

Vtrans_RTT11 = min(16,1) = 1