

Segundo Certamen (Tiempo: 90 min.)

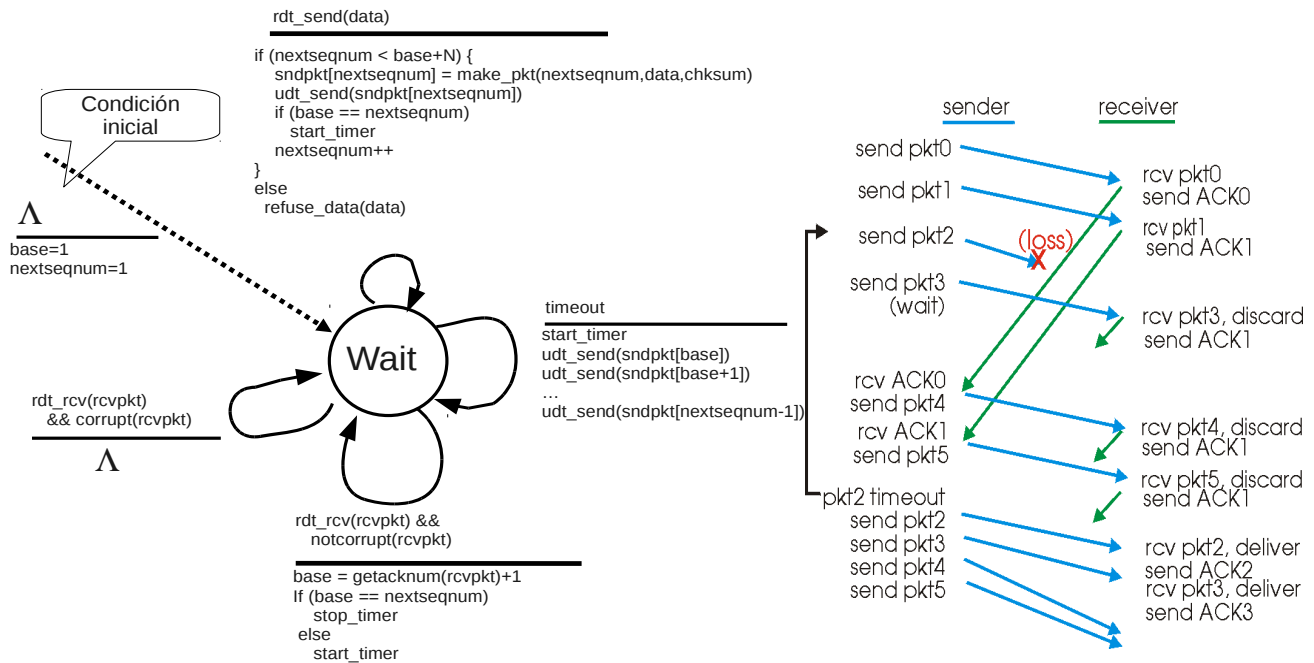
Si algo no está claro, haga un supuesto razonable, lo anota, y responda conforme a ello.

1.- (50 puntos) Responda en forma breve y clara:

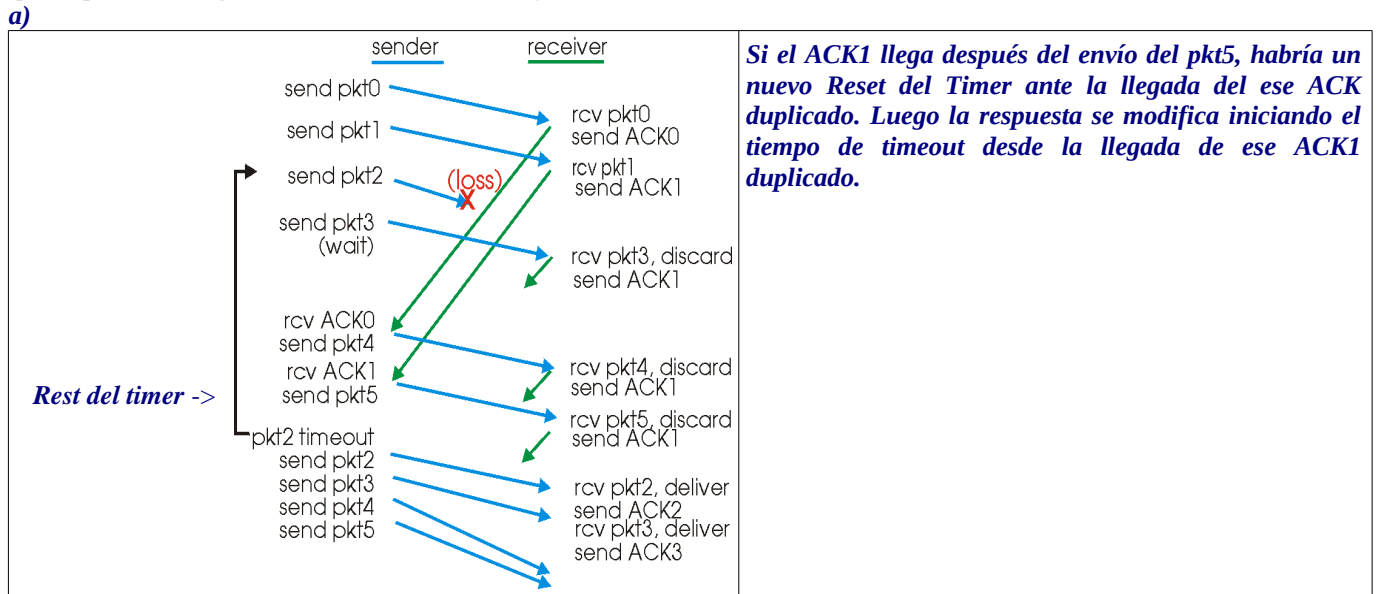
- a) Dé **dos** razones para explicar por qué UDP es un protocolo más rápido que TCP.
*UDP no requiere establecimiento de la conexión, TCP sí; luego UDP puede iniciar la transmisión antes. UDP no reduce su tráfico ante situaciones de congestión; mientras TCP pospone el envío de datos, UDP mantiene su envío.
TCP retrasa la entrega o el envío de datos cuando se pierde uno previo. UDP acepta las pérdidas y éstas no retrasan el envío o entrega de datos posteriores.*
- b) En cada capa lo conveniente es que cada protocolo dependa sólo de la información de su encabezado. ¿Cumple TCP este principio? Explique.
No. TCP entrega un segmento al socket que corresponda a la 4-tupla (IP_origen, Puerto_origen, IP_destino, Puerto_destino), luego TCP ocupa las direcciones IP origen y destino las cuales no pertenecen al encabezado de TCP.
- c) Un navegador corriendo en computador C trabaja en modo no persistente y establece dos conexiones para bajar dos objetos distintos de un mismo servidor web S. ¿Cómo sabe la capa de transporte en C a qué socket corresponde cada respuesta proveniente de S?
Si bien ambos socket provienen de los mismos IP y puerto origen y tienen igual IP destino, los mensajes de ambas conexiones si diferencian en el puerto destino. Basado en este puerto la capa de transporte lo entrega al socket asociado a cada conexión.
- d) ¿Por qué UDP es usado a menudo en aplicaciones multimedia en lugar de TCP?
Por que UDP puede ofrecer una tasa de datos estable a pesar de tener alguna tasa de paquetes errados. Las aplicaciones multimediales toleran pérdida de algunos paquetes y privilegian tasa de datos estable.
- e) ¿De qué depende el tiempo que TCP espera antes de retransmitir un segmento (tiempo de timeout)? ¿Cómo parte el mecanismo de estimación de este tiempo?
El timeout depende del RTT promedio estimado y de un estimador de las variaciones de RTT. Como al partir no se tiene mediciones de RTT, se usa un valor fijo.
- f) ¿Qué problema obligó la incorporación de números de secuencia para **lograr transferencias confiables** de datos?
Se incorporaron números de secuencia para distinguir mensajes (datos o acuses de recibos) duplicados.
- g) Mencione 2 ventaja de Go-Back-N respecto a Selective Repeat. (Obs. No basta con mencionar una diferencia, ésta debe ser favorable en algún sentido indicado por usted).
*i) Go-Back-N requiere menor memoria en el receptor. Esto es favorable en dispositivos receptores pequeños (poco hardware).
ii) Go-Back-N ocupa sólo un timer lo cual es conveniente en transmisores pequeños.

En general Go-Back-N es más simple de programar y ocupa menos recursos.*
- h) ¿Qué información del protocolo es enviada durante el Handshake de establecimiento de conexión en TCP?
Se intercambian los números de secuencias iniciales y los tamaños de ventana de recepción.
- i) Mencione dos razones que expliquen por qué los navegadores actuales trabajan en modo no persistente cuando bajan varios objetos desde un servidor web.
*i) En condiciones de congestión, se logra una mayor fracción de la capacidad disponible.
ii) En caso de pérdida del segmento de un objeto, su retransmisión no retrasa la transferencia del otro.*
- j) Cuando una aplicación UDP envía 10000 bytes en una única instrucción de envío (o escritura), el proceso receptor también lo lee en una única operación de recepción (o lectura). ¿Pasa lo mismo en TCP? Explique.
No. Mientras UDP es orientado al bloque, TCP es un flujo de bytes. TCP divide los 10000 bytes según el MSS del enlace, luego el receptor deberá hacer varias operaciones de lectura.

2.- (25 puntos) Usando la máquina de estados para la operación de Go-Back-N en el escenario de la figura, responde:



- a) En la figura muestre usted desde cuándo comenzó a medirse el tiempo que genera el timeout para la retransmisión del paquete 2. ¿Cómo se modifica su respuesta si el ACK1 enviado ante el recibo de pkt3, llega entre el envío del pkt5 y antes del timeout?
- b) ¿Cuál es el tamaño de la ventana de transmisión en el escenario mostrado?
- c) Muestre una modificación de la máquina de estados finita para reducir el tiempo de re-envío del paquete 2 de manera que el protocolo siga funcionando bien en éste y otros casos.



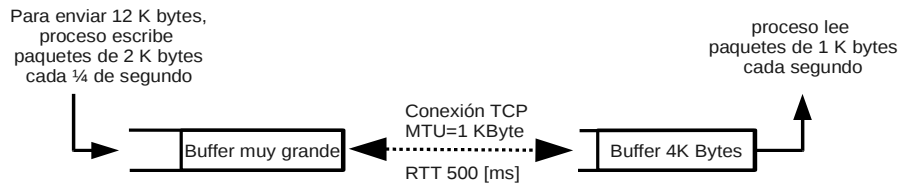
- b) El tamaño de la ventana de transmisión es de 4 paquetes.
- c) Ante la condición `rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)` se puede hacer algo al estilo de TCP:


```

if (base <= getacknum(rcvpkt)) { /* Sólo para ACK nuevos hacemos algo, en otro caso dejamos todo como estaba */
  base = getacknum(rcvpkt)+1
  if (base == nextseqnum) /* no hay ACK pendiente */
    stop_timer
  else
    start_timer /* hay ACK pendiente */
}

```

3.- (25 puntos) En la figura adjunta se muestra el buffer transmisor y buffer receptor de TCP en una conexión entre dos procesos. Mientras el buffer transmisor es muy grande, el receptor es de sólo 4 K bytes. Suponiendo que los enlaces en ambos extremos son de muy alta tasa ($t_{transmisión}$ despreciable), determine:



a) Tiempo que esta aplicación demora en transmitir los 12 K bytes desde que TCP envía el primer segmento para establecer conexión hasta que el proceso receptor lee el último paquete. Considere que el proceso transmisor comienza a enviar datos tan pronto la conexión es establecida y el proceso receptor está esperando por la llegada del primeros 1 K bytes de datos. Suponga que no hay pérdidas en TCP y los retardos en un sentido y en otro son iguales.

b) Si sólo se pierde el décimo paquete de datos enviado por TCP y éste detecta su pérdida por la llegada de tres acuses de recibo duplicados ¿cuál es su respuesta a a)? Ayuda: Haga un diagrama ordenado para responder en cada caso.

a)

El lado transmisor escribe datos en buffer a razón de 8 K bytes por segundos, luego en 1,5 [s] ya ha escrito los 12KB.

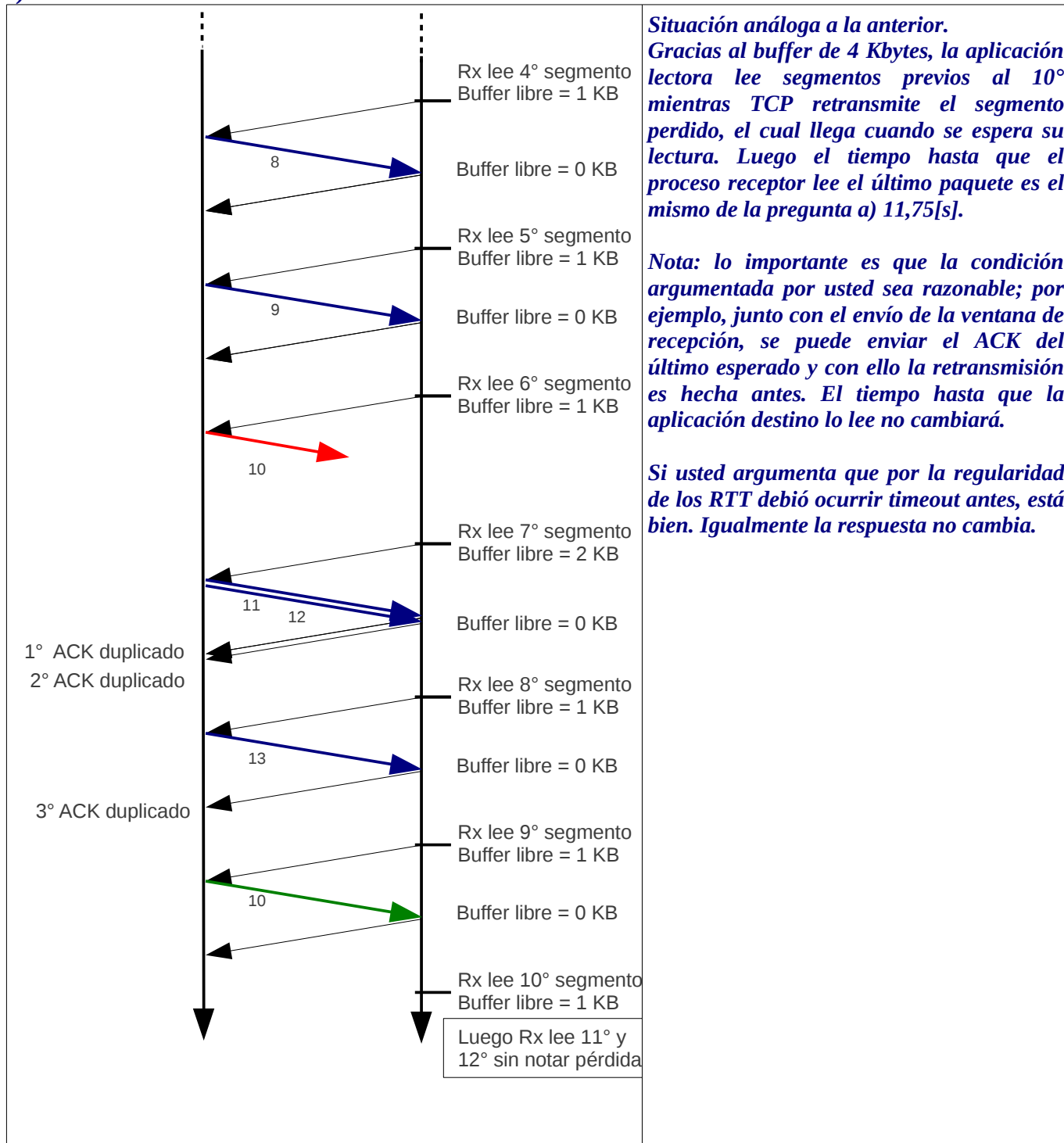
La aplicación receptora lee 1 KB por segundo, luego el buffer receptor se llena y el control de flujo limita el envío del transmisor.

Como la transmisión está limitada por la lectura del receptor el tiempo total será:

Para establecer conexión y leer primer segmento 1,5 RTT + 1 segundo para leer cada segmento restante.

Total 0,75 [s] + 11[s] = 11,75[s]

b)



Situación análoga a la anterior. Gracias al buffer de 4 Kbytes, la aplicación lectora lee segmentos previos al 10° mientras TCP retransmite el segmento perdido, el cual llega cuando se espera su lectura. Luego el tiempo hasta que el proceso receptor lee el último paquete es el mismo de la pregunta a) 11,75[s].

Nota: lo importante es que la condición argumentada por usted sea razonable; por ejemplo, junto con el envío de la ventana de recepción, se puede enviar el ACK del último esperado y con ello la retransmisión es hecha antes. El tiempo hasta que la aplicación destino lo lee no cambiará.

Si usted argumenta que por la regularidad de los RTT debió ocurrir timeout antes, está bien. Igualmente la respuesta no cambia.