(V.0.6, Spanish 1/12/2015)

Practical Exercises in Computer Networks and Distributed Systems

Commented exercises about TCP (WIP)

This document contains solved and commented exercises that have been part of recent term exams

Exercises

1. Considerad el mecanismo de control de flujo TCP. Asumid que el último ACK enviado por el receptor TCP tiene los siguientes campos: ACK = 12500, WS = 4000 y no usa ninguna opción TCP ¿Cuál de las siguientes opciones **no es** un mensaje válido que pueda ser enviado por el transmisor, dado este escenario? (Esto es, si el transmisor enviase uno de esos mensajes, el receptor no tendría suficiente espacio en el buffer de recepción (Receive Buffer) para procesar el mensaje recibido. Prestad atención al hecho de que el transmisor podría ya haber transmitido algunos bytes más allá del número de secuencia (SN) 12500. De acuerdo con esto, el número de secuencia del siguiente segmento transmitido puede ser mayor de 12500).

- a. Segmento de 4000 bytes con SN 12501
- b. Segmento de 1500 bytes con SN 12500
- c. Segmento de 1000 bytes con SN 15100
- d. Segmento de 4000 bytes con SN 12500

En el instante en que este segmento ACK fue transmisitido por el receptor, éste disponía de 4000 bytes en su buffer de entrada y, mediante el número de ACK acepta hasta el byte 12499 inclusive, por tanto, el siguiente byte esperado es el 12500 y el ultimo será el 12500 + 4000 – 1 = [12500, 16499], por tanto el bloque de datos máximo acceptable resultante es:

Bloque de datos máximo acceptable = **[12500, 16499]**

Para el momento en el que el transmisor recibe el segmento mencionado, puede haber consumido parte de la ventana, habiendo enviado algunos bytes, pero, en ningún caso debería superar el límite calculado anteriormente, de acuerdo con esto, analicemos cada una de las opciones para encontrar la que no es correcta:

a. Bloque de datos resultante: [12501, 12501+ 4000 - 1] = [12501, 16500], el cual, **está fuera de rango** al compararlo con el bloque de datos máximo acceptable. Esta es la respuesta que hay que marcar.
b. [12500, 12500 + 1500 – 1] = [12500, 13999]: aceptable
c. [15100, 15100 + 1000 - 1] = [15100, 16099]: aceptable
d. [12500, 15100 + 4000 - 1] = [12500, 16499]: aceptable

---

2. Considerad la fórmula de cálculo del RTT estimado (EstimatedRTT) con $\alpha = 0{,}9$ incluida en el algoritmo original de transmisión adaptativa TCP. Asumid que en una conexión TCP, en este instante, el EstimatedRTT es de 150 ms y que las siguientes tres muestras del RTT (SampleRTT en ms) son: 130, 180 y 39,2 ms ¿Cuál es el valor del SRTT final?
   - a. 150 ms
   - b. 180 ms
   - c. 140 ms
   - d. 135 ms
   - e. Un valor distinto de los incluidos en los apartados anteriores

   El Rtt estimado (EstimatedRTT) para el canal A->B de una conexión TCP entre A y B se calcula mediante la formula siguiente:

   EstimatedRtt[n + 1] = α ·EstimatedRtt[n] + (1 - α) ·SampleRtt[n] [1]

   En esta formula, n toma los valores 0, 1, …, éstos indexan cada uno de los ciclos *datos enviados/ack devuelto*, comenzando por el primero (Índice n = 0). En el enunciado, el conjunto de muestras del Rtt medidas por el transmisor en cada ciclo es el siguiente:

   SampleRtt[n: 0..2] = {130, 180, 39.82} ms; la condición inicial para la evaluación de la fórmula recursiva es: EstimatedRtt[0] = 150ms y el parámetro α = 0,9. Calculemos EstimatedRtt[1], EstimatedRtt[2] y EstimatedRtt[3]:

   EstimatedRtt[1] = 0,9 · EstimatedRtt[0] + 0,1 ·SampleRtt[0]
   EstimatedRtt[1] = 0,9 · 150 + 0,1 ·130 = 148 ms

   EstimatedRtt[2] = 0,9 · EstimatedRtt[1] + 0,1 ·SampleRtt[1]
   EstimatedRtt[2] = 0,9 · 148 + 0,1 ·180 = 151,2 ms

   EstimatedRtt[3] = 0,9 · EstimatedRtt[2] + 0,1 ·SampleRtt[2]
   EstimatedRtt[3] = 0,9 · 151,2 + 0,1 ·39,82 = 140,06 ms, **escogemos pues la respuesta c.**

   El enunciado pide el cálculo del "SRTT" final; SRTT (Smoothed Round Trip Time) es otra denominación del EstimatedRTT, la cual, hace referencia al hecho de que la fórmula recursiva [1] con α = 0.9, desde el punto del tratamiento digital de señales (DSP) se comporta como un filtro digital paso bajo y, por tanto, produce un efecto de suavizado (Smoothing) de la variable de entrada SampleRtt.

   Abundando sobre este hecho, debemos observar que el parámetro **α=0,9** en [1] aplica un peso muy grande al valor actual de la variable EstimatedRtt (Índice n-1) y, por tanto, 1 - α = 0.1, es decir, el peso aplicado a la muestra SampleRtt es mucho menor que el peso aplicado al valor actual estimado EstimatedRtt. Esta formula aplica poco peso a las muestras nuevas y mucho peso al valor actual de la formula, así, produce un efecto de filtrado sobre los valores grandes de SampleRtt, es decir, no sigue los cambios bruscos, de ahí el efecto de filtrado de las frecuencias altas, lo que se conoce como filtrado paso bajo (Las frecuencias bajas *sí pasan*).

---

3. Teniendo en cuenta la fórmula anterior y el algoritmo de Karn-Partridge ¿Cuál sería el RTO (Retransmission Timeout) resultante del caso anterior?

- a. 270 ms
- b. 280 ms
- c. 360 ms
- d. 300 ms
- e. 560 ms
- f. Un valor distinto de los incluidos en los apartados anteriores

El RTO correspondiente a una estimación del Rtt (EstimatedRtt) es el doble de éste ultimo:

$$\mathbf{RTO[n] = 2 \cdot EstimatedRtt[n]} \qquad \mathbf{[2]}$$

Por tanto, RTO[3] = 2 · EstimatedRtt[3] = 2 · 140 ms= 280 ms. **Marcamos la respuesta b.**

---

4. En la conexión TCP de la pregunta anterior, el transmisor envía un segmento TCP, el cual, se pierde debido a congestión en un router, por tanto, el *timer de retransmisión (RTO) se dispara.* Esto ocurre otras dos veces ¿Cómo calcula TCP el RTO programado en la última retransmisión?

- a. Fórmula aplicada en la pregunta 4
- b. Fórmula aplicada en la pregunta 4 con RTO = 2 x EstimatedRTT
- c. TCP no calcula el RTO en caso de retransmisión
- d. Backoff exponencial basado en el último RTO aplicado a la conexión TCP

La formula [2] para el cálculo del RTO se aplica solo si no hay retransmisiones, en caso de que haya retransmisiones, el transmisor deja de tomar muestras del Rtt (SampleRtt) y, por tanto, deja de estimar el Rtt (EstimatedRtt), con lo cual, también deja de calcular el RTO mediante [2], ya que, según hemos explicado, no dispone del EstimatedRtt[n]. En este caso, el transmisor calcula el RTO[n] usando el valor anterior del RTO, RTO[n-1] y efectúa el cálculo doblando éste:

En caso de retransmisión: $\mathbf{RTO[n] = 2 \cdot RTO[n-1]} \qquad \mathbf{[3]}$

Elegimos la respuesta **d.**; el resto de las respuestas son falsas y, *backoff exponencial* basado en el ultimo RTO es justo lo que expresa la formula anterior [3]: crecimiento exponencial de la secuencia de RTO en la que cada nuevo valor surge de doblar el anterior.