Protocolos punto a punto

Resolución de ejercicios vistos en clase

30.08.2017

1. Primer ejercicio

1.1. Enunciado

Dado un enlace punto a punto a la luna de 1Mbps con un delay de 1.25 segundos:

a. ¿Cuántos bits entran en el canal?

b. Asumiendo que se separan los bits en frames de largo fijo de 1Kb ¿Cuántos frames entran en el cable? ¿Y si usara frames de 2Kb?

1.2. Resolución

a. La ecuación que determina cuantos bits entran en el canal esta determinada por:

- $C_{vol} = V_{tx} * Delay$.

Dados entonces $V_{tx} = 1Mbps$ y $Delay = 1{,}25$ segundos tenemos que entran 1,25Mb en el canal.

b. Para frames de largo fijo 1Kb tenemos que entran: $\frac{1{,}25Mb}{1Kb} = 1250$ frames. En caso de ser frames de 2Kb serían exactamente la mitad, 625 frames.

2. Segundo ejercicio

2.1. Enunciado

Calcule el eficiencia del frame tomando en cuenta solo el overhead impuesto por las siguientes técnicas de framing:

a. Largo fijo.

b. Campo de 16 bits en el encabezado indicando el largo del frame.

c. Delimitadores de 8 bits usando bit-stuffing.

2.2. Resolución

a. Siendo $\eta_{frame} = \frac{largo\ de\ los\ datos}{largo\ total\ del\ frame}$, este caso tenemos que: $\eta_{frame} = \frac{largo\ de\ los\ datos}{largo\ de\ los\ datos} = 1$

b. Para campos en el encabezado tenemos que: $\eta_{frame} = \frac{largo\ del\ frame\ -|\ campo_{size}\ |}{largo\ del\ frame} = \frac{largo\ del\ frame - 16}{largo\ del\ frame}$

c. Este último caso es variable ya que depende de la cantidad de bits que haya que incorporar al frame debido al stuffing. Siendo n la cantidad de bits, nos queda $\eta_{frame} = \frac{largo\ del\ frame\ -\ 8\ -\ n}{largo\ del\ frame}$

3. Tercer ejercicio

3.1. Enunciado

Una estrategia para detectar y/o corregir errores es transmitir los datos en bloques de n filas con k bits por fila y agregar un bit de paridad a cada fila y columna. El bit de paridad de la última fila también lo es para todos los bits de paridad. ¿Qué tipos de errores detecta este esquema? ¿Qué tipos corrige?

3.2. Resolución

Vamos a contestar estas preguntas de modo analítico. Empezamos por el caso de los errores que detecta, separando los casos por cantidad de bits que se modificaron:

1. **1 bit:** Está claro que si hay un solo bit que se modifica durante la transmisión, éste va a ser detectado por el bit de paridad correspondiente a la fila en la cual el bit haya sido modificado. Es exactamente el caso de paridad simple.

2. **2 bits:** En caso de que haya dos errores tenemos dos posibilidades. Que los dos bits sean de la misma fila o que sean de distintas filas:

   a. Misma fila: En este caso también dividimos en dos posibilidades:

      1) Los dos bits que se modifican implican que debería haber un cambio en el bit de paridad y por ende el bit de paridad de la fila lo detecta.
      2) El bit de paridad de la fila sigue manteniendo su validez. Sin embargo, es necesario que dos bits de paridad a nivel columnas detecten el error de las dos columnas correspondientes.

   b. Distinta fila: En caso de que los bits que se modifican sean de distintas filas, los bits de paridad de cada una de las dos filas correspondiendes van a capturar el error.

   $\Rightarrow$ Cualquier modificación de 2 bits de la matriz va a ser detectado.

3. **3 bits:** Para el caso de 3 bits tenemos varias posibilidades. Si son todos de la misma fila el caso es análogo al anterior. Si son todos de distintas filas también. Ahora analicemos el caso de que haya 2 bits modificados en la misma fila y un bit en una segunda fila. Es facíl ver que en este caso el bit de paridad de la fila que tiene un solo bit modificado va a detectar el error.

4. Para el caso de 4 bits, encontramos un contraejemplo. Supongamos que se modifican 2 bits de la misma fila i y otros dos bits de la misma fila j, y además los bits que se modifican en i y en j pertencen a las dos mismas columnas k y z. O sea, se modifican i_k, i_z, j_k y j_z. Bajo este escenario, es posible que los bits de paridad tanto de las filas i y j, como de las columnas k y z mantengan su validez.

$\Rightarrow$ Por definición, si el análisis es correcto tenemos que la `distancia de hamming` es 3. O sea entre cualquier dos `codewords` de la codificación hay, como mínimo, 4 bits de diferencia.

Aunque sería más riguroso demostrarlo matemáticamente, si tomamos por válido que la `distancia de hamming` para esta codificación es 4, esta propiedad nos permite aseverar que el código propuesto puede corregir solamente los errores de 1 bit.

4. Cuarto ejercicio

4.1. Enunciado

Diseñe posibles conjuntos de frames para los siguientes tipos de protocolos, asumiendo que se detectan errores usando CRC (No hace falta aclarar el largo de los campos):

a. Stop & Wait

b. Sliding Window con GoBackN usando *Piggybacking*

c. Sliding Window con ACK Selectivo

4.2. Resolución

a. Para el primer caso podríamos definir un frame para el emisor y otro para el receptor de la siguiente forma:

$$\text{Emisor: } \#SEQ(1bit); Datos; Checksum$$
$$\text{Receptor: } \#ACK(1bit); Checksum$$

b. Para el segundo caso, al haber *Piggybacking* todos los frames pueden llegar a tener que cumplir simultáneamente los roles de emisión y recepción. Entonces proponemos un único frame como el siguiente:

$$\#SEQ; \#ACK; Datos; Checksum$$

c. Para el último caso, a diferencia del punto anterior tenemos reconocimento selectivo. Para estos casos se recomienda, por un tema de eficiencia, que el frame receptor reconozca los frames tanto acumulativa como individualmente. Entonces nos quedarían dos frames como los siguientes:

$$\text{Emisor: } \#SEQ; Datos; Checksum$$
$$\text{Receptor: } \#ACK; \#SACK; Checksum$$

5. Quinto ejercicio

5.1. Enunciado

Un protocolo sobre un enlace punto a punto de 1Mbps y 0.25 segundos de tiempo de propagación, trabaja con ventana deslizante con GoBackN usando frames de largo fijo 2KB y un CRC de 16bits para detectar errores.

a. Calcule cuáles son los tamaños de ventana de emisión y recepción óptimos.

b. ¿Cuántos bits hacen falta para secuenciar los frames?

c. Calcule cuánto tiempo es necesario para transmitir 20Mb de datos asumiendo que no hay errores.

5.2. Resolución

a. Es un protocolo de ventana deslizante GoBackN. Esto nos indica que por definición RWS = 1. Para el tamaño de la ventana de emisión tenemos: $SWS = \frac{V_{tx} * 2 * Delay(Frame)}{|Frame|} = \frac{1Mbps * 2 * 0{,}252s}{2Kb} =$ 252 frames.

b. $SWS + RWS \leq \#Frames \Rightarrow \#SEQ \geq 253 \Rightarrow$ con 8 bits alcanza para secuenciar los frames.

c. Veamos que nos piden transmitir 20Mb. Pero además sabemos que **no hay errores** en la transmisión. Entonces, como ya calculamos el tamaño de ventana que maximiza la eficiencia del protocolo podemos asumir que la eficiencia se aproxima a 1. Con todos estos datos, podemos asumir que podemos transmitir todo el tiempo, que no vamos a bloquearnos esperando reconocimientos y que no va a haber errores. Entonces, bajo estas condiciones, transmitir 20Mb no es otra cosa que el $Delay(20Mb)$ (+ Delay(2Kb) para esperar el último ACK):

$Delay(20Mb) = V_{tx}(20Mb) + T_{prop} = \frac{20Mb}{1Mbps} + 0,25s = 20,25s$. Si a esto le sumamos el último tiempo delay, la respuesta final es 20,502s.

6. Sexto ejercicio

6.1. Enunciado

Un protocolo confiable punto a punto que usa sliding window, opera sobre un canal de 10 Mbps, usa SACK y un frame emisor de 5Kb como el siguiente:

#SEQ (8bits) ; Datos ; Checksum

a. Proponga un frame para el receptor.

b. ¿Cuál es el valor de delay para el cual el protocolo presenta un 100 % de eficiencia?

c. Si el delay fuera de 1 seg ¿Cuántos bits deberían ocupar los números de secuencia de manera de maximizar la eficiencia?

6.2. Resolución

a. Al no presentarse requerimientos que ameriten el uso de piggybacking, no vamos a usar esta técnica para el framing. El protocolo va a ser asimétrico, en el sentido de que el frame emisor y el recetor van a ser distintos.

El número de secuencia del frame emisor ocupa 8 bits, esto implica que tanto el campo ACK del receptor como el campo SACK deben ocupar la misma cantidad de bits: 8 bits para ACK y 8 bits para SACK. Esto es porque es necesario poder referenciar en los reconociminetos (ACKs) a cada frame que envía el emisor. **Nota:** no es necesario para la correctitud del algoritmo de ventana deslizante usar SACK y ACK cuando hay reconocimientos selectivos. Sin embargo se estila usar de este modo porque mejora el desempeño del protocolo.

El frame receptor propuesto sería entonces:

#ACK(8bits) ; #SACK(8bits) ; Checksum (16bits)

b. Para encontrar el valor de delay óptimo hay que tratar de llenar el canal y que no se desperdicie tiempo estando bloqueado sin poder transmitir. Veamos que datos tenemos:

- Tengo 256 frames posibles porque el campo de #SEQ es de 8 bits.
- Pero, para evitar el problema de las reencarnaciones, solo 128 frames podrán ser enviados pues se usa SACK $\Rightarrow$ SWS $= 128 =$ RWS.

Luego, usamos los datos que tenemos en la formula de ventana de emisión óptima.

$$1 = \frac{Ttx(SWS)}{RTT(F)} = \frac{Ttx(F) * SWS}{Delay(Fe) + Delay(Fr)} = \frac{(5Kb/10Mbps) * 128}{Delay(5Kb) + Delay(32bits)} = 1 \Rightarrow$$

Resolviendo se obtiene que 0.031 segundos aprox. es el valor del delay que maximiza la eficiencia del protocolo.

c. Para resolver este ejercicio se recalcula el tamaño de la ventana de emisión para ajustarla al delay requerido:

$$SWS = \frac{V_{tx} * 2 * Delay}{|Frame|}$$

Reemplazando los datos y resolviendo se obtiene que SWS = 4000. Luego, #Frames $\geq$ 8000 y tomando el techo del $log_2(8000)$ se obtiene que se necesitan 13 bits.