Redes Deber 1

Andrés Martínez 00213046

1. Calcule el tiempo total requerido para transferir un archivo de 1000 KB en los siguientes casos, asumiendo un RTT de 100ms, un tamaño de paquete de 1 KB de datos, y un "handshaking" inicial de 2xRTT antes de que los datos sean enviados.

Tomar en cuenta que:

$1KB = 1 \times 10^3 Bytes = 8 \times 10^3 bits$

$1000\ KB = 1 \times 10^6 Bytes = 8 \times 10^6 bits\ RTT =$

$100ms = 100 \times 10^{-3} s$

$handshaking = 2 \times RTT = 2 \times 100ms = 200ms = 200 \times 10^{-3} s$

- **El ancho de banda es de 1.5 Mbps, y los paquetes de datos pueden ser enviados continuamente.**

  Dado que los datos pueden ser enviados continuamente, asumimos que los 1000 KB serán enviados en una sola tanda (TransferSize = 1000 KB); y además, solo tendremos que tomar en cuenta el tiempo de ida + el handshaking, es decir: RTT/2+handshaking.

$$transferTime = \frac{RTT}{2} + \frac{TransferSize}{BandWidth} + handshaking$$

$$transferTime = 100 \frac{}{} \frac{}{} \times 210^{-3} + 18.5 \times \times 1010^{6}{}_{6} bitsbits\ {}_{-3} s = \mathbf{5.58s}$$
$$+ 200 \times 10$$

- **El ancho de banda es de 1.5 Mbps, pero al terminar de enviar cada paquete de datos debemos esperar 1 RTT antes de enviar el siguiente.**

  En este caso, ya no podremos enviar continuamente, sino que solo podremos enviar paquetes de 1 KB en cada RTT (TransferSize = 1 KB). Además, tomamos en cuenta el tiempo de ida y vuelta (RTT).

$$transferTime = RTT + \frac{TransferSize}{BandWidth}$$

$$transferTime = 100 \times 10^{-3} s + 1 \frac{8 \times 10_3 bits}{} .5 \times 10_6 bits = 0.105s$$

Calculamos que si para enviar 1 KB de datos necesitamos 0.105s, necesitaremos 1000 x 0.105s = 105s para enviar 1000 KB. A esto sumamos el handshaking: $TotalTransferTime = 105s + 200 \times 10^{-3}s = \mathbf{105.2}\boldsymbol{s}$

- **El ancho de banda es infinito, lo que significa que el tiempo de transmisión es cero, y pueden enviarse 20 paquetes en cada RTT.**

  Si solo pueden enviarse 20 paquetes en cada RTT, quiere decir que se envían 20 * 1 KB = 20 KB de paquetes en cada RTT. Por lo tanto:

  $transferTime = RTT \times envíos + hanshaking$

  $$transferTime = 100 \times 10^{-3} \times \frac{1000KB}{20KB} + 200 \times 10^{-3} = \mathbf{5}.\ \mathbf{2}\boldsymbol{s}$$

- **El ancho de banda es infinito, y durante el primer RTT podemos enviar un paquete, durante el segundo RTT podemos enviar 2 paquetes, durante el tercer RTT podemos enviar 4, y así sucesivamente.**

  Si sabemos que 1 paquete = 1 KB, entonces debe realizarse una sumatoria de la siguiente manera:

  $1KB + 2KB + 4KB + \cdots + nKB = 1000\ KB$

  Esta sumatoria de números equivale a la siguiente fórmula:

  $2^{n+1} - 1 = 1000$

  Al despejar obtenemos que $n = 9$ para enviar todos los paquetes. Así que, finalmente nuestra fórmula quedaría así:

  $transferTime = handshaking + RTT \times 9$

  $transferTime = 200 \times 10^{-3}s + 100 \times 10^{-3}s \times 9 = \mathbf{1}.\ \mathbf{1}\boldsymbol{s}$

2. **Una propiedad de las direcciones es que son únicas. ¿Qué otras propiedades podrían ser útiles para las direcciones de red? ¿Se le ocurren situaciones donde las direcciones de red podrían no ser únicas?**

   Las direcciones de red tienen la característica de ser únicas y pueden proporcionar información sobre las características del dispositivo al que pertenecen. Además, es importante que estas direcciones estén estructuradas de manera ordenada y sigan un formato establecido para garantizar su compatibilidad con otras direcciones.

En algunos casos, las direcciones de red pueden repetirse si se utilizan en diferentes redes privadas. Por ejemplo, dos hogares podrían estar usando la misma dirección IPv4 para sus redes privadas, lo que permite la reutilización de direcciones IPv4 debido a la escasez de estas direcciones en todo el mundo.

3. **Para cada una de las siguientes operaciones en un servidor de archivos remoto, discuta si es más probable que sea sensible al *delay* o al *bandwidth*.**

   - Al abrir un archivo, lo que se realiza es simplemente la acción de abrirlo, sin necesidad de leer su contenido. Por esta razón, el ancho de banda no es relevante, pero la rapidez de entrega del pequeño mensaje es crucial, por lo que esta operación es sensible al retraso.
   - Al leer el contenido de un archivo, se transfiere una gran cantidad de datos a través de la red, por lo que el ancho de banda es el factor más importante a considerar en este caso.
   - Al listar los contenidos de un directorio, la cantidad de datos transferidos no es significativa, por lo que la velocidad de entrega es la principal preocupación, lo que hace que esta operación sea sensible al retraso.
   - Al mostrar los atributos de un archivo, la cantidad de información transmitida es insignificante, por lo que la rapidez de entrega es crucial y esta operación también es sensible al retraso.

4. **Suponga que cierto protocolo de comunicación implica una sobrecarga por paquete de 100 bytes para los headers. Enviamos 1 millón de bytes de datos usando este protocolo; sin embargo, cuando un byte de datos es corrompido, todo el paquete que lo contiene se pierde. Dé el número total de sobrecarga + bytes perdidos para los tamaños de paquetes de datos de: 1000, 5000, 10 000 y 20 000 bytes. ¿Cuál de estos es óptimo?**

   $DataSended = 1 \times 10^6\ bytes$

   $$TotalOverhead = overhead \times \frac{DataSended}{DataSize}$$

   $TotalSize = TotalOverhead + LoseBytes \rightarrow LoseBytes = DataSize$

   - DataSize = 1000

   $$TotalOverhead = 100bytes \times 1 \times \frac{10^6\ bytes}{1000\ bytes} = 1 \times 10^5 bytes$$

   $TotalSize = 1 \times 10^5 bytes + 1000bytes = 1.01 \times 10^5 bytes$

   - DataSize = 5000

   $$TotalOverhead = 100bytes \times \frac{1 \times 10^6\ bytes}{5000\ bytes} = 20\,000bytes$$

$TotalSize$ = 20 000$bytes$ + 5000$bytes$ = 25 000 $bytes$

- DataSize = 10 000

$$TotalOverhead = 100bytes \times \frac{1 \times 10^6\ bytes}{10\ 000\ bytes} = 10\ 000\ bytes$$

$TotalSize$ = 10 000$bytes$ + 10 000$bytes$ = **20 000 *bytes***

- DataSize = 20 000

$$TotalOverhead = 100bytes \times \frac{1 \times 10^6\ bytes}{20\ 000\ bytes} = 5000\ bytes$$

$TotalSize$ = 5000$bytes$ + 20 000$bytes$ = 25 000 $bytes$

El tamaño óptimo, que minimiza la sobrecarga y la pérdida de bytes, es de 10,000 bytes.

5. **Suponga que queremos transmitir el mensaje: *11001001* y protegerlo de errores usando el CRC polinomial $x^3 + 1$**

Sabemos que: $C(x) = x^3 + 1 = 1001$, lo cual quiere decir que k = 3.

- **Use división polinomial larga para determinar el mensaje que debería transmitirse.**

Primero determinamos $M(x)$, el cual es:

$M(x) = (1 \times x^7) + (1 \times x^6) + 0 + 0 + (1 \times x^3) + 0 + 0 + 1$

$M(x) = x^7 + x^6 + x^3 + 1$

Ahora multiplicamos M(x) por x^k, es decir:

$T(x) = (x^7 + x^6 + x^3 + 1)\ x^3 = x^{10} + x^9 + x^6 + x^3 = 11001001000$

Luego, dividimos T(x) entre C(x):

La división binaria nos da 011

Obtenemos un residuo de *11*. Esto lo debemos restar a T(x):

$T(x) - 11 = 11001001000 - 11 = 11001001011$

Por lo tanto, concluimos que el mensaje que debe transmitirse es: ***11001001011*.**

- **Suponga el bit del extremo izquierdo se invierte en tránsito. ¿Cuál es el resultado de recibir el cálculo CRC?**

Si se diera este cambio tendríamos lo siguiente:

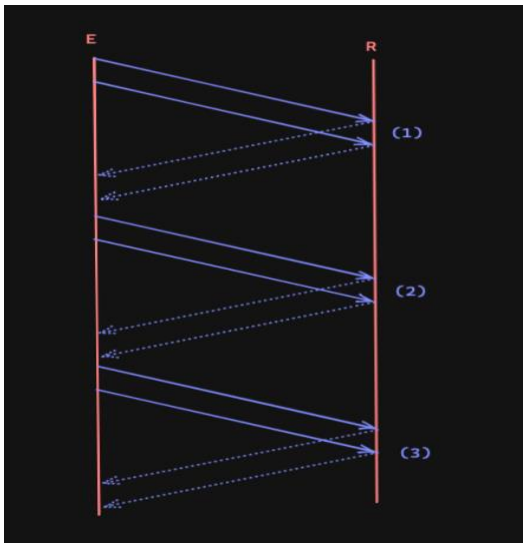
$P(x)$ = 01001001011

Al dividir esto por C(x) obtenemos:

Al obtener un residuo de *10*, se concluye que ocurrió un error durante la transmisión del mensaje. Sin embargo, CRC es capaz de corregir errores de un solo bit siempre y cuando los coeficientes de los términos x^k y x^0 no sean cero. Por lo cual este error puede corregirse.

6. **En una transmisión Stop-and-Wait, suponga que tanto el emisor como el receptor retransmiten su último frame inmediatamente en recibo de un ACK duplicado o frame; esta estrategia es superficialmente razonable porque al recibir dicho duplicado es probable que signifique que el otro lado ha experimentado un timeout.**

   - **Dibuje una línea de tiempo mostrando qué pasaría si el primer data frame es de alguna manera duplicado, pero ninguno se pierde. ¿Cuánto tiempo se mantendrán los duplicados? Esta situación se conoce como "The Sorcerer's Apprentice bug".**



El frame (1) es enviado y el receptor envía el ACK [1] correspondiente. Sin embargo, se produce una duplicación del frame (1), lo que provoca que el receptor reenvíe el ACK [1]. Al recibir el primer ACK [1], el emisor concluye que el frame (1) ha sido transmitido con éxito y procede a enviar el frame (2). Pero nuevamente, se recibe un ACK [1] debido a la duplicación del frame anterior. Debido a la estrategia de transmisión utilizada, dos ACKs duplicados provocan la retransmisión inmediata del último frame enviado, en este caso, el frame (2). Esto lleva a que el frame (2) se duplique y se envíe al receptor. Este proceso se repite sucesivamente hasta que se finaliza la comunicación.

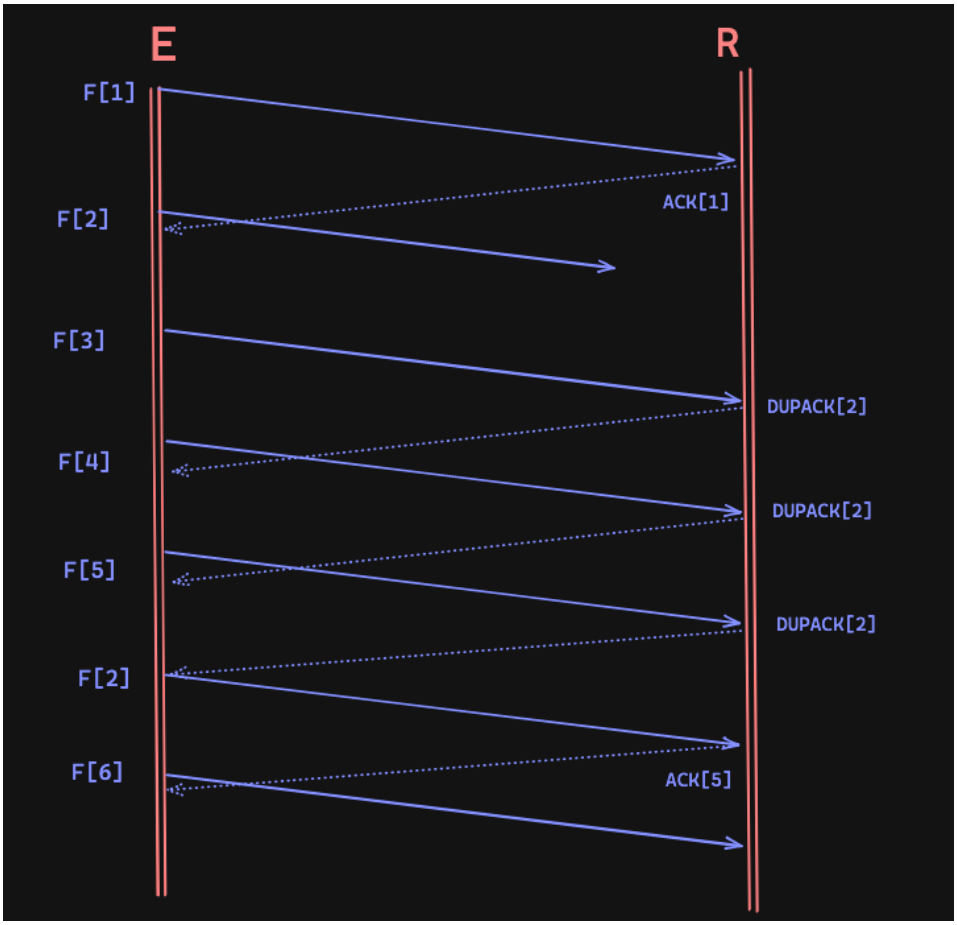
   - **Suponga que, como los datos, los ACKs son retransmitidos si no hay respuesta durante el timeout. Suponga también que ambos lados usan el mismo intervalo de timeout. Identifique un escenario probable razonable para que se active "The Sorcerer's Apprentice bug".**

Imaginemos que se ha enviado el FRAME[1] y el receptor lo ha recibido, pero su ACK[1] se ha retrasado. Como resultado, el emisor ha retransmitido el FRAME[1], pero luego ha recibido rápidamente el ACK[1]. En ese momento, se ha transmitido el FRAME[2], pero ha llegado un ACK[1] durante ese proceso, lo que ha provocado la duplicación de dos ACK[1]. Como resultado, el último frame, es decir el FRAME[2], se ha transmitido inmediatamente. Después, se ha recibido el ACK[2] y se ha transmitido el FRAME[3], pero durante ese proceso también ha llegado un ACK[2] duplicado

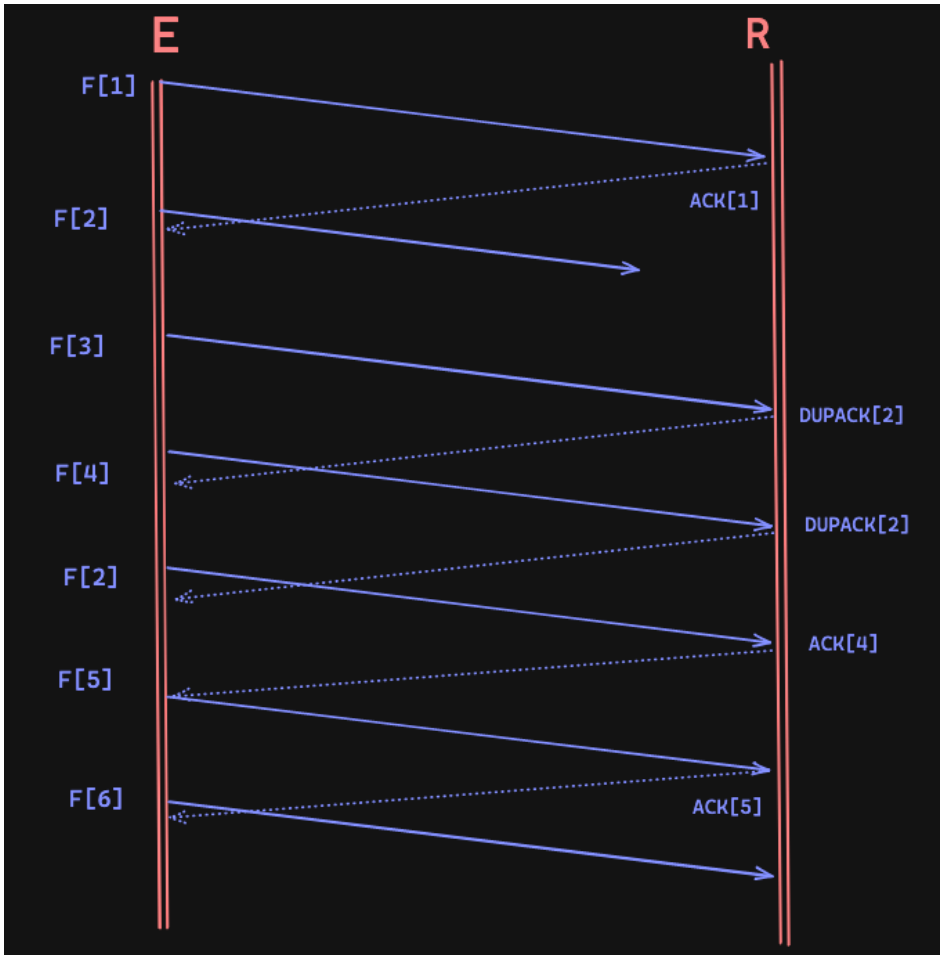
del FRAME[2], lo que ha causado la retransmisión del FRAME[3]. Este proceso se ha repetido sucesivamente hasta que se ha completado la transmisión.

7. **Dibuje un diagrama de línea de tiempo para el algoritmo de Sliding Window con SWS = RWS = 4 frames para las siguientes dos situaciones. Asuma que el receptor envía un ACK duplicado si no recibe el frame esperado. Por ejemplo, envía DUPACK[2] cuando espera ver el FRAME[2] pero recibe en su lugar el FRAME[3]. Además, el receptor envía ACK acumulativo después de recibir todos los frames esperados. Por ejemplo, envía ACK[5] cuando recibe el frame perdido FRAME[2] después de ya haber recibido el FRAME[3], FRAME[4] y el FRAME[5]. Use un intervalo de tiempo de 2 × RTT.**

   - **El frame 2 se pierde. La retransmisión toma lugar al agotarse el timeout. (como es usual).**



   - **El frame 2 se pierde. La retransmisión toma lugar cuando se recibe el primer DUPACK o cuando se agota el timeout. ¿Reduce este esquema el tiempo de transacción? Note que algunos protocolos end-to-end usan un esquema similar para una retransmisión rápida.**


E
R
F[1]
ACK[1]
F[2]
F[3]
DUPACK[2]
F[4]
DUPACK[2]
F[2]
ACK[4]
F[5]
F[6]
ACK[5]