Universidad San Francisco de Quito

Redes + Lab.

Deber #1

Luis Cagigal (00211793)

NRC: 4797

1. **Calculate the total time required to transfer a 1000KB file in the following cases, assuming an RTT of 100 ms, a packet size of 1KB data, and an initial 2RTT of "handshaking" before data is sent:**

   a. **The bandwidth is 1.5Mbps, and data packets can be sent continuously.**

$$N = number\ of\ total\ packets = \frac{filesize}{packetsize} = \frac{1000KB}{1KB} = 1000\ packets$$

$$Bandwidth(kbps) = \frac{1.5Mbps}{8} = 187.5\ Kbps$$

$$TP = Transfer\ time\ per\ packet = \frac{packetsize(kb)}{bandwidth(kbps)} = \frac{1KB}{187.5Kbps} = 5.33 \times 10^{-3} sec$$

$$F = File\ transfer\ time = N \times TP = 1000 \times 5.33 \times 10^{-3} = 5.33\ sec$$

$$RTT = 100\ ms$$

$$T = Total\ time = Initial\ RTT \times F = 200\ ms\ \times 5.33\ sec = 5.533\ sec$$

   b. **The bandwidth is 1.5Mbps, but after we finish sending each packet, we must wait one RTT before sending the next.**

El tiempo por paquete se mantiene igual:

$$TP = 5.33 \times 10^{-3} sec$$

Lo que cambia es el tiempo de traspaso de todo el archivo en sí:

$$F = N \times (TP + RTT) = 1000 \times (5.33 \times 10^{-3} + 0.1)$$

$$F = 1000 \times 0.1053 = 105.33\ sec$$

Por ende, el tiempo total también cambia:

$$T = Initial\ RTT \times F = 200\ ms\ \times 105.33\ sec = 105.533\ sec$$

c. The bandwidth is "infinite", meaning that we take transmit time to be zero, and up to 20 packets can be sent per RTT.

$$RTTs\ per\ packet = RTTp = \frac{N}{packets\ per\ RTT} = \frac{1000}{20} = 50\ RTTs$$

$$F = File\ transfer\ time = RTTs \times RTT = 50 \times 100\ ms = 5\ sec$$

$$T = 200ms + F = 5.2\ sec$$

d. The bandwidth is infinite, and during the first RTT we can send one packet, during the second RTT, we can send two packets, during the third we can send four, and so on.

Este concepto se puede representar como una secuencia geométrica de la siguiente forma, donde n es el número de RTTs:

$$S(n) = 2^n - 1\ packets$$

Para los paquetes de este caso:

$$1\ RTT + 2\ RTT + 4\ RTT = 7\ RTTs$$

$$7\ RTTs \times RTT = 7 \times 100\ ms = 0.7\ sec$$

$$T = 700\ ms + 200\ ms = 0.9\ sec$$

Se demoraría 0,9 segundos por grupo de paquetes aproximadamente hasta completar el archivo.

2. **One property addresses is that they are unique; if two nodes had the same address it would be impossible to distinguish between them. What other properties might be useful for network addresses to have? Can you think of any situations in which network addresses might not be unique?**

Las direcciones pueden tener otras propiedades como capacidad de redireccionamiento haciendo que el traspaso de datos sea más eficiente, deben tener propiedades de consistencia que permitan que funcionen sin errores independientemente del dispositivo o propiedades de seguridad para evitar accesos no autorizados a los datos.

Por ejemplo, en una VPN se pueden asignar a los dispositivos una misma dirección que se mostrará hacia afuera. Sin embargo, las distinciones de estas direcciones se abarcan dentro de la VPN en sí misma, evitando confusiones.

3. **For each of the following operations on a remote file server, discuss whether they are more likely to be delay sensitive or bandwidth sensitive:**
    a. **Open a file**

    Delay sensitive: Se demora más tiempo estableciendo la conexión que transmitiendo los datos, por ende, el servidor y tiempo de respuesta es lo que afecta en esta transacción.

    b. **Read the contents in a file**

Bandwidth sensitive: Dependiendo del ancho de banda, el traspaso de los contenidos de dicho archivo será más rápido o más lento. Una vez establecida la conexión, el delay ya no presentaría tanto conflicto como el ancho de banda para esta operación.

c. List the contents of a directory

Delay sensitive: Acceder a los metadatos de un sistema de carpetas requiere constante comunicación entre cliente y servidor, por ende, la conexión debe ser estable y depende del delay que exista.

d. Display the attributes of a file

Delay sensitive: Nuevamente, debido a que esta transacción requiere de los metadatos del archivo, requiere de una constante comunicación entre cliente y servidor, por lo que el delay es lo que más afecta en este caso.

4. **Suppose that a certain communications protocol involves a per-packet overhead of 100 bytes for headers. We send 1 million bytes of data using this protocol; however, when one data byte is corrupted, the entire packet containing it is lost. Give the total number of overhead + loss bytes for packet data sizes of 1000, 5000, 10000 and 20000 bytes. Which of these sizes is optimal?**

   Para cada caso, se utiliza la siguiente expresión para obtener el número de paquetes totales necesarios para completar el millón de bytes:

$$N = number\ of\ packets = \frac{Totalbytes}{packetsize}$$

$$N_{1000} = \frac{1000000}{1000} = 1000\ packets$$

$$N_{5000} = \frac{1000000}{1000} = 200\ packets$$

$$N_{10000} = \frac{1000000}{1000} = 100\ packets$$

$$N_{20000} = \frac{1000000}{1000} = 50\ packets$$

Ahora, utilizamos la siguiente expresión para obtener, para cada caso, la pérdida total:

$$L = total\ loss = \frac{1\times10^8}{N} + N$$

$$L_{1000} = \frac{1\times10^8}{1000} + 1000 = 101{,}000\ bytes$$

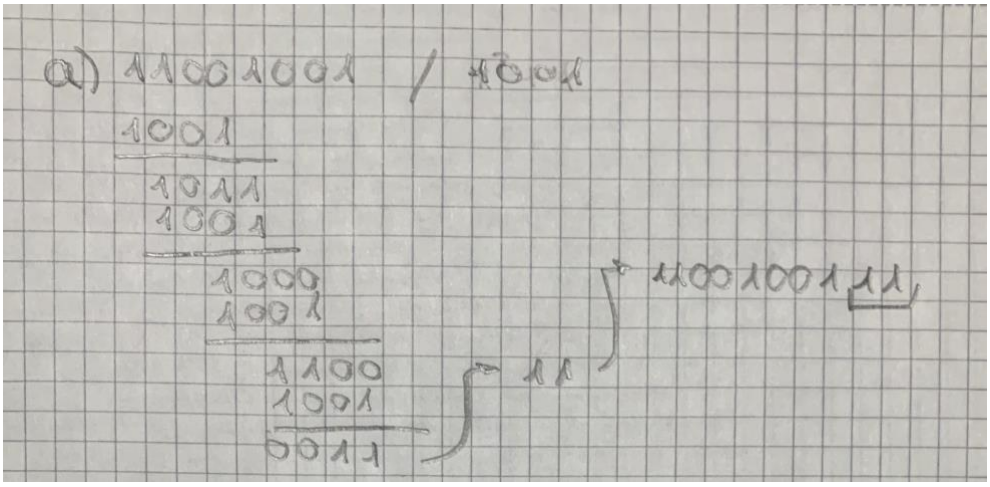
$$L_{5000} = \frac{1\times10^8}{5000} + 5000 = 25{,}000\ bytes$$

$$L_{10000} = \frac{1\times10^8}{10000} + 10000 = 20{,}000\ bytes$$

$$L_{20000} = \frac{1\times10^8}{20000} + 20000 = 25{,}000\ bytes$$

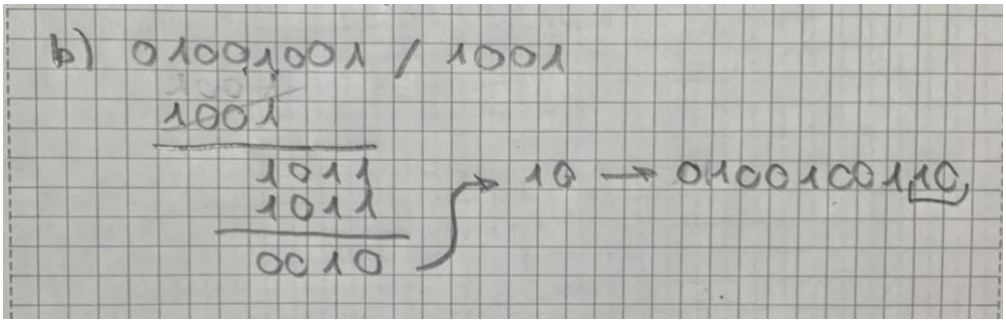
El tamaño de 10,000 es el más óptimo debido a que es el que menos datos perdería.

5. **Suppose we want to transmit the message 11001001 and protect it from errors using the CRC polynomial $x^3 + 1$:**
    a. **Use polynomial long division to determine the message that should be transmitted.**



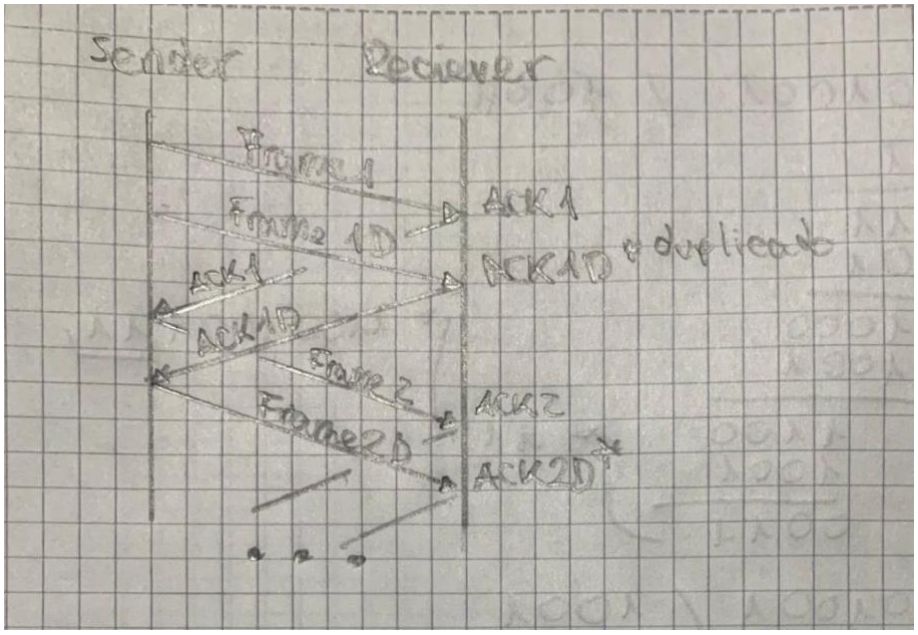
El mensaje sería 1100100111.

b. Suppose the leftmost bit gets inverted in transit. What is the result of the receiver's CRC calculation?



6. In stop-and-wait transmission, suppose that both sender and receiver retransmit their last frame immediately on receipt of a duplicate ACK or data frame; such a strategy is superficially reasonable because receipt of such a duplicate is most likely to mean the other side has experienced a timeout.

a. Draw a timeline showing what will happen if the first data frame is somehow duplicated, but no frame is lost. How long the duplications continue? This situation is known as the Sorcerer's Apprentice bug.



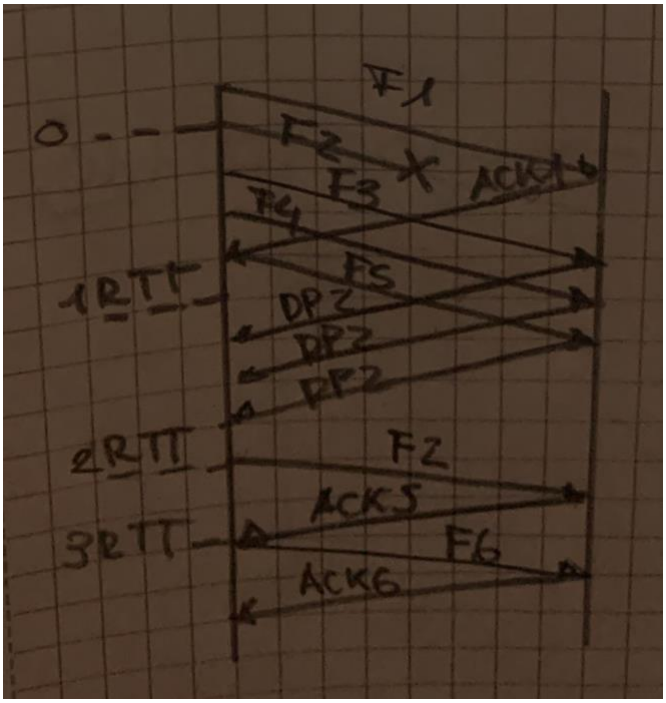
Los datos duplicados seguirían generándose indefinidamente, hasta que se crucen entre ellos hacia su destinatario, ya sea el Frame o el ACK correspondiente.

b. **Suppose that, like data, ACKs are retransmitted if there is no response within the timeout period. Suppose also that both sides use the same timeout interval. Identify a reasonably likely scenario for triggering the Sorcerer's Apprentice bug.**

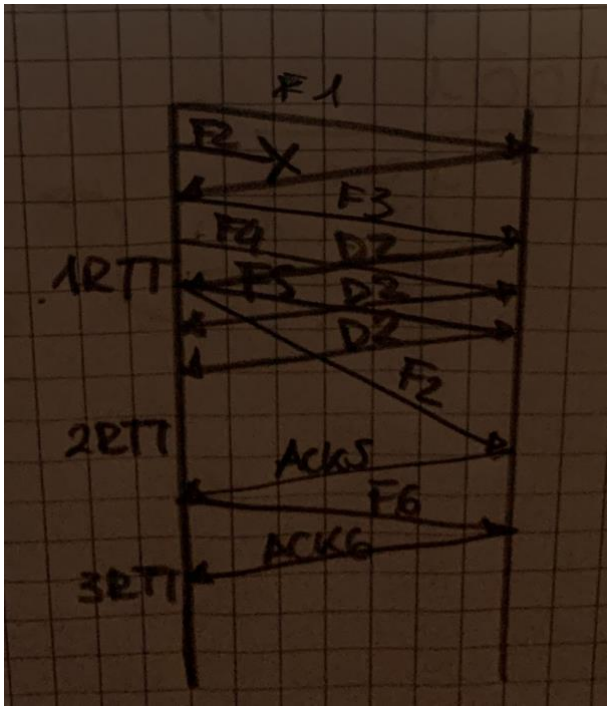
Pasaría que existirían multiples copias de los mismos datos circulando en la red, ocasionando una congestión. Al recibir tarde los ACK, el sender enviará continuamente los datos hasta recibir el ACK correspondiente. Al seguir enviando estos datos, el receiver los asumirá como nuevos frames, y enviará los ACKs para esos frames como si fueran nuevos.

7. **Draw a timeline diagram for the sliding window algorithm with SWS = RWS = 4 frames for the following two situations. Assume the receiver sends a duplicate acknowledgement if it does not receive the expected frame. For example, it sends DUPACK[2] when it expects to see FRAME[2] but receives FRAME[3] instead. Also, the receiver sends a cumulative acknowledgement after it receives all the outstanding frames. For example, it sends ACK[5] when it receives the lost FRAME[2] after it already received FRAME[3], FRAME[4], and FRAME[5]. Use a timeout interval of about 2RTT.**

   a. **Frame 2 is lost. Retransmission takes place upon timeout (as usual).**



   b. **Frame 2 is lost. Retransmission takes place either upon receipt of the first DUPACK or upon timeout. Does this scheme reduce the transaction time? Note that some end-to-end protocols (e.g., variants of TCP) use a similar scheme for fast retransmission.**


F1
F2
F3
F4
D2
1RTT
F5
D2
D2
F2
2RTT
Acks
F6
ACK6
3RTT