Pamela Mena – 00211564

Tarea 1

1. **Calculate the total time required to transfer a 1000-KB file in the following cases, assuming an RTT of 50 ms, a packet size of 1 KB data, and an initial 2 × RTT of "handshaking" before data is sent:**
   a. **The bandwidth is 1.5 Mbps, and data packets can be sent continuously.**

$$packet\ size = 1\ KB \quad RTT = 50\ ms \quad file\ size = 1000\ KB \quad bandwidth = 1.5\ Mbps$$

$$Propagation\ Delay = \frac{RTT}{2} = \frac{50\ ms}{2} = 25\ ms\ \left(\frac{1\ s}{1000\ ms}\right) = 0.025\ s$$

$$Transmission\ time = \frac{file\ size}{bandwidth} = \frac{1000\ KB}{1.5\ Mbps} = \frac{1000\ KB \times 8 \times 1024\ bits}{1.5 \times 10^6\ bps} = 5.461\ s$$

$$Handshaking = 2\ \times RTT = 2\ \times 50\ ms = 100\ ms\ \left(\frac{1\ s}{1000\ ms}\right) = 0.1\ s$$

$$Total\ Time = handshaking + propagation\ delay + transmission\ time$$

$$Total\ Time = 0.1\ s + 0.025\ s + 5.461\ s$$

$$Total\ Time = 5.586\ seconds$$

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

$$Initial\ Handshaking = 2\ \times RTT = 2\ \times 50\ ms = 100\ ms\ \left(\frac{1\ s}{1000\ ms}\right) = 0.1\ s$$

$$1000\ packets, waits\ for\ 999\ RTTs$$

$$Handshaking = RTT = 50\ ms = 50\ ms\ \left(\frac{1\ s}{1000\ ms}\right) = 0.05\ s$$

$$Total\ Time = 5.586\ s + 999 \times RTT$$

$$Total\ Time = 5.586\ s + 999 \times 0.1\ s$$

$$Total\ Time = 105.486\ seconds$$

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

$$1\ RTT \rightarrow 20\ packets$$

$$1000\ packets\ transmitted\ (only\ 20\ per\ RTT) = \frac{1000}{20} = 50\ RTT$$

$$\rightarrow last\ RTT\ transmission\ is\ 0 = \ 49\ RTT\ total$$

$$Propagation\ Delay = \frac{RTT}{2} = \frac{49\ RTTs}{2} = 24.5\ ms\left(\frac{1\ s}{1000\ ms}\right) = 0.0245\ s$$

$$Initial\ Handshaking = 2\ \times RTT = 2\ \times 50\ ms = 100\ ms\left(\frac{1\ s}{1000\ ms}\right) = 0.1\ s$$

$$Total\ Time = initial\ handshaking + propagation\ delay + \ 49\ RTTs$$

$$Total\ Time = 0.1\ s + 0.0245\ s + 49\ RTTs\ \times 0.05\ s$$

$$Total\ Time = 2.574\ seconds$$

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.

Initial	1 packet
1st RTT	2 packets
2nd RTT	4 packets
3rd RTT	8 packets
4th RTT	16 packets
5th RTT	32 packets
6th RTT	64 packets
7th RTT	128 packets
8th RTT	256 packets
9th RTT	489 packets

$$Handshaking\ time = 2\ \times RTT = 2\ \times 50\ ms = 100\ ms\left(\frac{1\ s}{1000\ ms}\right) = 0.1\ s$$

$$Propagation\ Delay = \frac{RTT}{2} = \frac{50\ RTTs}{2} = 25\ ms\left(\frac{1\ s}{1000\ ms}\right) = 0.025\ s$$

$$Total\ Time = handshaking\ time + propagation\ delay + 9\ RTTs$$

$$Total\,Time = 0.1\,s + 0.025\,s + 9\,RTTs\ \times 0.05\,s$$

$$Total\,Time = 0.575\,seconds$$

2. **One property of addresses is thar 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?**

Implementar una jerarquía de direcciones, este esquema puede organizar la red en grupos lógicos dependiendo del nivel de importancia, lo que manejar y escalar se vuelve más eficiente.

Designar direcciones administrativas para identificar el host de la red en lugar de permitir la dirección asignada de fábrica (identificador).

Otra característica importante son la longitud fija frente a la longitud de variable y la variable absoluta frente a la relativa.

Un ejemplo donde las direcciones de red pueden no ser únicas, es cuando se hace una llamada a un número gratuito para una empresa grande, es posible que todas las llamadas tengan la misma dirección (que ya no es única).

Otro ejemplo de una dirección no única es cuando se necesitan para alcanzar servidores equivalentes.

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.**

      Abrir un archivo es delay sensitive ya que la cantidad de intercambio de datos es menor durante la operación.

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

      Leer el contenido de un archivo puede ser de los dos tipos: delay sensitive o bandwidth sensitive. Va a depender del tamaño del archive, si el archive es más grande entonces va a ser bandwidth sensitive, y si el archivo es pequeño es más probable que sea delay sensitive.

   c. **List the content of a directory.**

      La lista del contenido de un directorio es delay sensitive debido a la cantidad de subdirectorios (tamaño pequeño), pero si el tamaño es más grande puede llegar a ser bandwidth sensitive.

   d. **Display the attributes of a file.**

La visualización de los atributos de un archivo es delay sensitive porque el tamaño de todos los atributos es relativamente pequeño.

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 overheads + loss bytes for packet data sizes of 1000, 5000, 10000, and 20000 bytes. Which of these sizes is optimal?**

$$packet\ number = \frac{1 \times 10^6}{data\ size}$$

$$total\ overhead = 100\ bytes\ \times packet\ number$$

$$total\ overhead = 100\ bytes\ \times \frac{1 \times 10^6}{data\ size}$$

$$loss = data\ size$$

$$total\ size = total\ overhead + loss$$

$$total\ size = 100\ bytes\left(\frac{1 \times 10^6}{data\ size}\right) + data\ size$$

1000 bytes:

$$total\ size = \frac{1 \times 10^8}{1000} + 1000$$
$$total\ size = 1.01 \times 10^5$$

5000 bytes:

$$total\ size = \frac{1 \times 10^8}{5000} + 5000$$
$$total\ size = 2.5 \times 10^4$$

10000 bytes:

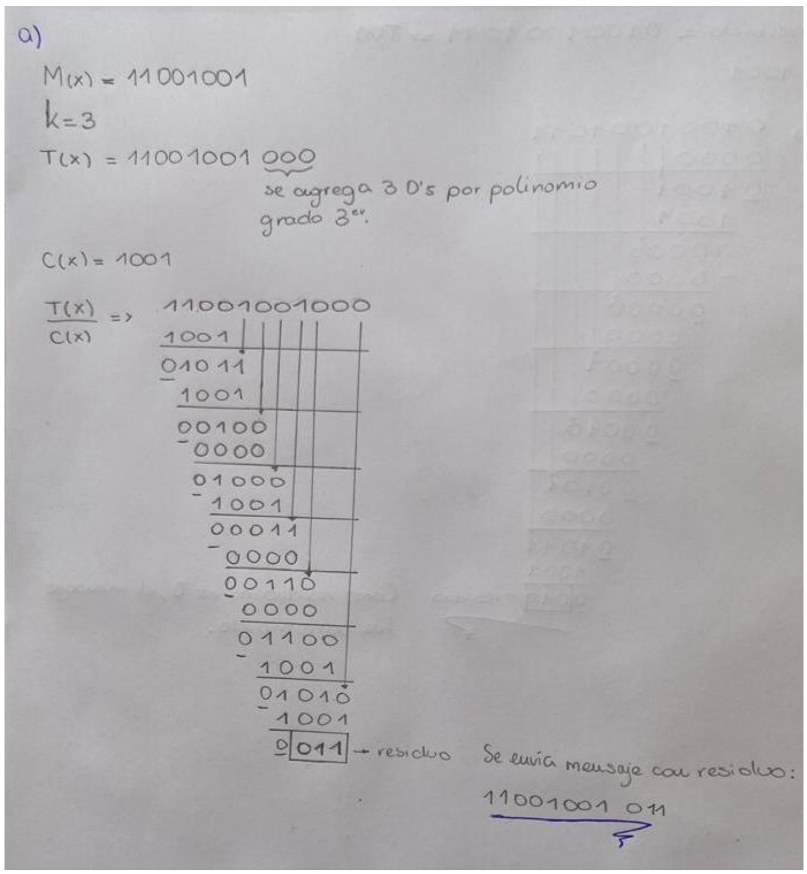
$$total\ size = \frac{1 \times 10^8}{10000} + 10000$$
$$total\ size = 2.0 \times 10^4$$

20000 bytes:

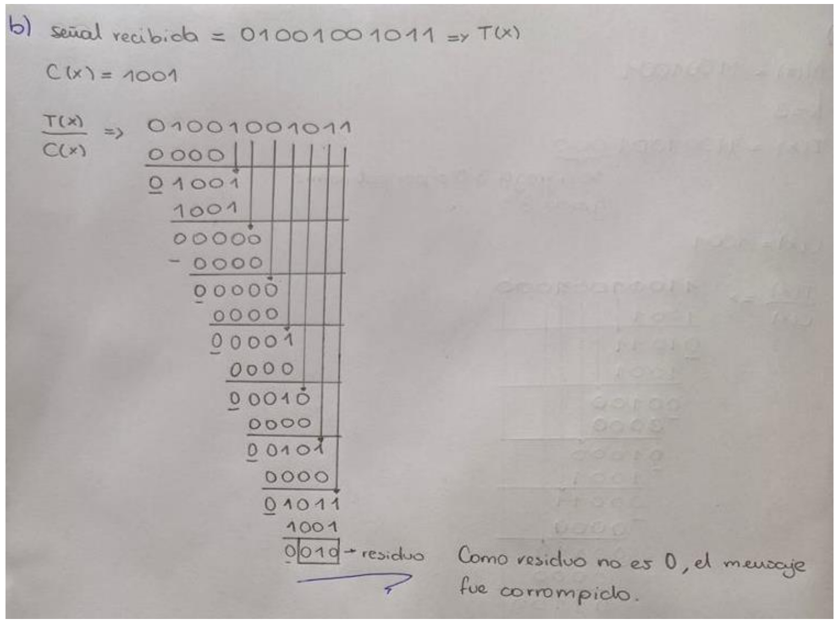
$$total\ size = \frac{1 \times 10^8}{20000} + 20000$$
$$total\ size = 2.5 \times 10^4$$

El tamaño óptimo de los datos del paquete será de 10000 bytes debido a que el número total de sobrecargas + los bytes perdidos son solo 2.0×10^4.

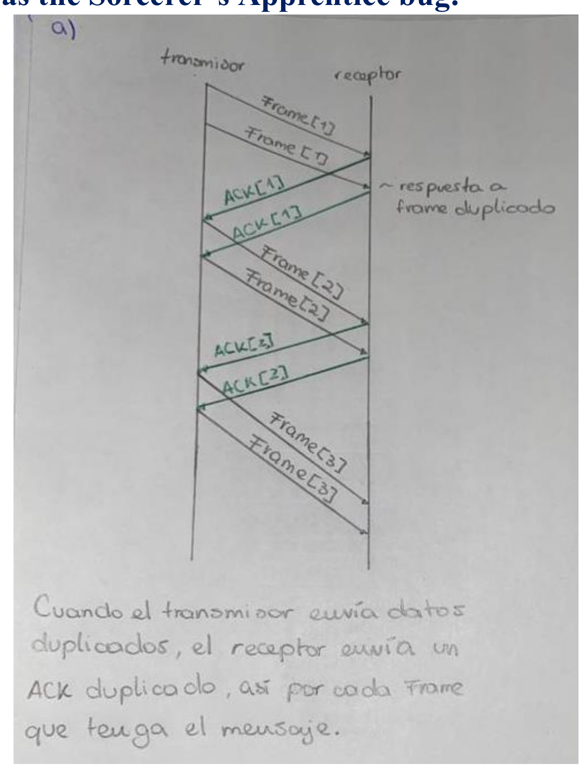
5. **Suppose we want to transmit the message 11001001 and protect it from errors using the CRC polynomial $x^3 + 1$.**
   a. **Use polynomial long division to determine the message that should be transmitted.**



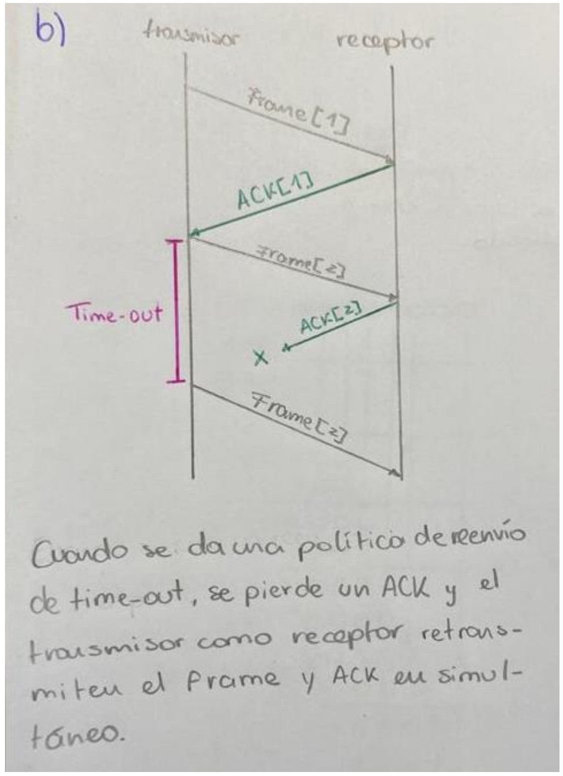
   b. **Suppose the leftmost bits gets inverted in transit. What is the result of the receiver's CRC calculation?**



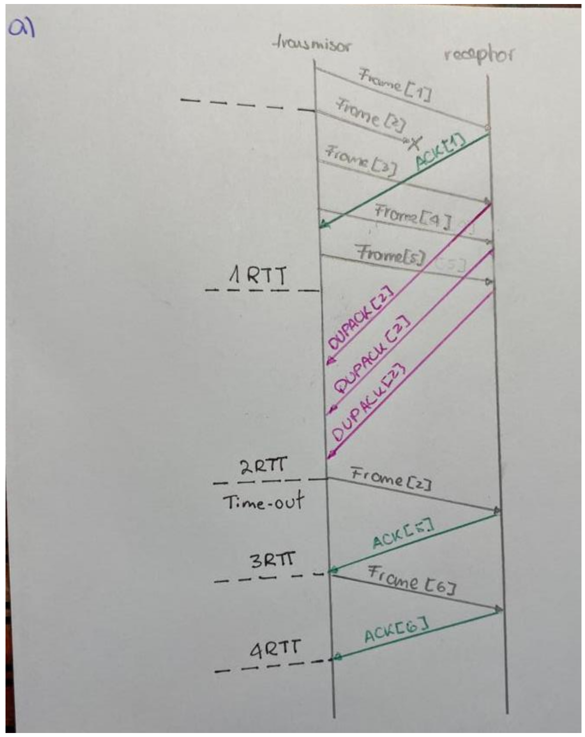
6. **In stop-and-wait transmission, suppose that both sender and receiver retransmit their last frame immediately on receipt of duplicate ACK or data frame; such a strategy is superficially reasonable because receipt of such a duplicate is most likely to mean the other side has experienced a timeout.**
   a. **Draw a timeline showing what will happen if the first data frame is somehow duplicated, but no frame is lost. How long will the duplications continue? This situation is known as the Sorcerer's Apprentice bug.**



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



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 duplicates 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 FRAME[2] after it already received FRAME[3], FRAME[4], and FRAME[5]. Use a timeout interval of about 2 x RTT.**
    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.**