1- Consider

- Propagation and queueing → 50 ms
  - ↳ due to ↳ RTT/2

If the packet size is 1KB → 10^3 bytes

1 byte → 8 bits, so the packet size is 8000 bits

→ Calculate (packet size)/ Tlbps

a) T= 8000 bits / 1.5Tlbps → 5.33 s

→ Add P & Q, but consider also handshaking

5.33 + 0.2 + 0.05 ⇒ 5.58 s

b) Considering RTT = 100 ms, if between the packet 1 and the packet 1000 there are 999 RTTs

→ 999 · (0,1) + 5,58 → 99,9 + 5,58 = 105,48

c) With handshaking → 2×RTT = 200 ms

→ Packet size of 1KB, considering 1000 packets, but 20 per RTT

⇒ 1000/20 → 50 RTT, with infinite bandwith

→ 49.5 RTTs, plus handshaking

⇒ 49,5 (0,1s) + 0,2 = 5,15 [s]

Consider that last batch arrives 0.5 RTT later here

d) If we send 2^n packets in the n RTT, we already have sent $2^{n+1}-1$ packets.

$2^{n+1} - 1 = 100 \rightarrow n = [\ln(1001)/\ln(2)] - 1$

$n \rightarrow 8.96 \approx 9$

→ with 0.5 RTT last batch, asuming infinite bandwidth and adding handshaking

→ 0.2 s + 9.5 (0.1 s) = 1,15 [s]

ESTILO

2. The point could be to make addresses hierarchical. Conside-ring different locations by level (customer, area, city, coun-try).

→ Address can be distinguished for being administrative-ly assigned, against those assigned by factory, used by ethernet.

→ There are other atributes to make a difference, which could be the fixed length and variable length, but at the same time the relative length in data.

→ Fixed length implies equal values of subnet ad-dresses in an adress block. However, in va-riable length those values are different without any form of structure in the number of values on each subnet address.

One thing could be to call a type of number for a large retailer, considering any of dozens of phone may an-swer. This is applied for phone with the same non-unique address.

→ An application for these addresses could be for reaching any of several equivalent servers.

3. Considering each one of the cases, there are some answers:

a) Delay-sensitive for opening a file

→ There is an exchange of data, but the amount of data is usually small (it doesn't affect the bandwidth), when the data is exchanged.

b) Bandwidth-sensitive for reading the contents of a file

→ This applies for large files, depending on the size of the file and characteristics of the data channel. In practice most of the files are small, beco-ming more focused on the delay.

c) Delay-sensitive for listing the contents of a directory

→ This could differ from the directory size, but ge-nerally directories are of modest size, smaller.

d) Delay-sensitive to display the attributes of a file

→ This is because those attributes are much smaller than the file itself.

4. Consider then, the packet amount

→ 10^6/ data size

→ The total overhead implies the previous amount, using 100 bytes

→ $100(10^6/\text{data size})$

→ The loss is simply data size here, so the total size is:

→ Total size = data size + $100 \cdot (10^6/\text{data size})$

For different sizes, it is the result.

Data size	overhead + loss
1000	$(10^8/1000) + 1000 \rightarrow 101000$
5000	$(10^8/5000) + 5000 \rightarrow 25000$
10 000	$(10^8/10000) + 10000 \rightarrow 20000$
20 000	$(10^8/20000) + 20000 \rightarrow 25000$

Based on the overhead + loss, the 1000 data size case is optimal.

5. a) Considering $T(x) = 11001001$, with a value $k=3$, a derivation could be $T(x) = 11001001\ 000$

→ Applying division

→
```
           11010011
     1001 | 11001001000
            1001
             1011
             1001
               1000
               1001
                 1100
                 1001
                  1010
                  1001
                    11  ← CRC
```

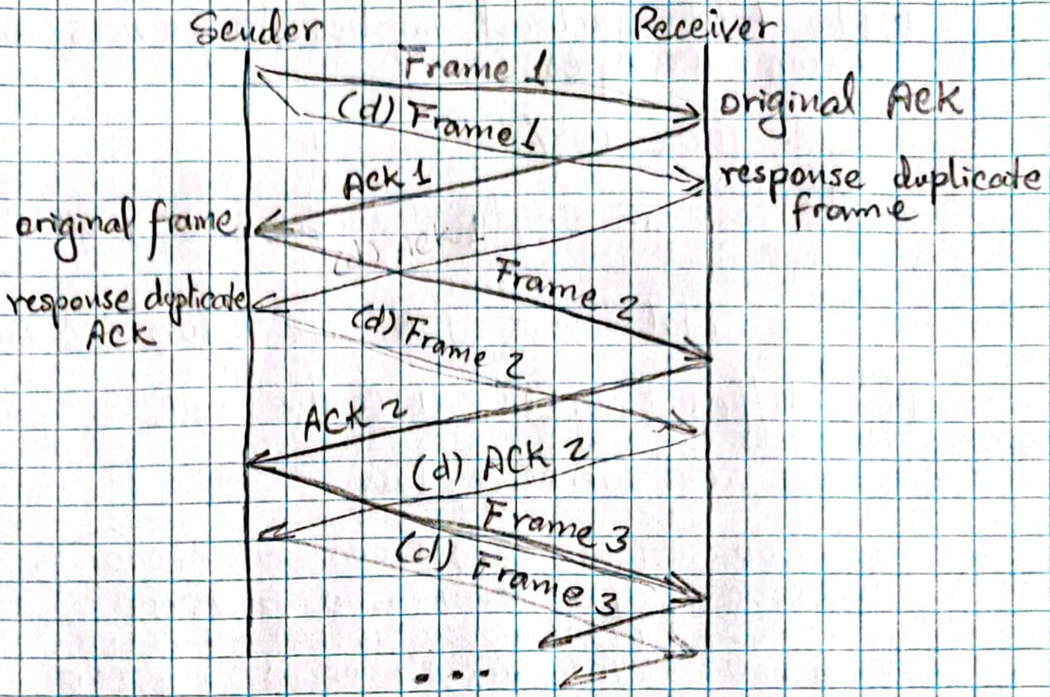
Answer: message sent is 11001001 011

b) with the leftmost bit inverted, the next result is:

```
          1000001
    1001 | 1001001011
           1001
              001011
                1001
                  10
```

→ with 10, the remainder is not zero, applies a hence ERROR!

6. a) The timeline then implies a duplication that could continue for the next frames, maybe with no ending, as it shows the next diagram.



b) In that case, a duplicate data frame must cross in the network with previous ACK for it.

→ If sender and receiver apply a resend-on-timeout strategy, with the same timeout interval

→ with an Ack lost, both sender and receiver will indeed retransmit the same time.

→ with retransmisions synchronized enough that they cross in the network, that would depend on other factors, helping to have modest latency delay or slow host

→ In right conditions, the phenomenom of the Sorcerer can be reproduced, in a reliable way.

7) Considering both cases, the transaction time could be reduced in terms of applied RTTs, considering the second case better.