CS425A: Assignment 2

Shrilakshmi S K (211012)

February 16, 2024

1 Question 1

Instructions to run the code:

1.1 Unmodified code:

Go to directory `cs425a2` and run the following commands in terminal:

```
g++ q1_code.cpp
./a.exe
```

Enter the frame size n and message size k. The code will output two examples on which the code is tested and their message, CRC, transmission message and validity.

1.2 Modified code:

Go to directory `cs425a2` and run the following commands in terminal:

```
g++ q1_modified_code.cpp
./a.exe
```

The code will output the message, CRC, transmission message before and after introducing error and validity.

2 Question 2

Consider a scenario where the window size is defined as $n = 2^k$, and the sender dispatches a sequence of frames numbered $(0, 1, ..., n-1)$ in this order. Assuming all frames are successfully received, the receiver then sends an acknowledgment back and anticipates the subsequent sequence of frames to follow the same numerical order.

Imagine a situation where the acknowledgment signal fails to reach the sender due to interference or another issue, leading the sender's timer to expire. Consequently, the sender retransmits the same sequence of frames from the previous window. Given that the receiver was expecting the next sequence, it mistakenly incorporates these repeated frames into the new window, creating an unintended duplication of the first window and causing an error.

On the other hand, when the window size is set to $n = (2^k - 1)$, this issue is avoided. In this setup, if the sender transmits a sequence of frames $(0, 1, 2, ...n-1)$ and faces the same scenario as before, the receiver expects frame number n next. However, if the sender begins to resend from the first frame (i.e., frame 0) following a lost acknowledgment, the receiver's expectation of frame n prevents the previous error. This discrepancy in expectation ensures that the duplication error encountered with a window size of 2^k does not occur, leading to a preference for limiting the window size to $2^k - 1$.

3 Question 3

In a communication system using k-bit sequence numbering, the total possible unique frame numbers are limited to 2^k. We examine scenarios where the size of the sliding window can either permit or prevent overlapping of consecutive windows.

Assuming an overlapping scenario, the initial window is comprised of frames numbered $\{0, 1, 2, \ldots, w-1\}$, followed by a window with frames $\{w, w+1, \ldots, 0 \ldots\}$. If the receiver acknowledges the receipt of first frame and prepares for second frame, but the sender fails to receive this acknowledgment due to a lossy channel and times out, it will retransmit first frame starting

with frame 0. The receiver, expecting second frame, may erroneously accept this frame 0 in buffer as the frame of second frame, resulting in error due to the receiver's misalignment.

In case of non-overlapping windows, it can be confidently asserted that ambiguity is nonexistent, as each frame can be distinctly mapped to a specific window. Take, for example, the first batch of w frames, which are allocated to first frame; subsequently, the ensuing w frames, distinct from the initial batch, are categorically assigned to second frame.

The key is to determine a window size that prevents overlap, ensuring each window contains exactly w frames, totaling $2w$ distinct frames for two consecutive windows. Since $2w$ must be less than or equal to 2^k, it follows that $w \leq 2^{k-1}$. Therefore, in a Selective-Repeat ARQ protocol with k-bit sequence numbers, the largest permissible window size is 2^{k-1}.

4 Question 4

U is required to be greater than or equal to 50%. The limitation on the variable a is deduced from the efficiency equation of the stop-and-wait flow control, given by $U = \frac{1}{1+2a}$.

Since U must be equal to or greater than 50%, we start by setting up the inequality $\frac{1}{1+2a} \geq \frac{1}{2}$. To solve for a, we take the reciprocal of both sides resulting in $1 + 2a \leq 2$. Simplifying further, we find $2a \leq 1$, which gives us $a \leq \frac{1}{2}$.

Next, we apply this constraint to a, which is defined by the ratio $\frac{t_{prop}}{t_{frame}}$, where t_{prop} is given as 20 ms. We require that the frame transmission time t_{frame} be at least 40 ms to satisfy the condition $\frac{1}{2} \geq \frac{20ms}{t_{frame}}$.

If S_F represents the frame size in bits, the relationship $40ms \leq t_{frame} = \frac{S_F}{(4kbps)}$ is established. Rearranging the terms and converting the rate to bps, we find $40 \cdot 10^{-3} \leq \frac{S_F}{(4000bps)}$, leading to the conclusion that 160 bits $\leq S_F$.

Therefore, the frame size must be at least 160 bits to achieve a minimum efficiency of 50% in the stop-and-wait protocol.

5 Question 5

5.1 Part a

Given the probability of a single bit error as $p = 10^{-3}$, the probability of a specific bit being error-free is $1-p = 0.999$. Let E_i represent the event where the i^{th} bit encounters an error. Consequently, $P(E_i) = p$ and its complement, $P(E_i') = 1 - p$, indicating the bit is not in error.

We want the probability that the first four bits are all correct, denoted by $P(E_1' \cap E_2' \cap E_3' \cap E_4')$, which translates to the simultaneous absence of errors in the first, second, third, and fourth bits. Given that these error events are independent, the combined probability is the product of their individual probabilities:

$$P(E_1' \cap E_2' \cap E_3' \cap E_4') = \prod_{i=1}^{4} P(E_i') = \prod_{i=1}^{4} (1-p) = (1-p)^4 = (1-10^{-3})^4$$

Using binomial approximation, this simplifies to $1 - 4 \cdot 10^{-3} = 0.996$.

5.2 Part b

The occurrence of at least one error is considered the opposite of the event where no errors occur. Denote by F the event where at least one error is present. The probability of F, noted as $P(F)$, was established in the initial part of the analysis. Our focus is solely on $P(F)$:

$$P(F) = 1 - P(\bar{F}') \approx 1 - 0.996 = 0.004 = 4 \cdot 10^{-3}$$

5.3 Part c

With a parity bit added, a 5-bit frame is considered. While a parity check detects odd numbers of bit inversions, it fails to detect even numbers, as with 2 or 4 errors. The probability of such undetected errors, where there are exactly two or exactly four errors, is sought.

For exactly 2 errors, the number of possible pairs of erroneous bits is determined by $\binom{5}{2}$, since we have 5 bits to consider for errors, including the parity

bit. Each bit error occurs independently, so the joint probability of two distinct bits, i and j, both being erroneous is $P(E_i \cap E_j) = P(E_i) \cdot P(E_j) = p \cdot p = p^2$.

However, to ensure that there are precisely two errors, we must also consider the probability of no errors occurring in the other bits. This is represented by the probability of the complement of error events for all bits other than i and j, $P\left(\bigcap_{k \neq i,j} E'_k\right)$, which equals $\prod_{k \neq i,j} P(E'_k) = (1-p)^3$.

Combining these probabilities, we get the probability of exactly 2 errors as $\binom{5}{2} \cdot p^2 \cdot (1-p)^3$, which simplifies to approximately $10 \cdot 10^{-6} \cdot 0.997$, resulting in $9.97 \cdot 10^{-6}$.

For the case of exactly 4 errors, the calculation is similar. We find the number of ways to choose 4 erroneous bits from 5, which is $\binom{5}{4}$. The probability is thus $5 \cdot p^4 \cdot (1-p)$, considering there are 4 errors and 1 bit without error, resulting in $4.995 \cdot 10^{-12}$.

Thus, the overall probability that the parity bit will not detect an error is the sum of the probabilities of getting 2 or 4 errors:

$$9.97 \cdot 10^{-6} + 4.995 \cdot 10^{-12} \approx 9.97 \cdot 10^{-6}$$

6 Question 6

The message M is first appended with a number of zeros equivalent to the $|P|$ minus one, which is standard practice in CRC computations. Specifically, five zeros are added to M. Following this, the augmented message is divided by P using modulo-2 division. The outcome of this division yields both a quotient and a remainder; the quotient is 10110110 and the remainder is 11010. The remainder from this operation is the CRC value, which is 11010. Figure 1 shows the computations.

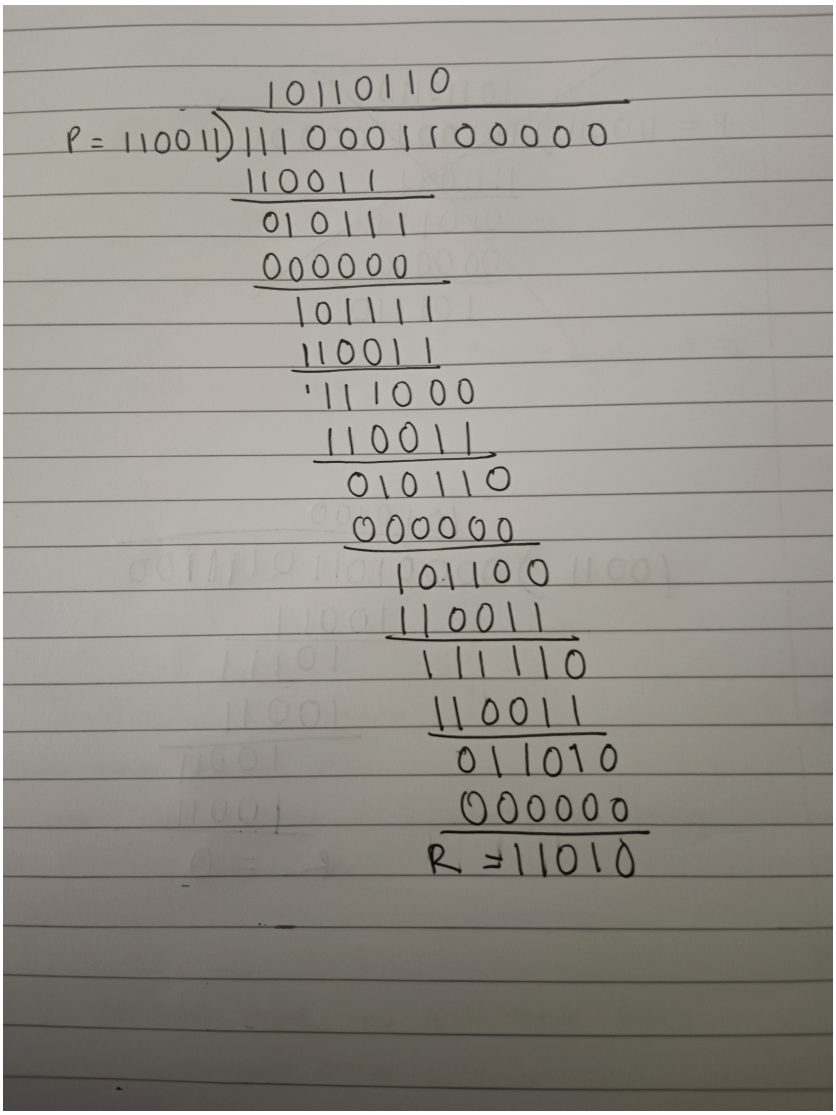
10110110
P = 11001) 11100011000000
11001 1
01011 1
00000 0
10111 1
11001 1
01110 0
00000 0
11100 0
11001 1
00101 0
00000 0
10100 0
11001 1
01101 0
11001 1
01100 0
00000 0
R = 11010

Figure 1:

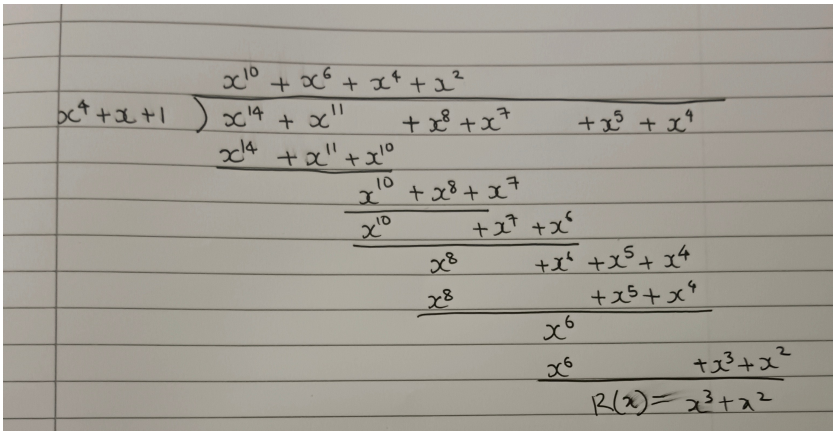

Figure 2:

7 Question 7

7.1 Part a

Converting the binary string into a polynomial.

$$M = 10010011011$$

$$M(X) = X^{10} + X^7 + X^4 + X^3 + X + 1$$

Figure 2 shows the division of $X^4 \cdot M(X)$ by $P(X)$.

We get remainder polynomial $R(X) = X^3 + X^2$. Converting it back to binary string, we get remainder $R = 1100$. We encode the message by appending R to M.

$$MR = 100100110111100$$

7.2 Part b

Since the error pattern $E = 100010000000000$ suggests that 1st and 5th bit are flipped, we flip 1st and 5th bit of MR to get what was actually received. Let the received message be A. Then $A = MR$ `xor` $E = 000110110111100$

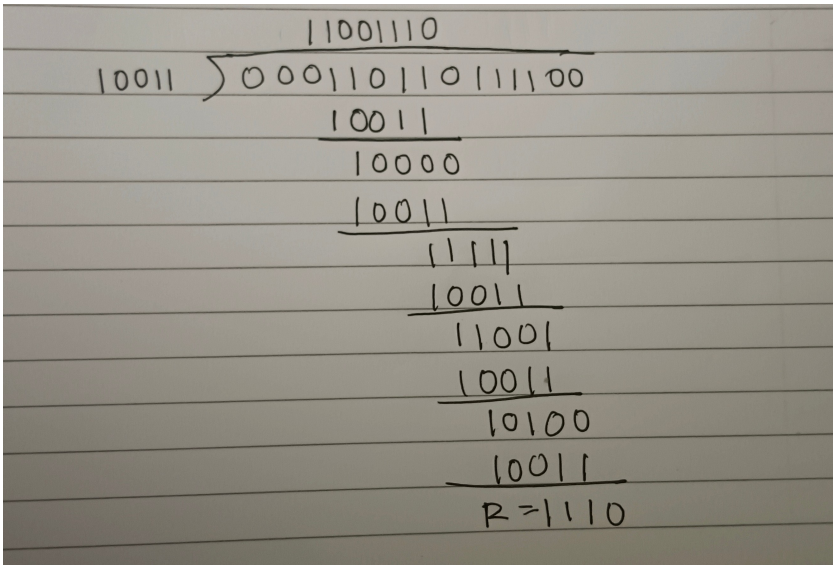

Figure 3:

Error detection is performed by dividing the received bit pattern by $P = 10011$. The presence of an error is indicated if the remainder from this division is non-zero. Given that a non-zero remainder is observed, an error is immediately identified. The modulo-2 division is depicted in Figure 3.

The non-zero remainder $R = 1110$ equates to $R(X) = X^3 + X^2 + X$ and confirms the detection of an error.

7.3 Part c

In this part, $E = 100110000000000$. Then received message $A = MR$ `xor` $E = 000010110111100$

Error detection is performed by dividing the received bit pattern by $P = 10011$. The presence of an error is indicated if the remainder from this division is non-zero. Given that a non-zero remainder is observed, an error is immediately identified. The modulo-2 division is depicted in Figure 4.

The zero remainder $R = 0000$ equates to $R(X) = 0$ does not detect an error.

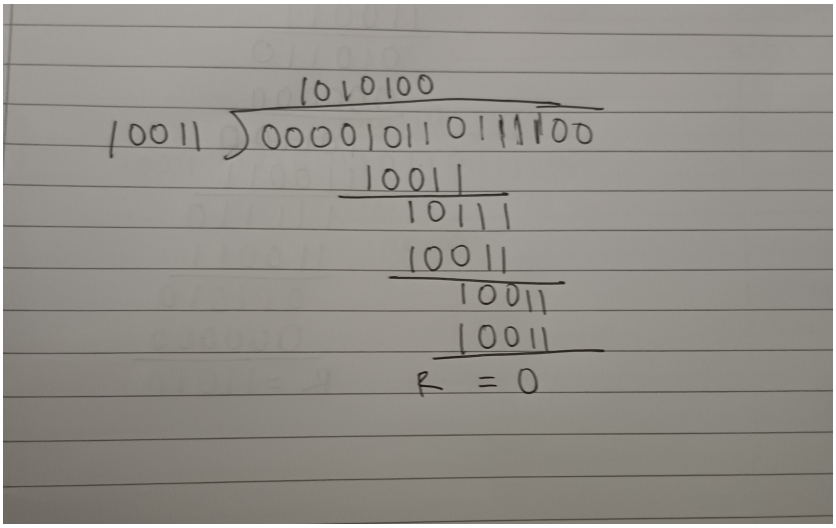
1010100
10011) 00001011011100
10011
10111
10011
10011
10011
R = 0

Figure 4: