Sarthak Kalankar 210935

CS425: Assignment 2

Problem 1

The folder contains 2 (.cpp) files :-

- program.cpp
- modified_program.cpp

1st Part

This takes care of the initial part of the question i.e. it generates a n-bit frame for the given data(k bits) and CRC pattern(n-k+1 bits).

Process of running-

- Run ./script_main.sh in the terminal.
- It will run the code on 2 sample inputs.
- Then a prompt will come if you want to enter more data.
- Enter 'Y' if you want to enter more data.
- If you enter 'N', program will turn off, similar behaviour if a wrong input is given.

2nd Part

This takes care of the latter part of the question i.e. it generates a 15-bit frame for the generated data(k = 10 bits) and CRC pattern(110101).

Process of running-

- Run ./script_modified.sh in the terminal.
- It will run the desired code, giving the generated data and the 15-bit frame.
- Then it introduces random errors in 3 bits, you can change this no. in line 74 of modified_program.cpp .
- Then it tells us if the error is being detected or not.

Problem 2

In the Go-Back-N ARQ protocol using k-bit sequence numbers, there arises a practical limitation when considering the window size. If one were to set the window size equal to 2^k, it creates a scenario where ambiguity can severely impact the protocol's reliability. For example, when the sender transmits the p^{th} frame and subsequently receives an acknowledgment (RR_p), it is permitted to send frames up to 2^k, potentially saturating the receiver's buffer. Upon consuming all the frames, the receiver might send RR_p again. This situation leads to confusion at the sender's end, as it cannot determine whether RR_p indicates:

- A cumulative acknowledgment for all frames being correctly received.
- A request for retransmission of all frames due to errors, implying none were correctly processed.

Both cases will give the same acknowledgment i.e RR_p.

The full use of 2^k sequence numbers in Go-Back-N ARQ leads to ambiguity, as no sequence number is left unused, blurring the line between complete success and total failure in transmission. Limiting the window size to $2^k - 1$ resolves this, ensuring clear communication by avoiding sequence number confusion and enhancing protocol reliability and error resistance.

Problem 3

In the **Selective-Reject ARQ mechanism**, which uses k-bit sequence numbers, the maximum window size must be such that it prevents any ambiguity in the identification of frames. It is essential to ensure that the window size allows for the unique identification of both new and retransmitted frames without confusion.

The Selective-Reject ARQ mechanism handles errors by allowing the receiver to selectively reject individual frames that have been received in error, rather than rejecting a whole block of frames, which is the case with other mechanisms such as Go-Back-N ARQ. Therefore, it is crucial to have a sufficient range of sequence numbers to avoid overlaps between windows of frames being transmitted and acknowledged.

With k-bit sequence numbers, we can have 2^k unique sequence numbers. To avoid any ambiguity, the window size should be **at most half of the total** sequence numbers, which leads us to a maximum window size of 2^{k-1}. This window size ensures that at any given time, the number of frames that have been sent but not yet acknowledged does not exceed the sequence number space available, thereby preventing any overlaps or confusion about whether a frame is new or a retransmission.

Consider the case of a Selective-Reject ARQ mechanism that utilizes a 3-bit sequence number. The sequence numbers range from 0 to $2^3 - 1$, which gives us a total of 8 unique sequence numbers. The maximum window size in this scenario should be less than half the total number of sequence numbers to avoid overlaps, hence the window size must be $2^{3-1} = 4$.

To illustrate this with an example, let us assume a window size of 5, which is more than the maximum allowed size, and observe the potential issue:

- Station X sends frames numbered $0, 1, 2, 3, 4$ to station B.
- Station Y receives all five frames and sends a cumulative acknowledgment for frame 5 (RR_5).
- Due to a noise burst, the acknowledgment RR_5 is lost.
- Station X times out and retransmits frame 0.
- Station Y has already advanced its receive window to accept frames $5, 6, 7, 0, 1$, assuming that frame $5, 6, 7$ have been lost and that frame 0 is a new frame, it accepts the frame 0 which is a duplicate one.

Now, lets take the window size to be 4:

- Station X sends frames numbered $0, 1, 2, 3$ to station B.
- Station Y receives all five frames and sends a cumulative acknowledgment for frame 4 (RR_4).
- Due to a noise burst, the acknowledgment RR_4 is lost.
- Station X times out and retransmits frame 0.
- Station Y has already advanced its receive window to accept frames $4, 5, 6, 7$. So. the receiver gets to know that the acknowledgment has been lost and it can send the acknowledgment again.

This scenario shows that using a window size greater than 2^{k-1} can lead to ambiguities and errors in frame identification. Thus, it reinforces the rule that the maximum window size in a Selective-Reject ARQ mechanism using k-bit sequence numbers is 2^{k-1}.

Problem 4

In stop-and-wait flow control, the efficiency U is defined by the relation:

$$U = \frac{1}{1 + 2a}$$

where a is the ratio of the propagation time T_{prop} to the frame transmission time T_{frame}, expressed as:

$$a = \frac{T_{\text{prop}}}{T_{\text{frame}}}$$

For a system where the propagation time T_{prop} is given as 20 ms, we want to ensure that the efficiency U is at least 50%. To find the minimum frame size S_f that satisfies this efficiency, we set up the inequality:

$$U \geq 50\% = \frac{1}{2}$$

Taking the reciprocal of the efficiency relation and setting the inequality, we get:

$$\frac{1}{U} \leq 2$$

$$1 + 2a \leq 2$$

$$2a \leq 1$$

$$a \leq \frac{1}{2}$$

Now, we incorporate the given value of T_{prop} and solve for the frame size S_f, assuming the transmission rate is 4 kbps. We express T_{frame} as:

$$T_{\text{frame}} = \frac{S_f}{4 \times 10^3}$$

Substituting a with $\frac{T_{\text{prop}}}{T_{\text{frame}}}$ and the values into the efficiency formula, we find:

$$\frac{20 \times 10^{-3}}{\left(\frac{S_f}{4 \times 10^3}\right)} \leq 0.5$$

Solving for S_f, we arrive at the conclusion that:

$$S_f \geq 160 \text{ bits}$$

Thus, the frame size must be at least **160 bits** to achieve the required 50% efficiency in stop-and-wait flow control.

Problem 5

Part 1

The event of all bits being received correctly can be expressed as:

$$p(\text{All correct}) = (1-p)^4 = (1-10^{-3})^4 \approx 1 - 4 \times 10^{-3} = 0.996$$

Part 2

The probability of having at least one error is the complement of having no errors:

$$p(\text{Atleast 1 Correct}) = 1 - p(\text{All Correct})$$

$$p(\text{Atleast 1 Correct}) \approx 1 - 0.096 = 0.004$$

Part 3

With the inclusion of a parity bit, the total number of bits becomes five. The parity bit mechanism fails to detect an even number of errors. Therefore, we consider the cases with 2 and 4 errors:

$$p(\text{Undetected Error}) = p(\text{Exactly 2 errors}) + p(\text{Exactly 4 errors})$$

$$p(\text{Undetected Error}) = \binom{5}{2}(1-p)^3 \times p^2 + \binom{5}{4}(1-p) \times p^4$$

$$p(\text{Undetected Error}) = \binom{5}{2}(1-10^{-3})^3 \times 10^{-6} + \binom{5}{4}(1-10^{-3}) \times 10^{-12} \approx 9.97 \times 10^{-6}$$

Problem 6

Given a message $M = 11100011$ and a polynomial $P = 110011$, we calculate the CRC as follows:

1. **Prepare the Message**: Append $|P| - 1 = 6 - 1 = 5$ zeros to M, resulting in $M' = M00000 = 1110001100000$.

2. **Perform Division**: The modulo-2 division process is shown below:

	10110110	(Quotient)
$P = 110011$	1110001100000	(Dividend)
	110011	(Subtract)
	010111	
	000000	(Subtract)
	101111	
	110011	(Subtract)
	111000	
	110011	(Subtract)
	010110	
	000000	(Subtract)
	101100	
	110011	(Subtract)
	111110	
	110011	(Subtract)
	011010	
	000000	(Subtract)
	11010	(Remainder/CRC)

3. **Obtain the CRC**: The remainder from the division is the CRC. For the given M and P, the CRC is found to be **11010**.

4. **Result**: The transmitted message will be M followed by the CRC, which is 1110001111010. This message should now be perfectly divisible by P with no remainder if there are no errors.

By including the CRC in the message, we ensure that any errors in transmission can be detected by performing the same division at the receiver's end. If the remainder is zero, the message is considered error-free. If there is a remainder, it indicates that the message has been corrupted during transmission.

Problem 7

Part (a)

Given a binary message and a pattern sequence, we calculate the CRC as follows:

1. The pattern sequence is given by the polynomial $P(x) = x^4 + x + 1$.
2. We convert the binary message into a polynomial sequence, $M(x) = x^{10} + x^7 + x^4 + x^3 + x + 1$
3. We append four zero bits to $M(x)$ to account for the degree of $P(x)$ (which is 4), giving us $T(x) = x^{14} + x^{11} + x^8 + x^7 + x^5 + x^4$.
4. We divide $T(x)$ by $P(x)$ using polynomial long division.

The polynomial long-division process is as follows:

$$
\begin{array}{r}
 \\
x^4 + x + 1) \\
 \\
 \\
 \\
 \\
 \\
 \\
 \\
\end{array}
\begin{array}{l}
x^{10} + x^6 + x^4 + x^2 \\
x^{14} + x^{11} + \quad\; x^8 + x^7 + x^5 + x^4 \\
\underline{x^{14} + x^{11} + x^{10}} \\
\quad\; x^{10} + x^8 + x^7 \quad\; + x^5 + x^4 \\
\quad\; \underline{x^{10} \quad\; + x^7 + x^6} \\
\quad\quad\; x^8 + \quad\; x^6 + x^5 + x^4 \\
\quad\quad\; \underline{x^8 \quad\quad\quad\; + x^5 + x^4} \\
\quad\quad\quad\quad\; x^6 \\
\quad\quad\quad\quad\; \underline{x^6 + x^3 + x^2} \\
\quad\quad\quad\quad\quad\; x^3 + x^2
\end{array}
$$

5. The remainder of this division is $R(x) = x^3 + x^2$.
6. We convert the remainder $R(x)$ into the 4-bit binary sequence $R = 1100$.
7. Finally, we concatenate the remainder R to the original message M to form the transmitted message $T = M||R = 100100110111100$.

This transmitted message includes the CRC and can be sent over the network. The receiver can perform the same division to check for errors. If the remainder is zero, the message is considered error-free.

Part (b)

1. **Prepare the Message**: The received message $W = T \oplus E$ (E is the error pattern). $W = 100100110111100 \oplus 100010000000000 = 000110110111100$

2. **Perform Division**: The modulo-2 division process is shown below:

	11001110	(Quotient)
$P = 10011$	000110110111100	(Dividend)
	10011	(Subtract)
	10000	
	10011	(Subtract)
	0011111	
	10011	(Subtract)
	11001	
	10011	(Subtract)
	10100	
	10011	(Subtract)
	1110	(Remainder/CRC)

3. **Obtain the CRC**: The remainder from the division is 1110 i.e. non-zero. So, **an error will be** detected.

Part (c)

1. **Prepare the Message**: The received message $W = T \oplus E$ (E is the error pattern). $W = 100100110111100 \oplus 100110000000000 = 000010110111100$

2. **Perform Division**: The modulo-2 division process is shown below:

	1010100	(Quotient)
$P = 10011$	000010110111100	(Dividend)
	10011	(Subtract)
	010111	
	10011	(Subtract)
	010011	
	10011	
	0000	(Remainder/CRC)

3. **Obtain the CRC**: The remainder from the division is 0. So, **no error** will be detected.