Date: 16.02.2024

CS425A: COMPUTER NETWORKS

Assignment-2

Submitted by:
Anay Sinhal
EXY23017

QUESTION 1:

Please refer "Q1_code.cpp" uploaded as a separate file.

Compilation:

- Ensure g++ is installed.
- In the terminal, navigate to the file's directory.
- Compile with g++ -o q1Sol Q1_code.cpp.

Running:

- Execute with ./q1Sol on Unix/Linux or q1Sol.exe on Windows.
- Input the polynomial P and message bit-length k as prompted.
- For the given question, $P = 110101$ and $k = 10$.

QUESTION 2:

The Go-back-N ARQ protocol restricts the sender's window size to $2^k - 1$ rather than 2^k to prevent sequence number ambiguity. With k-bit sequence numbers, a window size of 2^k would allow the sequence numbers to wrap around and restart from zero within a single window, making it impossible to distinguish between new frames and retransmissions. This limitation ensures a clear separation between the current window of frames being transmitted and the next, preventing the receiver from incorrectly accepting a frame from the new window as a retransmitted frame from the previous window.

For example, if a sender transmits 7 frames (in a scenario with $2^3 - 1 = 7$ window size) and begins receiving acknowledgments, there's no risk of mistaking a new frame for an old one due to the sequence numbers wrapping around. This clear separation maintains the integrity of the data transmission process.

QUESTION 3:

In Selective-Reject ARQ using k-bit sequence numbers, the maximum window size is 2^{k-1}. This restriction is necessary to avoid confusion over whether an acknowledgment is for a new frame or a retransmitted frame when sequence numbers wrap around.

For example, consider a system using 3-bit sequence numbers, which allows for 8 distinct sequence numbers (0 to 7). With a window size of $2^{3-1} = 4$, we can have a maximum of 4 frames unacknowledged at any time. If we were to use all 8 sequence numbers, frame 0 could be confused as a new frame or a retransmission of the 8th frame due to sequence number wrap-around. Therefore, a window size of 2^{k-1} ensures a safe margin to distinguish between frames within the sequence number space.

QUESTION 4:

For a channel with a data rate of 4 kbps and a propagation delay of 20 ms, to achieve an efficiency of at least 50% using the stop-and-wait ARQ protocol, we can derive the minimum frame size as follows:

Efficiency (U) in stop-and-wait ARQ is given by:

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

where a is the ratio of the propagation delay to the transmission time.

Given that the minimum efficiency is 50%, we have:

$$\frac{1}{1 + 2a} \geq \frac{1}{2}$$

The propagation delay (τ) is 20 ms (or 20×10^{-3} seconds), and the bit rate (R) is 4 kbps (or 4×10^{-3} bits per second). If x is the number of bits in a frame, then the transmission time (T_t) is $\frac{x}{R}$. Therefore, $a = \frac{\tau}{T_t}$.

Substituting τ and R into a gives us:

$$a = \frac{20 \times 10^{-3}}{x \,/\, 4 \times 10^{-3}}$$

Substituting a into the efficiency formula and solving for x yields:

$$\frac{1}{1 + 2 \times \frac{20 \times 10^{-3}}{x \,/\, 4 \times 10^{-3}}} \geq \frac{1}{2}$$

$$\frac{1}{1 + \frac{40 \times 10^{-3}}{x \,/\, 4 \times 10^{-3}}} \geq \frac{1}{2}$$

$$1 + \frac{160}{x} \leq 2$$

$$\frac{160}{x} \leq 1$$

$$\boldsymbol{x \geq 160}$$

Therefore, the frame size must be at least 160 bits to maintain an efficiency of 50% or higher.

QUESTION 5:

For a frame consisting of one character with 4 bits and a bit error probability of 10^{-3}, we can calculate the probabilities as follows:

(a) The probability P_{no_error} that the received frame contains no errors is the probability that each bit is received without error:

$$P_{no\_error} = (1 - error\_probability)^{number\_of\_bits}$$

$$P_{no\_error} = (1 - 10^{-3})^4$$

$$P_{no\_error} = 0.999^4$$

$$P_{no\_error} \approx 0.996$$

(b) The probability $P_{atleast\_one\_error}$ that the received frame contains at least one error is the complement of P_{no_error}:

$$P_{atleast\_one\_error} = 1 - P_{no\_error}$$

$$P_{atleast\_one\_error} \approx 1 - 0.996$$

$$P_{atleast\_one\_error} \approx 0.004$$

(c) The probability $P_{undetected\_errors}$ that a frame is received with errors that are not detected, given a parity bit is added for error detection, is calculated by considering even number of errors in a 5-bit frame (4 data bits + 1 parity bit), since an odd number of errors would be detected by the parity bit. The formula for undetected errors with a parity bit, based on the binomial probability distribution, is given as follows:

$$P_{undetected\_errors} = P_{2\ bits\ in\ error} + P_{4\ bits\ in\ error}$$

Where,

$$P_{2\ bits\ in\ error} = \binom{5}{2}\ (10^{-3})^2\ (1 - 10^{-3})^{5-2}$$

$$P_{4\ bits\ in\ error} = \binom{5}{4}\ (10^{-3})^4\ (1 - 10^{-3})^{5-4}$$

When we calculate these probabilities and sum them up, we get:

$$P_{undetected\_errors} = \binom{5}{2}\ (10^{-3})^2\ (1 - 10^{-3})^3 + \binom{5}{4}\ (10^{-3})^4\ (1 - 10^{-3})$$

$$P_{undetected\_errors} = 10 \times (10^{-3})^2 \times (0.999)^3 + 5 \times (10^{-3})^4 \times 0.999$$

$$P_{undetected\_errors} = 10 \times 10^{-6} \times 0.997002999 + 5 \times 10^{-12} \times 0.999$$

$$P_{undetected\_errors} \approx 9.97 \times 10^{-6} + 4.995 \times 10^{-12}$$

$$P_{undetected\_errors} \approx 9.97 \times 10^{-6}$$

QUESTION 6:

Given the polynomial $P = 110011$ and the message $M = 11100011$, we append zeros equivalent to the degree of $P - 1$ to the message M and perform a binary division to find the CRC.

The steps are as follows:

1. Append five zeros to M: 11100011 → 1110001100000.
2. Divide by P using binary long division (XOR operation).

The binary long division process:

```
             10110110
P = 110011 | 1110001100000
             110011
              010111
              000000
               101111
               110011
                111000
                110011
                 010110
                 000000
                  101100
                  110011
                   111110
                   110011
                    011010
                    000000
                     11010
```

CRC = 11010

QUESTION 7:

(a) The initial step in encoding the message using CRC involves translating the binary sequence into polynomial notation. For the message $M = 10010011011$, this translates to the polynomial expression $M(x) = x^{10} + x^7 + x^4 + x^3 + x + 1$.

Next, we must prepare the message for the division by multiplying it by x^4, which corresponds to padding the original message with four zeros to account for the degree of the generator polynomial $P(x) = x^4 + x + 1$.

The multiplication yields $M'(x) = x^{14} + x^{11} + x^8 + x^7 + x^5 + x^4$. We then proceed with the division of $M'(x)$ by $P(x)$, which is the core step in determining the CRC.

$$
\begin{array}{ll}
P(x) = x^4 + x + 1 & \underline{x^{10} + x^6 + x^4 + x^2} \\
 & x^{14} + x^{11} + x^8 + x^7 + x^5 + x^4 = x^4 \cdot M(x) \\
 & \underline{x^{14} + x^{11} + x^{10}} \\
 & \qquad\qquad \underline{x^{10} + x^8 + x^7} \\
 & \qquad\qquad\qquad\qquad \underline{x^8 + x^6 + x^5 + x^4} \\
 & \qquad\qquad\qquad\qquad\qquad \underline{x^6} \\
 & \qquad\qquad\qquad\qquad\qquad\qquad x^3 + x^2 = R(x)
\end{array}
$$

- The dividend is expanded to $x^{14} + x^{11} + x^8 + x^7 + x^5 + x^4$.
- The divisor $P(x)$ remains $x^4 + x + 1$.
- The result of the division provides us with the remainder polynomial $R(x) = x^3 + x^2$.

Upon converting $R(x)$ back to binary form, we obtain $R = 1100$. The final encoded message, which combines the original message with the calculated remainder, is given by $MR = 100100110111100$. This encoded sequence integrates both the data and error-checking elements, ready for transmission.

(b) To ascertain the received sequence, we invert the first and fifth bits of the encoded message MR, yielding:

$W = 000110110111100$

Alternatively, this result could be achieved by performing an XOR operation between the error pattern and MR. This method leverages the principle that XOR-ing with 1 flips the bit, whereas XOR-ing with 0 leaves it unchanged.

The next step is to validate the presence of an error by dividing the received sequence W by the polynomial $P(x)$. For this calculation, we use the binary representation of $P(x)$, which is $P = 10011$, simplifying the polynomial to its binary equivalent.

Executing the binary division (or modulo 2 division) and observing the remainder, we find:

$$
\begin{array}{rl}
 & 11001110 \\
\cline{2-2}
P = 10011 & \big)\, 000110110111100 \\
 & \underline{10011} \\
 & \;\; 10000 \\
 & \;\; \underline{10011} \\
 & \qquad 11111 \\
 & \qquad \underline{10011} \\
 & \qquad\;\; 11001 \\
 & \qquad\;\; \underline{10011} \\
 & \qquad\quad 10100 \\
 & \qquad\quad \underline{10011} \\
 & \qquad\quad \mathbf{1110}
\end{array}
$$

Given the **non-zero** remainder $R = 1110$, which in polynomial form is $R(x) = x^3 + x^2 + x$, we conclude that an **error** is present in the received sequence.

(c) Determining the received sequence involves XOR-ing the transmitted pattern with the given error pattern. The operation is as follows:

$W = (100100110111100) \oplus (100110000000000)$

Hence, the result is:

$W = 000010110111100$

To verify the existence of an error, we divide W by the binary equivalent of $P(x)$, which is $P = 10011$. This step is identical to the procedure in the previous part, using binary strings for brevity and clarity.

Performing the binary division, we calculate:

```
             1010100
P = 10011 | 000010110111100
          | 10011
              10111
              10011
                 10011
                 10011
                 0000
```

The division yields a remainder of $R = 0000$, which translates to $R(x) = 0$ in polynomial terms. This indicates that the error introduced does not affect the remainder and thus remains undetected by this error-checking method.