**Technologie Sieciowe**

**Lista 3**

Lazarenko Arina

257259

**Cel listy:**

Zadanie 1

* Zaprezentować ramkowania zgodnie z zasadą „rozpychania bitów”, oraz obliczyć i wstawić pola kontrolnego CRC.
* Program powinien działać również w drugą stronę, tzn. przekształcać plik wynikowy do wersji oryginalnej.

Zadanie 2

* Zasymulować ethernetowej metody dostępu do medium transmisyjnego.

**Zadanie 1**

Obliczamy pola kontrolnego CRC w następujący sposób:

11010011101110 000 <--- 14 bitów danych + wyzerowane bity

1011 <--- 4-bitowy dzielnik CRC

01100011101110 000 <--- wynik operacji XOR

1011

00111011101110 000

1011

00010111101110 000

1011

00000001101110 000

1011

00000000110110 000

1011

00000000011010 000

1011

00000000001100 000

1011

00000000000111 000

101 1

00000000000010 100

10 11

-------------------

00000000000000 010 <--- CRC

**Implementacja**

* Tworzymy pole CRC, zależnego od długości dzielnika
* W zależności od długości dzielnika, dodajemy odpowiednią ilość zer
* Dalej wykonujemy operacje XOR po wszystkich bitach.
* Ramkowanie polega na wstawianiu między fragmenty komunikatu flagi "01111110" oraz pola kontrolnego CRC. Rozpychanie bitów polega na wstawieniu 0, w przypadku wcześniejszego wystąpienia pięciu jedynek
* Odkodowanie pliku odbywa się w sposób analogiczny.

**Przykład**

Ramkowaniu poddany zostaje następujący komunikat

1101001110111

W wyniku otrzymujemy:

011111101101001011111100011101011111101011000011111101011

Dzielnik to

1011

Długość ramki to 4.

Analiza wyniku:

**01111110** 1101001 **01111110** 0011101 **01111110** 1011000 **01111110** 1011

Zaznaczone fragmenty to flaga rozpoczęcia nowego komunikatu. Bez nich komunikat wygląda następująco:

1101 **001** 0011 **101** 1011 **000** 1 **011**

Zaznaczone trójelementowe ciągi znaków to sumy kontrolne CRC. Reszta to nadawany komunikat. Jest on zgodny z oryginałem.

Funkcja odkodowująca zadany ciąg znaków, zwraca oryginalny komunikat, czyli działa dokładnie odwrotnie.

Input:

011111101101001011111100011101011111101011000011111101011

Output:

1101001110111

Funkcja sprawdzająca poprawność zadanej wyżej ramki zwraca w wyniku komunikat:

Ramka jest prawidłowa

**Wnioski z zadania pierwszego**

Tworzenie symulacji pozwoliło na zapoznanie się z procesem formowania ramki, zapoznanie się z jej najważniejszymi elementami, zrozumienie techniki nadziewania bitami, oraz oswojenie się z algorytmem obliczającym sumy kontrolne

**Zadanie 2**

**Implementacja**

Tworzymy medium transmisyjne, które umożliwia nam rozpoczęcie wysyłania sygnału.

Tworzymy host, który będzie przesyłał informacje, oraz sprawdzał czy nie występują zaburzenia.

W głównym module programu utworzonych zostaje kilku hostów którzy próbują przesłać konkretną informacje.

**Przykładowe wywołanie**

[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ ][ ][ ][ ][ ][A][ ][ ][B][ ][ ][ ][ ][C][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ ][ ][ ][ ][A][A][A][B][B][B][ ][ ][C][C][C][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ ][ ][ ][A][A][A][#][#][B][B][B][C][C][C][C][C][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ ][ ][A][A][ ][#][#][ ][#][B][#][#][C][C][C][C][C][ ][ ][ ][ ][ ][ ][ ][ ]

[ ][A][A][ ][#][#][ ][ ][#][#][#][#][#][C][C][C][C][C][ ][ ][ ][ ][ ][ ][ ]

[A][A][ ][#][#][ ][ ][#][ ][#][#][ ][#][#][ ][C][C][C][C][ ][ ][ ][ ][ ][ ]

[A][ ][#][#][ ][ ][#][ ][#][#][ ][ ][ ][#][#][ ][C][C][C][C][ ][ ][ ][ ][ ]

[ ][#][#][ ][ ][#][ ][#][#][ ][ ][ ][ ][ ][#][#][ ][C][C][C][C][ ][ ][ ][ ]

[#][#][ ][ ][#][ ][#][#][ ][ ][ ][ ][ ][ ][ ][#][#][ ][C][C][C][C][ ][ ][ ]

[#][ ][ ][#][ ][#][#][ ][ ][ ][ ][ ][ ][ ][ ][ ][#][#][ ][C][C][C][C][ ][ ]

[ ][ ][#][ ][#][#][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][#][#][ ][C][C][C][C][ ]

[ ][#][ ][#][#][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][#][#][ ][C][C][C][C]

[#][ ][#][#][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][#][#][ ][C][C][C]

[ ][#][#][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][#][#][ ][C][C]

[#][#][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][#][#][ ][C]

[#][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][#][#][ ]

[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][#][#]

[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][#]

[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

A, B, C oznaczają wysłane sygnały, a # oznacza zakłócenia jakie napotkały one na swojej drodze

**Wnioski z zadania drugiego**

Dzięki wykonanej symulacji zaobserwowaliśmy proces rozchodzenia się wiadowości nadawanych przez różne urządzenia sieciowe przez jedno medium. Zbadaliśmy działanie algorytmu CSMA/CD i pokazaliśmy, że dzięki podwajaniu czasu oczekiwania pozwalia on przy niewielkiej ilości prób uniknąć dalszych kolizji.

**Wnioski**

* Obliczanie pola kontrolnego CRC pozwala stwierdzić, czy transmisja była poprawna.
* Metoda ta jest szeroko wykorzystywana do wykrywania błędów przypadkowych, powstających np. podczas transmisji danych cyfrowych przez łącza telekomunikacyjne.
* Ramkowanie pozwala na podział komunikatu na mniejsze fragmenty.
* W drugim zadaniu możemy zauważyć, że gdy dwa sygnały nachodzą na siebie występują wtedy zakłócenia. Gdyby nadawany został jeden komunikat, zakłócenia by nie istniały.