Sprawozdanie nr 3

Technologie sieciowe

Laboratorium piątek 9:15

Weronika Jasiak
236733

1. Cele

Celem wykonania pierwszego zadania jest zaprezentowanie ramkowania zgodnie z zasadą „rozpychania bitów”, oraz obliczanie i wstawianie pola kontrolnego CRC. Program powinien działać również w drugą stronę, tzn. przekształcać plik wynikowy do wersji oryginalnej. Celem wykonania drugiego zadania jest zasymulowanie ethernetowej metody dostępu do medium transmisyjnego.

2. Realizacja

2.1. Zadanie 1

Obliczanie pola kontrolnego CRC odbywa się w następujący sposób:

```
11010011101110 000 <--- 14 bitów danych + 3 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
```

Listing 1. (źródło: Wikipedia)

Realizacja w kodzie:

Tworzenie pola CRC, zależnego od długości dzielnika:

```
protected CRC(String divisor) {

      this.divisor = divisor;

}
```

Listing 2.

W zależności od długości dzielnika, dodajemy odpowiednią ilość zer (analogicznie jak w listingu 1.)

```
int crcSize = divisor.length() - 1;

for (int i = 0; i < crcSize; i++) {
      frame.append("0");
}
```

Listing 3.

Dalej wykonujemy operacje XOR po wszystkich bitach. Metoda XOR wygląda następująco:

```
protected char XOR(char a, char b) {
      if(a == b) {
            return 0 ;
      } else {
            return 1;
      }
}
```

Listing 4.

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. Realizacja w kodzie:

```
for (String frame : frames) {
      String framePart = "";
      String crc = crcType.encode(frame);
      framePart += frame;
      framePart += crc;
      framePart = framePart.replace("11111", "111110");
      toReturn.append(flag).append(framePart);
}
```

Listing 5.

Odkodowanie pliku odbywa się w sposób analogiczny.

2.2. Zadanie 2

Tworzymy medium transmisyjne, które umożliwia nam rozpoczęcie wysyłania sygnału.

Realizacja w kodzie:

```
public void startSignal(int id,String message){

      if(signals[id]==null) {

            signals[id] = new Signal(message,"start");

      }

}
```

Listing 6.

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

Realizacja w kodzie:

```
if(!collision){
      for(int i=0; i<Cable.signals.length; i++){
            if(Cable.signals[id]==null){
                  c.startSignal(id, message);
                  try {
                        Thread.sleep(300);
                  } catch (InterruptedException e) {
                        e.printStackTrace();
                  }
            }
            else if(!Cable.signals[id].getMessage().equals("@")){
                  try {
                        Thread.sleep(300);
                  } catch (InterruptedException e) {
                        e.printStackTrace();
                  }
                  collision=false;
                  wait=true;
                  break;
            }
            else{
                  success=false;
                  collision=true;
                  break;
            }
      }
}
```

Listing 7.

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

```
Host a = new Host("A",c,3);

Host b = new Host("B",c,9);

Host s = new Host("S",c,15);
```

Listing 8.

3. Przykładowe przebiegi

3.1 Zadanie 1

Ramkowaniu poddany zostaje następujący komunikat:

```
1101001110111
```

Listing 9.

W wyniku otrzymujemy:

```
01111110110100101111110001110101111110101100001111110101 1
```

Listing 10.

Dzielnik to:

```
1011
```

Listing 11.

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 w listingu 10 ciąg znaków, zwraca oryginalny komunikat z listingu 9, czyli działa dokładnie odwrotnie.

Wejście:

```
01111110110100101111110001110101111110101100001111110101 1
```

Listing 12.

Wyjście:

```
1101001110111
```

Listing 13.

Funkcja sprawdzająca poprawność zadanej wyżej ramki zwraca w wyniku komunikat:

```
Ramka jest prawidłowa
```

Listing 14.

3.2. Zadanie 2

Przykładowe wywołanie:

```
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][B][ ][ ][ ][ ][ ][S][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][B][B][B][ ][ ][ ][S][S][S][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][B][B][B][B][B][ ][S][S][S][S][S][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][B][B][B][B][B][B][@][S][S][S][S][S][S][ ]
[ ][ ][ ][ ][ ][ ][B][B][B][B][B][B][B][@][@][@][S][S][S][S][S][S]
[ ][ ][ ][ ][B][B][B][B][B][B][@][@][@][@][@][S][S][S][S][S]
[ ][ ][ ][B][B][B][B][B][ ][@][@][@][ ][@][@][@][ ][S][S][S]
[ ][ ][B][B][B][B][B][ ][@][@][@][ ][ ][ ][@][@][@][ ][S][S]
[ ][B][B][B][B][B][ ][@][@][@][ ][ ][ ][ ][ ][@][@][@][ ][S]
[B][B][B][B][B][ ][@][@][@][ ][ ][ ][ ][ ][ ][ ][@][@][@][ ]
[B][B][B][B][ ][@][@][@][ ][ ][ ][ ][ ][ ][ ][ ][ ][@][@][@]
[B][B][B][ ][@][@][@][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][@][@]
[B][B][ ][@][@][@][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][@]
[B][ ][@][@][@][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][@][@][@][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[@][@][@][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[@][@][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[@][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
```

Listing 15.

`B, S` oznaczają wysłane sygnały, a @ oznacza zakłócenia jakie napotkały one na swojej drodze.

4. 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.