

# **Laboratorium**

## **Technologii Sieciowych**

Temat:

Badanie topologii grafów sieciowych uwzględniając  
przypływ pakietów.

Autor: Bartosz Banasik  
Informatyka, semestr: 4  
Prowadzący: dr Łukasz Krzywiecki

# 1.Wstęp

## 1.1 Cele pracy

Celem pracy jest badanie topologii sieci pod wpływem zadanych macierzy natężeń. W pierwszej kolejności przyjrzymy się samej topologii. Następnie będziemy badać średnie opóźnienie pakietu. Na koniec zajmiemy się określeniem niezawodności sieci pod wpływem zadanych parametrów.

## 2. Propozycja topologii sieci

Określmy funkcję przepustowości 'c' jako maksymalna ilość bitów, którą można wprowadzić do kanału komunikacyjnego w ciągu sekundy.

Określmy funkcję przepływu 'a' jako faktyczną liczbę pakietów, które wprowadza się do węzła komunikacyjnego.

### 2.1 Propozycja

Mamy Graf  $G=\langle V,E \rangle$ , gdzie  $|V| = 10$ ,  $|E| < 20$ .

Dla zadanej macierzy natężeń N:

10									
x	1	3	2	2	1	2	0	1	0
2	x	2	1	1	2	3	0	0	0
1	2	x	4	2	2	1	0	0	0
3	1	1	x	3	2	2	0	0	0
2	3	1	4	x	3	1	0	0	0
3	2	1	2	2	x	4	0	0	0
1	2	3	4	2	2	x	0	0	0
0	0	0	0	0	0	0	x	0	0
0	0	0	0	0	0	0	0	x	0
0	0	0	0	0	0	0	0	0	x

Topologia grafu wygląda następująco:

$G = ([1, 2, 3, 4, 5, 6, 7, 8, 9, 10],$   
 $[\{1,2\}, \{2,3\}, \{3,4\}, \{4,5\}, \{5,6\}, \{6,7\}, \{7,8\}, \{8,9\}, \{9,10\}, \{1,10\}, \{1,3\},$   
 $\{3,5\}, \{5,7\}, \{2,4\}, \{4,6\}, \{6,8\}, \{1,8\}])$

Dla poszczególnych kanałów komunikacyjnych wartości prezentują się następująco:

1 - 2 Przeptyw: 12000 c(v) = 170000 a(v) = 8  
 2 - 3 Przeptyw: 12000 c(v) = 170000 a(v) = 8  
 3 - 4 Przeptyw: 12000 c(v) = 170000 a(v) = 8  
 4 - 5 Przeptyw: 19500 c(v) = 170000 a(v) = 13  
 5 - 6 Przeptyw: 7500 c(v) = 170000 a(v) = 5  
 6 - 7 Przeptyw: 16500 c(v) = 170000 a(v) = 11  
 7 - 8 Przeptyw: 4500 c(v) = 170000 a(v) = 3  
 8 - 9 Przeptyw: 0 c(v) = 170000 a(v) = 0  
 9 - 10 Przeptyw: 1500 c(v) = 170000 a(v) = 1  
 1 - 10 Przeptyw: 1500 c(v) = 190000 a(v) = 1  
 1 - 3 Przeptyw: 12000 c(v) = 117000 a(v) = 8  
 3 - 5 Przeptyw: 22500 c(v) = 117000 a(v) = 15  
 5 - 7 Przeptyw: 19500 c(v) = 117000 a(v) = 13  
 2 - 4 Przeptyw: 24000 c(v) = 170000 a(v) = 16  
 4 - 6 Przeptyw: 24000 c(v) = 170000 a(v) = 16  
 6 - 8 Przeptyw: 6000 c(v) = 170000 a(v) = 4  
 1 - 8 Przeptyw: 10500 c(v) = 170000 a(v) = 7

## 2.2 Testowanie propozycji<sup>1</sup>

Tutaj skupimy się na średnim opóźnieniu pakietu danym wzorem

$$T = \frac{1}{G} * \sum_e \left( \frac{a(e)}{\frac{c(e)}{m} - a(e)} \right)$$

Gdzie: 'G' - suma wszystkich elementów macierzy natężeń, 'm' - średnia wielkość pakietu w bitach.

Dla macierzy natężeń N:

Output: Średnie opóźnienie pakietu :  
 0.017288494854991232

```

10
x 1 3 2 2 1 2 0 1 0
2 x 2 1 1 2 3 0 0 0
1 2 x 4 2 2 1 0 0 0
3 1 1 x 3 2 2 0 0 0
2 3 1 4 x 3 1 0 0 0
3 2 1 2 2 x 4 0 0 0
1 2 3 4 2 2 x 0 0 0
0 0 0 0 0 0 0 x 0 0
0 0 0 0 0 0 0 0 x 0
0 0 0 0 0 0 0 0 0 x
    
```

<sup>1</sup> Przykładowa implementacja na listingu nr 1.

Dla macierzy natężeń N:

Output:      Średnie      opóźnienie      pakietu      :  
0.021216267048914612

10
x 1 3 2 2 1 2 0 1 0
2 x 2 1 1 2 3 1 3 2
1 2 x 4 2 2 1 2 3 4
3 1 1 x 3 2 2 1 2 3
2 3 1 4 x 3 1 0 0 0
3 2 1 2 2 x 4 1 2 0
1 2 3 4 2 2 x 0 2 0
0 2 0 0 1 0 0 x 0 0
0 4 3 2 0 3 2 0 x 0
2 0 0 0 1 0 0 1 0 x

Wnioski: Dla macierzy natężeń o większej sumie przysyłanych pakietów w sieci opóźnienie jest większe. W naszym przypadku 2 razy większe.

## 2.3 Zastosowanie metody Monte Carlo do badania niezawodności sieci.<sup>2</sup>

Niech miarą niezawodności sieci będzie prawdopodobieństwo tego, że w dowolnym przedziale czasowym, nierozspójniona sieć zachowuje  $T < T_{\max}$ . Prawdopodobieństwo nieuszkodzenia każdej krawędzi podawane jest jako input. Tak samo jak maksymalna wartość opóźnienia.

Macierz N powinna się znajdować w pliku o nazwie 'Dane.txt' zapisanym w katalogu projektu programu.

Macierz N :

10
x 1 3 2 2 1 2 0 1 0
2 x 2 1 1 2 3 1 3 2
1 2 x 4 2 2 1 2 3 4
3 1 1 x 3 2 2 1 2 3
2 3 1 4 x 3 1 0 0 0
3 2 1 2 2 x 4 1 2 0
1 2 3 4 2 2 x 0 2 0
0 2 0 0 1 0 0 x 0 0
0 4 3 2 0 3 2 0 x 0
2 0 0 0 1 0 0 1 0 x

Input: 0.9 0.017

Output: Badanie niezawodności sieci ...

Przeprowadzonych testów: 1000

Ilość sukcesów: 87

Ilość rozspojnień: 14

Ilość przekroczonych opoźnień: 899

Testowana niezawodność sieci wynosi: 8.7 %

---

<sup>2</sup> Przykładowa implementacja na listingu nr 2.

Dla zmienionych danych:

Macierz N:

10
x 1 3 2 2 1 2 0 0 0
2 x 2 1 1 2 3 0 0 0
1 2 x 4 2 2 1 0 0 0
3 1 1 x 3 2 2 0 0 0
2 3 1 4 x 3 1 0 0 0
3 2 1 2 2 x 4 0 0 0
1 2 3 4 2 2 x 0 0 0
0 0 0 0 0 0 0 x 0 0
0 0 0 0 0 0 0 0 x 0
0 0 0 0 0 0 0 0 0 x

Input: 0.9 0.017

Output: Badanie niezawodności sieci ...

Przeprowadzonych testow: 1000

Ilość sukcesow: 748

Ilość rozspojnien: 13

Ilość przekroczonych opoznien: 239

Testowana niezawodność sieci wynosi: 74.8 %

Wnioski: Wraz ze wzrostem ilości pakietów przesyłanych po naszej sieci rośnie średnie opóźnienie pakietu, a co za tym idzie ilość przekroczonych opóźnień rośnie.

## 2.4 Wybrane fragmenty kodu.

Poniżej zamieszczę listingi z wybranymi fragmentami kodu, wraz z krótkim opisem. Wszystkie kawałki kodu zostały zaprojektowane w języku Java. Na listingu pierwszym znajduje się funkcja licząca średnie opóźnienie pakietu dla danego grafu. 'MEdge' jest krawędzią, 'edges' to zbiór wszystkich krawędzi w grafie. Funkcja getCapacity() zwraca wartość funkcji c(e) dla danej krawędzi, zaś funkcja getNumberOfPackets() zwraca wartość a(e) dla danej krawędzi.

```
double avgPacketDeley(){
    int G = 0;
    double sum = 0;
    for(MEdge edge : edges) {
        sum += edge.getNumberOfPackets() * 1.0 / (edge.getCapacity() * 1.0 /
SIZEOFPACKET * 1.0 - edge.getNumberOfPackets());
    }
    for(int i : packets) {
        G += i;
    }
    return (1.0/G*1.0) * sum;
}
```

*Listing 1*

Na listingu drugim zawarty jest fragment kodu odpowiedzialny za testowanie naszej sieci w celu określenia niezawodności. Badamy ją pod kątem nierozspójnienia oraz zachowania maksymalnego opóźnienia pakietu. Jako parametr funkcja przyjmuje prawdopodobieństwo nieuszkodzenia krawędzi oraz maksymalne opóźnienie pakietu.

Funkcja `testGenTraffic()` jako argument przyjmuje graf i na nim realizuje macierz natężeń `N`.

```
private static void test(double p, double t_max) {
    int succes = 0;
    int failures = 0;
    int overload = 0;
    for(int i = 0; i < 1000; i++) {
        MGraph test = genTestGraph();
        testGenTraffic(test);
        test.test(p);
        if(test.isConsistent()) {
            if(test.avgPacketDeley() < t_max) {
                succes++;
            } else overload++;
        } else failures++;
    }
    System.out.println("Ilosc sukcesow: "+succes);
    System.out.println("Ilosc rozspojnien: "+failures);
    System.out.println("Ilosc przekroczonych opoznien: "+overload);
    double reliability = (1000.0 - (failures + overload)*1.0)/(1000.0) * 100.0;
    System.out.println("Testowana niezawodnosc sieci wynosi: "+ reliability+" %");
}
```

*Listing 2*