Sprawozdanie 2

Józef Piechaczek

2019-03-26

1 Zadanie 1

1.1 Wstęp

Celem zadania 1. jest rozpatrzenie niezawodności sieci o modelu $S =< G, H >$ w czasie podzielonym na interwały. Podstawowy model składa się z grafu $G =< V, E >$ zwierającego 20 wierzchołków oznaczonych przez $v(i)$, dla $i = 1, ..20$, a zbiór E zawiera 19 krawędzi $e(j, j + 1)$, dla $j = 1, ..., 19$. Zbiór H zawiera funkcję niezawodności h przyporządkowującą każdej krawędzi $e(j, k)$ ze zbioru E wartość 0.95 oznaczającą prawdopodobieństwo nieuszkodzenia tego kanału komunikacyjnego w dowolnym interwale.

1.2 Symulacja

W celu oszacowania niezawodności sieci posłużę się programem, korzystającym z metody Monte Carlo, napisanym w języku Java, z pomocą biblioteki JGraphT. W celu wizualizacji grafów posłużę się narzędziem Graphviz. Symulacja będzie polegała na wykonaniu odpowiedniej liczby prób na danym grafie i porównaniu liczby prób udanych do liczby wszystkich. Pojedyncza próba będzie składać się z:

1. Utworzenia grafu
2. Testowania połączeń
3. Sprawdzenia spójności

Testowanie połączeń będzie polegać na wylosowaniu pesudolosowej liczby z zakresu $[0; 1]$ i porównaniu tej liczby z podaną niezawodnością - jeśli liczba przekroczy niezawodność oznacza to, że dane połączenie zostanie przerwane.

Po przetestowaniu połączeń, sprawdzam czy istnieje połączenie pomiędzy dowolnymi wierzchołkami - jeśli nie, oznacza to że graf jest niespójny, a sieć oparta na takim grafie zawodna.

Niezawodność sieci obliczam ze wzoru $P = \frac{A}{\Omega}$, gdzie A oznacza liczbę prób zakończonych powodzeniem (sieć spójna), a Ω liczbę wszystkich prób.

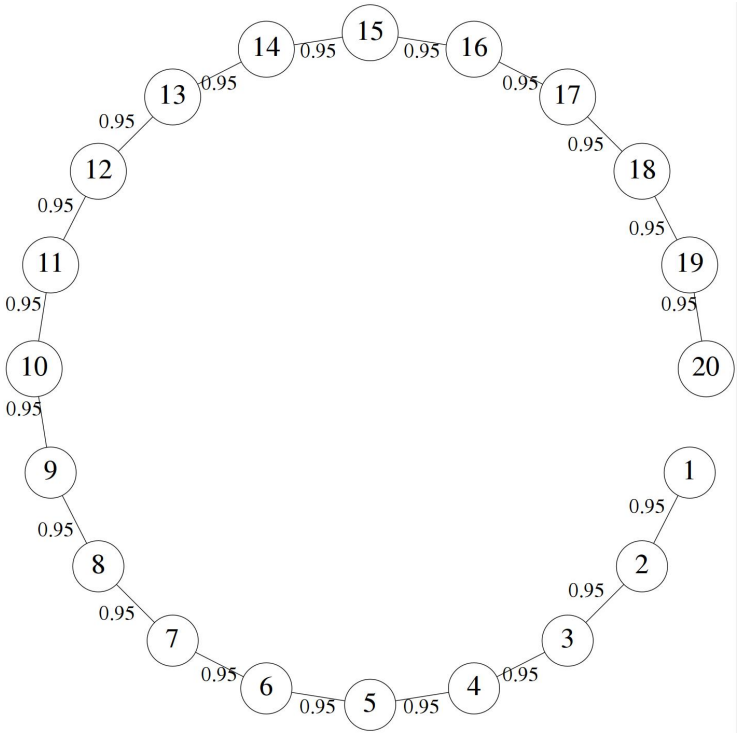
1.3 Wyniki

1.3.1 Test nr 1

W teście nr 1 korzystamy z modelu podstawowego, podanego we wstępie. Po wykonaniu symulacji uzyskujemy następujące wyniki:

	Liczba prób		
	1,000000	10,000,000	100,000,000
Niezawodność	0.377773	0.3775255	0.37736323

Model sieci:

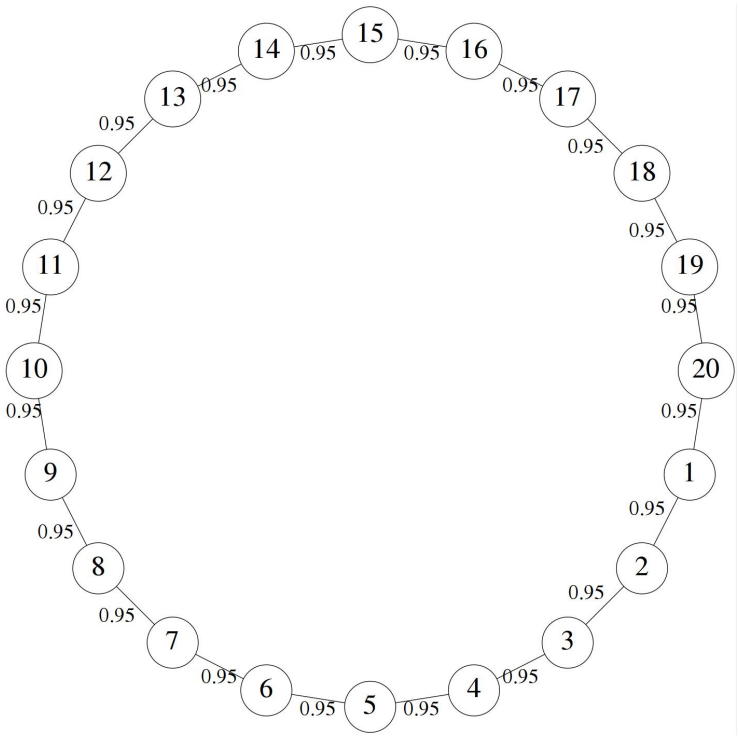


1.3.2 Test nr 2

W teście nr 2 korzystamy z modelu podstawowego z dodatkowym połączeniem pomiędzy wierzchołkami 1 i 20 o niezawodności wynoszteochęącej 0.95. Po wykonaniu symulacji uzyskujemy następujące wyniki:

	Liczba prób		
	1,000,000	10,000,000	100,000,000
Niezawodność	0.735969	0.735715	0.73583365

Model sieci:



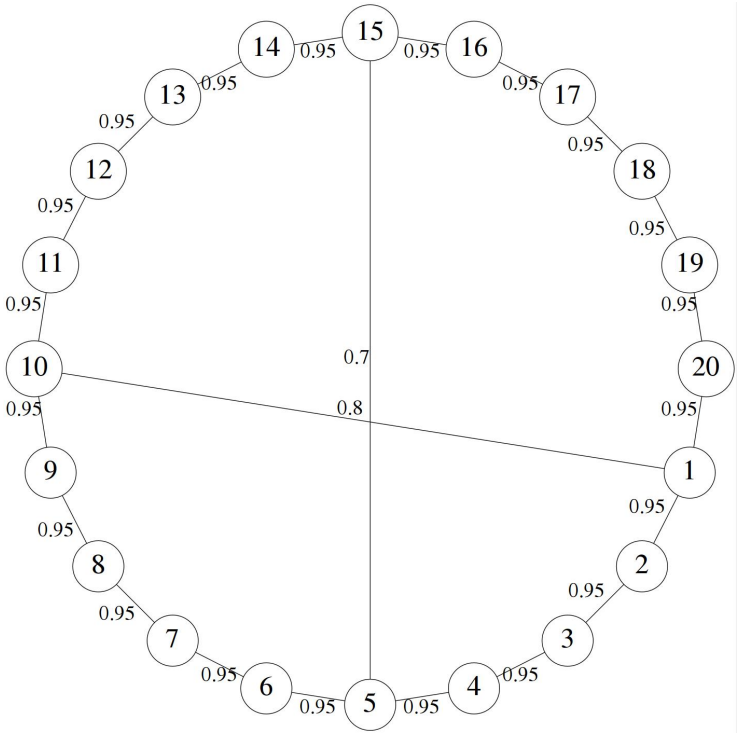
1.3.3 Test nr 3

W teście nr 3 korzystamy z modelu z poprzedniego test z dodatkowymi połączeniami pomiędzy wierzchołkami 1 i 10 oraz 5 i 15 o niezawodnościach wynoszących odpowiednio 0.8 i 0.7. Po wykonaniu symulacji uzyskujemy następujące wyniki:

Liczba prób

	1,000,000	10,000,000	100,000,000
Niezawodność	0.870627	0.8706126	0.87050279

Model sieci:

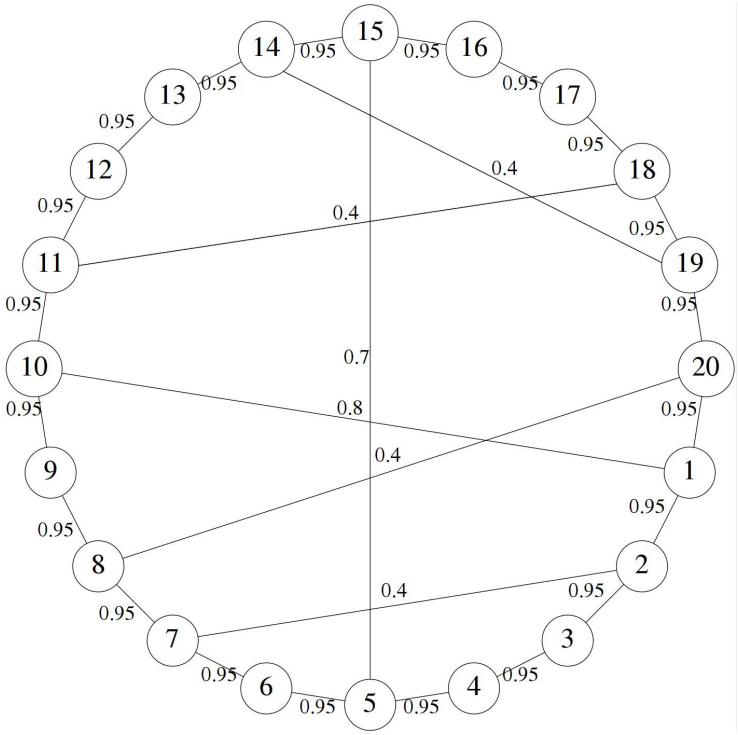


1.3.4 Test nr 4

W teście nr 4 korzystamy z modelu z poprzedniego testu z dodatkowymi czterema połączeniami, o niezawodności wynoszącej 0.4, pomiędzy losowymi wierzchołkami. Po wykonaniu symulacji uzyskujemy następujące wyniki:

	Liczba prób		
	1,000,000	10,000,000	100,000,000
Niezawodność	0.902696	0.9030207	0.90292538

Przykładowy model sieci:



1.4 Omówienie wyników

Dla bardziej skomplikowanych sieci problemem jest analityczne policzenie niezawodności. Jednak dla sieci z testu nr 1 można obliczyć ją w prosty sposób. Sieć pozostaje spójna jeśli żaden z kanałów komunikacyjnych nie jest przerwany, czego prawdopodobieństwo wynosi $P = 0,95^{19} \approx 0.37735360$.

Obliczone prawdopodobieństwo jest bardzo zbliżone do uzyskanego wyniku, a na dokładność wpływa między innymi jakość użytego generatora pseudolosowego.

Jak można zauważyć, w teście nr 2, po dodaniu kanału pomiędzy wierzchołkami 1 i 20 prawdopodobieństwo znacząco wzrasta - wynika to z faktu, iż w modelu drugim po zniszczeniu jednego kanału komunikacyjnego sieć wciąż zachowuje spójność.

Podobnie w teście nr 3 i nr 4 dodawanie krawędzi zwiększa niezawodność sieci. W teście nr 4 na niezawodność sieci wpływa również to, które wierzchołki łączą wylosowane krawędzie - im większa odległość pomiędzy wylosowanymi wierzchołkami, tym większa niezawodność sieci.

2 Zadanie nr 2

2.1 Obliczanie średniego opóźnienia

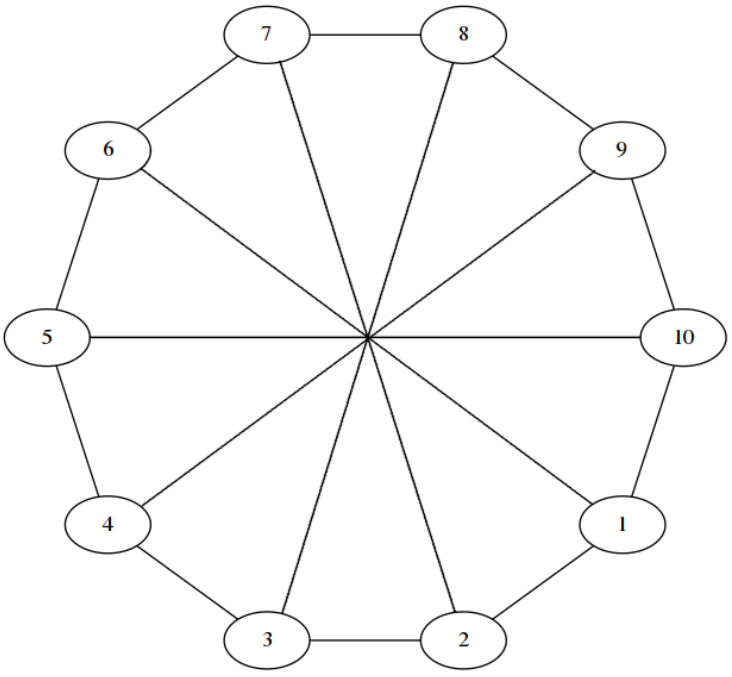
W celu testowania sieci posłużyłem się programem napisanym w Javie przy użyciu biblioteki JGraphT. Program na wejście przyjmuje niezbędne dane takie jak: połączenia pomiędzy wierzchołkami, wielkość pakietu, oraz macierz natężeń strumienia pakietów. Program analizuje spójność sieci, oblicza funkcję przepływu oraz średnie opóźnienie pakietu ze wzoru:

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

W celu obliczenia funkcji przepływu, program szuka najkrótszej ścieżki pomiędzy wierzchołkami za pomocą algorytmu Dijkstry, a następnie liczy ilość pakietów przechodzących przez dany wierzchołek.

2.2 Przykładowy model sieci

W zadaniu posłużyłem się modelem sieci o topologii siatki zgodnie z poniższym schematem:



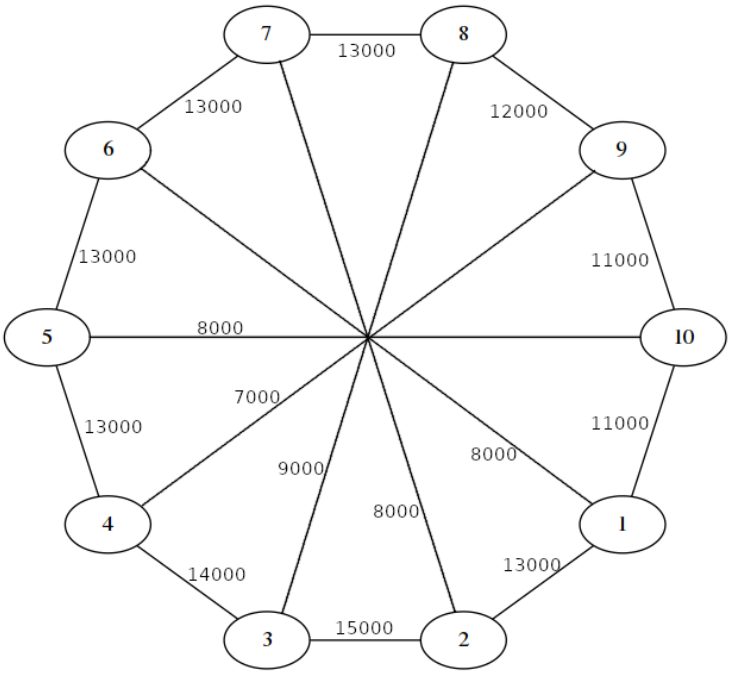
Wartość przesyłanego pakietu ustawiłem na $10B$. Dla podanego modelu sieci początkowo ustawiłem poniższą macierz natężeń (liczby pakietów):

$$N = [n(i,j)], \forall i,j; i \neq j : n(i,j) = 1000$$

Przepustowość na kanałach komunikacyjnych wynosi:

$$\forall_{e \in E} : c(e) = 200000 \frac{B}{s}$$

W celu policzenia funkcji przepływu, czyli faktycznej liczby pakietów wprowadzanej do poszczególnych kanałów komunikacyjnych posłużyłem się wcześniej wspomnianym programem. Otrzymałem następujący wynik:

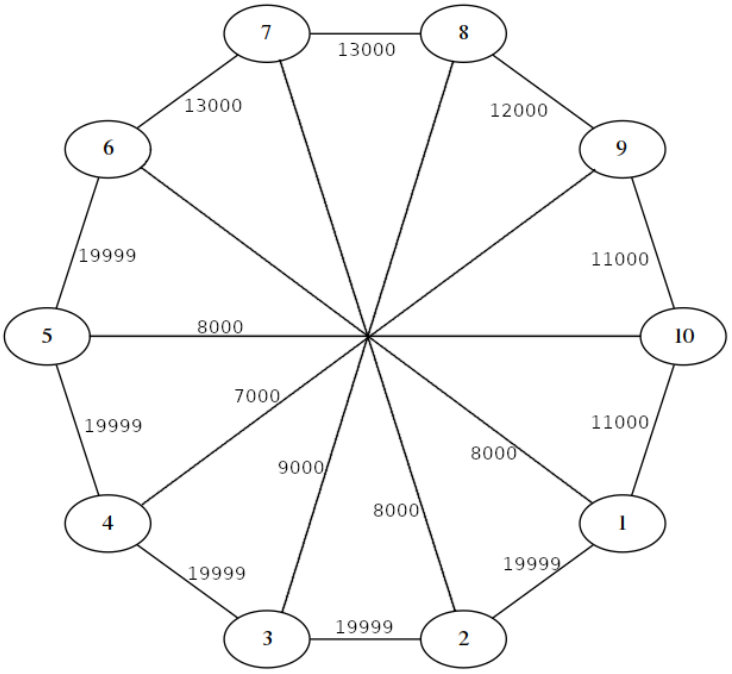


Jak można zauważyć, ilość przesyłanych pakietów na żadnym kanale nie przekroczyła przepustowości (200000B = 20000 pakietów).

Średnie opóźnienie pakietu w podanym modelu sieci wyniosło:

$$T = 0.000023s$$

W przypadku próby modyfikowania sieci tak, aby łączny przepływ na wierzchołkach zbliżał się do przepustowości, opóźnienie sieci znacząco wzrasta. Przykładowo, przy następujących wartościach łącznego przepływu:



średnie opóźnienie pakietu znacząco wzrasta do wartości $T = 0.833s$

2.3 Testowanie sieci

W celu testowania sieci posłużyłem się poprzednim programem rozbudowanym dodatkowo o testowanie nieuszkodzenia krawędzi. Program przyjmuje dodatkowo parametry p (prawdopodobieństwo nieuszkodzenia krawędzi) oraz tmax (maksymalne średnie opóźnienie pakietu). Po utworzeniu sieci program testuje spójność sieci, za pomocą sposobu opisanego w pierwszym zadaniu. Jeśli sieć pozostaje spójna, program testuje wysyłanie pakietów, oblicza całkowity przepływ i średnie opóźnienie. Pojedynczy test kończy się sukcesem, jeśli sieć pozostaje spójna, a średnie opóznienie jest mniejsze niż $tmax$.

2.4 Przykładowy test

W calu testowania przykładowej sieci posłużę się modelem sieci z zwiększoną liczbą pakietów z poprzedniego zadania. Wartość średniego opóźnienia dla podanego modelu wyniosła $T = 0,769s$. Doświadczenia wykonuję dla 100000 testów. Wyniki doświadczenia umieszczam w poniższej tabeli:

P	1	0.95	0.50
TMax	0.77	0.77	0.77
Niezawodność	1	0.1439	0.0002

W powyższym modelu łączny przepływ był bardzo zblizony do przpustowości sieci. Oznacza to, że w przypadku przerwania chociaż jednego węzła komunikacyjnego istnieje bardzo duże prawdopodbieństwo, że sieć nie będzie potrafiła obsłużyć użytkowników, co można zauważyć w powyższej tabeli - wraz z zmniejszaniem prawdopobdobieństwa nieuszkodzenia kanłów komunikacyjnych niezawodność sieci znacząco się zmniejsza.