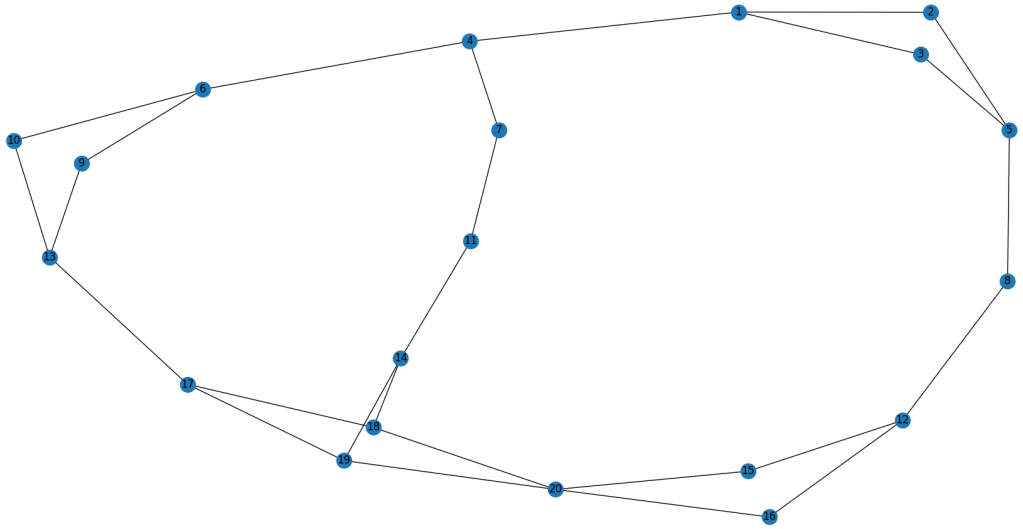
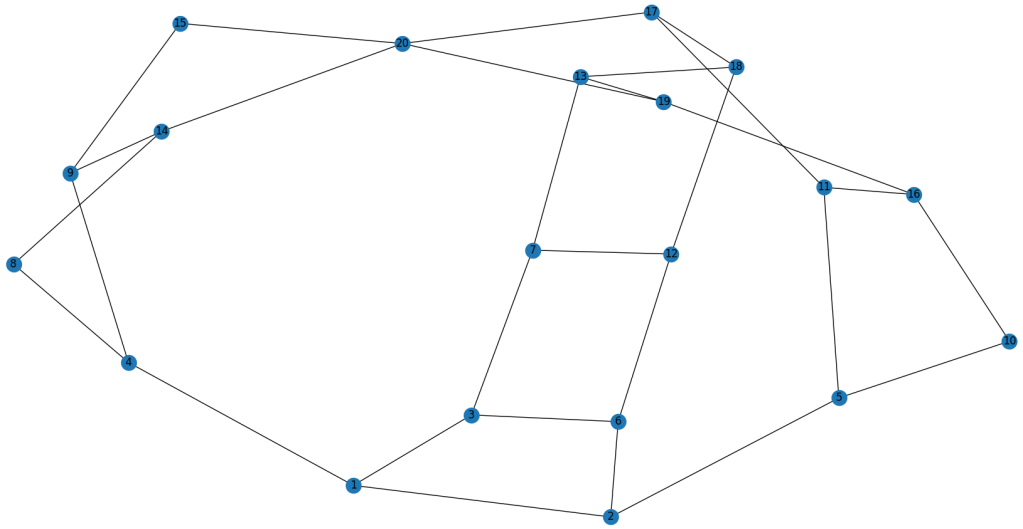
Użyte Topologie





Podpunkt 1

generowanie macierzy natężeń

```
def gen_matrix(self):
    #Macierz natężeń
    matrix = [[random.randint(1,9) if x!=y else 0 for x in range(20)] for y in range(20)]
    return matrix
```

Program losuje liczbę wysyłanych pakietów od 1 do 9, stosując zasadę że wierzchołek nie może wysyłać pakietów sam do siebie

funkcja przepustowości

maksymalna liczba bitów jest ustawiona standardowo na 5000

funkcja przepływu

```
for i,row in enumerate(self.matrix):
    for idx,packets in enumerate(row):
        try:
            while(packets>0):
                path = self.shortest_path(i+1,idx+1,1)
                path_edges = [self.find_edge((path[x-1],path[x])) for x in range(1,len(path))]
                packets_sent =  math.ceil(min([(edge.c/self.m)-edge.a for edge in path_edges])-1)
                if(packets_sent>packets):
                    packets_sent = packets
                for edge in path_edges:
                    edge.a+=packets_sent
                packets-=packets_sent
        except nx.NetworkXNoPath as e:
            pass
```

program przechodzi po wszystkich wartościach w macierzy natężeń i próbuje wysyłać po jednym pakiecie przez najkrótszą ścieżkę.Jeżeli znajdzie taką ścieżkę to sprawdza ile pakietów maksymalnie może wysłać, a następnie zwiększa przepływ krawędzi, które są w najkrótszej ścieżce

```
def shortest_path(self,v_sc,v_dst,packets):
    new_graph = nx.Graph()
    #krawedzie dla ktorych ( packets + a(e) )*m < c(e)
    new_edges = [edge.v for edge in self.edges if(edge.enabled and (packets+edge.a)*self.m<edge.c)]
    new_graph.add_nodes_from(self.nodes)
    new_graph.add_edges_from(new_edges)
    return nx.shortest_path(new_graph,v_sc,v_dst)
```

funkcja szuka najkrótszej ścieżki, z dostępnych krawędzi dla których liczba (pakietów + przepływ)*wielkość pakietu jest mniejsza od przepustowości

Przykładowe wyniki dla pierwszej topologii (średnia wielkość pakietu = 10b)

```
=== MACIERZ NATĘŻEŃ ===
[0, 6, 4, 2, 4, 9, 9, 2, 9, 3, 7, 9, 6, 6, 8, 4, 5, 9, 3, 8]
[9, 0, 2, 3, 6, 6, 2, 1, 3, 1, 8, 5, 2, 9, 2, 2, 9, 2, 5, 5]
[7, 7, 0, 6, 4, 8, 4, 7, 9, 2, 8, 3, 2, 2, 2, 9, 9, 6, 5, 8]
[3, 7, 6, 0, 2, 7, 4, 3, 4, 8, 8, 9, 8, 7, 2, 2, 8, 8, 7, 2]
[1, 2, 2, 4, 0, 6, 4, 9, 1, 9, 8, 7, 9, 8, 4, 6, 1, 8, 2, 1]
[9, 3, 2, 9, 7, 0, 2, 2, 3, 1, 7, 4, 8, 4, 5, 2, 6, 3, 3, 7]
[1, 8, 8, 2, 7, 7, 0, 9, 2, 1, 1, 8, 2, 9, 6, 6, 1, 1, 8, 3]
[8, 8, 9, 6, 3, 9, 7, 0, 8, 6, 8, 9, 7, 3, 4, 7, 5, 4, 6, 9]
[6, 1, 9, 1, 6, 2, 1, 1, 0, 2, 7, 6, 3, 6, 2, 6, 6, 1, 1, 5]
[2, 7, 5, 9, 1, 2, 5, 8, 4, 0, 5, 8, 4, 8, 9, 3, 1, 8, 7, 8]
[6, 1, 2, 1, 4, 5, 8, 4, 2, 6, 0, 2, 9, 2, 5, 6, 6, 7, 6, 8]
[8, 5, 4, 5, 7, 4, 4, 2, 1, 9, 7, 0, 2, 3, 1, 3, 4, 1, 8, 8]
[3, 1, 2, 4, 1, 3, 5, 8, 1, 7, 5, 7, 0, 4, 8, 9, 3, 3, 6, 8]
[8, 4, 5, 9, 7, 1, 9, 1, 9, 9, 8, 4, 8, 0, 4, 2, 7, 5, 7, 3]
[6, 4, 5, 3, 7, 2, 2, 6, 4, 3, 1, 4, 5, 9, 0, 4, 5, 9, 2, 3]
[3, 1, 5, 7, 4, 1, 5, 9, 7, 4, 8, 1, 5, 5, 1, 0, 8, 8, 3, 1]
[9, 9, 7, 7, 1, 2, 1, 7, 8, 2, 6, 9, 8, 2, 8, 8, 0, 6, 9, 8]
[5, 9, 6, 9, 7, 2, 4, 5, 8, 8, 4, 1, 7, 3, 4, 3, 4, 0, 5, 5]
[2, 2, 6, 8, 2, 7, 8, 1, 2, 1, 2, 6, 4, 8, 9, 7, 1, 7, 0, 4]
[8, 4, 5, 2, 6, 5, 2, 5, 8, 2, 3, 5, 3, 5, 8, 6, 6, 5, 6, 0]
=== Funkcja Przepływu ===
(1, 2):386
(1, 3):120
(1, 4):499
(2, 5):345
(3, 5):82
(4, 6):350
(4, 7):335
(5, 8):399
(6, 9):228
(6, 10):87
(7, 11):307
(8, 12):379
(9, 13):231
(10, 13):101
(11, 14):296
(12, 15):328
(12, 16):68
(13, 17):380
(14, 18):258
(14, 19):79
(15, 20):378
(16, 20):113
(17, 18):387
(17, 19):41
(18, 20):443
(19, 20):66
```

można tu zauważyć, że krawędź (1,4) osiągnęła swój limit przepustowości

Przykładowe wyniki dla drugiej topologii (średnia wielkość pakietu = 10b)

```
=== MACIERZ NATĘŻEŃ ===
[0, 9, 3, 9, 5, 4, 3, 2, 6, 2, 3, 9, 5, 7, 9, 6, 5, 4, 7, 9]
[8, 0, 4, 5, 6, 9, 4, 2, 9, 9, 4, 1, 8, 6, 8, 8, 6, 7, 3, 1]
[7, 2, 0, 3, 1, 3, 8, 6, 6, 1, 9, 7, 7, 5, 1, 7, 1, 2, 9, 8]
[4, 3, 6, 0, 8, 3, 5, 7, 4, 8, 1, 8, 4, 4, 8, 6, 3, 9, 2, 7]
[5, 2, 4, 5, 0, 5, 8, 7, 5, 5, 7, 7, 2, 6, 6, 9, 1, 9, 9, 3]
[1, 1, 2, 7, 6, 0, 8, 3, 7, 8, 1, 5, 2, 7, 6, 9, 2, 4, 2, 6]
[7, 8, 3, 2, 8, 5, 0, 2, 7, 8, 5, 8, 9, 6, 6, 1, 7, 9, 1, 2]
[8, 4, 4, 3, 1, 1, 7, 0, 9, 4, 2, 9, 6, 3, 7, 8, 8, 8, 1, 4]
[2, 8, 7, 2, 6, 1, 4, 4, 0, 7, 8, 7, 4, 6, 6, 8, 2, 4, 6, 1]
[6, 1, 5, 9, 4, 4, 4, 6, 6, 0, 5, 4, 8, 4, 7, 2, 6, 6, 6, 4]
[3, 1, 8, 9, 1, 8, 5, 1, 7, 4, 0, 2, 9, 9, 9, 6, 6, 7, 2, 5]
[2, 8, 3, 7, 8, 2, 8, 9, 4, 1, 7, 0, 4, 4, 8, 7, 1, 3, 2, 5]
[1, 1, 2, 8, 2, 4, 8, 1, 1, 5, 5, 9, 0, 7, 7, 2, 6, 9, 6, 1]
[6, 7, 5, 8, 1, 9, 2, 1, 7, 5, 6, 3, 5, 0, 8, 3, 5, 8, 5, 1]
[7, 5, 3, 9, 3, 6, 5, 1, 8, 8, 6, 5, 6, 3, 0, 8, 2, 4, 6, 6]
[6, 8, 4, 1, 3, 5, 4, 1, 5, 8, 6, 9, 1, 5, 9, 0, 6, 2, 3, 4]
[8, 1, 2, 9, 5, 2, 1, 4, 2, 9, 9, 4, 6, 1, 8, 5, 0, 9, 4, 8]
[9, 4, 1, 2, 6, 9, 1, 6, 3, 8, 7, 2, 4, 5, 6, 7, 3, 0, 5, 4]
[9, 3, 6, 4, 7, 8, 5, 2, 4, 7, 7, 8, 8, 3, 2, 3, 2, 9, 0, 4]
[5, 1, 5, 6, 7, 2, 4, 9, 4, 6, 4, 1, 8, 9, 5, 6, 7, 1, 6, 0]
=== Funkcja Przepływu ===
(1, 2):417
(1, 3):198
(1, 4):410
(2, 5):302
(2, 6):230
(3, 6):50
(3, 7):196
(4, 8):215
(4, 9):181
(5, 10):205
(5, 11):156
(6, 12):167
(7, 12):71
(7, 13):209
(8, 14):174
(9, 14):83
(9, 15):95
(10, 16):175
(11, 16):55
(11, 17):217
(12, 18):141
(13, 18):61
(13, 19):253
(14, 20):268
(15, 20):132
(16, 19):225
(17, 18):192
(17, 20):233
(19, 20):226
```

dla drugiej topologii nie udało mi się zapełnić przepustowości, wynika z tego że jest bardziej niezawodna w kwestii przepustowości

Podpunkt 2

Mierząc niezawodność zostały użyte następujące parametry:
prawdopodobieństwo = 0.97
średnia wielkość pakietu = 10b
t_max = 1/100
w macierzy można było wpisać od 1-10 liczby pakietów
przepustowość dla każdej krawędzi wynosiła 10_000b

```
def reliability(self,t_max,rep):
    self.failures = 0
    success = 0
    delay = 0
    for i in range(rep):
        failures = self.failures
        self.gen_a()
        if(self.T<t_max and failures == self.failures):
            success += 1
        if(self.T>=t_max):
            delay += 1
    return [round(success/rep*100,2),round(self.failures/rep*100,2),round(delay/rep*100,2)]
```

funkcja mierząca niezawodność zlicza ile razy wystąpił sukces,ile razy opóźnienie przekroczyło wartość maksymalną, ile razy wystąpiła rozspójenie.

```
def update_graph(self):
    for edge in self.edges:
        edge.a = 0
        pr = random.random()
        edge.enabled = pr < self.pr

    self.graph.clear_edges()
    self.graph.add_edges_from([edge.v for edge in self.edges if edge.enabled])

def gen_a(self):
    self.update_graph()
    if(nx.is_connected(self.graph)==False):
        self.failures += 1
```

do starej funkcji generującej funkcje przepływo została dodana funkcja która aktualizuje graf zgodnie z prawdopodobieństwem niezawodności, a następnie zlicza rozspójnienie, jeśli wystąpiło. Na koniec tej funkcji jeszcze została dodana jedna linijka

```
self.T = (1 / self.G) * sum(edge.a / ((edge.c / self.m) - edge.a) for edge in self.edges)
```

```
C:\Users\Mateusz\Desktop\sieci>python xd.py --topologia 2 --podpunkt 2
Niezawodnosc sieci: 97.28%
Procent rozspojnien sieci: 1.36%
Procent przekroczenia opóźnienia: 1.36%

C:\Users\Mateusz\Desktop\sieci>python xd.py --topologia 2 --podpunkt 2
Niezawodnosc sieci: 97.8%
Procent rozspojnien sieci: 1.0%
Procent przekroczenia opóźnienia: 1.2%

C:\Users\Mateusz\Desktop\sieci>python xd.py --topologia 2 --podpunkt 2
Niezawodnosc sieci: 97.76%
Procent rozspojnien sieci: 1.12%
Procent przekroczenia opóźnienia: 1.24%

C:\Users\Mateusz\Desktop\sieci>python xd.py --topologia 1 --podpunkt 2
Niezawodnosc sieci: 83.28%
Procent rozspojnien sieci: 3.8%
Procent przekroczenia opóźnienia: 13.2%

C:\Users\Mateusz\Desktop\sieci>python xd.py --topologia 1 --podpunkt 2
Niezawodnosc sieci: 82.32%
Procent rozspojnien sieci: 3.4%
Procent przekroczenia opóźnienia: 14.68%

C:\Users\Mateusz\Desktop\sieci>python xd.py --topologia 1 --podpunkt 2
Niezawodnosc sieci: 80.88%
Procent rozspojnien sieci: 3.88%
Procent przekroczenia opóźnienia: 15.72%
```

dla każdej topologii wykonałem po 3 powtórzenia, oczywistym wnioskiem jest że topologia numer 2 jest bardziej niezawodna

Podpunkt 3

Mierząc niezawodność zostały użyte następujące parametry:
prawdopodobieństwo = 0.97
średnia wielkość pakietu = 10b
t_max = 1/100
w macierzy można było wpisać od 1-[5,10,15,20] liczby pakietów
przepustowość dla każdej krawędzi wynosiła 10_000b

```python
for i in range(5,21,5):
    print(f"max liczba pakietów w macierzy:{i}")
    n.max_packets = i
    n.matrix = n.gen_matrix(len(n.nodes))
    rel = n.reliability(t_max=t_max,rep=2500)
    print(f"Niezawodnosc sieci: {rel[0]}%")
    print(f"Procent rozspojnien sieci: {rel[1]}%")
    print(f"Procent przekroczenia opóźnienia: {rel[2]}%\n")
```

```
C:\Users\Mateusz\Desktop\sieci>python xd.py --topologia 2 --podpunkt 3
max liczba pakietów w macierzy:5
Niezawodnosc sieci: 98.92%
Procent rozspojnien sieci: 1.08%
Procent przekroczenia opóźnienia: 0.0%

max liczba pakietów w macierzy:10
Niezawodnosc sieci: 98.08%
Procent rozspojnien sieci: 1.0%
Procent przekroczenia opóźnienia: 0.92%

max liczba pakietów w macierzy:15
Niezawodnosc sieci: 88.48%
Procent rozspojnien sieci: 1.32%
Procent przekroczenia opóźnienia: 10.24%

max liczba pakietów w macierzy:20
Niezawodnosc sieci: 0.0%
Procent rozspojnien sieci: 0.92%
Procent przekroczenia opóźnienia: 99.56%


C:\Users\Mateusz\Desktop\sieci>python xd.py --topologia 1 --podpunkt 3
max liczba pakietów w macierzy:5
Niezawodnosc sieci: 96.16%
Procent rozspojnien sieci: 3.84%
Procent przekroczenia opóźnienia: 0.0%

max liczba pakietów w macierzy:10
Niezawodnosc sieci: 81.08%
Procent rozspojnien sieci: 4.04%
Procent przekroczenia opóźnienia: 15.52%

max liczba pakietów w macierzy:15
Niezawodnosc sieci: 0.0%
Procent rozspojnien sieci: 3.48%
Procent przekroczenia opóźnienia: 98.12%

max liczba pakietów w macierzy:20
Niezawodnosc sieci: 0.0%
Procent rozspojnien sieci: 3.8%
Procent przekroczenia opóźnienia: 99.56%
```

Tutaj także topologia numer 1 wypada gorzej, można zauważyć, że wraz ze wzrostem maksymalnej liczby pakietów do wpisania w macierzy wzrasta prawdopodobieństwo przekroczenia dopuszczalnego opóźnienia.
największą różnice pomiędzy tymi dwiema topologiami widać dla max_pakietow=15, gdzie topologia nr 2 ma 10% na opóźnienie, a topologia nr 1 aż 98%.Prawdopodobieństwo rozspójnienia sieci bez większych zmian

Podpunkt 4

Mierząc niezawodność zostały użyte następujące parametry:
prawdopodobieństwo = 0.97
średnia wielkość pakietu = 10b
t_max = 1/100
w macierzy można było wpisać od 1-10 liczby pakietów
przepustowość dla każdej krawędzi wynosiła 5_000,6_000,….,10_000

```
for i in range(5_000,11_000,1_000):
    for edge in n.edges:
        edge.c = i
    rel = n.reliability(t_max=t_max,rep=2500)
    print(f"Maksymalna przepustowość w krawędziach:{i}")
    print(f"Niezawodnosc sieci: {rel[0]}%")
    print(f"Procent rozspojnien sieci: {rel[1]}%")
    print(f"Procent przekroczenia opóźnienia: {rel[2]}%\n")
```

```
C:\Users\Mateusz\Desktop\sieci>python xd.py --topologia 2 --podpunkt 4
Maksymalna przepustowość w krawędziach:5000
Niezawodnosc sieci: 0.0%
Procent rozspojnien sieci: 1.2%
Procent przekroczenia opóźnienia: 99.56%

Maksymalna przepustowość w krawędziach:6000
Niezawodnosc sieci: 59.44%
Procent rozspojnien sieci: 1.08%
Procent przekroczenia opóźnienia: 39.72%

Maksymalna przepustowość w krawędziach:7000
Niezawodnosc sieci: 88.56%
Procent rozspojnien sieci: 1.16%
Procent przekroczenia opóźnienia: 10.44%

Maksymalna przepustowość w krawędziach:8000
Niezawodnosc sieci: 95.32%
Procent rozspojnien sieci: 0.8%
Procent przekroczenia opóźnienia: 3.88%

Maksymalna przepustowość w krawędziach:9000
Niezawodnosc sieci: 97.52%
Procent rozspojnien sieci: 0.72%
Procent przekroczenia opóźnienia: 1.76%

Maksymalna przepustowość w krawędziach:10000
Niezawodnosc sieci: 97.96%
Procent rozspojnien sieci: 1.36%
Procent przekroczenia opóźnienia: 0.68%


C:\Users\Mateusz\Desktop\sieci>python xd.py --topologia 1 --podpunkt 4
Maksymalna przepustowość w krawędziach:5000
Niezawodnosc sieci: 0.0%
Procent rozspojnien sieci: 3.56%
Procent przekroczenia opóźnienia: 99.68%

Maksymalna przepustowość w krawędziach:6000
Niezawodnosc sieci: 0.0%
Procent rozspojnien sieci: 3.4%
Procent przekroczenia opóźnienia: 99.44%

Maksymalna przepustowość w krawędziach:7000
Niezawodnosc sieci: 0.0%
Procent rozspojnien sieci: 3.32%
Procent przekroczenia opóźnienia: 98.08%

Maksymalna przepustowość w krawędziach:8000
Niezawodnosc sieci: 61.8%
Procent rozspojnien sieci: 3.72%
Procent przekroczenia opóźnienia: 35.2%

Maksymalna przepustowość w krawędziach:9000
Niezawodnosc sieci: 72.44%
Procent rozspojnien sieci: 4.12%
Procent przekroczenia opóźnienia: 24.0%

Maksymalna przepustowość w krawędziach:10000
Niezawodnosc sieci: 86.4%
Procent rozspojnien sieci: 3.64%
Procent przekroczenia opóźnienia: 10.24%
```

Tutaj także widać, że topologia numer 2 jest o wiele lepsza. Wraz ze wzrostem przepustowości zaobserwować można spadek prawdopodobieństwa przekroczenia opóźnienia.Warto zauważyć, że dla przepustowości 7000b niezawodność topologii numer 2 jest większa aż o 88% od niezawodności topologii numer 1.
Prawdopodobieństwo rozspójnienia sieci bez większych zmian

Podpunkt 5

Mierząc niezawodność zostały użyte następujące parametry:
prawdopodobieństwo = 0.97
średnia wielkość pakietu = 10b
t_max = 1/100
w macierzy można było wpisać od 1-10 liczby pakietów
przepustowość dla każdej krawędzi wynosiła 10_000

```
print(f"liczba krawedzi:{len(n.edges)}")
rel = n.reliability(t_max=t_max,rep=2500)
print(f"Niezawodnosc sieci: {rel[0]}%")
print(f"Procent rozspojnien sieci: {rel[1]}%")
print(f"Procent przekroczenia opóźnienia: {rel[2]}%\n")
for i in range(4):
    n.add_edge()
    n.add_edge()
    print(f"liczba krawedzi:{len(n.edges)}")
    rel = n.reliability(t_max=t_max,rep=2500)
    print(f"Niezawodnosc sieci: {rel[0]}%")
    print(f"Procent rozspojnien sieci: {rel[1]}%")
    print(f"Procent przekroczenia opóźnienia: {rel[2]}%\n")
```

program 4 razy dodaje dwie krawędzie i wyświetla niezawodność

```
C:\Users\Mateusz\Desktop\sieci>python xd.py --topologia 2 --podpunkt 5
liczba krawedzi:29
Niezawodnosc sieci: 97.88%
Procent rozspojnien sieci: 1.08%
Procent przekroczenia opóźnienia: 1.04%

liczba krawedzi:31
Niezawodnosc sieci: 99.24%
Procent rozspojnien sieci: 0.64%
Procent przekroczenia opóźnienia: 0.12%

liczba krawedzi:33
Niezawodnosc sieci: 99.12%
Procent rozspojnien sieci: 0.84%
Procent przekroczenia opóźnienia: 0.04%

liczba krawedzi:35
Niezawodnosc sieci: 99.36%
Procent rozspojnien sieci: 0.56%
Procent przekroczenia opóźnienia: 0.08%

liczba krawedzi:37
Niezawodnosc sieci: 99.36%
Procent rozspojnien sieci: 0.64%
Procent przekroczenia opóźnienia: 0.0%


C:\Users\Mateusz\Desktop\sieci>python xd.py --topologia 1 --podpunkt 5
liczba krawedzi:26
Niezawodnosc sieci: 86.16%
Procent rozspojnien sieci: 3.68%
Procent przekroczenia opóźnienia: 10.4%

liczba krawedzi:28
Niezawodnosc sieci: 87.92%
Procent rozspojnien sieci: 3.24%
Procent przekroczenia opóźnienia: 9.24%

liczba krawedzi:30
Niezawodnosc sieci: 97.52%
Procent rozspojnien sieci: 1.88%
Procent przekroczenia opóźnienia: 0.6%

liczba krawedzi:32
Niezawodnosc sieci: 98.84%
Procent rozspojnien sieci: 1.04%
Procent przekroczenia opóźnienia: 0.12%

liczba krawedzi:34
Niezawodnosc sieci: 98.6%
Procent rozspojnien sieci: 1.4%
Procent przekroczenia opóźnienia: 0.0%
```

zauważyć można, że wraz ze wzrostem liczby krawędzi maleje prawdopodobieństwo rozspójnienia oraz przekroczenia opóźnienia.Dodatkowo zauważyć można, że topologia numer 2 jest lepsza od pierwszej nawet wtedy kiedy ma mniejszą liczbę krawędzi

WNIOSKI:

bardzo ważne jest stworzenie dobrej topologii sieci, ponieważ jak można było zauważyć w powyższych przykładach, dobra topologia wypadała lepiej od gorszej nawet dla gorszych parametrów