SPRAWOZDANIE

|  |  |
| --- | --- |
| **Przedmiot:** | **Teoria i inżynieria ruchu teleinformatycznego (Z06-43d INZ007600P)** |
| **Prowadzący:** | mgr Michał Pleszkun |
| **Autorzy:** | Hanna Mielczarek (164699)  Tomasz Marciniak (213217)  Tomasz Obszarny (94800) |

Spis treści

[Wprowadzenie 2](#_Toc378347893)

[Środowisko 2](#_Toc378347894)

[Generatory 3](#_Toc378347895)

[Generator ruchu o rozkładzie Poissona 3](#_Toc378347896)

[Generator ruchu o rozkładzie Wykładniczym 4](#_Toc378347897)

[Generator ruchu On/Off 6](#_Toc378347898)

[Generator ruchu MMPP 6](#_Toc378347899)

[Literatura 7](#_Toc378347900)

# Wprowadzenie

Projekt zakładał stworzenie wielu symulacji sieci. Jako narzędzie należało wykorzystać darmowe środowisko OMNET++. Na początku zaimplementowano różne generatory ruchu, które posłużyły do przetestowania różnych algorytmów kontroli dostępu, profilowania oraz schedulerów. Dzięki temu stało się możliwe przeprowadzenie badań izebranie statystyk. Do projektu zostały załączone źródła.

## Środowisko

Zgodnie z wytycznymi do implementacji projektu wykorzystane zostało środowisko OMNET++. Jest ono darmowe dla celów niekomercyjnych i oparte na znanym IDE o nazwie Eclipse co zapewnia bogatą funkcjonalność oraz możliwość intalowania pluginów. Pozwala na implementację komponentów sieciowych w języku C++ zapewniając odpowiednie podstawowe komponenty (np. cMessage). Środowisko samo w sobie nie jest symulatorem a raczej środowiskiem do tworzenia symulacji.

Pozwala na tworzenie modeli sieci zapisywanych do pliku z rozszerzeniem ned za pomocą graficznego edytora lub ręcznie. Modele zawierają moduły, które można hierarchicznie zagnieżdzać.

Symulacje mogą być uruchamiane z poziomu środowiska lub linii poleceń. Można debugować kod. Do stworzenia prostej symulacji wystarczą pliki:

* \*.ned – modele sieci
* \*.cpp – implementacje komponentów
* \*.h – pliki nagłówkowe języka C++
* \*.ini – pliki konfiguracyjne (sterujące)

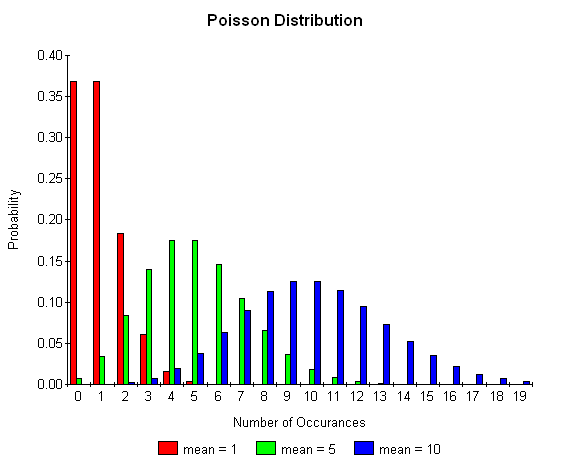
Do najważniejszych funkcji programu można zaliczyć:

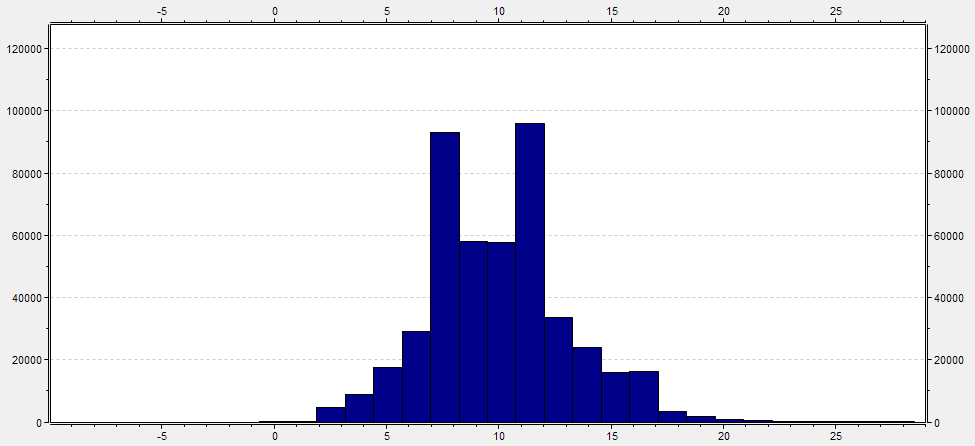
* Możliwość uruchamiania dowolnie skonfigurowanej symulacji
* Modelowanie sieci (protokoły, sieci: kablowe, bezprzewodowe oraz kolejkowe)
* Gromadzenie statystyk
* Prezentacja graficzna pracy symulowanego modelu (możliwość dostrajania działania)
* Prezentacja statystyk w formie wykresów

# Generatory

## Generator ruchu o rozkładzie Poissona

Generuje ruch pakietów w odstępach czasu opartych o rozkład Poissona, który jest dyskretnym rozkładem prawdopodobieństwa, wyrażającym prawdopodobieństwo szeregu wydarzeń mających miejsce w określonym czasie, gdy te wydarzenia występują ze znaną średnią częstotliwością i w sposób niezależny od czasu jaki upłynął od ostatniego zajścia tego zdarzenia.



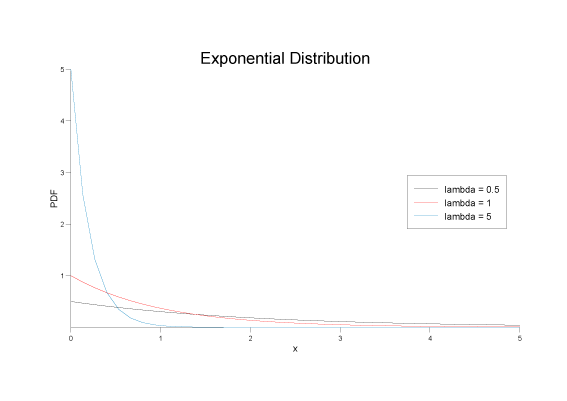


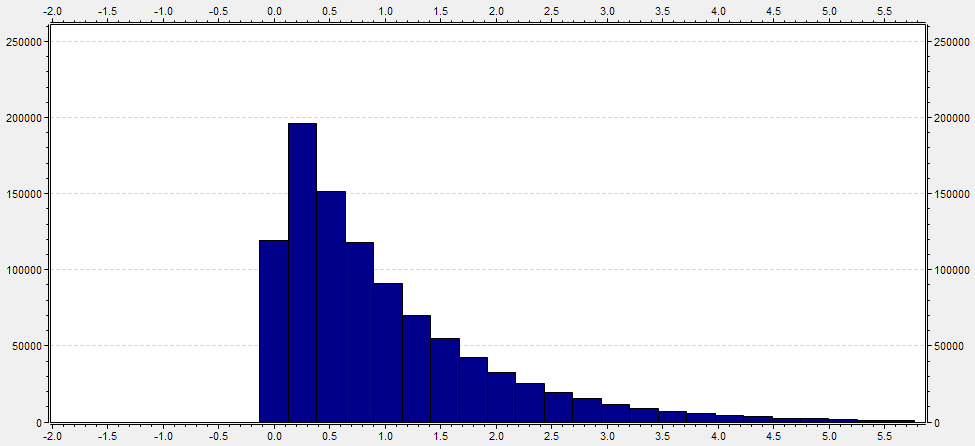
Jeśli oczekiwaną liczbą zdarzeń w przedziale czasu jest λ, to prawdopodobieństwo, że jest dokładnie *k* wystąpień, gdzie *k* jest nieujemną liczbą całkowitą jest równe

Aby wygenerować ruch o rozkładzie Poissona korzystamy ze sposobu podanego przez Knutha. Daje to złożoność liniową względem λ.

## Generator ruchu o rozkładzie Wykładniczym

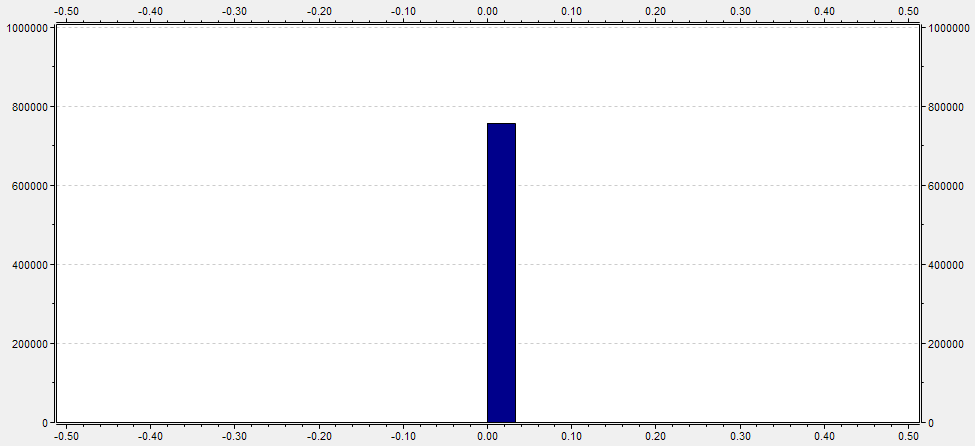
Generuje ruch w podobny sposób do generatora o rozkładzie Poissona z tą różnicą, że rozkład jest oparty o rozkład wykładniczy



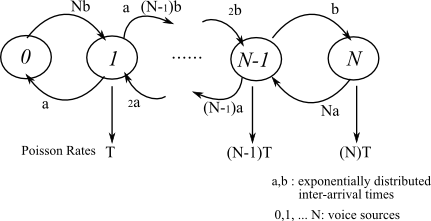


## Generator ruchu On/Off

Generuje ruch w sposób nie ciągły. Posiada dwa tryby: On i Off. Kiedy jest w trybie „On” wysyła pakiety ze stałym odstępem w czasie. W trybie off nie generuje pakietów i czeka na przejście w poprzedni tryb. W naszej implementacji podczas trybu Off generowane są pakiety wysyłane na samego siebie aby wzbudzić pracę generatora.



## Generator ruchu MMPP





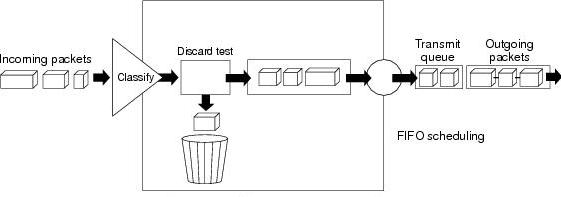
# Algorytmy

## Admission Control

Kontrola przyjęć (Admission control) jest procesem walidacji w systemach komunikacyjnych, gdzie przeprowadzane jest sprawdzenie przed nawiązaniem połączenia, czy dostępne zasoby są wystarczające aby dane połączenie obsłużyć. W celu zasymulowania tego modułu posłużyliśmy się 4 różnymi algorytmami:

* Tail Drop
* Front Drop
* RED (Random Early Detection)
* WRED (Weighted Random Early Detection)

### Tail Drop



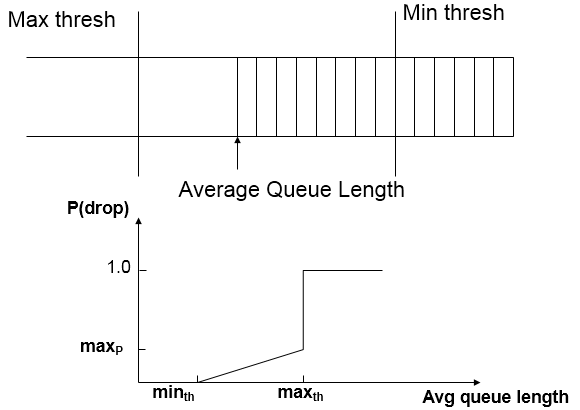
Tail drop jest prostym algorytmem zarządzania kolejkami, który często jest wykorzystywany w ruterach podczas podejmowania decyzji, które pakiety odrzucić. Każdy pakiet jest traktowany tak samo. Kiedy kolejka jest maksymalni zapełniona, kolejne nowe pakiety, które przychodzą od tamtej pory, są odrzucane aż do momentu, gdy zwolni się miejsce w kolejce.

### Front drop

Działanie algorytmu jest analogiczne do działania algorytmu Tail Drop. W wypadku wszystkie pakiety również traktowane są tak samo. Różnica między tymi dwoma algorytmami polega na tym, że w wypadku, gdy kolejka jest zapełniona, to pakiety, które najdłużej czekały w kolejce zostają odrzucone (z przodu kolejki), a nowe pakiety zostają przyjęte.

### Random Early Detection

Random Early Detection (RED), w przeciwieństwie do poprzednich dwóch algorytmów, jest algorytmem unikania zakleszczeń. Algorytm ten monitoruje średnią wielkość kolejki i odrzuca pakiety bazując na prawdopodobieństwach statystycznych. Jeżeli bufor jest prawie pusty, wszystkie przychodzące pakiety są akceptowane. W miarę jak kolejka rośnie wzrasta prawdopodobieństwo odrzucenia przychodzącego pakietu. Kiedy bufor jest pełny, prawdopodobieństwo osiąga 1, przez co wszystkie pakiety są odrzucane.



### Weighted Random Early Detection

Weighted Random Early Detection (WRED) działa na podobnej zasadzie co algorytm RED. Wprowadza tylko jedną zmianę. Niektóre pakiety mogą być ważniejsze od pozostałych, dlatego nie powinny być tak samo często odrzucane jak reszta pakietów. WRED zapewnia, że prawdopodobieństwo odrzucenia ważniejszych pakietów jest mniejsze w danej chwili od prawdopodobieństwa odrzucenia pakietów o standardowej wadze. Parametry dla poszczególnych klas pakietów są przez nas losowane.

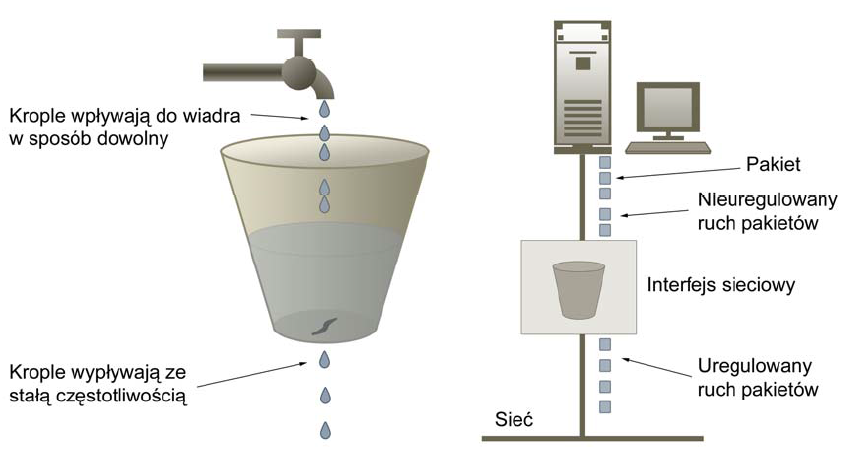
## Profilowanie ruchu

Mechanizm kształtowania ruchu zapobiega powstawaniu zatorów. Jego działanie opiera się o wygładzanie ruchu, który wypływa z węzła do sieci, przez co odbiorca transmisji nie zostanie zalany zbyt dużą ilością docierających do niego pakietów. Algorytmy wykorzystywane przy profilowaniu ruchu stosuje się przy wyprowadzaniu pakietów z kolejek. Oznacza to, że algorytmu te nie są same w sobie algorytmami kolejkowania, ale służą do pewnego zaplanowania procesu wysyłania pakietów. Zaimplementowane przez nas algorytmy profilowania ruchu to:

* Leaky Bucket
* Double Leaky Bucket
* Token Bucket
* Double Token Bucket

### Leaky Bucket

Algorytm ten umożliwia dokładne określenie, z jaką częstością dane będą pobierane z kolejek i przesyłane przez sieć. Natężenie przepływu danych objętego taką strategią charakteryzuje się płaskim przebiegiem. Gwarantuje to zabezpieczenie przed wystąpieniem przeciążeń. Podstawą działania algorytmu jest pojedynczy bufor – kolejka FIFO, która ma określoną maksymalną długość. Element algorytmu planujący wysyłanie pakietów wybiera je z kolejki przy zachowaniu zasady, że wyjście ma mieć stałe natężenie. Ideę działania algorytmu przedstawia poniższy schemat:

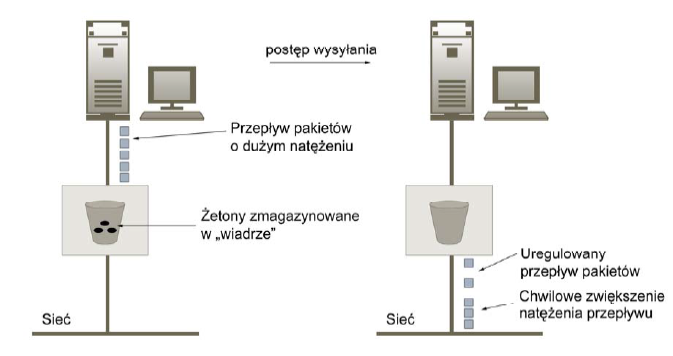


### Double Leaky Bucket

Działanie algorytmu jest analogiczne do poprzedniego. Różnica polega na tym, że w tym algorytmie połączone są dwa Leaky Buckety o różnych charakterystykach. Powoduje to zwiększoną odporność na przepełnienie buforów.

### Token Bucket

Algorytm ten podobnie jak Leaky Bucket również umożliwia określenie, z jakim natężeniem dane będą wysyłane. Jednak w odróżnieniu od poprzednio opisanego, pozwala, aby ruch wychodzący zmieniał się częściowo w zależności od stopnia nasilenia żądań wysyłania pakietów. Podstawą działania algorytmu jest bufor gromadzący żetony. Po przesłaniu pakietu pewna ilość wykorzystanych do tego celu tokenów ulega zniszczeniu. Efekt wygładzania przepływu i ograniczenie wykorzystania przepustowości otrzymuje się przez stałe w czasie generowanie nowych żetonów. Gdy nie ma aktualnie danych do wysłania, żetony są gromadzone, aż wypełnią bufor i mogą zostać użyte do wysłania zwiększonej liczby pakietów i obsłużenia nagłym wzrostów natężenia ruchu. Działanie algorytmu przedstawia poniższy schemat:



### Double Token Bucket

Działanie tego algorytmu jest analogiczne do działania algorytmu Double Leaky Bucket, z tą jedyną różnicą, że do jego utworzenia wykorzystywane są dwa Token Buckety.

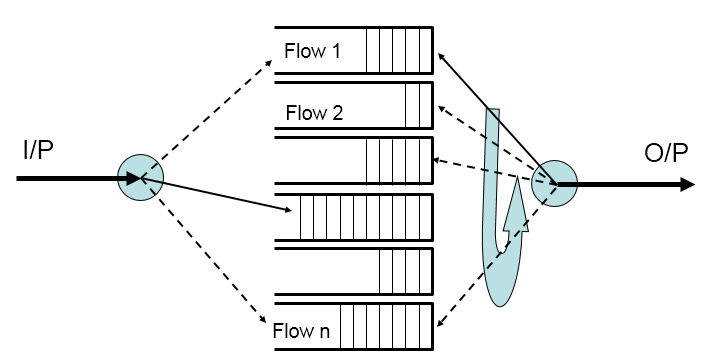
## Scheduler

Algorytmy kolejkowania są podstawowymi mechanizmami kształtowania ruchu w sieciach komputerowych. Umożliwiają bowiem utworzenie pewnych kategorii przepływu danych, do których można zastosować określone reguły dostępu do łącza. Wybrane przez nas algorytmy kolejkowania to:

* FQ (Fair Queuing)
* WFQ (Weighted Fair Queuing)
* RR (Round Robin)
* WRR (Weighted Round Robin)

### Fair Queuing

Algorytm sprawiedliwego kolejkowania emuluje dzielenie bitowe, obliczając kiedy każdy z pakietów powinien zostać wysłany, by czasy pokrywały się (czy tez były jak najbardziej zbliżone) z wysłaniem ostatniego bitu pakietu przez domniemany algorytm karuzelowy dzielący bitowo. W naszym przypadku algorytm stara się wyrównać długość pakietów wychodzących z poszczególnych kolejek. Idea działania algorytmu została przedstawiona na poniższym schemacie:

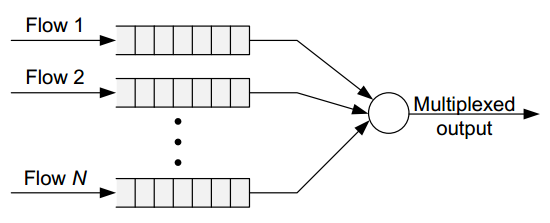


### Weighted Fair Queuing

Podobnie do algorytmu Fair Queuing z wyjątkiem nadawania poszczególnym kolejkom losowych wag. Dwa razy większa waga oznacza, że z kolejki w tym samym czasie mogą wyjść pakiety o dwa razy większej sumie długości.

### Round Robin

Round Robin zwany algorytmem karuzelowym jest najprostszym algorytmem szeregowania. Działanie algorytmu polega na cyklicznej obsłudze każdej z kolejek, jedna po drugiej. Pod pojęciem kolejki można również rozumieć przykładowo strumień lub połączenie sieciowe. W wypadku, gdy któraś z kolejek się zapełni, nie wpływa ona na jakość ruchu w pozostałych kolejkach. Sprawiedliwość może być zapewniona w wypadku, gdy wszystkie kolejki mają jednakową wagę oraz gdy pakiety mają jednakową długość.



### Weighted Round Robin

Weighted Round Robin jest zmodyfikowaną wersją zwykłego RR. Różni się ona ty, że pozwala przypisać wagę każdemu z połączeń (kolejek). Wagi te są normalizowane, po czym służą za wyznacznik, ile pakietów z danej kolejki może zostać zdjętych na raz podczas jednego okresu.

# Literatura

1. A. Grzech, Sterownie ruchem w sieciach teleinformatycznych, Oficyna Wydawnicza Politechniki Wrocławskiej, 2002
2. <http://inet.omnetpp.org>
3. <http://www.omnetpp.org/doc/omnetpp/Manual.pdf>
4. <http://www.zsk.p.lodz.pl/~morawski/Dyplomy/Praca%20dyplomowa%20p.%20Pijewskiego.pdf>
5. <http://www.znsi.aei.polsl.pl/materialy/SI63/SI63_2.pdf>