|  |
| --- |
|  |
| Teoria i inżynieria ruchu teleinformatycznego |
| projekt |

1/1/2016

|  |
| --- |
| Marcin Borowiński 203393  Thibault Dehacq 206960 |

# Wstęp

Zadaniem projektu jest zamodelowanie ruchu teleinformatycznego w wybranym programie. Do tego celu wybraliśmy silnik do gier „Unreal engine” wyprodukowany przez firmę Epic Games. Pozwala on na blokową implementację algorytmów oraz wizualizację obiektów 3D.

Ruch teleinformatyczny może być zaimplementowany w różne sposoby w zależności od potrzeb systemów lub użytkowników. Administratorzy oraz architekci sieci mogą mieć wpływ na wiele czynników, takich jak sposób generowania ruchu w sieci, kontrola przyjęć lub harmonogramowanie ruchu.

Na początek zdefiniujmy, czym jest ruch teleinformatyczny. Są to strumienie jednostek danych generowane w ramach procesów komunikacyjnych obsługiwane przez dane protokoły komunikacyjne. Z kolei protokół komunikacyjny jest zbiorem reguł i kroków postępowania wykonywanych w sposób automatyczny przez urządzenia komunikacyjne. Protokoły zapewniają łączność i wymianę danych pomiędzy urządzeniami. Protokoły dzielimy na klasyczne, internetowe oraz stosowane w automatyce przemysłowej.

Podstawowym zadaniem sterowania ruchem w sieciach teleinformatycznych jest dostarczenie, jakości usługi dla użytkownika w sieci oraz optymalne wykorzystanie dostępnych zasobów. Do podstawowych zadań związanych ze sterowaniem ruchem teleinformatycznym jest wyznaczanie tras, przeciwdziałania przeciążeniom sieci oraz sterowanie przepływem danych. Złożoność tych zadań rośnie wraz z urozmaiceniem grupy użytkowników w tej samej sieci teleinformatycznej.

W naszym przypadku będziemy mieć wpływ na sposób generowania ruchu, czyli odstępy między generowaniem kolejnych pakietów. Koncentrujemy się na rozkładzie Poissona, On-Off, wykładniczy oraz MMPP, które zostaną omówione później. Dodatkowo zostaną zaimplementowane oraz analizowane takie komponenty jak Admission Control, który jest procesem zatwierdzania w komunikacji pakietów. Innymi słowa sprawdza, czy podczas komunikacji pakietów są wystarczające środki. Scheduler, czyli algorytm kolejkowania, czyli, w jaki sposób są wysyłane oraz odbierane pakiety. Profilowanie ruchu, czyli algorytmy, które służą do kontrolowania ilości pakietów przechodzących przez dany węzeł w trakcie sekundy.

# Cel dokumentu

Dokument ma na celu wprowadzić osobę do problematyki ruchu teleinformaczycznego oraz opisanie i analizę zaimplementowanych algorytmów. Tego typu zadanie powinno wykazać - przynajmniej w teorii - jakie algorytmy się najlepiej sprawują w danej sytuacji.

# Wybrane środowisko

Środowiskiem wybranym do implementacji algorytmów jest Unreal Engine – obiektowy silnik gry, wyprodukowany przez firmę Epic Games. Atutami przesądzającymi o jego wyborze są:

* intuicyjna obsługa
* graficzna reprezentacja kodu
* ułatwiona integracja logiki z trójwymiarową wizualizacją
* wsparcie producenta
* dbałość firmy o społeczność użytkowników (community)

# Przedstawienie problemu

## Struktura programu

Z wykorzystaniem paradygmatu obiektowego środowiska, został utworzony zbiór klas (nazywanych w środowisku „Blueprints”):  
  
Klasy główne:

**Node** – jest to reprezentacja węzła, który wysyła pakiety do swoich odbiorców lub, w przypadku braku tych ostatnich, „konsumuje” je, usuwając z systemu. To właśnie w nim znajduje się większość algorytmów sterujących przepływem pakietów w sieci. W przypadku otrzymania pakietu węzeł tworzy na swojej liście nowy pakiet o losowych parametrach. Graficznie jest on reprezentowany przez walec szarego koloru. Posiada zbiór pakietów do wysłania, które są wyświetlane pionowo – tuż nad nim. Powiązania z węzłami-odbiorcami są oznaczone jasną strzałką.

**Packet** – model pakietu, którego najważniejsze zmienne to:  
DST : Node – Węzeł, do którego pakiet zmierza. Jeżeli pakiet znajduje się w kolejce, jego cel jest niezdefiniowany.  
SRC: Node – Węzeł, z którego pakiet pochodzi.  
Session id : Integer – ID sesji  
Priority : Integer – Priorytet pakietu, brany pod uwagę w przypadku algorytmu kolejki priorytetowej. Jest on zależny od numeru węzła docelowego na liście odbiorców węzła-nadawcy.  
Payload : String – wiadomość przekazywana w pakiecie

Klasy pomocnicze:

**MainMenuWidget** – menu główne, zawierające konfigurowalne ustawienia symulacji, które zostały opisane w dalszych częściach dokumentu

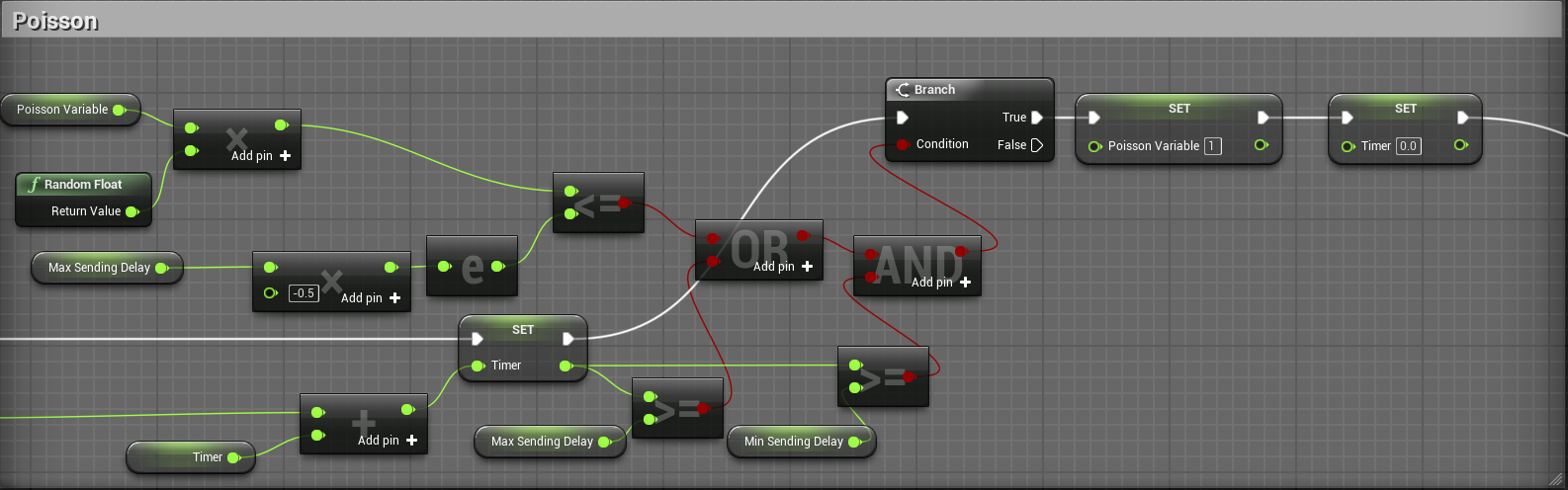
**MainMenuLevel** – prezentacja opakowująca menu główne

**GameInstance** – instancja gry, w której zapisywane są ustawienia użytkownika

**StartingPoint** – obiekt pojawiający się na początku wizualizacji, pobiera wybrane przez użytkownika ustawienia z instancji gry i na ich podstawie tworzy określoną liczbę węzłów obsługujących ruch według wskazanych algorytmów

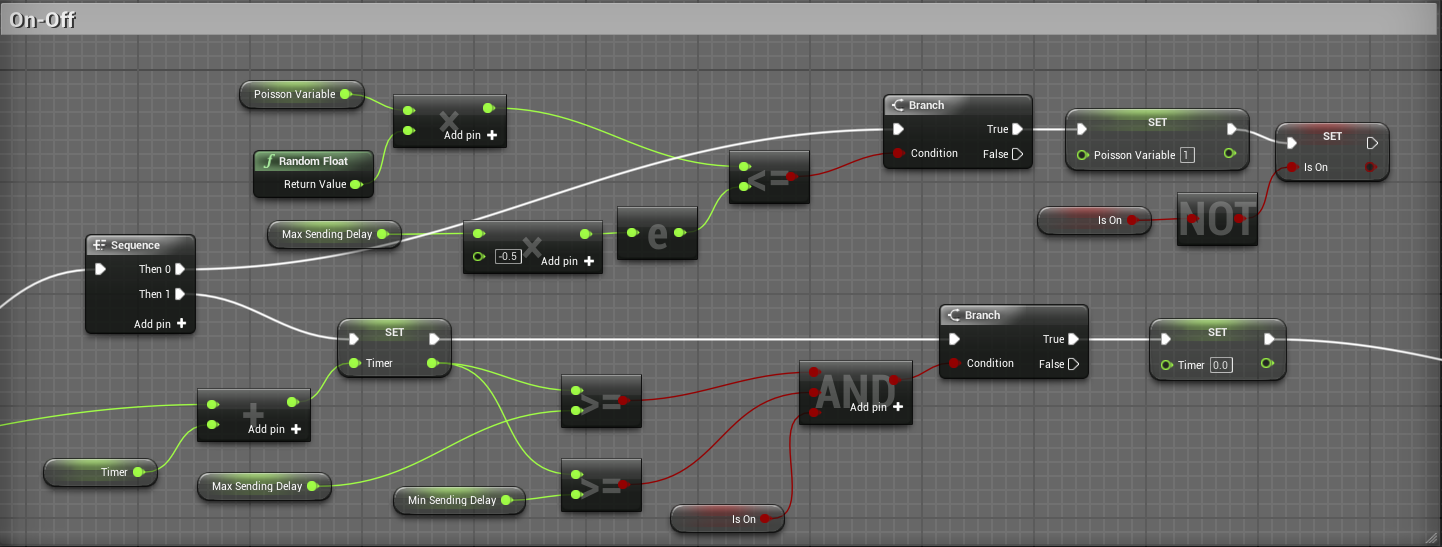
## Opis algorytmów

Pierwszym omawianym oraz implementowanym algorytmem jest generowanie ruchu według rozkładu Poissona. Określa on prawdopodobieństwo wysłania pakietu w danym momencie czasu. Na rozkład Poissona można wpływać za pomocą parametru λ. Im jest on większy tym bardziej rozkład przypomina normlany.



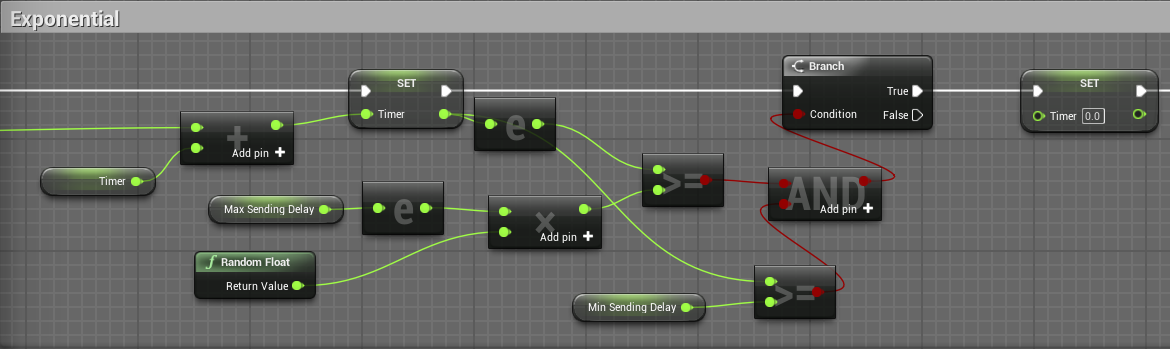
*Rysunek 3.1.1 przedstawiający blokową implementację algorytmu generującego ruch według rozkładu Poissona.*

Drugim implementowanym algorytmem jest generowanie ruchu według rozkładu On-Off. W którym istnieją tylko dwa stany. On jest stanem, w którym pakiety są wysyłane natomiast stan Off jest stanem przeciwnym, żaden pakiet nie jest wysyłany.



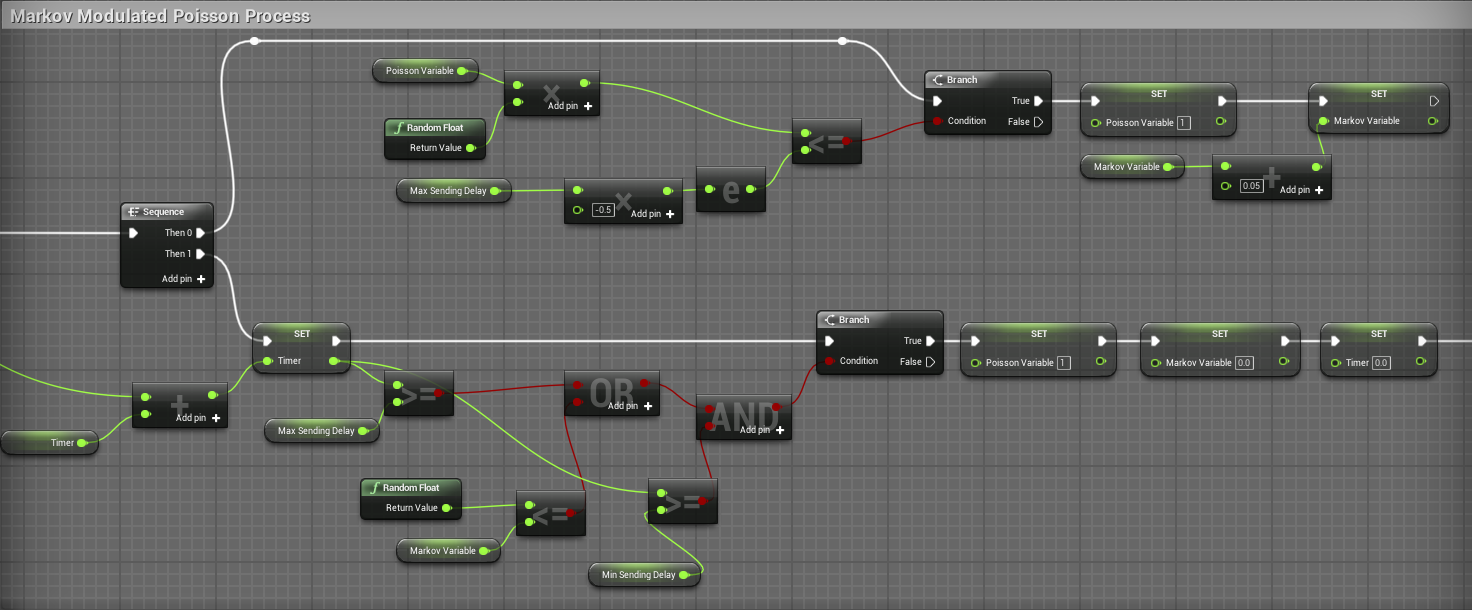
*Rysunek 3.1.2 Reprezentujący blokowa implementację algorytmu generującego ruch według rozkładu On-Off*

Trzecią implementacją jest generowanie ruchu teleinformatycznego według rozkładu wykładniczego. Opisuje fenomeny bez pamięciowe innymi słowa, prawdopodobieństwo nie zależy od stanu poprzedniego. Według rozkładu im dłuższy czas mija, tym prawdopodobieństwo na przykład wysłania pakietu jest większe. Parametrem tego rozkładu podobnie jak w rozkładzie Poissona jest λ. Wraz z jej wzrostem, spłaszczamy wykres.



Rysunek 3.1.3 Reprezentujący blokową implementację algorytmu generującego ruch według rozkładu Wykładniczego.

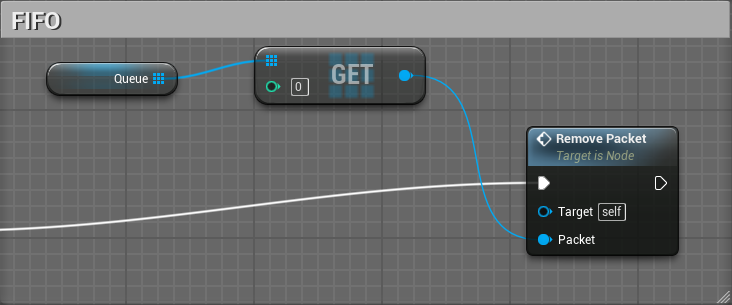
Ostatnim implementowanym algorytmem ruchu jest rozkład Markov Modulated Piosson Process. Jest to modyfikacja rozkładu Poissona. Każdy proces Poissona jest przełączany między ciągłym łańcuchem Markowa.



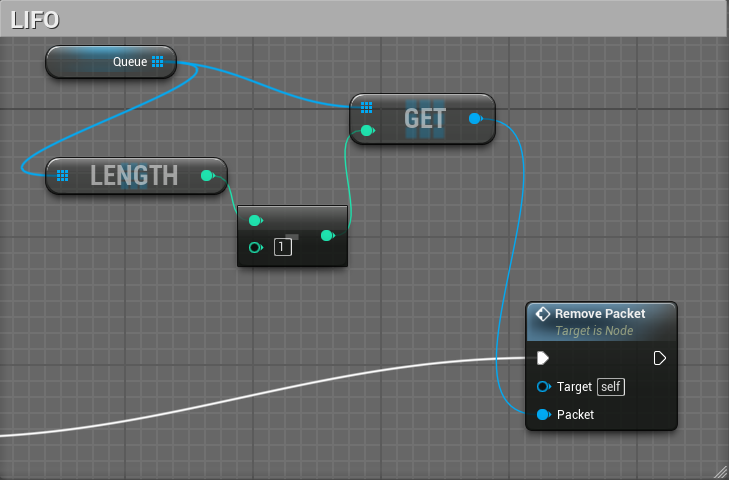
*Rysunek 3.1.4 Reprezentujący blokową implementację algorytmu generującego ruch według rozkładu MMPP*.

Algorytmy kolejkowania:

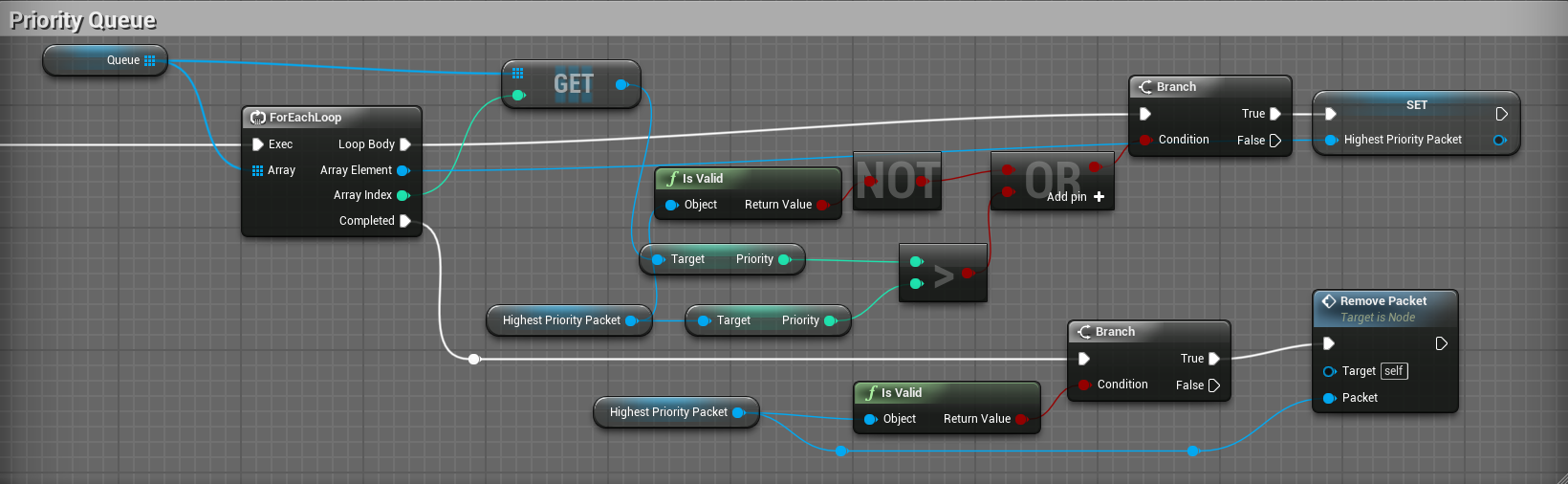
FIFO – czyli first in first out, polega na tym, że pierwszy pakiet który dochodzi do węzła jest pierwszym pakietem który z tego węzła zostanie wysłany.



LIFO – czyli last in first out, polega na tym, że ostatni pakiet który dochodzi do węzła jest pierwszym pakietem który zostanie z tego węzła wysłany.



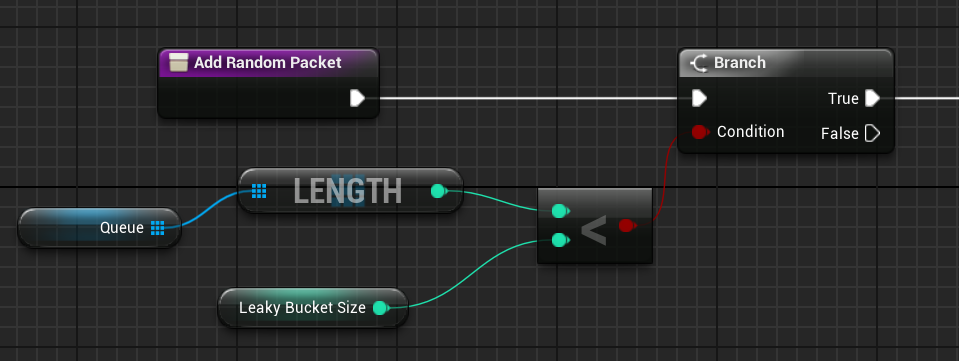
Kolejka priorytetowa – Zwykła kolejka LIFO lub FIFO, z dodatkiem priorytetów na pakietach. Sprawia to, iż pakiety a większym priorytecie są obsługiwane przed innymi pakietami.



WRR – czyli weighted round robin, polega na obsługiwaniu pakietów w sposób cykliczny, lecz pakiety o większej wadze ( większy priorytet) są obsługiwane wcześniej tak, aby waga wszystkich pakietów była taka sama.

Algorytmy kontroli przyjęcia:

Domyślną logiką systemu jest przyjmowanie do węzłów wszystkich pakietów przychodzących, ale został również opracowany algorytm Leaky Bucket, który pozwala na zdefiniowanie w węźle jego maksymalnej pojemności pakietów. Wszystkie pakiety nadmiarowe zostają odrzucone. Zostało to osiągnięte dzięki instrukcji warunku:



## Wyniki symulacji i ich analiza

Analiza implementowanego programu polegała na badaniu średnich czasów wysyłania pakietów w zależności od rozkładu generowanego ruchu oraz algorytmów kolejkowania. Podczas badań następujące zmienne nie były modyfikowane:

-Liczba węzłów = 5

-Minimalna liczba odbiorców = 3

-Minimalne opóźnienie wysłania = 5 [s]

-Liczba pakietów w węźle = 5

-Maksymalna liczba odbiorców = 3

-Maksymalne opóźnienie wysłania = 5 [s]

-Leaky bucket size = 5

Otrzymane wyniki dla rozkładu Poissona i kolejkowania FIFO, LIFO i priority queue.

|  |  |  |
| --- | --- | --- |
|  | | |
|  | Poisson |  |
|  | Fifo | Priority |
|  | 1.197234 | 1.061471 |
|  | 1.637968 | 1.880434 |
|  | 1.647933 | 2.240472 |
|  | 2.574592 | 2.515604 |
|  | 1.097964 | 1.651258 |
|  | 2.90669 | 3.282518 |
|  | 1.794568 | 1.0611812 |
|  | 3.122564 | 1.590284 |
|  | 1.745207 | 1.341135 |
|  | 1.613906 | 1.881645 |
|  | 1.075496 | 5.246584 |
|  | 2.689394 | 1.962145 |
|  | 1.126218 | 5.451645 |
|  | 1.829141 | 1.596452 |
|  | 1.578523 | 6.210081 |
|  | 2.086878 | 1.028013 |
|  | 1.216676 | 3.133764 |
|  | 1.925009 | 3.266399 |
|  | 5.268195 | 2.294565 |
|  | 1.368571 | 1.796452 |
|  | 1.350284 | 3.305465 |
|  | 4.966212 | 6.492654 |
| AVG | 2.08269195 | 2.7404646 |

|  |  |  |
| --- | --- | --- |
|  | On-Off |  |
|  | Fifo | Priority |
|  | 10.000181 | 10.054795 |
|  | 10.000181 | 11.946542 |
|  | 10.000181 | 17.465446 |
|  | 10.000181 | 10.165445 |
|  | 11.900373 | 10.065445 |
|  | 10.008787 | 10.000654 |
|  | 10.008787 | 9.9999416 |
|  | 10.008787 | 9.9999859 |
|  | 11.076088 | 30.827166 |
|  | 10.162033 | 10.000031 |
|  | 10.162033 | 10.000031 |
|  | 21.094698 | 9.999996 |
|  | 9.999973 | 9.999966 |
|  | 22.333279 | 60.217674 |
|  | 9.999962 | 9.999996 |
|  | 9.999962 | 9.999966 |
|  | 9.999962 | 10.000031 |
|  | 10.00034 | 10.000031 |
|  | 10.000054 | 10.000038 |
|  | 62.375015 | 14.2496411 |
| AVG | 13.9565429 | 14.2496411 |

|  |  |  |
| --- | --- | --- |
|  | Exponential |  |
|  | Fifo | Priority |
|  | 3.377654 | 4.022345 |
|  | 3.64528 | 4.334564 |
|  | 5.365449 | 5.864146 |
|  | 6.214654 | 6.046876 |
|  | 3.914465 | 6.754132 |
|  | 4.145654 | 5.511354 |
|  | 2.941064 | 5.165446 |
|  | 5.314654 | 4.823445 |
|  | 1.214564 | 4.776545 |
|  | 4.564984 | 6.795431 |
|  | 4.486541 | 3.4564214 |
|  | 6.066465 | 6.125987 |
|  | 3.455488 | 4.524984 |
|  | 2.798462 | 5.1846546 |
|  | 5.744564 | 6.3054654 |
|  | 4.785462 | 5.846545 |
|  | 4.245654 | 5.0546545 |
|  | 6.104654 | 4.452654 |
|  | 5.846548 | 6.624684 |
|  | 4.56546 | 3.056565 |
| AVG | 4.439886 | 5.23634495 |

|  |  |  |
| --- | --- | --- |
|  | MMPP |  |
|  | Priority | Fifo |
|  | 1.009026 | 1.02481 |
|  | 1.028465 | 1.055714 |
|  | 1.334564 | 1.114564 |
|  | 1.863375 | 1.984564 |
|  | 1.126544 | 1.000896 |
|  | 1.064435 | 1.132896 |
|  | 2.161108 | 2.684654 |
|  | 1.428645 | 1.007544 |
|  | 1.110044 | 1.112837 |
|  | 2.113308 | 1.016809 |
|  | 1.184652 | 1.249451 |
|  | 1.846522 | 2.498556 |
|  | 1.097465 | 1.112484 |
|  | 6.345546 | 1.034955 |
|  | 1.052121 | 1.062254 |
|  | 2.44584 | 1.02455 |
|  | 1.034565 | 3.984621 |
|  | 1.154565 | 1.334564 |
|  | 4.269458 | 1.721654 |
|  | 2.176242 | 1.999984 |
|  | 1.011754 | 1.099992 |
|  | 1.174555 | 1.134565 |
| AVG | 1.77421814 | 1.47240536 |

Wykres reprezentujący średni czas wysłania pakietu przy domyślnych parametrach programu.

Analiza otrzymanych wyników pozwala na wyciągnięcie następujących wniosków:

- Średni czas wysyłania w przypadku algorytmu LIFO zdaje się być minimalnie mniejszy niż średni czas w przypadku kolejki priorytetowej, co może być spowodowane mniejszą złożonością obliczeniową (dla LIFO)

- Porównując rozkłady generowania ruchu, najlepszym z nich jest MMPP, następnie Poisson. Rozkład wykładniczy radzi sobie dość dobrze, lecz czasy są już większe w porównaniu do dwóch poprzednich. Na końcu znajduje się generowanie ruchu według rozkładu On-Off, których czasy wysyłania pakietów są wyższe niż dla innych algorytmów.

- Algorytm kontroli przyjęcia pakietów Leaky Bucket powoduje straty pakietów (ponieważ odrzuca je, gdy kolejka jest przepełniona), ale może być przydatny w sieciach, w których taka strata nie jest istotnym czynnikiem

## 3.3 Podsumowanie

Projekt pozwolił poznać teorię i inżynierię ruchu teleinformatycznego w inny sposób niż wykłady. Implementacja algorytmów w takim programie jak Epic Unreal Engine pozwala nam na ciekawą wizualizacje ruchu teleinformatycznego. Dzięki temu, w łatwy sposób można zauważyć, który węzeł wysyła pakiety oraz kolejkowanie algorytmów.

Z przeprowadzonych badań oraz ich analizy, można się dowiedzieć, że LIFO oraz FIFO to najprostsze algorytmy kolejkowania. Natomiast najbardziej zaawansowanym rozkładem generowania ruchu (spośród wybranych) jest MMPP, który jest modyfikacją rozkładu Poissona.