# Symulacja Cyfrowa
# Raport końcowy

Marcin Wachowiak,
EiT, TMIB, 135670
Zadanie nr 49, K = 4, LR = 15,
metoda M1 – przeglądania działań, algorytm A6e – CSMA

## 1. Treść zadania:

W sieci bezprzewodowej stacje nadawcze konkurują o dostęp do łącza. W losowych odstępach czasu CGPk *k*-ta stacja nadawcza generuje pakiety gotowe do wysłania. Po uzyskaniu dostępu do łącza zgodnie z algorytmem A, *k*-ty terminal podejmuje próbę transmisji najstarszego pakietu ze swojego bufora. Czas transmisji wiadomości z *k*-tej stacji nadawczej do *k*-tej stacji odbiorczej wynosi CTPk. Jeśli transmisja pakietu zakończyła się sukcesem, stacja odbiorcza przesyła potwierdzenie ACK (ang. *Acknowledgment*) poprawnego odebrania wiadomości. Czas transmisji ACK wynosi CTIZ. Jeśli transmisja pakietu nie powiodła się, stacja odbiorcza nie przesyła ACK. Odbiór pakietu uznajemy za niepoprawny, jeśli w kanale transmisyjnym wystąpiła kolizja lub błąd. Przez kolizję rozumiemy nałożenie się jakiejkolwiek części jednego pakietu na inny pakiet (pochodzący z innego nadajnika). Dodatkowo każda transmisja pakietu może zakończyć się błędem TER. Brak wiadomości ACK po czasie (CTPk+ CTIZ) od wysłania pakietu jest dla stacji nadawczej sygnałem o konieczności retransmisji pakietu. Każdy pakiet może być retransmitowany maksymalnie LR razy. Dostęp do łącza w przypadku retransmisji opiera się na tych samych zasadach co transmisja pierwotna. Jeśli mimo LR -krotnej próby retransmisji pakietu nie udało się poprawnie odebrać, wówczas stacja nadawcza odrzuca pakiet i – jeśli jej bufor nie jest pusty – przystępuje do próby transmisji kolejnego pakietu.

**Metoda dostępu do łącza: A6e**
Protokół CSMA (ang. *Carrier Sense Multiple Access*) z wymuszaniem transmisji z prawdopodobieństwem 1 (ang. *1-persistent*) – po wygenerowaniu nowego pakietu, stacja nadawcza sprawdza zajętość kanału transmisyjnego (nasłuchiwanie kanału odbywa się co 0.5 ms). Jeśli kanał jest wolny przez okres dłuższy niż czas DIFS, to stacja podejmuje próbę przesłania swojego pakietu. W przypadku retransmisji, stacja nadawcza sprawdza stan kanału po losowym czasie CRP równym R*CTPk, gdzie R jest losową liczbą z przedziału od <0, (2^r -1)>, a *r* jest numerem aktualnej retransmisji (przy każdej retransmisji czas ten jest losowany ponownie). Wówczas uruchamiana jest taka sama procedura jak w przypadku transmisji pierwotnej (jeśli od tego momentu kanał pozostaje wolny przez czas DIFS, to po czasie DIFS pakiet jest retransmitowany). DIFS = 5 ms.

## 2. Założenia:

Za pomocą symulacji wyznacz:

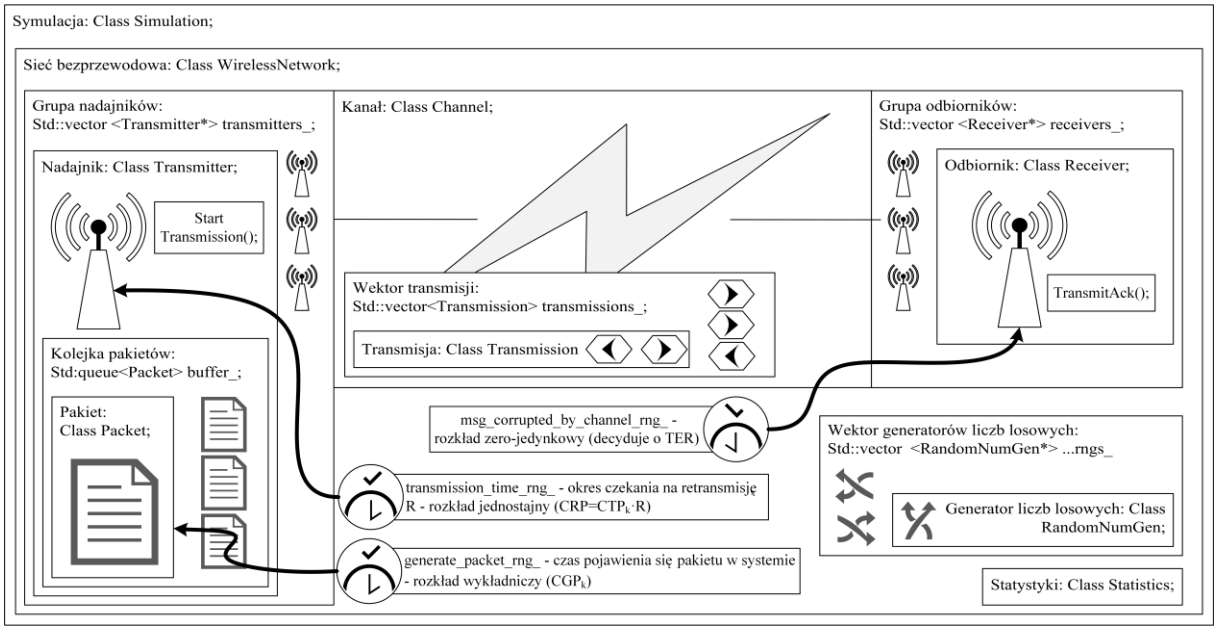
- Wartość parametru L, która zapewni średnią pakietową stopę błędów (uśrednioną po *K* odbiornikach) nie większą niż 0.1, a następnie:
- Pakietową stopę błędów w każdym z odbiorników mierzoną jako iloraz liczby pakietów straconych do liczby przesłanych pakietów,
- Średnią liczbę retransmisji pakietów,
- Przepływność systemu mierzoną liczbą poprawnie odebranych pakietów w jednostce czasu,
- Średnie opóźnienie pakietu, tzn. czas jaki upływa między pojawieniem się pakietu w buforze, a jego poprawnym odebraniem,
- Średni czas oczekiwania, tzn. czas między pojawieniem się pakietu w buforze, a jego opuszczeniem
- Sporządź wykres zależności średniej liczby retransmisji pakietów od parametru P
- Sporządź wykres zależności przepływności systemu oraz średniej i maksymalnej pakietowej stopy błędów w zależności od wartości L.

## 3. Opis modelu symulacyjnego:

Model symulacyjny został zestawiony w sposób możliwie odwzorowujący rzeczywisty przypadek. Cały model jest zawarty w klasie Simulation, która mieści w sobie odwzorowanie fizyczne sieci bezprzewodowej – Wireless Network, jak i dodatkowy obiekt klasy pomocniczej do zbierania statystyk systemu – kl. Statistics oraz wektory instancji generatorów liczb losowych – klasy RandomNumGenerator.

W skład sieci bezprzewodowej wchodzą pary nadajnik-odbiornik, które są ze sobą sprzężone. Wewnątrz każdego układu nadawczego znajduje się bufor – kolejka w której są przechowywane pakiety do wysłania. Pakiety reprezentowane są za pomocą klasy Packet, która posiada pola do przechowania momentów czasowych ich generacji. Wszystkie obiekty TX-RX współdzielą wspólny kanał w którym mogą być umieszczane transmisje. W przypadku rozpoczęcia nadawania pakietu lub potwierdzenia ACK, tworzone są obiekty transmisji, które są umieszczane w kanale. Kierunek oraz para biorąca udział w komunikacji są ustalane na podstawie flag w obiekcie transmisji. Stan łącza jest aktualizowany automatycznie i w zależności od liczby transmisji w kanale ustawiane są flagi wystąpienia kolizji lub niezakłóconego przekazu. Nadajniki i odbiorniki nie przechowują parametrów zakończonych transmisji, lecz w przypadku takiej operacji przekazują sformatowane dane do obiektu statystyk. W modelu obecnych jest kilka grup generatorów losowych. Każdy nadajnik posiada indywidualny generator momentów czasowych generacji pakietów $CGP_k$ o rozkładzie wykładniczym i intensywności lambda $\lambda$. W przypadku niepowodzenia transmisji każdy nadajnik posiada indywidualny generator mnożnika czasu uśpienia R o rozkładzie równomiernym i wartościach w zakresie $< 0; 2^{n\_retransmisji} - 1 >$. Ostatnim generatorem występującym w systemie jest generator o rozkładzie zero-jedynkowym z prawdopodobieństwie sukcesu $P$, który decyduje o tym czy transmisja nie została zakłócona przez kanał.

## 4. Schemat modelu symulacyjnego:



## 5. Opis klas systemu i ich atrybutów:

| Obiekt | Opis | Atrybuty |
|---|---|---|
| Symulacja<br>Implementacja<br>Simulation | Klasa nadrzędna w której mieszczą się wszystkie obiekty biorące udział w danej symulacji. Jest to klasa kontrolująca przeprowadzenie symulacji zgodnie z metodą: przeglądania działań. Organizuje i nadzoruje cykle przeglądania działań, steruje zegarem systemu. | • **`const size_t kInitialPhaseTimeLength_`** - czas trwania fazy początkowej<br>• **`double kPacketRngIntensity_`** - intensywność lambda generatora pakietów<br>• **`double kChannelProbabilityOfSuccess_`**- czas trwania fazy początkowej<br>• **`int minimal_time_step_`** - minimalny kwant czasowy symulacji (0,1ms)<br>• **`int channel_check_interval_`** - odstęp czasowy sprawdzania stanu kanału (5j = 0,5ms)<br>• **`bool was_event_triggered_`** - flaga obsługi zdarzenia<br>• **`size_t target_sim_time_`** - wymagana długość trwania symulacji<br>• **`size_t clock_`** - główny zegar symulacji<br>• **`WirelessNetwork* wireless_network_`** - wskaźnik na symulowaną sieć bezprzewodową<br>• **`spdlog::logger* logger_`** - wskaźnik do loggera wyświetlającego dane<br>• **`int simulation_run_index_`** - indeks symulacji<br><br>Wartości docelowe, których osiągniecie może wcześniej zakończyć symulację:<br>• **`float target_avg_packet_error_rate_`** - docelowa wartość pakietowej stopy błędów<br>• **`float target_avg_retransmission_count_`** - docelowa wartość średniej liczby retransmisji<br>• **`int target_total_received_packets_count_`** - docelowa liczba poprawnie odebranych pakietów<br>• **`int target_total_lost_packets_count_`** - docelowa liczba straconych pakietów<br>• Interfejs do zapisu danych fazy początkowej:<br>• **`bool save_initial_phase_data_`** - flaga decydująca o zapisie danych fazy początkowej do pliku<br>• **`std::string init_phase_data_filename_`** - nazwa pliku z zapisanymi danymi początkowymi<br>•<br>Wskaźniki na obiekty zdarzeń:<br>• **`CollisionEvent* collision_event_`** - wskaźnik do zdarzenia kolizji w kanale<br>• **`ChannelBusyEvent* channel_busy_event_`** - wskaźnik do zdarzenia zajętości kanału<br>• **`ChannelEmptyEvent* channel_empty_event_`** - wskaźnik do zdarzenia niezajętości kanału<br>• **`UpdateChannelStateEvent* update_channel_state_event_`** - wskaźnik do zdarzenia aktualizacji stanu kanału<br>• **`std::vector<Event*> generate_packet_events_`** - wektor wskaźników na zdarzenia generacji nowego pakietu |

| | | |
|---|---|---|
| | | • **std::vector<Event*> end_ack_listening_events_** - wektor wskaźników na zdarzenia zakończenia nasłuchiwania potwierdzenia ACK<br>• **std::vector<Event*> end_ack_transmission_events_** - wektor wskaźników na zdarzenia zakończenia transmisji ACK<br>• **std::vector<Event*> end_packet_transmission_events_** - wektor wskaźników na zdarzenia zakończenia transmisji pakietu<br>• **std::vector <StartRetransmissionEvent*> start_retransmission_events_** -wektor wskaźnikó na zdarzenia rozpoczęcia retransmisji pakietu<br>• **std::vector <StartTransmissionEvent*> start_transmission_events_** - wektor wskaźników na zdarzenia rozpoczęcia transmisji pakietu<br>• **std::vector <TransmitAckEvent*> transmit_ack_events_** - wektor wskaźników na zdarzenia rozpoczęcia transmisji ACK<br>• **std::vector <SuccessfulPacketTransmissionEvent*> successful_packet_transmission_events_** - wektor wskaźników na zdarzenia poprawnej transmisji pakietu<br>• **std::vector <RemoveLostPacketEvent*> remove_lost_packet_events_** - wektor wskaźników na zdarzenia usunięcia pakietu po LR retransmisji<br>• **std::vector <ScheduleRetransmissionEvent*> schedule_retransmission_events_** - wektor wskaźników na zdarzenia uśpienia nadajnika po nieudanej transmisji pakietu<br>• **std::vector<Event*> end_tx_sleep_events_** - wektor wskaźników na zdarzenia zakończenia uśpienia po nieudanej transmisji<br>• **std::vector <StartChannelObservingEvent*> start_channel_observing_events_** - wektor wskaźników na zdarzenia rozpoczęcia obserwacji stanu kanału<br>• **std::vector <ProbeChannelStateEvent*> probe_channel_state_events_** - wektor wskaźników na zdarzenia próbkowania stanu kanału<br>• **std::vector <MarkEmptyChannelStateEvent*> mark_empty_channel_state_events_** - wektor wskaźników na zdarzenia zaznaczenia wolnego stanu kanału i rozpoczęcia regularnego sprawdzania stanu co 0,5ms<br>• **std::vector <ReduceTXDownCounterEvent*> reduce_tx_down_counter_events_** - wektor wskaźników na zdarzenia redukcji licznika stanu niezajętości po sprawdzeniu wolnego stanu kanału<br>• **std::vector <ResetTXDownCounterEvent*> reset_tx_down_counter_events_** - wektor wskaźników na zdarzenia resetowania licznika czasu nie zajętości do wartości domyślnej po stwierdzeniu zajętości kanału |

| | | |
|---|---|---|
| | | Generatory pseudolosowe:<br>• **std::vector <RandomNumGen*> packet_generation_rngs_** - wektor wskaźników na generatory czasu pojawienia się pakietów w systemie<br>• **std::vector <RandomNumGen*> packet_transmit_time_rngs_** - wektor wskaźników na generatory czasu trwania transmisji<br>• **std::vector <RandomNumGen*> sleep_time_rngs_** - wektor wskaźników na generatory mnożnika czasu uśpienia nadajnika<br>• **RandomNumGen* msg_corrupted_by_channel_rng_** - wskaźnik na generator flagi nie zniekształcenia transmisji przez kanał |
| Obiekt<br>Statystyki (Wyniki symulacji)<br>Implementacja<br>Statistics | Klasa posiadająca pola używane do przechowywania i aktualizacji wyników symulacji (parametrów sieci). | • **Simulation* simulation_ptr_** - wskaźnik na instancję symulacji<br>• **std::vector<int> successfully_received_packets_vector_** - wektor liczby poprawnie przesłanych pakietów (dla każdej pary TX-RX)<br>• **std::vector<int> lost_packets_vector_** - wektor liczby straconych pakietów (dla każdej pary TX-RX)<br>• **size_t tot_retran_count_** - licznik całkowitej liczby retransmisji<br>• **size_t tot_packet_await_time_** - licznik całkowitego czasu oczekiwania pakietów<br>• **size_t tot_packet_delay_** - licznik całkowitego czasu opóźnienia pakietów |
| Obiekt<br>Sieć bezprze-wodowa<br>Implementacja<br>Wireless Network | Klasa gromadząca wszystkie elementy sieci działające na tym samym obszarze w obrębie jednego kanału. | • **const int kRxTxCount_** - liczba par nadajników i odbiorników w systemie<br>• **const int kAckMsgTime_** - długość trwania transmisji potwierdzenia ACK (10j = 1ms)<br>• **const int kTxInitDownCounterVal_** - początkowa wartość licznika czasu niezajetości (55j = 5,5ms)<br>• **std::vector <Transmitter*> transmitters_** - wektor wskaźników na obiekty nadajników<br>• **std::vector <Receiver*> receivers_** - wektor wskaźników na obiekty odbiorników<br>• **Channel * channel_** - wskaźnik na obiekt kanału radiowego<br>• **Statistics* statistics_** - wskaźnik na obiekt do zbierania statystyk |
| Obiekt<br>Kanał Radiowy (Łącze) | Kanał przechowuje na czas transmisji wszelkie wiadomości, które zostały wysłane przez stacje nadawcze lub odbiorcze. Na podstawie zawartości | • **bool is_update_enabled_** - flaga zezwalająca na aktualizację stanu kanału<br>• **bool is_channel_busy_** - flaga zajętości kanału<br>• **bool was_transmit_error_** - flaga wystąpienia kolizji oznaczająca błąd w przekazie |

| | | |
|---|---|---|
| Implementacja<br>Channel | i stanu flag (historii) tego wektora podejmowana jest decyzja o wolnym stanie, zajętości lub wystąpieniu błędu w kanale. Po upływie czasu trwania wiadomości jest ona usuwana z kanału. | • **std::vector <Transmission*> transmissions_** - wektor wskaźników na transmisje obecne w kanale<br><br>Kanał jest wolny gdy brak wskaźnika na obiekt transmisji w wektorze - flagi są wyzerowane. Kanał jest zajęty gdy występuje jeden obiekt transmisji i flaga błędu jest wyzerowana. Kanał jest zablokowany przez kolizję gdy pojawiły więcej niż dwie transmisje. Transmisje są zaznaczane jako błędne (flaga transmit_error) do czasu opróżnienia wektora i nie odnoszą skutku. |
| Obiekt<br>Nadajnik<br>Implementacja<br>Transmitter | Klasa reprezentująca stację nadawczą, generuje pakiety gotowe do transmisji w odstępach czasu $CGP_k$, w przypadku dostępnego łącza dokonuje próby transmisji. Obiekt jest odpowiedzialny za realizację protokołu transmisji według algorytmu i przechowywania danych z nią związanych. Posiada mechanizm obserwujący zajętość kanału i wyzwalający możliwość nadawania po upłynięciu czasu DIFFS w przypadku pierwotnej transmisji lub R*CTPk w przypadku retransmisji Zawiera liczniki średniego czasu pakietów w kolejce oraz ilości retransmisji aktualizowane po usunięciu pakietu z kolejki. Generuje liczby losowe dotyczące czasu próby retransmisji na podstawie numeru retransmisji - r. | • **const int kTxId_** - id nadajnika<br>• **const int kMaxRetranCount_** - maksymalna liczba retransmisji<br>• **const int kTxInitDownCounterVal_**- początkowa wartość lokalnego licznika czasu niezajetości<br>• **std::queue <Packet*> buffer_**- kolejka wskaźników na wygenerowane pakiety<br>• **int transmission_time_** - zmienna pamiętająca czas trwania ostatniej próby transmisji<br>• **int curr_cln_time_down_counter_**- licznik bieżącego czasu niezajętości kanału<br>• **int curr_retran_num_** - licznik bieżącego numeru retransmisji<br>• **int wait_time_multpl_** - mnożnik czasu uśpienia nadajnika<br>• **bool was_tran_ok_**- flaga oznaczająca otrzymanie ack – poprawną transmisję<br>• **bool is_busy_**- flaga zajętości nadajnika, oznaczająca nadawanie<br>• **Receiver* coupled_receiver_** - wskaźnik na sprzężony z instancją odbiornik<br>• **bool was_channel_empty_** - flaga pamiętająca ostatni stan kanału<br>• **bool is_listening_finished_** - flaga oznaczająca koniec nasłuchiwania ACK i wyzwalająca odpowiednie zdarzenia<br>• **bool is_active_**- flaga aktywności nadajnika – jej brak oznacza uśpienie po nieudanej transmisji<br>• **bool is_probing_active_**- flaga zezwalająca na sprawdzenie stanu kanału (co 0,5ms)<br>• **bool is_channel_observed_** - flaga oznajmiająca czy kanał jest obserwowany przez nadajnik<br>• **size_t transmit_timestamp_** - zmienna ustalana przy pierwszej transmisji kanału, używana w obliczeniu czasu oczekiwania pakietu |
| Obiekt<br>Odbiornik<br>Implementacja | Klasa reprezentująca stację odbiorczą, odpowiedzialna za przechwytywanie danych przesyłanych przez kanał i informowanie o pomyślnej transmisji. | • **const int kRxId_** - id odbiornika<br>• **bool was_packet_received_** - flaga oznaczając odbiór transmisji pakietu (może być uszkodzona)<br>• **bool was_packet_uncorrupted_** - flaga brak uszkodzenia transmisji przez kolizję |

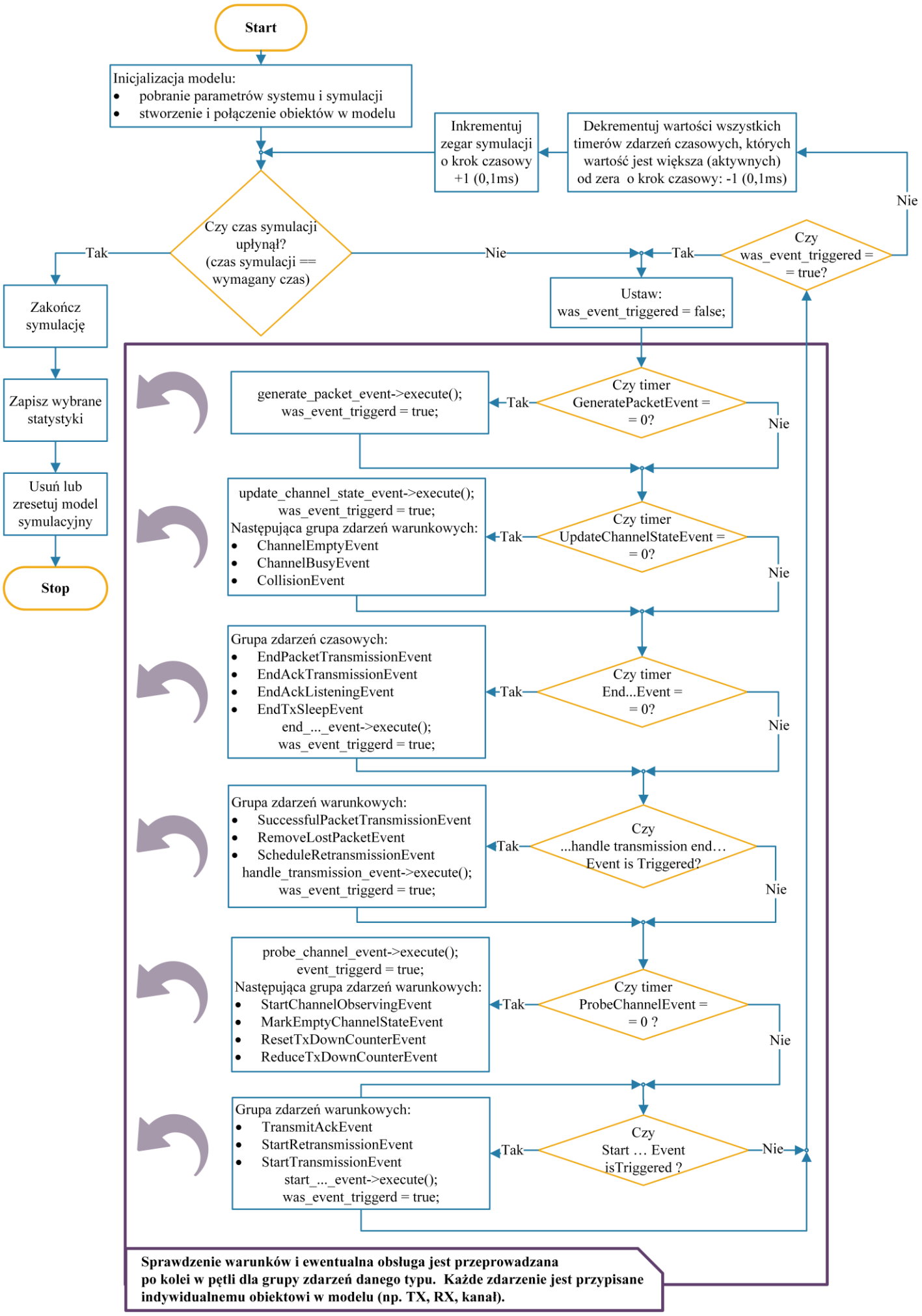
| | | |
|---|---|---|
| Receiver | | • **`bool was_uncorrupted_by_channel_`** - flaga oznaczająca brak zniekształcenia transmisji przez kanał<br>• **`Transmitter* coupled_transmitter_`** - wskaźnik na sprzężony nadajnik |
| Obiekt<br>Pakiet<br>Implementacja<br>Packet | Klasa przedstawiająca pojedyncze pakiety gotowe do transmisji. Pakiety są ustawione w kolejkę FIFO w nadajniku. Każdy pakiet zmienną określającą moment czasowy w którym został wygenerowany. | • **`size_t gen_timestamp_`** - moment czasowy generacji pakietu, wykorzystywany do wyznaczania czasów oczekiwania w kolejce |
| Obiekt<br>Transmisja<br>Implementacja<br>Transmission | Klasa pomocnicza przechowująca podstawowe dane dotyczące transmisji wiadomości: pakietu lub potwierdzenia ACK. Zawiera podstawowe informacje użyteczne do identyfikacji i obsługi stanu kanału.<br><br>Dodana w celu usprawnienia realizacji wykrywania i obsługi procesu kolizji lub zajętości oraz identyfikacji stacji urządzeń docelowych | • **`int tran_time_`** - czas trwania transmisji<br>• **`bool is_packet_`** - flaga oznaczająca typ transmisji (0 – ACK)<br>• **`Transmitter* transmitter_`** - wskaźnik na nadajnik, w zależności od kierunku nadawca lub odbiorca<br>• **`Receiver* receiver_`**- wskaźnik na odbiornik, w zależności od kierunku nadawca lub odbiorca |

## 6. Metoda symulacyjna:

Symulacja została zrealizowana na podstawie metody przeglądania działań, która polega na iteracyjnym sprawdzeniu czy dane zdarzenie czasowe lub warunkowe może zostać wykonane. W przypadku obsługi zdarzenia ustawiana jest flaga, która powoduje ponowny przegląd wszystkich zdarzeń w pętli i ich ewentualne wykonanie, aż do momentu gdy w danej chwili czasowej nie zachodzi już żadne zdarzenie. Wtedy zegar symulacji jest inkrementowany i sytuacja się powtarza aż do spełnienia kryteriów zakończenia.

## 7. Schemat blokowy pętli symulacyjnej:

Ze względu na dużą liczbę zdarzeń zostały one pogrupowane w bloki funkcjonalne dla przejrzystości. W modelu i implementacji zostały wyeliminowane zdarzenia zagnieżdżone - hybrydy czasowo-warunkowe. Mimo, że zdarzenia są przedstawione w jednym bloku towarzyszom im osobne warunki sprawdzające czy zdarzenia mogą zajść.


Start
Inicjalizacja modelu:
• pobranie parametrów systemu i symulacji
• stworzenie i połączenie obiektów w modelu
Inkrementuj zegar symulacji o krok czasowy +1 (0,1ms)
Dekrementuj wartości wszystkich timerów zdarzeń czasowych, których wartość jest większa (aktywnych) od zera o krok czasowy: -1 (0,1ms)
Czy czas symulacji upłynął? (czas symulacji == wymagany czas)
Tak
Nie
Zakończ symulację
Zapisz wybrane statystyki
Usuń lub zresetuj model symulacyjny
Stop
Czy was_event_triggered = = true?
Tak
Nie
Ustaw: was_event_triggered = false;
Czy timer GeneratePacketEvent = = 0?
generate_packet_event->execute();
was_event_triggerd = true;
Czy timer UpdateChannelStateEvent = = 0?
update_channel_state_event->execute();
was_event_triggerd = true;
Następująca grupa zdarzeń warunkowych:
• ChannelEmptyEvent
• ChannelBusyEvent
• CollisionEvent
Grupa zdarzeń czasowych:
• EndPacketTransmissionEvent
• EndAckTransmissionEvent
• EndAckListeningEvent
• EndTxSleepEvent
Czy timer End...Event = = 0?
end_..._event->execute();
was_event_triggerd = true;
Grupa zdarzeń warunkowych:
• SuccessfulPacketTransmissionEvent
• RemoveLostPacketEvent
• ScheduleRetransmissionEvent
Czy ...handle transmission end… Event is Triggered?
handle_transmission_event->execute();
was_event_triggerd = true;
Czy timer ProbeChannelEvent = = 0 ?
probe_channel_event->execute();
event_triggerd = true;
Następująca grupa zdarzeń warunkowych:
• StartChannelObservingEvent
• MarkEmptyChannelStateEvent
• ResetTxDownCounterEvent
• ReduceTxDownCounterEvent
Grupa zdarzeń warunkowych:
• TransmitAckEvent
• StartRetransmissionEvent
• StartTransmissionEvent
Czy Start … Event isTriggered ?
start_..._event->execute();
was_event_triggerd = true;
Sprawdzenie warunków i ewentualna obsługa jest przeprowadzana po kolei w pętli dla grupy zdarzeń danego typu. Każde zdarzenie jest przypisane indywidualnemu obiektowi w modelu (np. TX, RX, kanał).

## 8. Lista zdarzeń w modelu symulacyjnym:

**Zdarzenia czasowe:**

| Zdarzenie | Opis | Algorytm |
|---|---|---|
| Generacja nowego pakietu w nadajniku k | Zdarzenie o czasie generowanym przez źródło $CGP_k$ oznaczające pojawienie się nowego pakietu gotowego do wysłania. | 1. Umieść pakiet na końcu kolejki w nadajniku k z jego parametrami (momentem wygenerowania)<br>2. Zaplanuj kolejne zdarzenie: wylosuj czas kolejnego zdarzenia za pomocą generatora wykładniczego i ustaw czas zdarzenia na wylosowaną wartość |
| Aktualizacja stanu kanału | Zdarzenie ustawiające flagę aktualizacji kanału, regularnie wykonywane co najmniejszy krok czasowy symulacji. | 1. Ustaw flagę **`is_update_enabled = true;`**<br>2. Zaplanuj kolejne zdarzenie: ustaw czas zdarzenia na najmniejszy kwant czasu symulacji (1 jednostkę = 0,1ms) |
| Próbkowanie stanu kanału przez nadajnik | Zdarzenie odpowiedzialne za ustawienie flagi, która pozwala nadajnikowi spróbkować stan kanału i w wykonać odpowiednie operacje z lokalnym licznikiem czasu niezajetości. Zdarzenie wspomagające realizację regularnego sprawdzania kanału co 0,5ms (5 jednostek) w przypadku obecności pakietu do wysłania. | 1. Ustaw flagę w nadajniku: **`is_probing_active = true;`**<br>2. Zaplanuj kolejne zdarzenie: ustaw czas zdarzenia na okres sprawdzania kanału przez TX (0,5ms = 5 jednostek) |
| Koniec trwania transmisji pakietu | Zdarzenie oznaczające koniec obecności transmisji pakietu w kanale radiowym. Po zakończeniu jej trwania ustawiane są odpowiednie flagi w adresowanym odbiorniku | 1. Odczytaj flagi stanu kanału, i wylosuj wartość TER – czy nastąpił błąd w kanale za pomocą generatora z rozkładem zero-jedynkowym<br>2. Ustaw flagi statusu po zakończeniu transmisji w odbiorniku na uzyskane wartości:<br>a. **`was_packet_received`**,<br>b. **`was_packet_uncorrupted`**,<br>c. **`was_uncorrupted_by_channel`**<br>3. Usuń transmisję z kanału<br>4. Dezaktywuj zdarzenie – ustaw zegar na wartość -1 |

| | | |
|---|---|---|
| Koniec trwania transmisji ACK | Zdarzenie tożsame z zakończeniem obecności transmisji potwierdzającej odbiór pakietu w kanale radiowym. Po jej zakończeniu ustawiane są odpowiednie flagi w nadajniku, do którego transmisja była adresowana. | 1. Ustaw w adresowanym transmiterze flagę: **`was_tran_ok = true;`**<br>2. Usuń transmisję z kanału<br>3. Dezaktywuj zdarzenie – ustaw zegar na wartość -1 |
| Koniec nasłuchiwania ACK | Zdarzenie oznaczające zamknięcie okna czasowego w którym mogło zostać otrzymane potwierdzenie otrzymania pakietu. Wykonanie zdarzenia oznacza brak otrzymania ACK w wymaganym czasie. | 1. Inkrementuj licznik retransmisji TX: **`curr_retran_num++;`**<br>2. Ustaw flagę w nadajniku: **`is_listening_finished = true;`**<br>3. Dezaktywuj zdarzenie – ustaw zegar na wartość -1 |
| Koniec czasu uśpienia po nieudanej transmisji/retransmisji | Zdarzenie reprezentujące reaktywację nadajnika po czasie uśpienia spowodowanym nieudaną transmisją lub retransmisją. Ustawiane są odpowiednie flagi umożliwiające ponowną obserwację kanału. | 1. Ustaw flagę w nadajniku: **`is_active = true;`**<br>2. Dezaktywuj zdarzenie – ustaw zegar na wartość -1 |

**Zdarzenia warunkowe:**

| Zdarzenie | Opis | Algorytm |
|---|---|---|
| Kanał pusty | Zdarzenie oznaczające, że kanał jest pusty i aktualizujące jego flagi stanu | Jeśli aktualizacja stan kanału jest aktywna i wektor wiadomości w kanale jest pusty: **`is_update_enabled == true && transmissions().empty()`**<br>1. Resetuj flagi stanu kanału:<br>a. **`is_update_enabled == false;`**<br>b. **`is_channel_busy == false;`**<br>c. **`was_transmit_error == false;`**<br>2. Dezaktywuj flagę umożliwiającą aktualizację stanu kanału:<br>**`is_update_enabled == false;`** |

| | | |
|---|---|---|
| Kanał zajęty | Zdarzenie równorzędne z zajętością kanału i ustawiające odpowiednie flagi w kanale | Jeśli aktualizacja stan kanału jest aktywna i wektor wiadomości w kanale zawiera minimum jedną transmisję: **`is_update_enabled == true && transmissions().size() >= 1`**<br>1. Ustaw flagi zajętości kanału: **`is_channel_busy == true;`**<br>2. Dezaktywuj flagę umożliwiającą aktualizację stanu kanału:<br>**`is_update_enabled == false;`** |
| Kolizja w kanale | Zdarzenie tożsame z wystąpieniem kolizji w kanale. Odpowiedzialne za ustawienie flag oznaczających uszkodzenie wszystkich transmisji w kanale. | Jeśli w liczba transmisji w kanale jest większa od 1 i nie występuje flaga błędu transmisji:<br>**`transmissions().size()>1 && was_transmit_error == 0`**<br>1. Ustaw flagę błędu w kanale (kolizji) : **`was_transmit_error = true;`** |
| Rozpocznij obserwację kanału | Zdarzenie odpowiedzialne za zaplanowanie momentu rozpoczęcia próbkowania kanału w przypadku gdy jest dostępny pakiet do wysłania i nadajnik jest aktywny. | Jeśli kanał nie jest obserwowany, nadajnik jest aktywny i niezajęty transmisją oraz w buforze znajduje się pakiet do wysłania: **`is_channel_observed == false && is_active == true && is_busy == false && buffer().empty() == false`**<br>1. Ustaw czas zdarzenia „Próbkowanie stanu kanału przez nadajnik” na najbliższą chwilę czasową (1 jednostkę czasu = 0,1ms)<br>2. Ustaw flagę obserwacji kanału: **`is_channel_observed = true;`** |
| Zaznacz pusty stan kanału | Zdarzenie oznaczające zapamiętanie początkowego stanu kanału w celu poprawnego obliczenia wymaganego czasu niezajetości kanału. | Jeśli flaga próbkowania kanału jest aktywna i flaga ostatniego stanu kanału w nadajniku wskazuje, że kanał był zajęty oraz aktualnie kanał jest wolny:<br>**`is_probing_active == true && was_channel_empty == false && is_channel_busy == false`**<br>1. Ustaw flagę pamięci stanu kanału: **`was_channel_empty == true;`**<br>2. Ustaw flagę próbkowania kanału na nieaktywną: **`is_probing_active == false`** |
| Zresetuj licznik czasu niezajętości | Zdarzenie odpowiadające za zresetowanie licznika czasu niezajetości oraz flag pamięci stanu kanału w nadajniku w przypadku gdy kanał jest zajęty przez inną transmisję. | Jeśli flaga próbkowania jest aktywna, transmiter nie jest zajęty ani uśpiony oraz kanał jest w stanie zajętości: **`is_probing_active == true && is_active == true && is_busy == false && is_channel_busy == true`**<br>1. Zresetuj flagę pamiętającą stan kanału w nadajniku: **`swas_channel_empty == false;`**<br>2. Ustaw flagę próbkowania kanału jako nieaktywną: **`is_probing_active == false`** |

| | | |
|---|---|---|
| Zredukuj licznik czasu niezajętości | Zdarzenie odpowiedzialne za zredukowanie wymaganego licznika czasu niezajetości o regularny odstęp czasowy sprawdzenia kanału w przypadku gdy kanał jest wolny i w ostatnim momencie sprawdzenia jego stan również taki był ( 0,5ms = 5 jednostek) | Jeśli flaga próbkowania jest aktywna, transmiter nie jest zajęty ani uśpiony oraz kanał jest wolny: **`is_probing_active == true && is_active == true && is_busy == false && is_channel_busy == false`**<br>1. Redukuj licznik czasu niezajętości o odstęp sprawdzania kanału:<br>**`curr_cln_time_down_counter_ -= channel_check_interval; (5j)`**<br>2. Ustaw flagę próbkowania kanału jako nieaktywną: **`is_probing_active == false;`** |
| Udana transmisja pakietu | Zdarzenie równoważne z obsługą następującą po otrzymaniu potwierdzenia poprawnej transmisji ACK od odbiornika. Zapisane zostają odpowiednie statystyki, pakiet zostaje usunięty oraz następuje reset flag i pozostałych parametrów nadajnika w celu organizacji następnych prób transmisji. | Jeśli w nadajniku są ustawione flagi poprawnego odebrania oraz stanu zajętego oczekiwaniem nadajnika: **`was_tran_ok == true && is_busy == true`**<br>1. Jeśli faza początkowa minęła oblicz i zapisz wybrane statystyki dotyczące transmisji: czas oczekiwania pakietu, opóźnienie pakietu oraz średnią liczbę retransmisji. Inkrementuj liczbę poprawnie przesłanych pakietów dla nadajnika K w statystykach.<br>2. Usuń pierwszy pakiet z kolejki<br>3. Zresetuj wszystkie flagi i liczniki parametrów transmisji nadajnika do wartości domyślnych (numer retransmisji, czasy pierwszej transmisji, flagi niezajętości kanału i liczniki czasu wolnego kanału, itp.) Nadajnik ponownie od początku musi rozpocząć obserwację kanału.<br>4. Dezaktywuj powiązane zdarzenie „Koniec Nasłuchiwania ACK”, ustaw czas zdarzenia na -1 |
| Zaplanuj retransmisję pakietu | Zdarzenie występujące w przypadku gdy nie powiodła się transmisja oraz istnieje możliwość retransmisji – liczniki prób nie przekroczyły wartości dopuszczalnej. Nadajnik zostaje dezaktywowany na określony przez losowanie czas. Po upływie tego czasu wznawia normalne funkcjonowanie. | Jeśli flaga oznaczająca koniec nasłuchiwania ACK jest aktywna oraz flaga poprawnej transmisji nie jest zaznaczona oraz numer retransmisji pakietu nie jest większy od dopuszczalnej wartości: **`is_listening_finished == true && was_tran_ok == false && curr_retran_num_ < kMaxRetranCount`**<br>1. Wygeneruj mnożnik czasu uśpienia korzystając z równomiernego generator liczb losowych o zakresie: $r = < 0; 2^{n\_retransmisji} - 1 >$. Wyznacz czas uśpienia jako: $r \cdot CTP_k$<br>2. Ustal czas powiązanego zdarzenia „Koniec czasu uśpienia po nieudanej transmisji” na obliczoną wartość (jeśli czas uśpienia == 0, ustal czas uśpienia na 1 w celu ograniczenia liczby zdarzeń zachodzących w tym samym momencie czasowym)<br>3. Dezaktywuj powiązane zdarzenie „Próbkowanie stanu kanału” ustalając jego czas na -1<br>4. Zresetuj wszystkie lokalne flagi stanu kanału i liczniki obserwacji na wartość domyślną (false) |

| | | |
|---|---|---|
| | | 5. Ustal flagi aktywności nadajnika: **`is_busy == false, is_active == false;`**<br>6. Dezaktywuj flagę skończenia nasłuchiwania ACK:<br>**`is_listening_finished == false`** |
| Usuń stracony pakiet | Zdarzenie oznaczające przekroczenie dopuszczalnej liczby retransmisji danego pakietu, co powoduje konieczność usunięcia straconego pakietu i reset nadajnika w celu próby przesłania pozostałych pakietów. | Jeśli flaga oznaczająca koniec nałuchiwania ACK jest oraz flaga poprawnej transmisji nie jest zaznaczona oraz numer retransmisji pakietu jest równy lub większy od dopuszczalnej wartości:<br>**`is_listening_finished == true && was_tran_ok == false && curr_retran_num_ >= kMaxRetranCount`**<br>1. Jeśli faza początkowa minęła zapisz wybrane statystyki: czas oczekiwania pakietu oraz inkrementuj licznik straconych pakietów dla danej pary TX-RX<br>2. Usuń pakiet z początku kolejki<br>3. Zresetuj wszystkie flagi i liczniki parametrów transmisji nadajnika do wartości domyślnych (numer retransmisji, czasy pierwszej transmisji, flagi niezajętości kanału i liczniki czasu wolnego kanału, itp.) Nadajnik ponownie od początku musi rozpocząć obserwację kanału. |
| Rozpocznij transmisję pakietu | Zdarzenie odpowiadające spełnieniu warunków na rozpoczęcie transmisji pakietu i tożsame z rozpoczęciem pierwszej próby przekazania pakietu przez nadajnik. Zapisywany jest moment czasowy pierwszej próby nadania w celu zbierania statystyki czasu oczekiwania pakietu. | Jeśli wartość lokalnego licznika czasu niezajętości i wartość momentu czasowego pierwszej próby wysłania są równe 0 oraz transmiter nie jest zajęty:<br>**`curr_cln_time_down_counter == 0 && transmit_timestamp == 0 && is_busy == false`**<br>1. Losuj czas trwania transmisji za pomocą generatora równomiernego w zakresie {10, 20, … 100}ms<br>2. Ustal moment czasowy pierwszej transmisji na bieżącą chwilę czasową<br>3. Stwórz transmisję z odpowiednimi parametrami i wstaw ją do kanału<br>4. Ustaw flagę zajętości w nadajniku: **`is_busy == true;`** Dezaktywuj flagi obserwacji i próbkowania kanału.<br>5. Ustal czas i wskaźniki powiązanego zdarzenia „Koniec transmisji pakietu” na wylosowany czas transmisji i dodaną transmisję<br>6. Ustal czas powiązanego zdarzenia „Koniec nasłuchiwania ACK” na: $CTP_k + CTIZ + 1j(min\ kwant\ czasu)$<br>7. Dezaktywuj zdarzenie „Próbkowanie stanu kanału przez nadajnik” – ustal czas na -1 |

| | | |
|---|---|---|
| Rozpocznij transmisję pakietu | Zdarzenie odpowiadające spełnieniu warunków na rozpoczęcie retransmisji pakietu i tożsame z rozpoczęciem kolejnej próby nadawania przez nadajnik. | Jeśli wartość lokalnego licznika czasu niezajętości jest równa 0 i wartość momentu czasowego pierwszej próby wysłania jest różna od 0 oraz transmiter nie jest zajęty:<br>**`curr_cln_time_down_counter == 0 && transmit_timestamp != 0 && is_busy == false`**<br>1. Losuj czas trwania transmisji za pomocą generatora równomiernego w zakresie {10, 20, … 100}ms<br>2. Stwórz transmisję z odpowiednimi parametrami i wstaw ją do kanału<br>3. Ustaw flagę zajętości w nadajniku: **`is_busy == true;`** Dezaktywuj flagi obserwacji i próbkowania kanału.<br>4. Ustal czas i wskaźniki powiązanego zdarzenia „Koniec transmisji pakietu” na wylosowany czas transmisji i dodaną transmisję<br>5. Ustal czas powiązanego zdarzenia „Koniec nasłuchiwania ACK” na: $CTP_k + CTIZ + 1j(min\ kwant\ czasu)$<br>6. Dezaktywuj zdarzenie „Próbkowanie stanu kanału przez nadajnik” – ustal czas na -1 |
| Rozpocznij transmisję potwierdzenia ACK | Zdarzenie oznaczające poprawne odebranie transmisji pakietu przez odbiornik, który wysyła zwrotne potwierdzenie poprawnego otrzymania wiadomości – ACK. | Jeśli wszystkie flagi statusu otrzymanej transmisji są ustawione:<br>**`was_packet_received == true && was_packet_uncorrupted == true && was_uncorrupted_by_channel == true`**<br>1. Jeśli faza początkowa się skończyła inkrementuj licznik poprawnie odebranych pakietów danej pary TX-RX<br>2. Stwórz transmisję ACK z odpowiednimi wskaźnikami i czasem transmisji równym $CTIZ$ = 1ms (10 jednostek) i wstaw ją na kanał.<br>3. Ustal czas powiązanego zdarzenia „Koniec trwania transmisji ACK” na czas $CTIZ$<br>4. Resetuj lokalne flagi statusu otrzymanej transmisji do wartości domyślnej (false) |

## 9. Generatory liczb losowych:

W symulacji wykorzystane są 3 typy generatorów liczb losowych o różnych rozkładach prawdopodobieństwa losujące następujące parametry:

- $CGP_k$ – czas generacji nowego pakietu – rozkład wykładniczy o intensywności L, $\lambda$
- $CTP_k$ – czas trwania transmisji pakietu – rozkład równomierny o wartościach: {10, 20, 30, …, 100} ms
- $R$ – mnożnik czasu uśpienia po nieudanej transmisji/retransmisji – rozkład równomierny o zakresie wartości $< 0; 2^{n\_retransmisji} - 1 >$.
- $TER$ – wystąpienie błędu w kanale i uszkodzenie transmisji – rozkład zero-jedynkowy o prawdopodobieństwie sukcesu 1- P, gdzie P = 0,8.

Wymagane generatory zostały zaimplementowane w klasie RandomNumGen i stosując odpowiednie operacje przekształceń i zaokrągleń zakresy generatorów zostały dostosowane do wymagań.

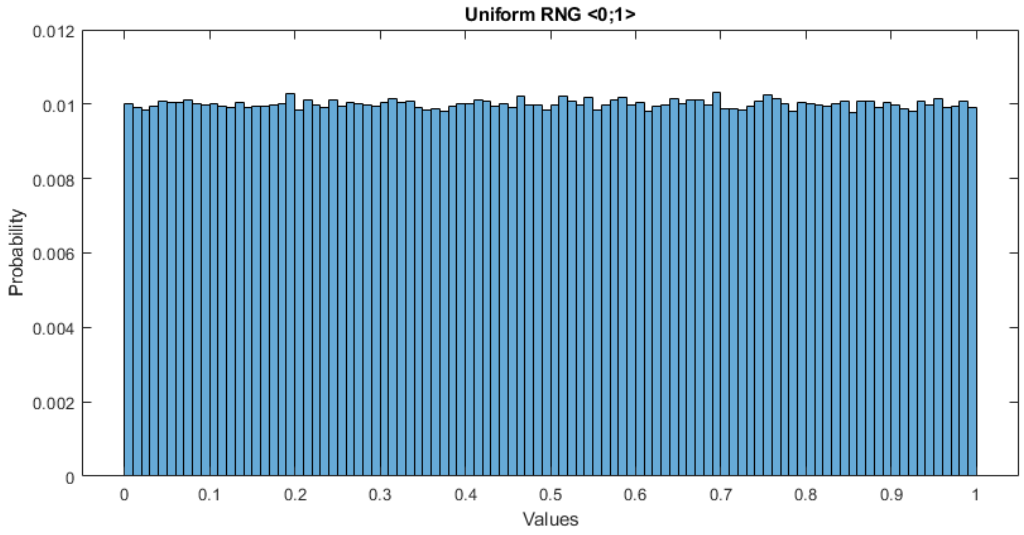
**Algorytmy generacji i zaokrąglania wartości losowych oraz histogramy generatorów:**

- Generator o rozkładzie równomiernym i zakresie: $< 0; 1 >$, funkcja $rand()$:

Wartości początkowe: **M** = 2147483647.0; **A** = 16807; **Q** = 127773; **R** = 2836;

```
1. kH = kernel_ / kQ_;
2. kernel_ = kA * (kernel_ - kQ_ * kH) - kR_ * kH;
3. if (kernel_ < 0) { kernel_ = kernel_ + static_cast<int>(kM_); }
4. return kernel_ / kM_;
```

Histogram wartości zwracanych przez generator:



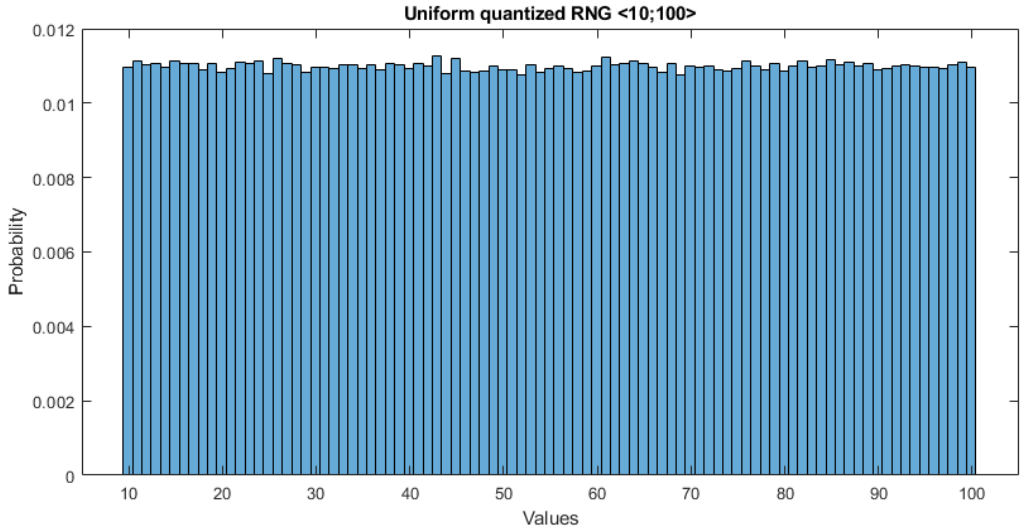

- Generator o rozkładzie równomiernym i zakresie $< min; max >$, $rand(int\ min, int\ max)$:

```
1. return rand()*(max-min)+min;
```

Dodana funkcja zaokrąglająca do wartości całkowitych: randRoundedInt( int min, int max):

- `return static_cast<int>(floor(rand()*(max - min + 1)) + min)`

Histogram wartości zwracanych przez generator (ustalony zakres 10 do 100):



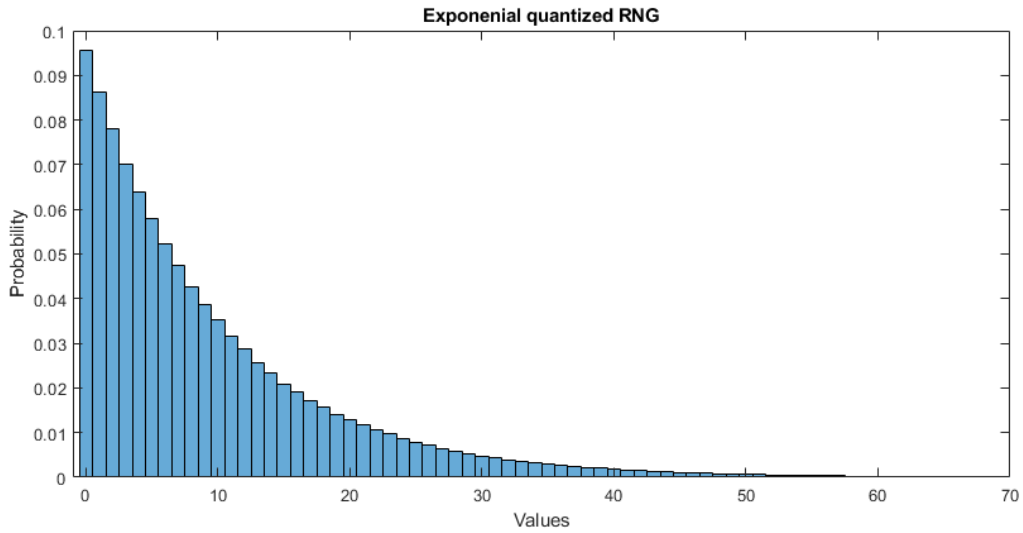

- Generator o rozkładzie wykładniczym o intensywności λ, funkcja randExp(λ):

1. `return -(1.0 / lambda) * log(rand());`

Dodana funkcja zaokrąglająca do wartości całkowitych: randExpRoundedInt():

- `return static_cast<int>(floor(-(1.0 / lambda) * log(rand())));`

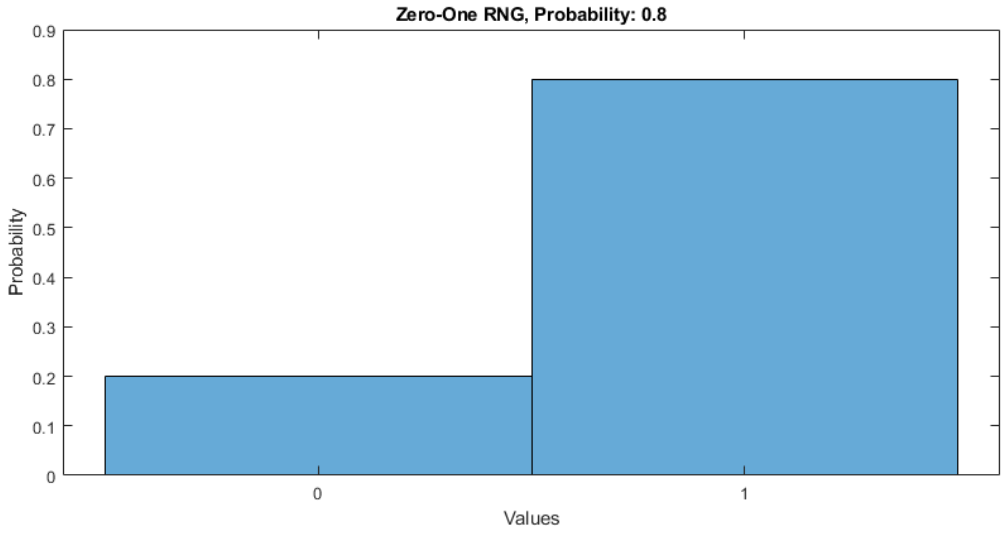
Histogram wartości zwracanych przez generator dla λ = 0,005:




- Generator o rozkładzie zero-jedynkowym z prawdopodobieństwem sukcesu P, randZeroOne(P):

1. `if(rand() < probability) {return true;}`
2. `else {return false;}`

Histogram wartości zwracanych przez generator dla P = 0,8:




W projekcie przewidziane zostało 10 symulacji dla sukcesywnie inkrementowanych wartości parametru lambda. Każda z symulacji była wykonywania 15 krotnie z różnymi ziarnami. Dla jednego przebiegu symulacji wymagana jest następująca liczba ziaren: $3 \cdot K_{TX\&RX} + 1$ (w każdym nadajniku potrzebne jest indywidualne ziarno dla generatora momentu pojawienia się pakietu, czasu transmisji, czasu uśpienia, oraz jedno ziarno dla całego kanału transmisyjnego decydujące o błędzie wprowadzonym przez kanał). Dla 16 nadajników i odbiorników dla jednego przebiegu symulacyjnego wymagane jest 49 ziaren. W sumie dla jednego całego przebiegu ewaluującego wartość parametru lambda wymagane było $10 \cdot 15 \cdot 49 = 7350$ wartości ziaren.

**Metoda generacji i przechowywania ziaren:**

- Ziarna generatorów losowych są generowane na podstawie pierwotnej wartości wprowadzonej przez użytkownika w jednej pętli. Między kolejnymi ziarnami występuje odstęp 100000 losowań.
- Po wygenerowaniu odpowiedniej liczby dla jednego danego przebiegu symulacji wartości są zapisywane w jednej linijce do pliku, oddzielone średnikami. (N linijek = N uruchomień symulacji)
- W przypadku uruchomienia symulacji pobierane są ziarna z N-tej linijki pliku tekstowego, następnie odczytane wartości są przekładane do wektora i w stałej kolejności ustawiane w odpowiednich instancjach generatorów liczb losowych.

## 10. Parametry wywołania symulacji:

Na podstawie początkowych analiz koniecznie okazało się zmodyfikowanie parametrów symulowanego systemu za zgodą prowadzącego.

Przyczyny i uzasadnienie wprowadzonych modyfikacji:

- Redukcja liczby dopuszczalnych retransmisji z 15 na 6 – ze względu na bardzo duże czasy uśpienia które były losowane według formuły $CTP_k \cdot URNG < 0; 2^{n\_retransmisji} - 1 >$ pakietowa stopa błędów była na niskim poziomie, rzędu tysięcznych części. Zwiększenie liczby nadajników nie powodowało odpowiednio proporcjonalnego wzrostu błędów i wymagało ekstremalnie długich czasów stabilizacji.
- Zwiększenie liczby par nadajnik-odbiornik z 4 do 16. W celu zapewnienia odpowiednio dużego zakresu pakietowej stopy błędów dla właściwej analizy i wywołania odpowiednio dużej liczby kolizji i strat pakietów.

Po obserwacji odpowiedzi systemu na różne wartości parametrów. Symulacja została uruchomiona z następującym zestawem parametrów:

$$N_{symulacji} = 10 \qquad T_{\min step} = 0{,}1\ [ms]$$

$$N_{powtórzeń} = 15 \qquad T_{sym} = 1600\ [s]$$

$$K_{TX-RX} = 16 \qquad T_{init\ phase} = 600\ [s]$$

$$LR_{max} = 6 \qquad DIFS = 5\ [ms]$$

$$P_{chan} = 0{,}8 \qquad Kernel_{init\ val} = 10032000$$

$$\lambda = 2{,}22 \cdot 10^{-4}\ do\ 2{,}31 \cdot 10^{-4}\ (krok\ co\ 10^{-6})$$

## 11. Testowanie poprawności działania modelu:

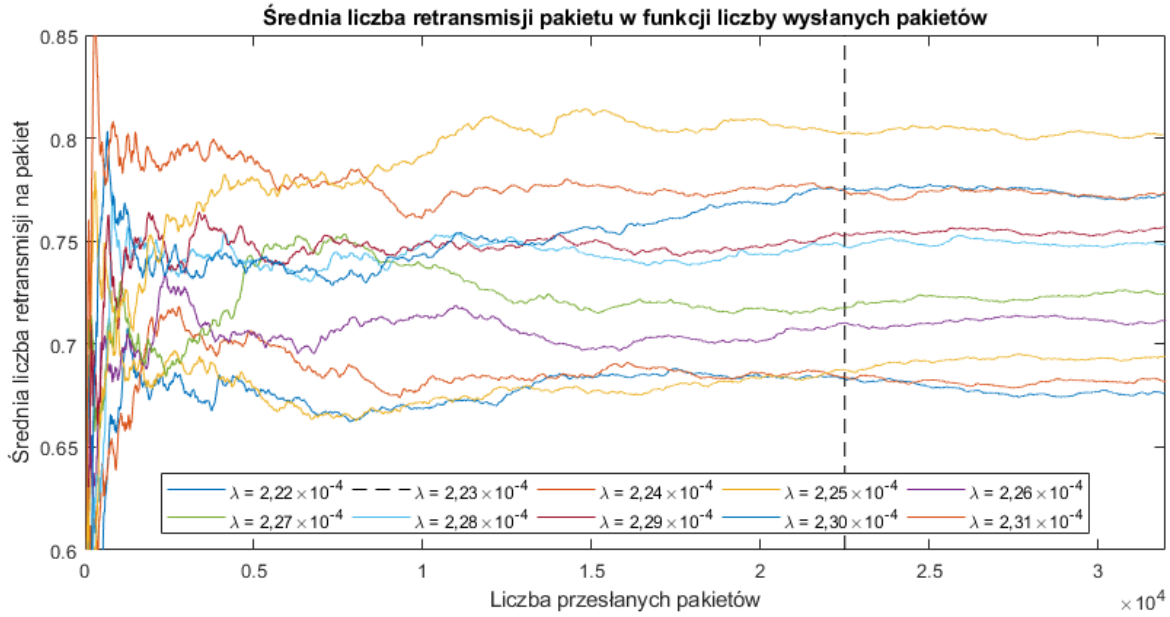
Weryfikacja poprawności działania symulacji odbywała się na kilku etapach:

- Skrupulatnej analizie krokowej poddawane były losowe przedziały czasowe symulacji w celu sprawdzenie czy zdarzenia występują odpowiednio po sobie w ustalonej sekwencji i nie występują jednocześnie zdarzenia odpowiedzialne za różne działania tego samego obiektu.
- Sprawdzone zostały wartości graniczne i reakcja systemu na ich wystąpienie. (przykładowo: usunięcie pakietu po nieudanych retransmisjach). Zweryfikowane zostały również momenty uśpienia i ponownej obserwacji kanału przez nadajniki.
- Porównanie wyników końcowych symulacji z innymi rozwiązaniami innych studentów, którzy posiadali ten sam algorytm po dostosowaniu parametrów systemu. Otrzymane wyniki zawierały się w zakresie tolerancji spowodowanym różnymi ziarnami początkowymi oraz ograniczeniem jednocześnie zachodzących zdarzeń.

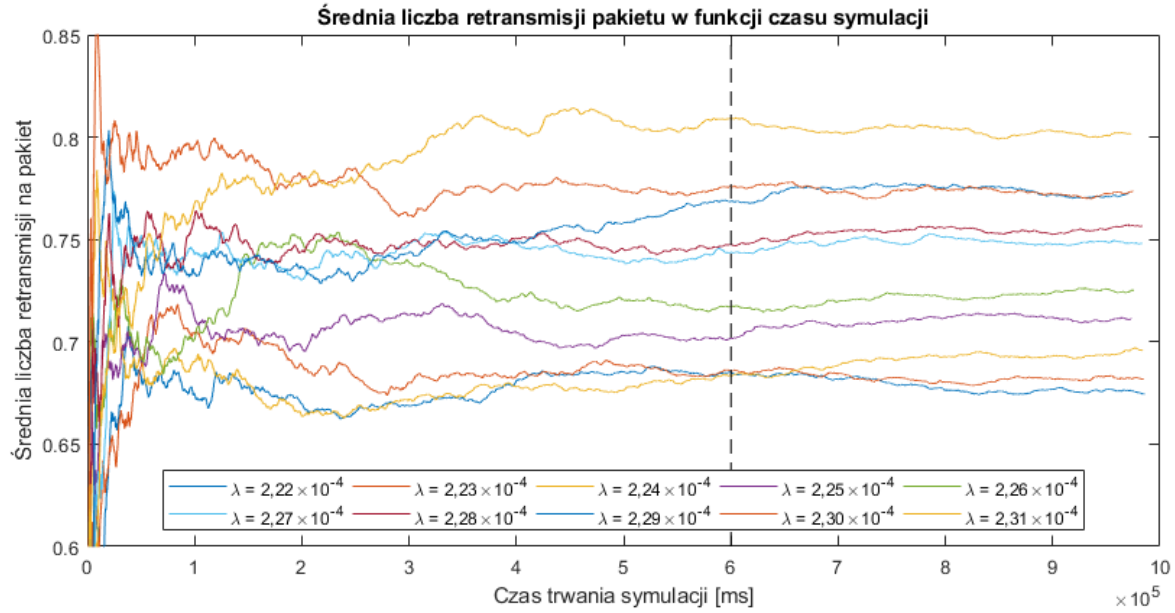
## 12. Wyznaczenie fazy początkowej symulacji:

Po wcześniejszym ustaleniu zakresu wartości parametru lambda. Zostały wykonane symulacje bez pominięcia fazy początkowej, które zapisywały wartość średniej liczby retransmisji dla każdego poprawnie odebranego pakietu. Dla każdej z 10 wartości lambda symulacja została powtórzona 15 razy, a następnie wyniki zostały uśrednione.

Wykres zależności średniej liczby retransmisji pakietu w zależności od liczby wysłanych pakietów:



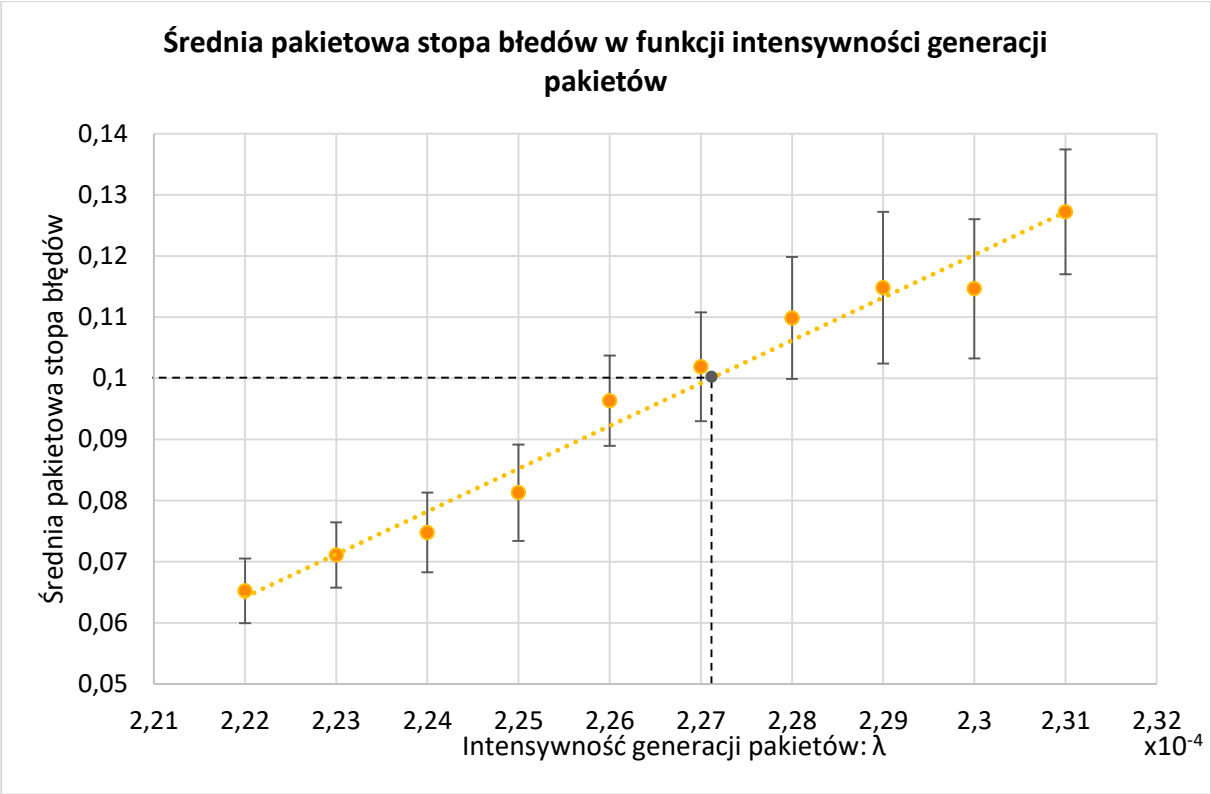

Dysponując uśrednionymi czasami wystąpienia kolejnych pakietów stworzony został również wykres zależności średniej liczby retransmisji pakietu w zależności od czasu symulacji:




Na podstawie zaobserwowanych przebiegów stabilizacji średniej liczby retransmisji pakietu, za moment względnej stabilizacji uznany został czas po wysłaniu 22500 pakietu lub moment po upływie 600 sekundy trwania symulacji.

### 13. Wyznaczenie wartości parametru lambda:

W celu wyznaczenia parametru lambda wykonane zostało 15 symulacji dla każdej z 10 wartości parametru lambda. Zebrane wartości średniej pakietowej zostały ponownie uśrednione, wyznaczone zostało odchylenie standardowe oraz przedziały ufności dla 95% z rozkładu T-Studenta.



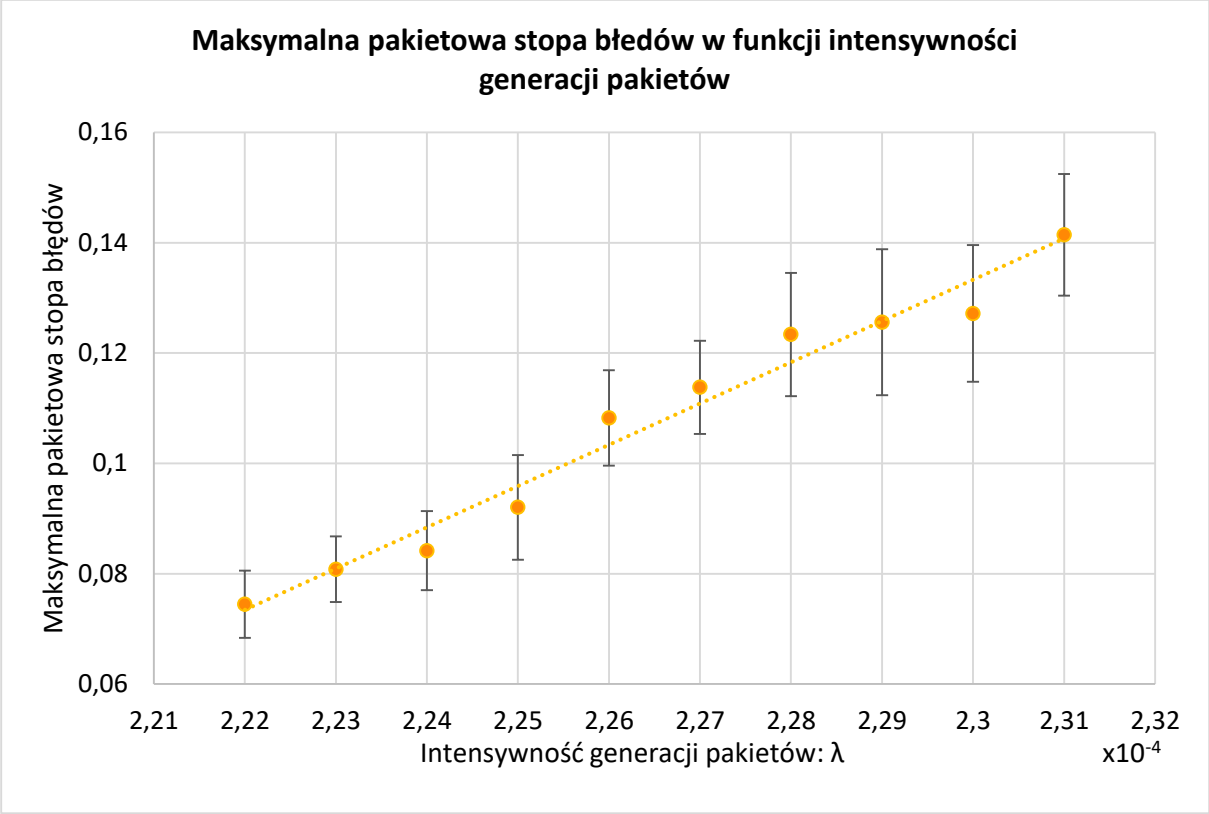

Na podstawie uzyskanej zależności zostały wykonane testy, które miały określić która z interpolacji dla zbioru danych okaże się najkorzystniejsza i dzięki której będzie można uzyskać pakietową stopę błędów na poziomie 0,1. Skuteczna okazała się interpolacja liniowa. Na jej podstawie wartość intensywności generacji pakietów została ustalona na poziomie $2{,}272 \cdot 10^{-4}$.

**14. Rezultaty serii symulacji z wyznaczonym parametrem λ:**

| Numer symulacji | Średnia pakietowa stopa błędów | Maksymalna pakietowa stopa błędów | Średnia liczba retransmisji na pakiet | Przepływność systemu [pakiet/s] | Średnie opóźnienie pakietu [ms] | Średni czas oczekiwania pakietu [ms] |
|---|---|---|---|---|---|---|
| **1** | 0,092 | 0,102 | 0,703 | 20,66 | 73,10 | 60,15 |
| **2** | 0,088 | 0,102 | 0,714 | 20,70 | 72,64 | 57,09 |
| **3** | 0,104 | 0,118 | 0,749 | 20,61 | 78,37 | 62,89 |
| **4** | 0,111 | 0,121 | 0,733 | 20,48 | 77,93 | 70,88 |
| **5** | 0,100 | 0,118 | 0,720 | 20,49 | 75,49 | 62,27 |
| **6** | 0,098 | 0,109 | 0,715 | 20,59 | 75,22 | 62,32 |
| **7** | 0,128 | 0,139 | 0,772 | 20,26 | 85,76 | 76,40 |
| **8** | 0,099 | 0,112 | 0,725 | 20,68 | 77,08 | 63,65 |
| **9** | 0,092 | 0,108 | 0,722 | 20,89 | 75,49 | 60,86 |
| **10** | 0,134 | 0,149 | 0,792 | 20,23 | 88,16 | 82,34 |
| **11** | 0,085 | 0,099 | 0,711 | 21,02 | 71,58 | 53,29 |
| **12** | 0,085 | 0,091 | 0,724 | 20,87 | 72,45 | 51,14 |
| **13** | 0,093 | 0,105 | 0,751 | 20,90 | 77,41 | 59,19 |
| **14** | 0,093 | 0,104 | 0,724 | 20,73 | 75,79 | 59,21 |
| **15** | 0,105 | 0,116 | 0,746 | 20,42 | 77,85 | 61,71 |
| **Średnia** | **0,100** | **0,113** | **0,733** | **20,63** | **76,95** | **62,89** |
| **Przedział ufności** | **±0,008** | **±0,008** | **±0,013** | **±0,12** | **±2,47** | **±4,36** |

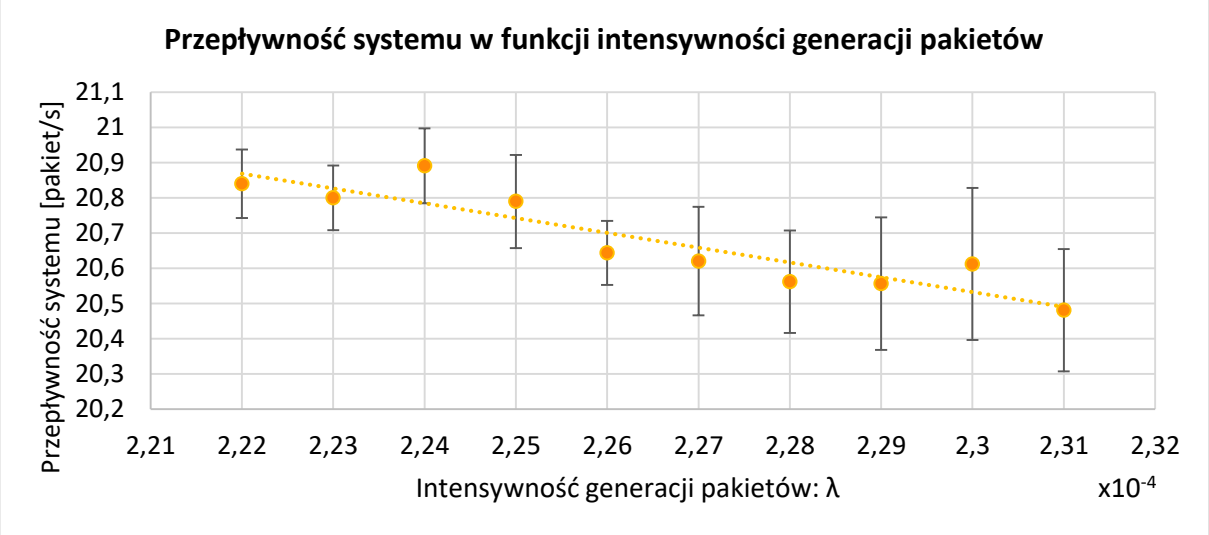
Na podstawie uśrednionych wyników widać, że stosując liniową estymatę udało się osiągnąć średnią pakietową stopę błędów na poziomie równym 0,1 dla parametru lambda określonego jako: $2{,}272 \cdot 10^{-4}$. Ten rodzaj przybliżenia może być słuszny tylko dla bardzo małych zmian wartości λ i obserwując zależność w szerszym zakresie stanie się bezużyteczny, ze względu na zmianę odpowiedzi systemu, która już nie będzie lokalnie proporcjonalna. W celu upewnienia się w poprawności obranej wartości
i zastosowanej interpolacji analiza zwiększona została liczba powtórzeń symulacji dla wybranej wartości lambda. Dla 50 powtórzeń i uśrednień średnia pakietowa stopa błędów wyniosła 0,0979.

Wykres zależności maksymalnej pakietowej stopy błędów od parametru lambda:

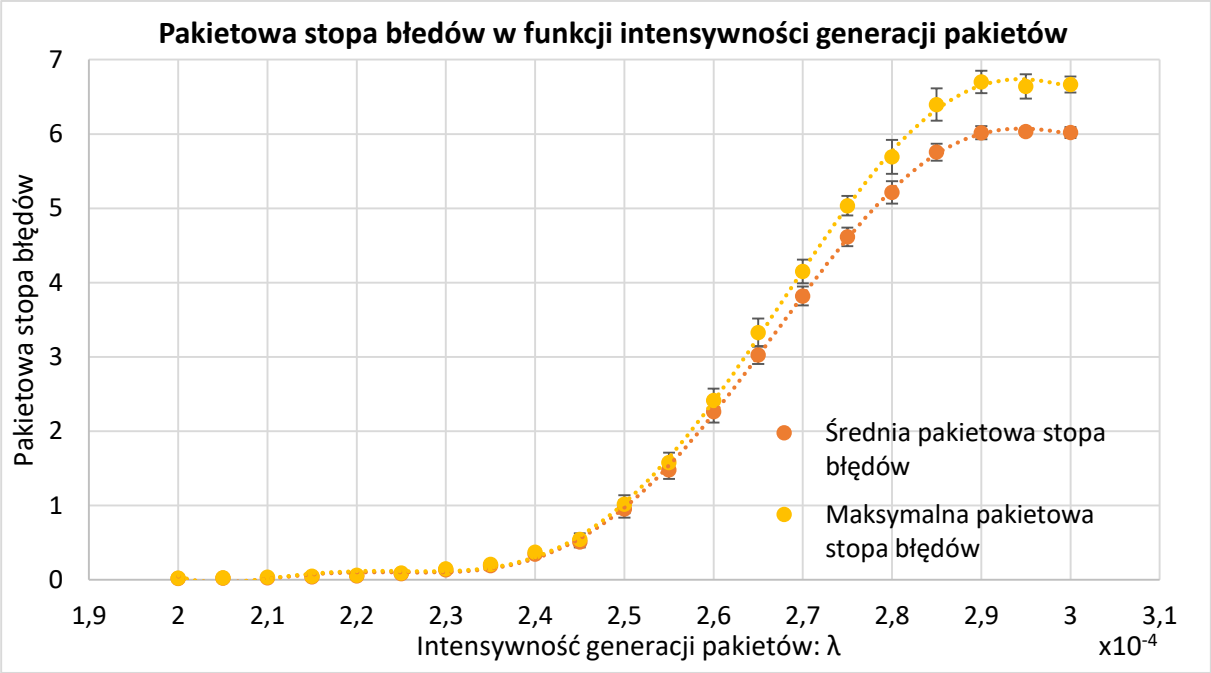



Na podstawie analizy maksymalnej pakietowej stopy błędu można zauważyć, że jest ona silnie skorelowana z średnią pakietową stopą błędu i charakteryzuje się zbliżoną sekwencją wartości. Różnią się one o około $0{,}9 \cdot 6.$ do $1{,}4 \cdot 10^{-6}$ jednostek.

Dla wyznaczonego zakresu wartości lambda została wyznaczona również przepływność systemu:

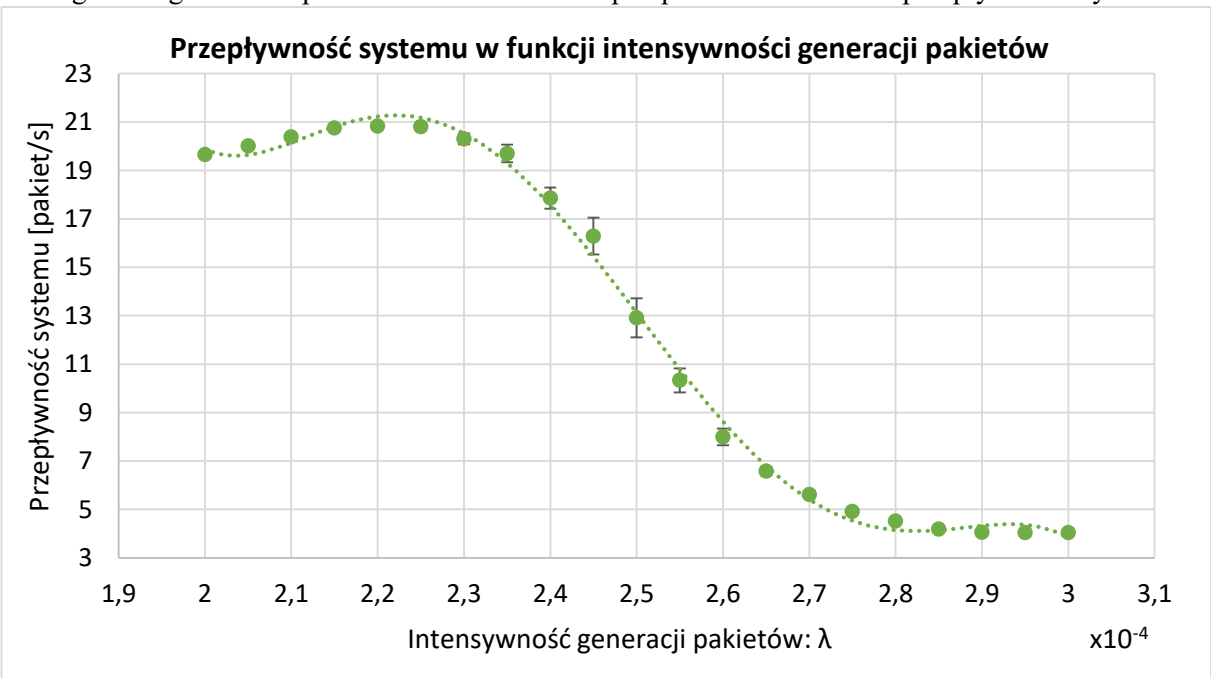


Niestety ze względu na małe zmiany przepływności trudno jest dostrzec wyraźne tendencje. W celu dokładniejszej analizy jak zmienia się przepływność systemu przeprowadzona została dodatkowa symulacja o szerszym zakresie parametru lambda. Dla 21 wartości intensywności generowania pakietu została przeprowadzone symulacje z 10 krotnym powtórzeniem, ze względu na redukcję czasu potrzebnego na wykonanie symulacji. Rezultaty średniej i maksymalnej pakietowej stopy błędu dla poszerzonego zakresu parametru lambda:




Dzięki zastosowaniu analizy o większym zakresie zmienności parametru lambda można zaobserwować zachowanie systemu dla zwiększającej się intensywności generowania pakietów. Wartości graniczne parametrów systemu, dla których pakietowa stopa błędów ulega nasyceniu. Przebieg tego parametru jest bardzo podobny do dystrybuanty rozkładu normalnego. Dla dalszego zwiększania wartości lambda wartość PER nie ulegała większym zmianom i pozostawała na poziomie około 6,5 jednostek – system się nasycał.

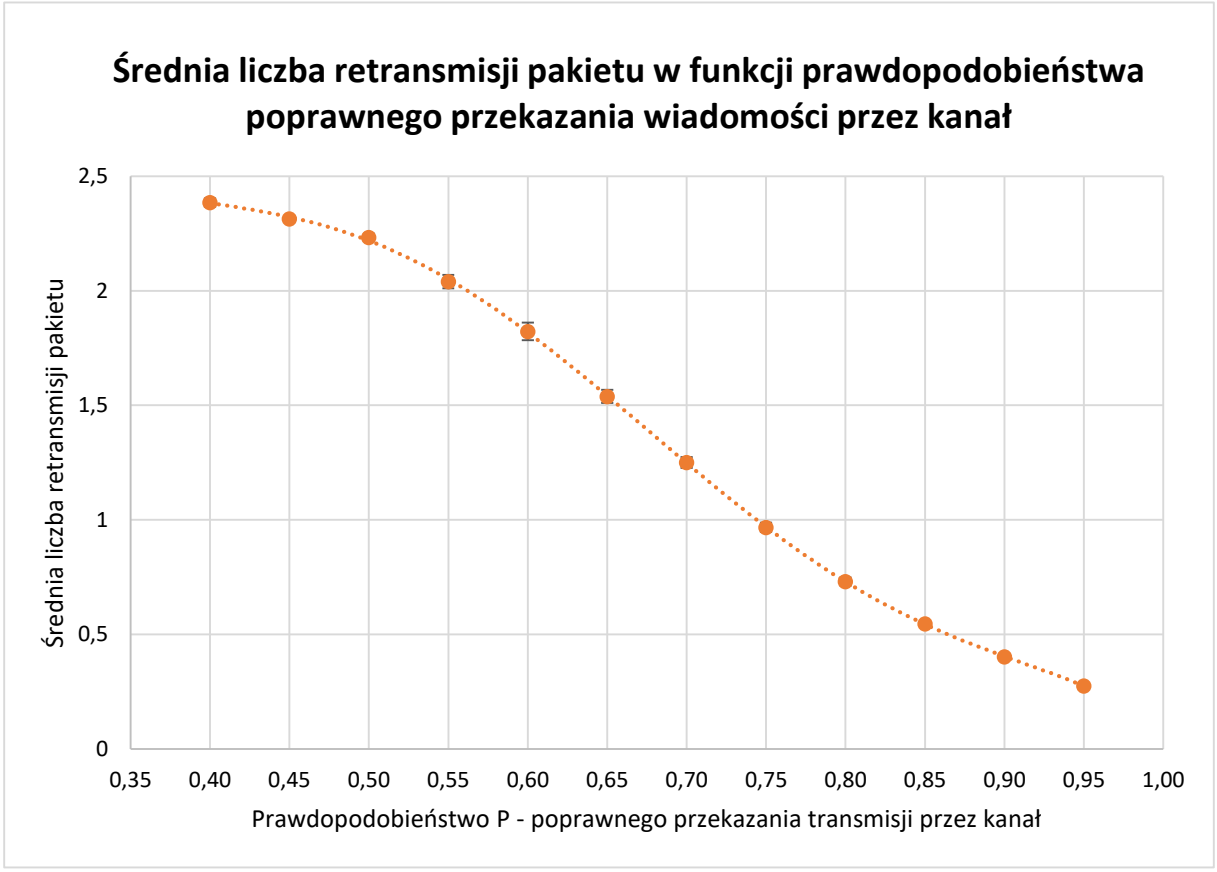
Dla tego samego zakresu parametru lambda została przeprowadzona analiza przepływności systemu:



Analizując przepływność w systemy w zależności od intensywności generacji pakietów, można ponownie zauważyć okres przejściowy podobny do charakterystyki przejściowej dystrybuanty rozkładu normalnego. Jednakże widoczne jest również lokalne maksimum w okolicach wartości $2{,}2 \cdot 10^{-4}$, gdzie sieć znajduje się w stanie optymalnym i ma największą przepływność. Następuje równowaga między ilością przychodzących pakietów i maksymalną częstotliwością wysyłania, która nie powoduje dużej liczby kolizji.

**15. Analiza zależności średniej liczby retransmisji pakietu od parametru P – prawdopodobieństwa poprawnego przesłania transmisji przez kanał.**

W celu sprawdzenia jak jakość kanału wpływa na średnią liczbę retransmisji pakietu dla ustalonej wcześniej wartości parametru lambda wykonana seria symulacji ze zmienną wartością parametru P. Parametr P, czyli prawdopodobieństwo poprawnego przekazania transmisji przez kanał, był zmieniany w zakresie od 0,4 do 0,95 z krokiem 0,05. Dla każdej wartości parametru symulacja była powtarzana 15 razy a wyniki zostały uśrednione. Wyznaczony został również przedział ufności dla rozkład T-Studenta, ale ze względu na małe wartości nie jest wyraźnie widoczny na wykresie.




Na podstawie wykresu można zaobserwować, że im większe prawdopodobieństwo sukcesu tym średnia liczba retransmisji jest mniejsza. Charakter tej zależności przypomina przejściową fazę arcusa cotangensa lub odwróconą w osi Y dystrybuantę rozkładu normalnego. Dla przedstawionej analizy dla wartości granicznych nie widoczne jest zjawisko nasycenia.

## 16. Wnioski:

- Większość zaobserwowanych przebiegów parametrów systemu jest bardzo zbliżona do dystrybuanty rozkładu normalnego, co może wskazywać na poprawnie zachodzące w symulacji procesy losowe.

- Im większa liczba elementów w systemie: nadajników-odbiorników oraz im większa dopuszczalna liczba retransmisji tym czas trwania fazy początkowej się wydłuża i system potrzebuje dłuższy okres na stabilizację.

- W przypadku bardzo małych zmian wartości parametru lambda lub P (prawdopodobieństwa poprawnego przekazania transmisji przez kanał) dopuszczalna wydaje się interpolacja liniowa w celu wyznaczenia pożądanych wartości.

- Dla małych zmian parametrów systemu ich wypływ na uzyskiwane statystyki jest bardzo trudno zauważalny ze względu na kilka procesy losowe. Dla większej dokładności należałoby wydłużyć czas symulacji lub uśrednić większą ich liczbę w celu redukcji wpływu różnych stanów początkowych i losowości na końcowe wyniki.

- Z przeprowadzonych symulacji systemu można zauważyć tendencję do nasycania się pakietowej stopy błędów oraz przepływności dla wartości spoza zakresu $2 \cdot 10^{-4}$ do $3 \cdot 10^{-4}$. Charakterystyka przejściowa zawiera się w bardzo wąskim zakresie, a dla pozostałych wartości system albo swobodnie przesyła pakiety bez większych problemów lub występuje duża ilość kolizji i system jest przeciążony. Warto również zwrócić uwagę na przewidywany początkowy kształt charakterystyki przepływności, który nie został ukazany w analizie. Można przypuszczać, że dla narastającej wartości parametru lambda – intensywności generacji pakietów system odpowiadałby stosunkowo proporcjonalnie na zwiększającą się intensywność napływu pakietów.

- Uzyskane rezultaty symulacji oraz rozrzuty przedziałów ufności pozwalają potwierdzić stosunkowo dobrą powtarzalność rezultatów symulacji. Dla wyznaczonego parametru lambda przedziały ufności stanową mniej niż 10% wartości wyniku. Przepływność systemu ustala się na poziomie 20,63 pakietów/s. Średni czas oczekiwania pakietu wniósł 62,89 ms natomiast czas opóźnienia pakietu 76,96 ms. Oznacza to, że przybliżony czas między rozpoczęciem przekazu a poprawnym odebraniem wyniósł 14,07 ms. W zależności od rozmiaru pakietu można by estymować przepływność bitową systemu. Otrzymane czasy wydają się być stosunkowo duże, a przepływność zestawionego sytemu mała w porównaniu do praktycznych zastosować. Może to być spowodowane dużymi czasami oczekiwania i wymaganymi czasami niezajetości.

- Istnieje możliwość wykorzystania rezultatów symulacji w celu znalezienia maksimum przepływności sytemu lub granicznej intensywności napływu nowych pakietów w celu optymalizacji istniejących rozwiązań lub dodania mechanizmów dostosowujących intensywność dla polepszenia przepływności pakietowej systemu.

- W celu przeprowadzenie symulacji musiała zostać zredukowana liczba nadajników z początkowych wartości równych około 30-50. Spowodowane to było zbyt dużą ilością zdarzeń. (każdy nadajnik wprowadzał dodatkowy zestaw 20 zdarzeń) Pozwoliło to na potwierdzenie, że metoda przeglądania działań nie jest optymalna dla systemów z dużą ilością zdarzeń i wraz ze wzrostem ich liczby, czas wykonania symulacji się wydłuża, ze względu na wzrost złożoności obliczeniowej pojedyńczej iteracji pętli.

- Istnieje możliwość testowania różnych konfiguracji dokonując małych modyfikacji w kodzie projektu, przykładowo ewaluacji systemu z kilkoma kanałami radiowymi.