Warszawa, czerwiec 2010

Symulator działania algorytmu Flow Random Early Drop (FRED)

Projekt z przedmiotu Sieci wielousługowe

Prowadzący: mgr inż. Piotr Wiśniewski

Autorzy: Rafał Malinowski, Cezary Zawadka

Zawartość

WSTĘP

Celem projektu było zaimplementowanie algorytmu kolejkowania Flow Random Early Drop (FRED) będącego rozwinięciem algorytmu Random Early Detection (RED). W tym celu stworzono symulator umożliwiający badanie obydwu algorytmów FRED i RED przetwarzających ruch o stałej lub zmiennej przepływności bitowej. Następnie przeprowadzono szereg symulacji potwierdzających wyższość algorytmu FRED nad algorytmem RED.

DZIAŁANIE ALGORYTMU FRED

Algorytm Flow Random Early Drop (FRED) jest rozwinięciem popularnego algorytmu kolejkowania Random Early Detection (RED). Podstawowy algorytm RED pozwala na unikanie przeciążenia poprzez losowe odrzucanie pakietów gdy średnia długość kolejki (avg, wyliczana za pomoca średniej kroczącej) przekroczy wartość progową (min_th). Prawdopodobieństwo odrzucenia pakietu rośnie wraz ze wzrostem średniej długości kolejki. Po przekroczeniu wartości maksymalnej (max_th) wszystkie pakiety są odrzucane.

Opisana implementacja algorytmu RED może prowadzić do niesprawiedliwego podziału pasma, gdy ruch oferowany na wchodzący do kolejki RED jest większy od szybkości portu wychodzącego. Dzieje się tak zwłaszcza wtedy, gdy połaczenia przychodzące różnią się typem ruchu(CBR/VBR), parametrem round trip time (rtt) lub szybkościa łącza.

Algorytm FRED pozwala na bardziej sprawiedliwy podział pasma w wyżej wymienionych przypadkach poprzez zapamiętywanie kilku dodatkowych parametrów dla każdego połączenia którego pakiety znajdują się w kolejce.

FRED wprowadza następujące parametry:

- ***max_q*** maksymalna ilość pakietów w kolejce dopuszczalna dla jednego połączenia
- ***min_q*** minimalna ilość pakietów w kolejce dopuszczalna dla jednego połączenia. Gdy połączenie ma mniej pakietów w kolejce, nie będą one losowo odrzucane.
- ***avgcq*** oznaczający średnią ilość zakolejkowanych pakietów na połączenie.
- ***qlen*** przechowujący ilość zakolejkowanych pakietów dla każdego połączenia.
- ***strike*** oznaczający ilość przypadków, gdy dane połączenie usiłowało przekroczyć maksymalną dopuszczalną ilość pakietów w kolejce.

FRED odrzuca przychodzące pakiety połaczeń dla których ilość zakolejkowanych pakietów przekracza ***max_q,*** dwukrotność ***avgcq*** lub pakiety połączeń które mają niezerową wartość ***strike*** i przekroczyły ***avgcq***. Nie odrzuca pakietów dla których ilość zakolejkowanych pakietów nie przekracza ***min_q.*** Ponadto algorytm FRED wylicza średnią długość kolejki **avg** zarówno po przyjściu nowego pakietu jak i po zdjęciu pakietu z kolejki (RED wylicza **avg** tylko w drugim przypadku). W pozostałych przypadkach (długośc kolejki > min_th) algorytm FRED działa identycznie jak RED odrzucając pakiety z prawdopodobieństwem proporcjonalnym do długości kolejki.

IMPLEMENTACJA

W ramach projektu stworzono w języku JAVA symulator wraz z następującymi węzłami:

- Fred – węzeł przekazujący pakiety z zaimplementowaną kolejką FRED.
- RED – węzeł przekazujący pakiety z zaimplementowaną kolejką RED.
- TCPSource – źródło generujące pakiety o charakterystyce podobnej do zachłannego połączenia TCP. Węzeł zwiększa szybkość nadawania pakietów co czas RTT. Po odnotowaniu straty pakietu (po upłynięciu czasu timeout) źródło zmniejsza okno nadawcze o połowę.
- UDPSource – źródło generujące ruch o stałej szybkości. Nie otrzymuje potwierdzeń, nie zwalnia w przypadku straty pakietu.
- Sink – węzeł docelowy wszystkich połączeń. Służy do odnotowania odebrania pakietu w logach. Informuje źródła TCPSource o odebraniu pakietu. Informowanie odbywa się poprzez wywołanie odpowiedniej metody obiektu TCPSource po upływie określonego czasu(równego różnicy czasu pomiędzy odebraniem pakietu przez Sink a nadaniem pakietu przez TCPSource). Nie jest nadawany pakiet z ACK, potwierdzenie nie może zostać stracone.

Ponadto zaimplementowano klasy łącza (Link), pakietów (UDPPacket, TCPPacket), zegara (Timer) oraz, w celach testowych węzeł FakeNode operujący na zasadzie „tail drop" (odrzucanie wszystkich pakietów po przepełnieniu bufora). W klasie Constants zdefiniowane są wszystkie zmienne globalne wykorzystywane w czasie symulacji.

Implementacja symulatora nie jest optymalna. Zrezygnowano ze mechanizmu zdarzeniowego na rzecz prostoty. W czasie symulacji wykonywana jest pętla w której każdy węzeł oraz łącze sprawdza, co powinno zostać wykonane.

PROCEDURA SYMULACJI

Aby przeprowadzić symulacje należy w pliku FREDmain.java, w metodzie main dodać odpowiednie węzły oraz połączenia. W pliku Constants.java można zmienić niektóre parametry sumulacji. Następnie trzeba skompilować i uruchomić program.

W czasie symulacji zdefiniowane źródła zaczynają nadawać pakiety w losowym czasie *(0<= t <= Constants . timeStart*), następnie po upływie czasu *Constants.timeStartToMesure* pakiety które zostaną odebrane przez węzły sink są odnotowywane w plikach log file. Nazwy plików powstają w następujący sposób: „log file_+<id wezla sink>+_+<nazwa wezla źródłowego>+_+< id wezla źódłowego>+.txt". Format pliku jest zgodny z fromatem danych CSV rozumianym prze MS Excel. (przy ustawieniach lokalnych dla Polski). Symulacja kończy się po czasie *Constants .timeToMesure.*

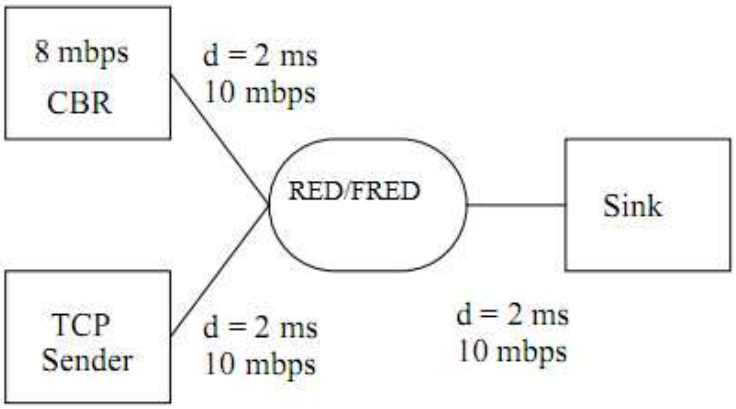
Wartości parametrów, dla których przeprowadzane były symulacje:

- *timeStart = 5s*
- *timeStartToMesure = 30s*
- *timeToMesure = 330s*
- *buffer_size = 32[pakietów]*
- *rozmiar pakietu: 1500B*

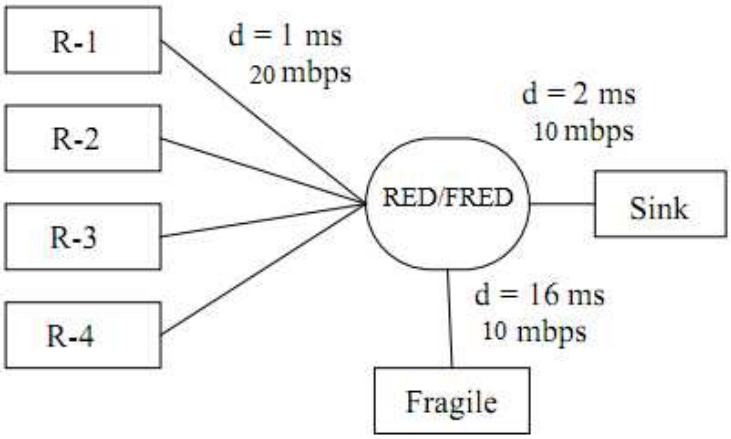
SCENARIUSZE SYMULACJI

Przeprowadzono 4 symulacje. Schematy połączeń wraz z parametrami przedstawiono poniżej:

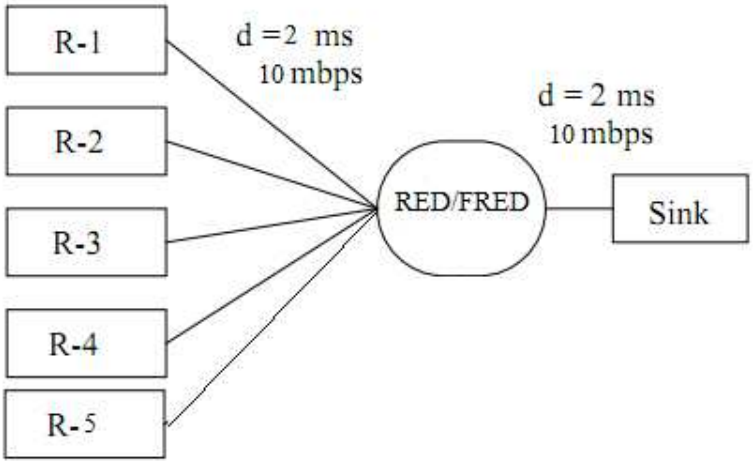
1. Źródło CBR oraz TCP:



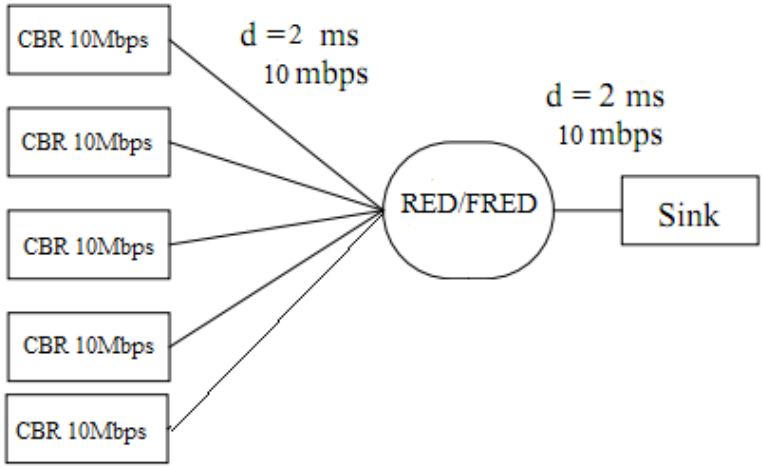
2. Źródła TCP, z których jedno ma większy czas RTT oraz mniejszą szybkość łącza



3. Równorzędne źródła TCP.



4. Równorzędne źródła CBR.



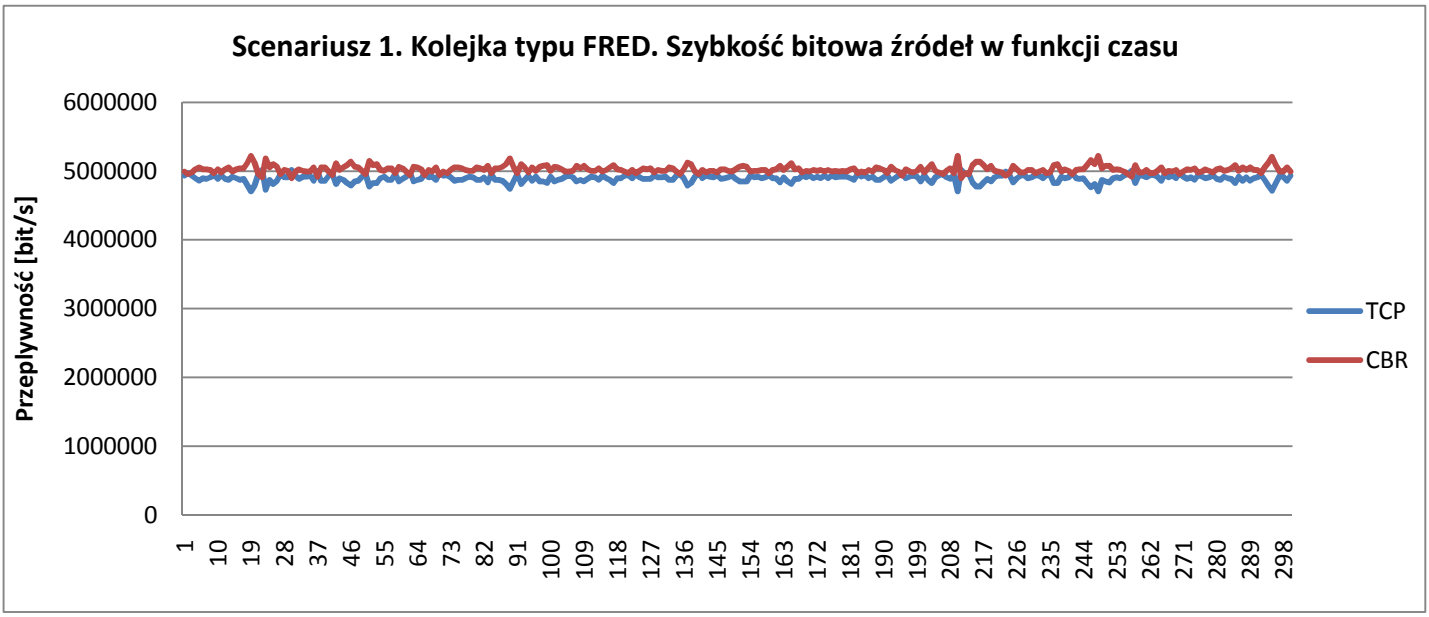
Wszystkie scenariusze przeprowadzono dla węzłów z algorytmem FRED oraz RED.

WYNIKI

Poniżej przedstawiono wyniki wszystkich symulacji. Tabele zawierają informacje na temat średniej przepływności bitowej w całym okresie pomiarowym dla każdego źródła. Wykorzystanie łącza oznacza utylizacje łącza RED/FRED <-> Sink.

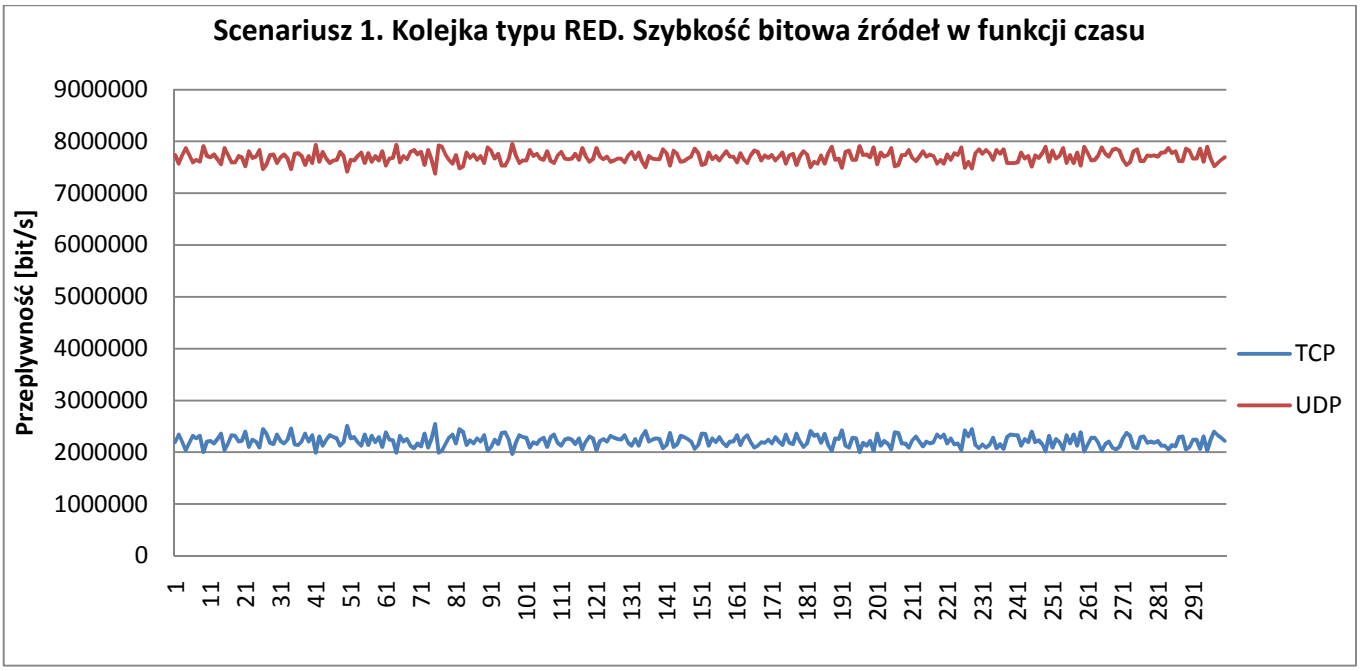
Scenariusz 1. Kolejka typu FRED.

Typ źródła:	Średnia przepływność [bit/s]
TCP	4893760
CBR	5023720
suma	9917480
Wykorzystanie łącza:	0,991748



Scenariusz 1. Kolejka typu RED.

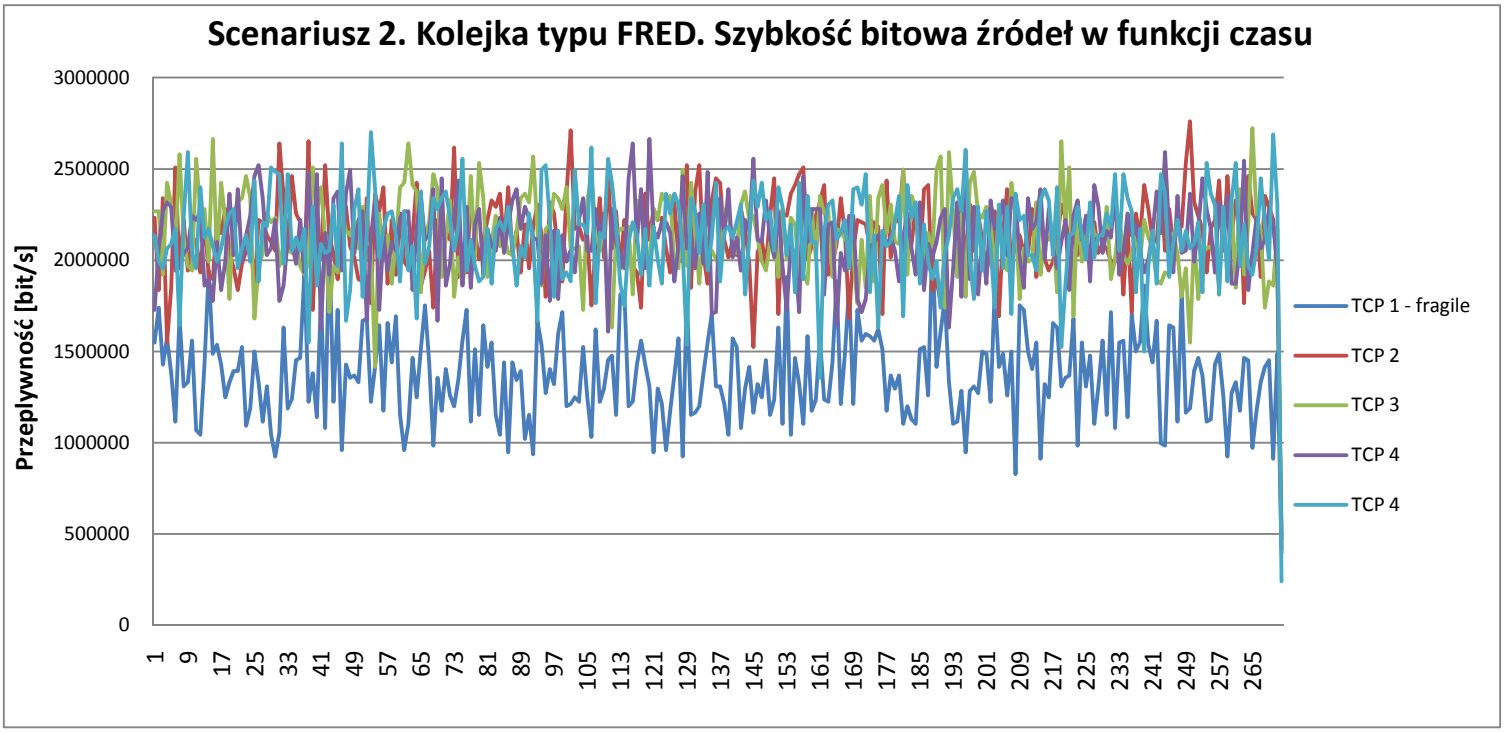
Typ źródła:	Średnia przepływność [bit/s]
TCP	2216960
CBR	7699880
suma	9916840
Wykorzystanie łącza	0,991684



Wyniki symulacji 1 potwierdzają sprawiedliwy podział pasma w przypadku algorytmu FRED. Źródło CBR ma zajmuje niewiele większą część pasma, co wynika z faktu, że źródło TCP zwalnia po stracie pakietu. W przypadku algorytmu RED źródło CBR zajmuje znaczną część pasma (niewiele mniejszą od ruchu generowanego przez to źródło - 8MB) ponieważ algorytm RED nie odrzuca pakietów do nieelastycznego połączenia z większym prawdopodobieństwem.

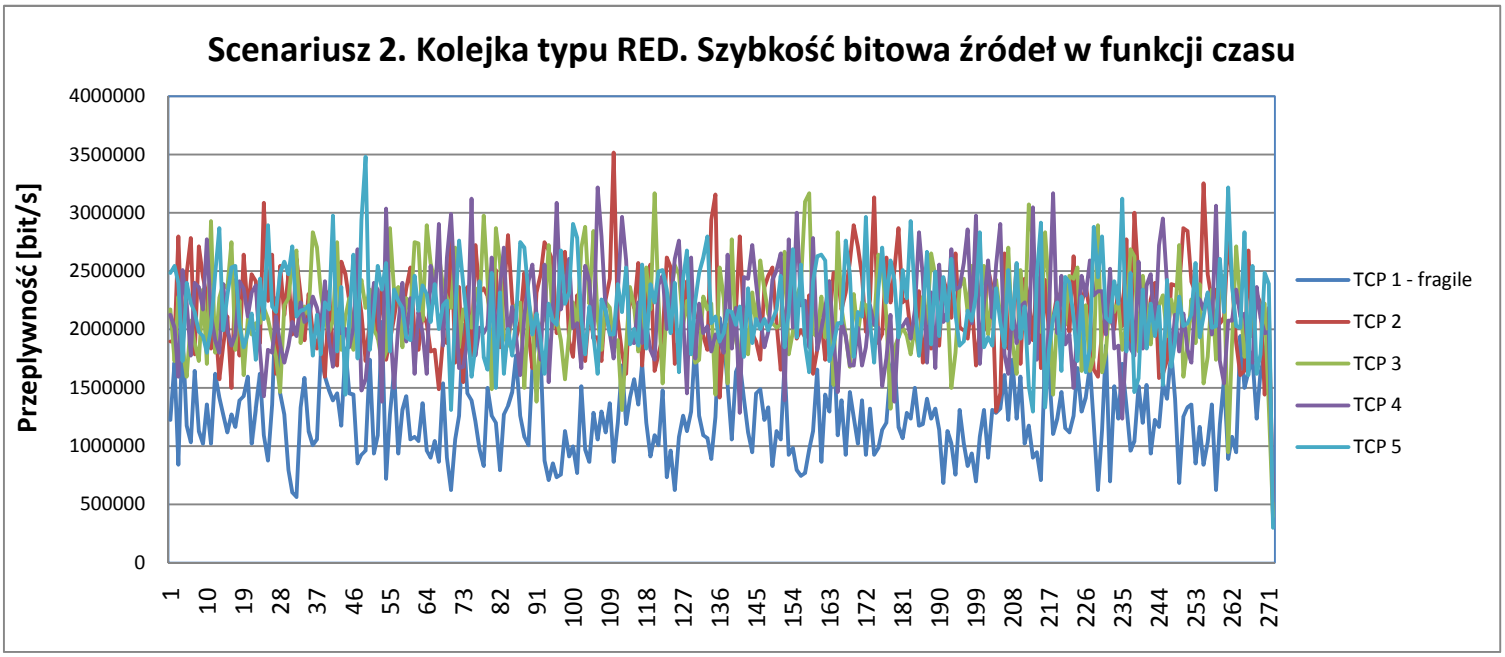
Scenariusz 2. Kolejka typu FRED.

Typ źródła:	Średnia przepływność [bit/s]
TCP 1 - fragile	1366368
TCP 2	2129912
TCP 3	2131412
TCP 4	2115309
TCP 5	2129735
suma	9872735
Wykorzystanie łącza	0,987274



Scenariusz 2. Kolejka typu RED.

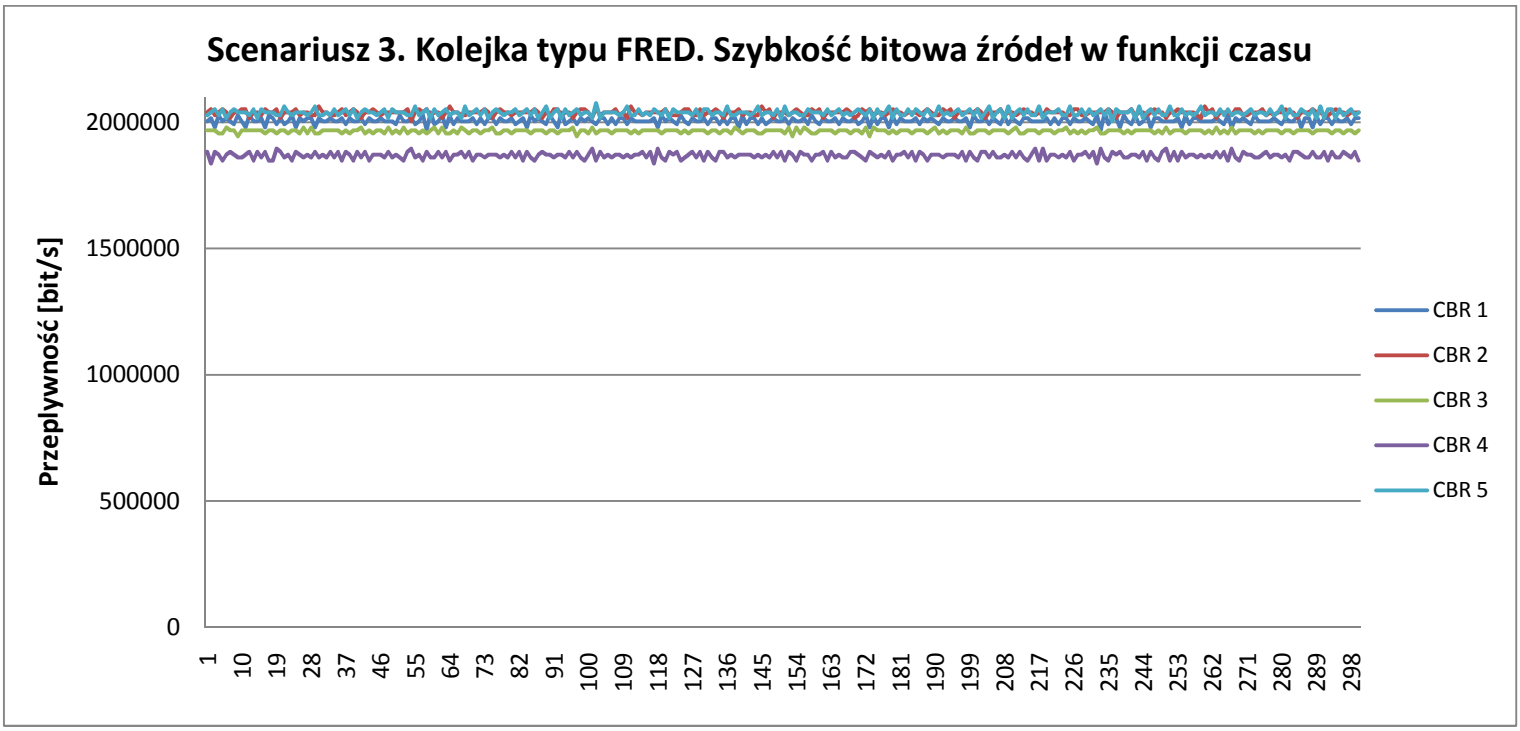
Typ źródła:	Średnia przepływność [bit/s]
TCP 1 - fragile	1237324
TCP 2	2160926
TCP 3	2152015
TCP 4	2127529
TCP 5	2176809
suma	9854603
Wykorzystanie łącza	0,98546



W przypadku, gdy jedno ze źródeł TCP ma znacznie większy parametr rtt oraz wolniejsze łącze od pozostały algorytm FRED oferuje bardziej sprawiedliwy podział pasma.

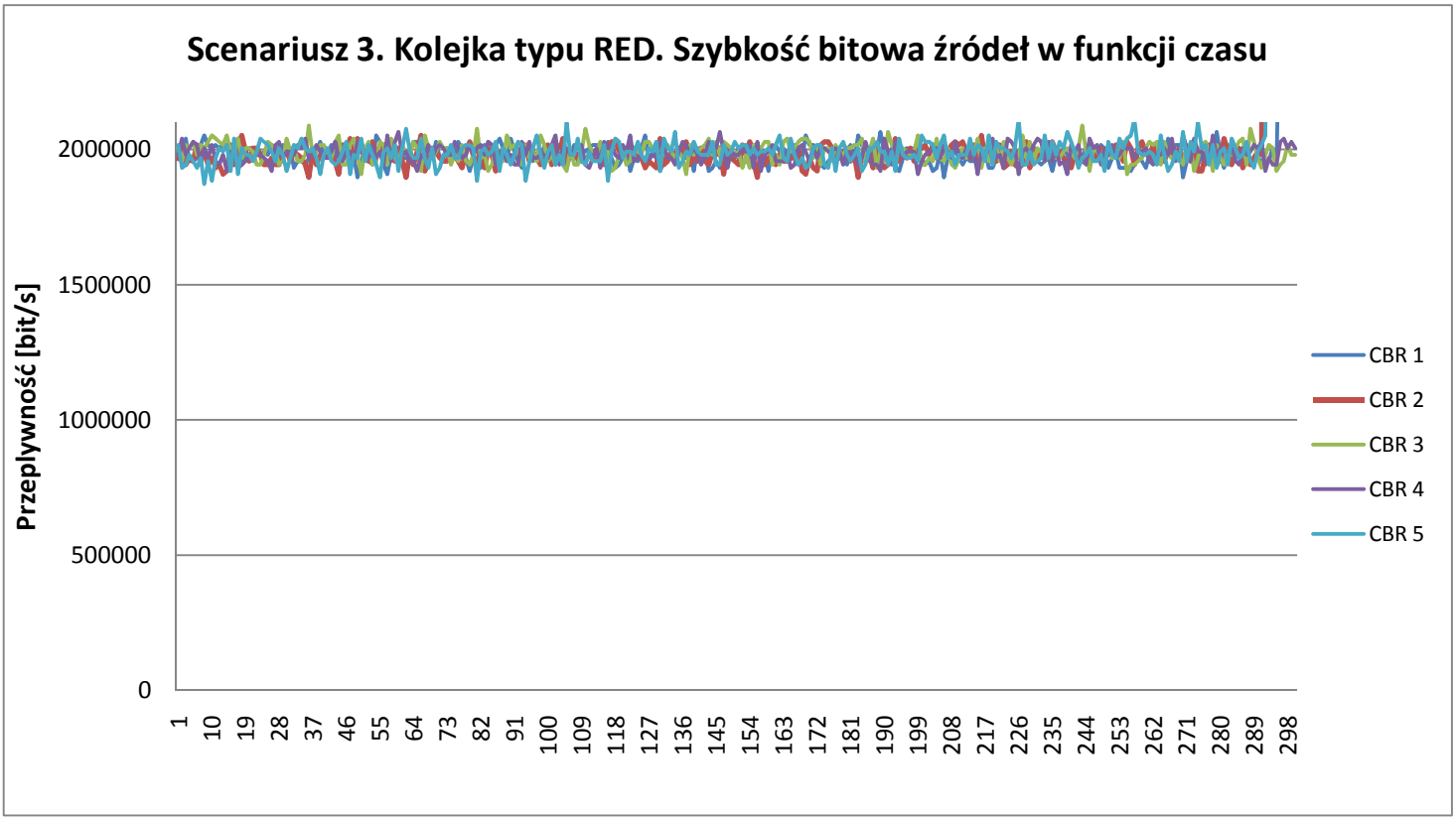
Scenariusz 3. Kolejka typu FRED.

Typ źródła:	Średnia przepływność [bit/s]
CBR 1	2038600
CBR 2	2006480
CBR 3	1965080
CBR 4	1868640
CBR 5	2038560
suma	9917360
Wykorzystanie łącza	0,991736



Scenariusz 3. Kolejka typu RED.

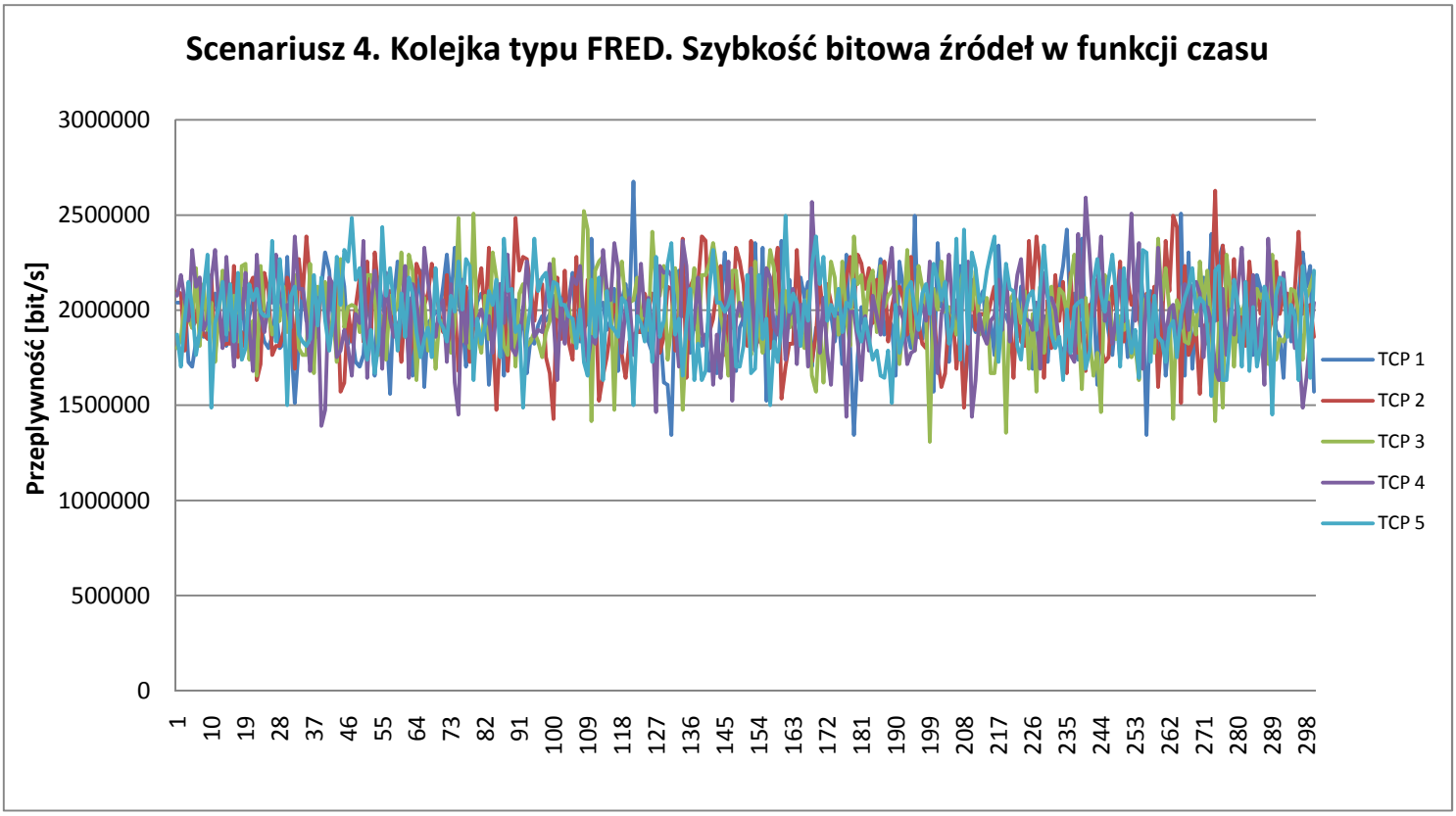
Typ źródła:	Średnia przepływność [bit/s]
CBR 1	2028601
CBR 2	2006480
CBR 3	1989480
CBR 4	1986280
CBR 5	1938649
suma	9949490
Wykorzystanie łącza	0,994949



W przypadku źródeł generujących pakiety ze stałą szybkością oraz identycznymi parametrami węzły z algorytmami FRED i RED obsługują ruch w podobny sposób. Utylizacja łącza jest bardzo wysoka. Zmienność szybkości pakietów w węźle FRED jest niewielka. Dla algorytmu RED średnie szybkości dla wszystkich źródeł są bardziej zbliżone niż w przypadku FRED.

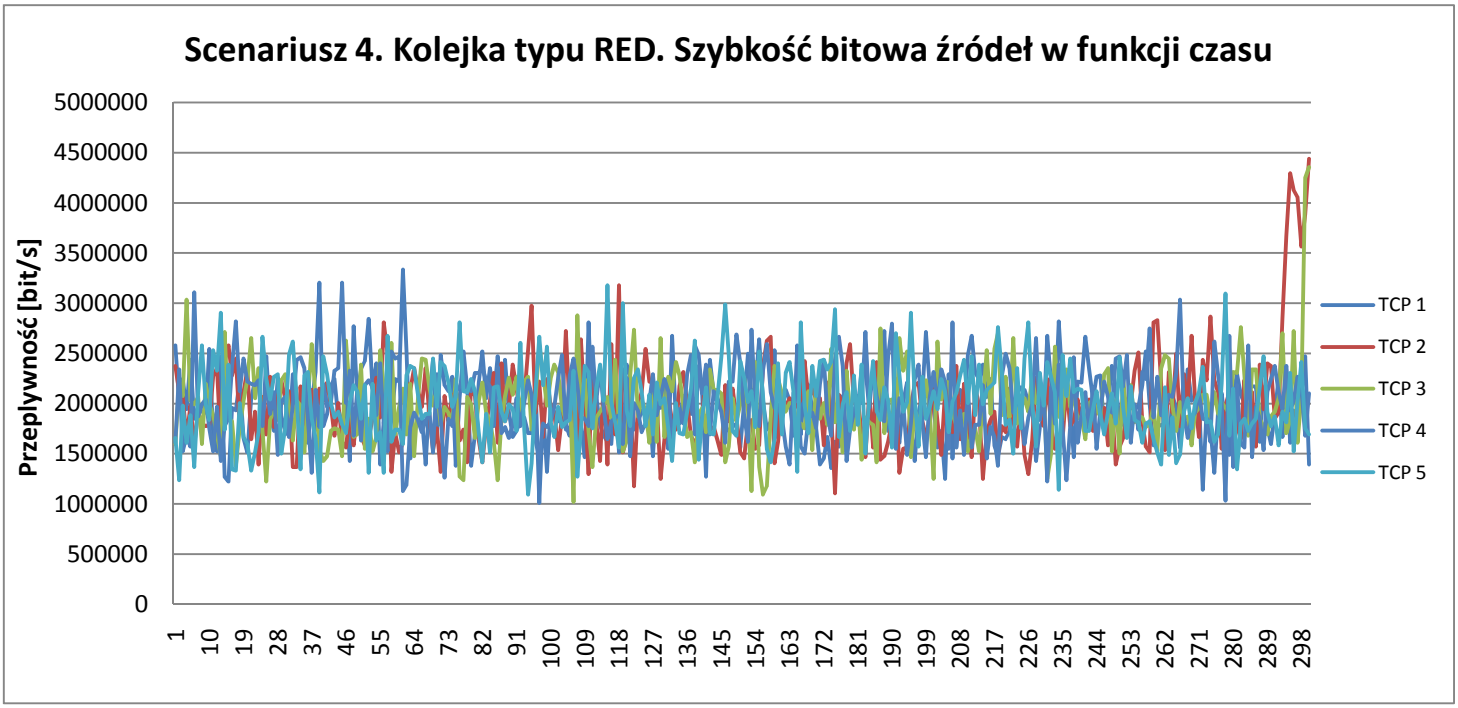
Scenariusz 4. Kolejka typu FRED.

Typ źródła:	Średnia przepływność [bit/s]
TCP 1	1961440
TCP 2	1988720
TCP 3	1980800
TCP 4	1979880
TCP 5	1979720
suma	9890560
%	0,989056



Scenariusz 4. Kolejka typu FRED.

Typ źródła:	Średnia przepływność [bit/s]
TCP 1	1978320
TCP 2	1988600
TCP 3	1980760
TCP 4	2003840
TCP 5	2000680
suma	9952200
%	0,99522



W przypadku źródeł TCP o identycznych parametrach węzły z algorytmami FRED i RED obsługują ruch w podobny sposób. Zmienność szybkości bitowej w węźle FRED jest znacznie mniejsza niż w przypadku RED. Utylizacja łącza oraz sprawiedliwość podziału pasma jest porównywalna w obu przypadkach.

PODSUMOWANIE

W trakcie realizacji projektu zaimplementowano symulator algorytmu Flow Random Early Drop (FRED) wraz ze źródłami pakietów (uproszczone TCP oraz UDP) oraz algorytmem Random Early Detection (RED). Następnie przeprowadzono 4 scenariusze symulacyjne porównując działanie obydwu algorytmów kolejkowania. Symulacje potwierdziły znacznie bardziej sprawiedliwy podział pasma (na łączu będącym wąskim gardłem) w węzłach z zaimplementowanym algorytmem FRED w sytuacji, gdy węzły źródłowe mają różne parametry działania. W przypadku, gdy źródła działały w sposób identyczny symulacje dla obydwu algorytmów dawały bardzo zbliżone rezultaty.

Należy zauważyć, iż prosta implementacja źródła o zmiennej szybkości (brak algorytmów slow start, fast recovery oraz fast retransmit, retransmisja typu go-back-n) obniża wiarygodność symulacji.

BIBLIOGRAFIA

1. Dong Lin and Robert Morris, *Dynamics of Random Early Detection*, ACM SIGCOMM 1997 Conference, strony 127-137