STUDIA INFORMATICA 2001
Volume 22 Number 3 (45)

Sławomir NOWAK
Instytut Informatyki Teoretycznej i Stosowanej PAN

WYKORZYSTANIE MECHANIZMU RED W REGULACJI RUCHU SIECIOWEGO

Streszczenie. W artykule przedstawiono charakterystykę algorytmu RED dla kontroli ruchu i zapobiegania przeciążeniom w sieciach z przełączaniem pakietów (w szczególności w sieci Internet). Przedstawiono zalety tego algorytmu na tle innych popularnie stosowanych rozwiązań, omówiono problem optymalnego doboru parametrów algorytmu oraz zbadano zachowanie się algorytmu RED dla wybranych modeli symulacyjnych.

APPLICATION OF THE RED MECHANISM TO NETWORK TRAFFIC CONTROL

Summary. The paper presents characteristics of the RED algorithm to control traffic intensity and to avoid congestion in packet-switched networks (especially in the Internet). It explains the features of the mentioned algorithm in the comparison to other popular approaches. The paper discusses the problem of influence of RED parameters for optimal performance and illustrates a behavior of the algorithm with the use of simulation models.

1. Wprowadzenie

W szybkich sieciach szerokopasmowych, w celu przechowania pakietów w chwilowych stanach przeciążeń, wykorzystuje się odpowiednio duże rozmiary buforów (kolejek). Jednocześnie w obecnych sieciach opartych na protokole TCP/IP reakcja na wystąpienie przeciążenia polega na retransmisji pakietów, na które nie otrzymano potwierdzenia w spodziewanym czasie (*time out*). Jest to tzw. mechanizm *implicit feedback*, czyli bez

jawnego sprzężenia zwrotnego. Brak potwierdzenia może być wynikiem zarówno usunięcia pakietu (skutkiem działania odpowiednich algorytmów kontroli przeciążeń w węzłach sieci), jak również zbyt dużym opóźnieniom podczas transmisji pakietu w sieci (związanych z kolejkowaniem w węzłach). Jest oczywiste, że stałe utrzymywanie dużej zajętości buforów pakietów w węzłach jest niekorzystne, gdyż w znaczącym stopniu zwiększa to średnie opóźnienie transmisji. Dlatego przy rosnącej popularności szybkich sieci komputerowych, niezwykle ważnym problemem jest zapewnienie dynamicznego mechanizmu, który utrzymywałby dużą przepustowość, ale niską średnią długość kolejek [2].

Sieć Internet bazuje na protokole TCP/IP, którego zalety przyczyniły się do jego ogromnej popularności. Jednak brak bezpośredniego (*explicit*) sprzężenia zwrotnego z węzła do źródła jest przyczyną, dla której problem efektywnej, dynamicznej kontroli przeciążeń w sieciach TCP/IP jest tak istotny. Ponieważ stacje prowadzące transmisje nie mają bezpośrednich informacji o stanie sieci, konieczne jest istnienie odpowiednich mechanizmów kontroli działających w węzłach. W dużym uproszczeniu, algorytmy te mają za zadanie zarządzać kolejkami pakietów przeznaczonych do przesłania, usuwając niektóre z nich w przypadku przepełnienia buforów lub zgodnie z innymi regułami zastosowanego algorytmu. Zaproponowano kilka takich mechanizmów [13, 14], zapewniających dużą przepustowość i niskie opóźnienia. Niektóre z nich znajdują zastosowanie w obecnych rozwiązaniach dla sieci Internet.

Algorytmy kontroli przeciążeń, działające w węzłach, mogą prowadzić najbardziej efektywną detekcję przeciążeń. Bramka (węzeł) może rozróżniać opóźnienia związane z propagacją sygnału od opóźnień związanych z kolejkowaniem. Tylko węzły mają bezpośredni wpływ na zachowanie kolejek w czasie. Węzeł sieci jest wykorzystywany przez wiele aktywnych połączeń o różnych czasach przejścia przez sieć, tolerancji na opóźnienie, wymaganiach na przepustowość itp. Także decyzje o długości i rozmiarze chwilowego dozwolonego przeciążenia najlepiej określać dla poszczególnych węzłów.

Najprostszym, powszechnie stosowanym w sieciach opartych na protokole TCP/IP, jest algorytm *Drop Tail*. Jego działanie polega na odrzucaniu wszystkich nadchodzących pakietów po osiągnięciu zadanego progu (maksymalnej możliwej długości kolejki). Rozwiązanie to ma jednak wiele wad, przedstawionych w dalszej części artykułu.

Wśród innych zaproponowanych rozwiązań mechanizmów kontroli przeciążeń najbardziej interesujący wydaje się RED (*Random Early Detection*). RED został zaproponowany po raz pierwszy w pracy [1]. Pozwala on na optymalne wykorzystanie dostępnego łącza, zapewniając przy tym niewielkie opóźnienia pakietów.

Ponieważ zasada działania RED jest dość uniwersalna, algorytm znajduje zastosowanie w kontroli średniej długości kolejek zarówno w sieciach wykorzystujących znacznik

wystąpienia przeciążenia, jak i w sieciach, gdzie protokoły nie uwzględniają znacznika przeciążenia i nie wykorzystują żadnych mechanizmów kontroli przeciążeń (kontrola realizowana za pomocą wyższych warstw protokołu).

Niektóre cechy algorytmu RED sprawiają, że jest on szczególnie użyteczny w sieciach TCP/IP (brak jawnego sprzężenia zwrotnego, usunięte pakiety są wystarczającym sygnałem wystąpienia przeciążenia dla warstwy transportowej protokołu).

Mechanizm RED monitoruje średnią długość kolejki dla każdej z kolejek wyjściowych, i losowo wybiera połączenia, które uznane zostają za przeciążone. Chwilowe przeciążenia są rozładowywane w okresowo zwiększających się kolejkach (średnia długość kolejki zmienia się nieznacznie). Dłużej trwające stany przeciążenia mają swoje odzwierciedlenie w zwiększeniu wyliczonej średniej długości kolejki, a w rezultacie część połączeń (lub wszystkie, po przekroczeniu maksymalnego progu długości kolejki) zostaje zmuszonych do redukcji swojego okna transmisji. Im większą część dostępnej przepustowości łącza zajmuje dane połączenie, tym większe jest prawdopodobieństwo, że połączenie to zostanie uznane za przeciążone.

2. Zasada działania algorytmu RED

Algorytm RED wykorzystuje średnią długość kolejki (obliczaną za pomocą filtru dolnoprzepustowego). Ta średnia długość kolejki jest porównywana do dwóch progów (*thresholds*): maksymalnego (max_{th}) i minimalnego (min_{th}). Ogólną zasadę działania algorytmu można zapisać następująco:

```
dla każdego przychodzącego pakietu:
    wylicz średnią długość kolejki avg
    if min_th<=avg<=max_th
        wylicz prawdopodobieństwo p_a
        z prawdopodobieństwem p_a:
            oznacz przychodzący pakiet
    else if max_th<=avg
        oznacz przychodzący pakiet
```

Gdy średnia długość kolejki przekracza próg maksymalny, każdy nadchodzący pakiet jest oznaczany jako przeciążony, jeśli jest poniżej progu minimalnego, wszystkie pakiety są transmitowane jako nieprzeciążone, a gdy średnia długość jest w przedziale pomiędzy progiem maksymalnym a minimalnym, każdy przychodzący pakiet jest oznaczany jako przeciążony z prawdopodobieństwem p_a, gdzie p_a jest funkcją średniej długości kolejki. Przez pojęcie „oznaczanie pakietów" rozumie się, w zależności od protokołu, usuwanie pakietów

(np. w sieciach TCP/IP) lub ustawianie w nagłówku pakietu bitu sygnalizującego wystąpienie przeciążenia (ECN - *Explicit Congestion Notification*).

Jak widać, rozwiązanie to korzysta z dwóch odrębnych algorytmów, z których pierwszy odpowiada za wyliczanie średniej długości kolejki, drugi określa prawdopodobieństwo, z jakim przybywające pakiety będą oznaczane jako przeciążone.

Wyliczanie średniej długości kolejki oparte jest na filtrze dolnoprzepustowym. W rezultacie krótkotrwałe zwiększenie długości kolejki, będące wynikiem chwilowego zwiększenia natężenia ruchu (*bursty traffic*), nie powoduje znacznego zwiększenia średniej długości kolejki, a w rezultacie strat pakietów. Średnią długość kolejki oblicza się korzystając z zależności (1):

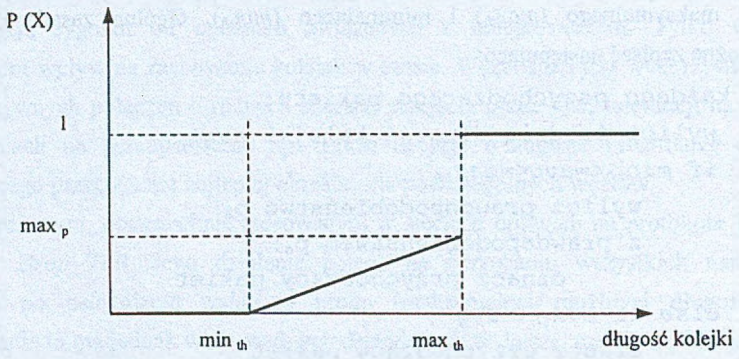
$$avg \leftarrow (1 - w_q)avg + w_q q \tag{1}$$

gdzie w_q jest stalą czasową filtru dolnoprzepustowego, a q jest chwilową długością kolejki.

Wyliczanie prawdopodobieństwa, z którym oznaczane są przychodzące pakiety przebiega zgodnie z zależnością (2):

$$\begin{aligned} p_b &\leftarrow \max_p (avg - \min_{th})/(\max_{th} - \min_{th}) \\ p_a &\leftarrow p_b/(1 - count \cdot p_b) \end{aligned} \tag{2}$$

gdzie *count* jest liczbą pakietów przesłanych przez węzeł od ostatnio oznaczonego jako przeciążony. Zależność tę ilustruje rys.1.



Rys. 1. Prawdopodobieństwo oznaczenia pakietu dla algorytmu RED
Fig. 1. RED marking packet function

2.1. Porównanie cech algorytmów RED i *Drop Tail* w sieci Internet

Powszechnie stosowany przy rozwiązywaniu przeciążeń w sieci Internet algorytm *Drop Tail* ma wiele wad, przyczyniających się w znacznym stopniu do obniżenia wydajności sieci. Można wśród nich wymienić:

- dla małych dopuszczalnych kolejek w węzłach niezdolność do wykrywania i przezwyciężania chwilowych stanów przeciążenia, a w przypadku dużych dozwolonych kolejek zbyt duże opóźnienia transmisji (szczególnie ważne dla usług wrażliwych na opóźnienia, takie jak telnet czy multimedialne usługi komunikacyjne),
- brak rozróżniania stanów: chwilowego i utrzymującego się stanu przeciążenia,
- efekt globalnej synchronizacji, który powstaje wtedy, gdy wiele połączeń w tym samym czasie otrzymuje informacje o przeciążeniu, w wyniku czego musi zredukować swoje okna transmisji,
- brak tolerancji dla połączeń charakteryzujących się wybuchowością (*bursty traffic*),
- niesprawiedliwy podział dostępnej przepustowości pomiędzy poszczególne połączenia.

Mechanizmu RED pozbawiony jest większości z wymienionych wad, a w szczególności charakteryzuje się następującymi cechami[2]:

- w zakresie kontroli ruchu algorytm gwarantuje, że średnia długość kolejki nie przekroczy określonego rozmiaru (dzięki usuwaniu przeciążonych pakietów),
- po wykryciu przeciążenia, reakcja w postaci zmniejszenia natężenia ruchu następuje przynajmniej po czasie zadziałania sprzężenia zwrotnego (*round trip*). Algorytm RED nie sygnalizuje stanu przeciążenia w chwilowych stanach zwiększenia natężenia ruchu (chwilowy wzrost długości kolejki),
- brak globalnej synchronizacji – intensywność, z jaką RED oznacza pakiety jako przeciążone, zależy od stopnia przeciążenia. Podczas nieznacznych przeciążeń prawdopodobieństwo usunięcia pakietu jest również niewielkie. Wzrost przeciążenia powoduje wzrost tego prawdopodobieństwa. Zapobiega to stanowi globalnej synchronizacji źródeł (dzięki oznaczaniu tak małej liczby pakietów, jak to możliwe),
- niewielka złożoność algorytmu RED, który może być łatwo implementowany w istniejących sieciach,
- duża skuteczność w wykorzystaniu przepustowości sieci – badania wykazały, że algorytm RED jest skuteczniejszy w wykorzystywaniu dostępnej przepustowości (*link utilization*) niż inne stosowane algorytmy (co wykazano m.in. w [1, 3]),
- optymalność (sprawiedliwość) – w dzielonym środowisku przepustowość jednego źródła zależy od realizacji żądań innych źródeł. Algorytm powinien sprawiedliwie rozdzielić dostępne pasmo pomiędzy poszczególne połączenia. W algorytmie RED

liczba oznaczanych pakietów danego połączenia jest proporcjonalna do wykorzystywanej przez to połączenie przepustowości. Algorytm RED dostarcza też mechanizmów kontroli i identyfikacji połączeń wykorzystujących nieproporcjonalnie dużą przepustowość, które to mechanizmy mogą być wykorzystane przez wyższe warstwy protokołów np. do zarządzania siecią,

- algorytm RED jest odpowiedni dla sieci z dużą liczbą aktywnych połączeń o różnych czasach reakcji na sprzężenie zwrotne.

2.2. Implementacja algorytmu

Jak pokazano w [1], algorytm RED może być implementowany w węzłach (routerach) nie powodując znacznego narzutu na liczbę instrukcji wykonywanych dla każdego przybywającego pakietu. Większość instrukcji związanych z algorytmem może być wykonywana równolegle w stosunku do czynności związanych z przełączaniem pakietów (obliczanie średniej długości kolejki oraz prawdopodobieństwo oznaczenia pakietu p_b).

Dla każdego przybywającego pakietu obliczana jest średnia długość kolejki zgodnie z zależnością (1). Można zauważyć, że jeśli w_q jest ujemną potęgą liczby 2, obliczenie tego równania można ograniczyć do jednej operacji przesunięcia i dwóch dodawania.

Jeśli pakiet przybywa do węzła, a średnia długość kolejki jest pomiędzy min_{th} a max_{th}, wartość prawdopodobieństwa jest obliczana zgodnie z zależnością (2). Można to zapisać następująco (3):

$$p_b \leftarrow C_1 avg - C_2$$

gdzie:

$$C_1 = \frac{\max_p}{\max_{th} - \min_{th}} \qquad (3)$$

$$C_2 = \frac{\max_p \cdot \min_p}{\max_{th} - \min_{th}}$$

Parametry C_1 i C_2 mogą zostać obliczone tylko raz, po zainicjalizowaniu pozostałych parametrów algorytmu: max_p, max_{th} i min_{th}. Gdy uda się dobrać parametry C_1 i C_2 jako ujemną potęgę liczby 2, wówczas podobnie jak w poprzednim przypadku, obliczenie prawdopodobieństwa można ograniczyć do prostych operacji przesunięcia i dodawania.

Zgodnie z najprostszą koncepcją algorytmu [1], kolejnym krokiem jest wylosowanie liczby z zakresu [0,1] i porównanie jej do obliczonego prawdopodobieństwa. Można jednak zrezygnować z tej operacji i ograniczyć się do odpowiednio dużej tablicy liczb losowych przechowywanych w pamięci. Kolejną możliwością zoptymalizowania szybkości działania

algorytmu jest określenie po każdym oznaczonym pakiecie następnego pakietu, który ma zostać zaznaczony (zamiast wykonywania operacji losowania dla każdego pakietu).

Algorytm RED znalazł swoje zastosowanie w wielu rzeczywistych rozwiązaniach sieciowych. Jako przykład można podać routery Cisco [7] (serii 1600, 2500, 3600, 4000, 7200 i in.). Wykorzystywany jest mechanizm WRED (*Weighted RED*) dostosowany do firmowych wymagań QoS (*Quality of Service*). Mechanizm ten wymaga osobnej konfiguracji [8], a jeśli nie zostanie przeprowadzona, domyślnie stosowany jest algorytm *Drop Tail*.

Algorytm RED, jako mechanizm uniwersalny dla sieci z przełączaniem pakietów, znajduje zastosowanie także w sieciach opartych na innych protokołach, np. w sieciach ATM [9].

Pewną modyfikacją algorytmu jest uwzględnianie przy obliczeniach nie liczby pakietów, lecz długości pakietów w bajtach. Zapobiega to jednakowemu traktowaniu pakietów o różnej długości, co może prowadzić do dedukcji okna transmisji dla połączeń o małych rozmiarach pakietów (np. telnet), które i tak mają niewielki wpływ na powstawanie przeciążeń (w porównaniu np. do przesyłów FTP).

Opracowano wiele wersji RED, które pozwalają osiągać lepszą wydajność sieci dla różnych przypadków i dla określonych klas ruchu. Porównanie skuteczności niektórych z nich zawiera [11]. Alternatywnym rozwiązaniem może być jednak przeznaczenie dla poszczególnych klas ruchu osobnych kolejek z algorytmem RED i dobranie dla każdej z nich odpowiednich parametrów [12].

2.3. Dobór parametrów algorytmu

Konsekwencją wynikającą z zastosowania algorytmu RED w sieci, zarówno opartych na TCP/IP, jak i dla innych protokołów, jest konieczność ustalenia przez administratora odpowiednich dla tego algorytmu parametrów: stałej w_q, maksymalnego prawdopodobieństwa oznaczenia pakietu max_p oraz minimalnego i maksymalnego progu kolejki (min_{th} i max_{th}). Dzięki temu jednak, korzystając z przeprowadzonych wcześniej badań, oraz oceniając przewidywany charakter ruchu, ma on możliwość optymalnego doboru tych parametrów w celu najlepszego wykorzystania łącza i zapewnienia odpowiednich parametrów jakościowych. Problem doboru parametrów jest omawiany w literaturze, m.in. w [1, 4, 5]. W przypadku algorytmu *Tail Drop* pozostaje do ustalenia tylko jeden parametr, którym jest rozmiar bufora.

Właściwe dobranie parametru w_q zależy od tego, jakie chwilowe zwiększenie natężenia ruchu można zaakceptować, bez odnotowywania stanu przeciążenia. Jeśli stała czasowa w_q jest zbyt duża, algorytm nie będzie mógł rozróżniać chwilowych stanów przeciążenia, a w rezultacie zmniejszy się stopień wykorzystania łącza. Jeśli w_q będzie zbyt małe, reakcja na wzrost długości kolejki będzie zbyt wolna, a w rezultacie nie będzie możliwe wykrycie

rozpoczynającego się stanu przeciążenia o trwałym charakterze. W praktyce zaleca się przyjmowanie wartości w_q w granicach pomiędzy 0,001 a 0,0042, a domyślną wartością zalecaną jest $w_q = 0{,}002$ [1, 5].

Optymalna wartość progów min_{th} i max_{th} zależy od wymaganej średniej długości kolejki, a to z kolei od zakładanego charakteru ruchu oraz dopuszczalnych opóźnień pakietów. Jeśli ruch charakteryzuje się wybuchowością, min_{th} musi być odpowiednio duże dla zapewnienia najlepszego wykorzystania łącza. Mniejsze wartości tego parametru zmniejszają opóźnienia pakietów, ale przyczyniają się do znacznie mniejszego wykorzystania dostępnego pasma. Problem optymalnego doboru tych parametrów jest wciąż przedmiotem badań.

Jeśli chodzi o parametr max_{th}, badania wykazały, że algorytm RED pracuje najefektywniej dla (max_{th} - min_{th}) większego, niż typowy wzrost średniej długości kolejki w czasie zadziałania sprzężenia zwrotnego (*round trip time*). W praktyce max_{th} powinno być przynajmniej dwukrotnością wartości min_{th} [1].

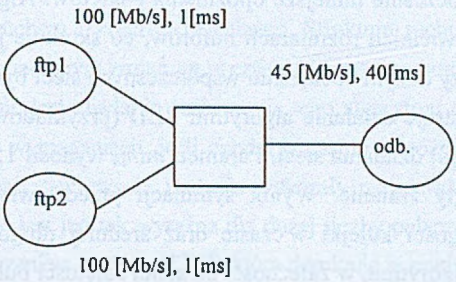
Określenie maksymalnego prawdopodobieństwa max_p, aby oznaczyć pakiet jako przeciążony, również jest przedmiotem badań. Ogólnie można powiedzieć, że wartość max_p jest zależna od poziomu zatłoczenia, który panuje w danej sieci. W pierwszej pracy [1] na temat algorytmu RED sugerowaną wartością było $max_p = 0{,}02$. Była to jednak wartość nieodpowiednia dla sieci wykorzystywanej przez bardzo dużą liczbę połączeń o stosunkowo małym natężeniu ruchu. Obecnie sugeruje się wartość $max_p = 0{,}1$ jako znacznie bardziej odpowiadającą rzeczywistemu ruchowi w sieci [5]. Nie wydaje się właściwe dalsze zwiększanie tej wartości. Gdy liczba przeciążonych (usuniętych) pakietów w sieci nadal będzie duża (mimo ustawienia $max_p = 0{,}1$), należałoby raczej zastanowić się nad zmianą innych parametrów ruchu lub konstrukcji samej sieci. Jednocześnie, zgodnie z [1], sugeruje się równomierny (*uniform*) rozkład zmiennej losowej.

3. Badania symulacyjne

Badania symulacyjne przeprowadzono z wykorzystaniem programu NS (*Network Simulator*) w wersji 2.0. NS był opracowywany i rozwijany od 1989 roku w ramach projektu VINT (*Virtual InterNetwork Testbed*). NS dostarcza wielu mechanizmów do symulacji sieci opartych na protokole TCP (zarówno przewodowych jak i bezprzewodowych).

Stworzona została konfiguracja testowa, dla przedstawienia działania algorytmu RED oraz porównania go z algorytmem *Drop Tail*. Konfiguracja testowa przedstawiona została na rys.2. Składa się z dwóch źródeł FTP, komunikujących się z jednym odbiorcą. Źródła przesyłają pakiet o wielkości 1000 bajtów, kiedy tylko pozwala im na to okno transmisji.

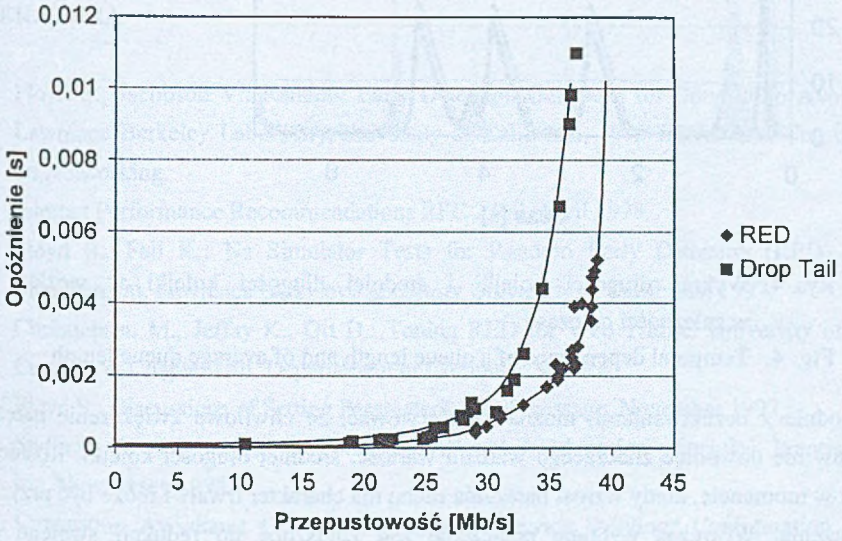
Pierwsze źródło FTP rozpoczyna transmisję w chwili rozpoczęcia symulacji, drugie po czasie t, który jest dobierany losowo. Maksymalne okno transmisji wynosi 240 bajtów.



Rys. 2. Model symulacyjny
Fig. 2. Simulation model

Parametry algorytmu RED są następujące: $max_p = 0{,}1$, $w_q = 0{,}02$, $max_{th}=3*min_{th}$.

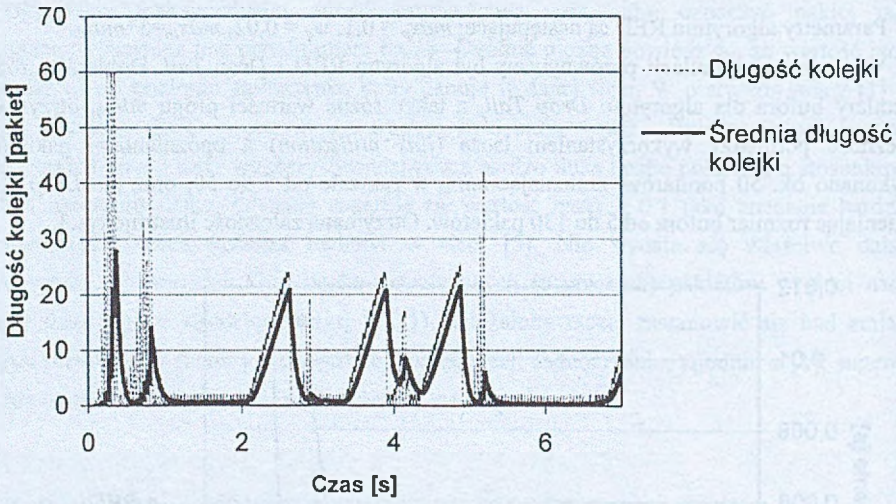
W pierwszej symulacji porównywany był algorytm RED i *Drop Tail*. Dobierając różne rozmiary bufora dla algorytmu *Drop Tail*, a także różne wartości progu min_{th}, otrzymano zależność pomiędzy wykorzystaniem łącza (*link utilization*) a opóźnieniami pakietów. Wykonano ok. 50 pomiarów, zmieniając min_{th} w zakresie od 5 do 50, oraz dla *Drop Tail*, zmieniając rozmiar bufora od 5 do 130 pakietów. Otrzymaną zależność ilustruje rys.3.



Rys. 3. Porównanie algorytmów Drop Tail i RED
Fig. 3. Comparing Drop Tail and RED algorithms

Dla badanego przypadku algorytm RED jest wyraźnie lepszy od *Drop Tail*. Dobierając odpowiednie parametry można uzyskać znacznie lepsze wykorzystanie dostępnego pasma transmisji, uzyskując jednocześnie mniejsze opóźnienia pakietów. Algorytmy zachowują się podobnie jedynie przy niewielkich rozmiarach buforów, co skutkuje jednak nieefektywnym wykorzystaniem łącza. Przy dużym obciążeniu współczesnych sieci byłoby to niepraktyczne.

Kolejna symulacja ilustruje działanie algorytmu RED (przykładowy przebieg czasowy). Czas symulacji wynosił 7 [s] działania sieci. Parametr min_{th} wynosił 15. Pozostałe parametry i topologia sieci nie uległy zmianie. Wynik symulacji przedstawiono na rys.4. Wykres przedstawia przebieg długości kolejki w czasie oraz średniej długości kolejki w czasie. Zgodnie z regułami tego algorytmu, w zależności od stanu zajętości bufora (długości kolejki), następuje odpowiednia reakcja w postaci odrzucenia pakietu, co powoduje konieczność jego retransmisji.



Rys. 4. Wykres długości kolejki i średniej długości kolejki w węźle w zależności od czasu

Fig. 4. Temporal dependence of a queue length and of average queue length

Zgodnie z oczekiwaniami, można zaobserwować, że chwilowe zwiększenie natężenia pakietów nie powoduje znaczącego wzrostu wartości średniej długości kolejki. Rośnie ona jednak w momencie, kiedy wzrost natężenia ruchu ma charakter trwały i może być przyczyną przeciążenia. Wówczas wybrane połączenie jest zmuszone do redukcji swojego okna transmisji.

4. Wnioski

Przedstawione symulacje dotyczą jedynie bardzo prostego modelu i nie oddają złożoności problemu sterowania ruchem i kontroli przeciążeń. Struktura ruchu internetowego jest bardzo złożona, ulega ona też znaczącym zmianom w czasie (np. rosnąca popularność usług o charakterze multimedialnym). Istotne jest zbadanie zachowania tego algorytmu dla różnych rodzajów ruchu sieciowego, o różnych wymaganiach, jeśli chodzi o pasmo i opóźnienia transmisji, a także dla rozbudowanych topologii sieciowych. Badania wykazały na przykład, że przewaga algorytmu RED nad *Drop Tail* nie jest już tak wyraźna dla dużej liczby połączeń o krótkim czasie trwania (jak np. transmisja związania z obsługą HTTP, która dominuje w sieci Internet) [4].

Przedstawione symulacje zwracają jedynie uwagę na zalety mechanizmu RED w stosunku do *Drop Tail*. Przy obecnej popularności sieci Internet, a także innych sieci opartych na protokołach z przełączaniem pakietów (ATM) problem maksymalnie efektywnego wykorzystania zasobów sieciowych przy zagwarantowaniu odpowiednich parametrów jakościowych, jest szczególnie istotny. Algorytm RED, dzięki swojej uniwersalności i dużym możliwościom konfigurowania, wydaje się obiecującym rozwiązaniem, czego dowodem mogą być liczne prace naukowe poświęcone tej tematyce, jak też jego implementacje w rzeczywistych urządzeniach sieciowych.

LITERATURA

1. Floyd S., Jacobson V.: Random Early Detection Gateways for Congestion Avoidance, Lawrence Berkeley Laboratory University of California, 1993 IEEE/ACM Transactions on Networking.
2. Internet Performance Recommendations RFC 2309, April 1998.
3. Floyd S., Fall K.: Ns Simulator Tests for Random Early Detection (RED) Queue Management. Lawrence Berkeley Laboratory University of California, 1997.
4. Christiansen M., Jeffay K., Ott D.: Tuning RED for Web Traffic. University of North Carolina at Chapter Hill, Departament of Computer Science,
5. Floyd S.: Discussions of Setting Parameters, email message, November 1997,
6. Stallings W.: High-speed networks TCP/IP and ATM design principles. Prentice-Hall, Inc. New Jersey 1998.
7. Congestion Avoidance Overview, Quality of Service Solutions Configuration Guide, Cisco Systems Inc, 1999.
8. Flow-Based RED, Cisco IOS Release 12.0(3)T, Cisco Systems Inc, 1999.
9. Elloumi O., Afifi H.: RED Algorithm in ATM Networks. Tech report, June 1997.

10. Fall K., Varadhan K.: The ns Manual. The VINT Project a Collaboration between researches at UC Berkeley, LBL, USC/ISI and Xerox PARC, Berkeley 2001.
11. Cnodder S., Elloumi O., Pauwels K.: Effect of different packet sizes on RED performance. Traffic and Routing Technologies project, Alcatel Corporate Research Center, Belgium.
12. Laalaoua R., Jędruś S., Czachórski T., Atmaca T.: Diffuzion model of RED control mechanism. Institut National des Telecommunications, France 1999.
13. Jain, R.: Congestion Control in Computer Networks: Issues and Trends. IEEE Network, May 1990.
14. Jain, R., and Ramakrishan, K.K.: Congestion Avoidance in Computer Network with a Conectionless Network Layer: Concepts, Goals, and Methodology. Proc. IEEE Comp. Networking Symp., Washington, D.C., April 1988.

Recenzent: Dr inż. Halina Kamionka-Mikuła

Wpłynęło do Redakcji 31 marca 2001 r.

Abstract

The paper presents characteristics of the Random Early Detection (RED) algorithm which is recommended by Internet Engineering Task Force as a queue management scheme for development and implementation, as it turns out that the end-to-end TCP window-based congestion control mechanism is not sufficient to ensure Internet stability and should by supplemented by router-based schemes. The algorithm RED is also useful for other networks based on packet-switching protocols.

The principle of RED mechanism is to start marking packets (setting congestion indicator bit or dropping, depending on current protocol) before the buffer is full, opposite to the politics of Tail Drop mechanism. The probability of discarding packets is given by the discard mark function, see Fig.1.

The RED algorithm avoids many problems that occur when Tail Drop is used as the congestion avoidance mechanism (e.g. global synchronization). The RED mechanism can be easy implement in switches (routers) with only a small add and shift instructions for each packet arrival.

When a network operator decides to use the RED algorithm as queue management mechanism he must set some parameters (maximum and minimum thresholds, maximum

marking probability and queue weight). By setting this set of parameters he can achieve desired network performance - packet delays and link utilization.

This paper presents some simulation. First simulation results. It compares RED algorithm with Drop Tail, and shows the advantage of the first (see Fig.3). The it displays the dynamic of the RED algorithm while shapeing network traffic shaper (see Fig.4).