Telekomunikacja Cyfrowa – Technologie i Usługi
Tom 8. Rok 2006/2007

Zliczanie danych TCP w przypadku odbioru przemieszanych segmentów

Andrzej Głowacz {glowacz@kt.agh.edu.pl}
Katedra Telekomunikacji Akademii Górniczo-Hutniczej – Kraków

STRESZCZENIE

Poprawne zliczanie przesłanych danych pozwala na efektywne szacowanie dostępnej przepustowości protokołu TCP oraz sterowanie przepływem danych. Niniejszy artykuł przedstawia analizę algorytmu zliczania danych zastosowaną w protokole TCP Westwood. Rozpatrując szczególne przypadki odbioru przemieszanych segmentów, wykazano, że wymieniona procedura może prowadzić do następujących błędów: ujemnej wartości szacowania, 100-procentowego chwilowego błędu próbkowania oraz niedoszacowania przepustowości. W rezultacie analizy została zaproponowana modyfikacja algorytmu zliczania danych dla zastosowania go w protokole TCP Cracow. Nowa metoda pozwala na znacznie dokładniejsze próbkowanie przepustowości i zwiększenie wydajności TCP dla przypadku odbioru przemieszanych segmentów.

ABSTRACT

TCP Data Counting for Out-of-Order Segments

Correct TCP data counting allows for efficient bandwidth estimation and congestion control. This paper presents an analysis of the data counting algorithm applied in TCP Westwood. Considering particular scenarios, where out--of-order data is received, we show that TCP Westwood data counting procedure can lead to following errors: negative bandwidth estimation, 100% momentary sampling error, and bandwidth underestimation. We propose a modified data counting algorithm for TCP Cracow. New method allows for significantly improved bandwidth sampling and improving TCP performance in out-of-order segments scenarios.

1. Wprowadzenie

Szacowanie ilości i jakości zasobów sieciowych jest w ostatnich latach wiodącym tematem wielu prac naukowych. Ten fakt wynika z rosnącej potrzeby zapewnienia efektywnych mechanizmów sterowania sieciami telekomunikacyjnymi. Mechanizmy te muszą działać w zróżnicowanych warunkach, a więc: dla różnych rodzajów sieci, topologi, prędkości transmisji oraz mechanizmów pośredniczących w różnych warstwach sieciowych. Znaczenie algorytmów szacowania jest czysto praktyczne – powinno zapewnić możliwie najwierniejsze odwzorowanie warunków panujących w sieci dla mechanizmów sterowania. Wiele współczesnych prac badawczych jest ściśle ukierunkowanych na praktyczne rozwiązania, np. realizowane w Katedrze Telekomunikacji AGH europejskie projekty zintegrowane DAIDALOS i DAIDALOS 2 dostarczające mechanizmy wspierające jakość usług oraz integrację sieci bezprzewodowych i przewodowych.

Protokół TCP, z racji swego umiejscowienia w warstwie transportowej, działa w oparciu o połączenia pomiędzy dwoma węzłami. W ten sposób jest on w działaniu niezależny od innych czynników np. typu sieci, jednak jego wydajność w istotny sposób zależy od tych czynników. Dla przykładu: TCP będzie działał zarówno w sieci przewodowej, jak i bezprzewodowej, ale błędy na niższych warstwach czy wpływ samopodobieństwa będą przenosić się na protokół w warstwie transportowej.

Dla TCP kluczowym parametrem decydującym o jego wydajności jest wielkość okna transmisyjnego. Jest ono ograniczone przez okno odbiorcy oraz okno przeciążeniowe po stronie nadawcy. Wielkość okna przeciążeniowego jest zmienna w czasie według reguły przyjętej dla konkretnej wersji protokołu, która najczęściej definiuje wielkość zmiany okna i zakres czasowy tej zmienności. Dodatkowe reguły sterowania oknem przeciążeniowym są wprowadzane do protokołu przez zasady retransmisji, które determinują działanie TCP w przypadku strat wysłanych segmentów. Również i tutaj, zasady retransmisji zależą od rozpatrywanej wersji protokołu.

Niezależnie od przyjętych mechanizmów sterowania, można założyć, iż dokładność szacowania zasobów sieciowych ma wpływ na efektywność sterowania. Rosnąca liczba przypadków zastosowań sprawia, że coraz więcej mechanizmów sieciowych musi adaptować się do zmiennych warunków. Naturalną konsekwencją tego faktu jest zwiększanie dokładności szacowania w celu zapewnienia lepszej adaptacji rozwiązań. Dla protokołu TCP oznacza to bardziej dokładne określanie strat segmentów, czasów transmisji oraz dostępnej przepustowości. W niniejszym artykule skupiono się nad ostatnią wielkością w przypadku scenariuszy z przemieszanymi segmentami dla protokołów TCP Westwood [3, 5, 7,] i TCP Cracow [4]

2. Szacowanie dostępnej przepustowości protokołu TCP

Protokół TCP w specyficzny sposób rywalizuje o dostęp do zasobów sieciowych na podstawie przebiegu wcześniejszej transmisji. Zazwyczaj to wielkość okna przeciążeniowego decyduje o poziomie nasilenia rywalizacji. W przypadku dostępnych zasobów sieciowych okno powinno być zwiększane, w przypadku nasycenia połączenia TCP okno powinno być utrzymywane, natomiast dla zmniejszających się zasobów okno powinno być redukowane. Poprawa szacowania dostępnej przepustowości może pozwolić na usprawnienie sterowania oknem we wszystkich wymienionych przypadkach. Pomijając efekt rywalizujących połączeń i wpływ innych warstw sieciowych, można powiedzieć, że idealny przypadek transmisji zachodzi, gdy okno przeciążeniowe jest równe, co do wartości iloczynowi dostępnej przepustowości (*BWE*) i opóźnienia transmisji (*RTT*). Iloczyn ten (*Bandwidth-Delay Product*, *BDP*) zmienia się szybko, gdyż zarówno dostępna przepustowość jak i opóźnienia nie pozostają na stałym poziomie.

Wśród protokołów TCP szacujących dostępną przepustowość interesujące rozwiązanie można odnaleźć w serii TCP Westwood. Szacowanie polega na obliczeniu próbek przepustowości, a następnie poddaniu ich dyskretnej filtracji dolnoprzepustowej po stronie nadawcy. W ten sposób oddziela się składowe: wolnozmienną i szybkozmienną. Składowa wolnozmienna odpowiada za nasilenie generowanego ruchu, zaś składowa szybkozmienna stanowi chwilowe wahania prędkości transmisji. W efekcie filtracji otrzymujemy oszacowanie dostępnej przepustowości dla danego połączenia. Z kolei samo obliczanie próbek opiera się na zliczaniu przesłanych danych w zadanej jednostce czasu.

3. Zliczanie przesłanych danych TCP

Skupiając się nad zliczaniem przesłanych danych przez algorytm *AckedCount* protokołu TCP Westwood, w kolejnych częściach artykułu rozważymy przypadek przemieszanych segmentów, tj. segmentów odbieranych w innej kolejności niż zostały nadane oraz zaproponujemy jego modyfikację.

3.1. TCP Westwood

Algorytm *AckedCount* protokołu TCP Westwood został przedstawiony w tabeli 1. Jego celem jest obliczenie liczby przesłanych danych (*cumul_ack*) dla skumulowanych potwierdzeń, także w przypadku przemieszania odbieranych segmentów. Procedura jest cyklicznie powtarzana dla kolejnych potwierdzeń, które są odbierane przez nadawcę. Istotne jest użycie numerów sekwencyjnych (SEQ) zawartych w nagłówkach segmentów TCP. Znając wartości SEQ dla ostatnio odebranego potwierdzenia (*current_ack_seqno*) oraz poprzedniego (*last_ack_seqno*) można obliczyć liczbę danych potwierdzanych. Jednakże mogą wystąpić odstępstwa od bezpośredniego obliczenia różnic SEQ. Tak dzieje się w przypadku odbierania zduplikowanych potwierdzeń, gdy nastąpi przemieszanie segmentów lub ich utrata. Wówczas odbiorca TCP wysyła potwierdzenie ostatniego poprawnie odebranego segmentu. Gdy zostanie odebrany oczekiwany segment, odbiorca wysyła skumulowane potwierdzenie uwzględniające dane, aż do ostatniego odebranego segmentu. W praktyce liczba przechowywanych segmentów zależy od wielkości bufora odbiorczego.

Tabela 1
Procedura zliczania przesłanych danych protokołu TCP Westwood

```
PROCEDURE AckedCount
cumul_ack = current_ack_seqno – last_ack_seqno;
(*)
if (cumul_ack == 0)
    accounted_for = accounted_for + 1;
    cumul_ack = 1;
endif
if (cumul_ack > 1)
    if (accounted_for ≥ cumul_ack)
(**)    accounted_for = accounted_for – cumul_ack;
        cumul_ack = 1;
    elseif (accounted_for < cumul_ack)
        cumul_ack = cumul_ack – accounted_for;
        accounted_for = 0;
    endif
endif
(***)
if (cumul_ack > 2)
    cumul_ack = 2;
endif
last_ack_seqno = current_ack_seqno;
return(cumul_ack);
END PROCEDURE
```

W przypadku przemieszania segmentów, należy uwzględniać przy szacowaniu dostępnej przepustowości jedynie bieżące segmenty. Liczba danych, które są wliczane do szacowania, będzie jednak mniejsza w przypadku, gdy nadejdzie zbiorcze potwierdzenie. Stąd użycie zmiennej określającej liczbę danych wliczonych do szacowania, a jeszcze niepotwierdzonych przez odbiorcę (*accounted_for*). W tabeli 1 gwiazdkami oznaczono miejsca modyfikacji algorytmu.

3.2. Przypadek nieaktualnego potwierdzenia

Zauważmy *(*)*, że jeżeli z jakichkolwiek przyczyn nadejdzie nieaktualne potwierdzenie, tj. z numerem SEQ mniejszym niż oczekiwane, wówczas

$$current\_ack\_seqno < last\_ack\_seqno,$$

a w rezultacie

$$cumul\_ack < 0.$$

W powyższym przypadku algorytm powinien niezwłocznie zwrócić wartość zerową. W przeciwnym przypadku ujemna wartość jest wliczana do szacowania, co w konsekwencji może doprowadzić nawet do ujemnej wartości szacowania[1]. Zatem w miejscu *(*)* należałoby użyć

```
(*)     if (cumul_ack < 0) return (0);
```

Powodem, dla którego nieaktualne potwierdzenie nie powinno być wliczane do szacowania dostępnej przepustowości jest fakt, że albo potwierdzane dane zostały już wliczone wcześniej albo nastąpiło wadliwe działanie protokołu.

3.3. Przypadek przemieszanych segmentów

Inny problem może wynikać z obliczenia *(**)*. Zauważmy, że w pewnych sytuacjach przyjęcie

```
accounted_for = accounted_for – cumul_ack;
```

może prowadzić do błędów.

Stwierdzenie to jest związane z przypadkiem przemieszanych segmentów, kiedy wymieniony fragment algorytmu będzie wykonywany. Aby je uzasadnić, przedstawmy pewną abstrakcję scenariusza symulacyjnego, w którym występują przemieszane segmenty. Rozważmy połączenie TCP pomiędzy parą węzłów: nadawcą i odbiorcą. Przemieszane segmenty mogą wystąpić w sieci, w której dane docierają różnymi trasami. Dzieje się tak zarówno w sytuacji działania mechanizmów równoważenia obciążenia, jak i mechanizmów doboru trasy. Przykładem może być sieć bezprzewodowa WLAN (*Wireless Local Area Network*), zgodna ze standardem IEEE 802.11g [2].

Ilustracja rozważanego scenariusza została przedstawiona na rysunku 1. Dla celów symulacji użyto symulatora sieciowego *ns-2* [6]. Eksperyment polegał na przeprowadzeniu 10 jednoczesnych połączeń FTP pomiędzy różnymi węzłami sieci ad-hoc IEEE 802.11g. Dla uproszczenia prezentacji przyjęto, że segmenty *SEQ = 2* i *SEQ = 4* są opóźnione i zostają odebrane w innej kolejności niż zostały nadane. Uproszczeniem jest także sposób numeracji, gdzie segmenty przedstawiono kolejno, a nie według numeracji bajtów danych. Także odległości na osiach czasu nie odzwierciedlają rzeczywistego upływu czasu, ale jedynie kolejność zdarzeń.

Niech A_{in} i A_{out} będą dla algorytmu wejściowymi i wyjściowymi parametrami *accounted_for*, zaś C_{in} i C_{out} odpowiednio wejściowymi i wyjściowymi wartościami *cumul_ack*. Załóżmy także, iż wielkość okna przeciążeniowego wynosi 8 segmentów, tj. $C_{\text{in}} = 8$ [segmentów]. Warto dodać, że powyższy przykład jest uproszczeniem zjawiska obserwowanego podczas badania sieci WLAN w symulatorze *ns-2*. Z reguły okno przeciążeniowe może osiągnąć znacznie większe wartości niż rozważana, co jednak byłoby trudne do jasnego przedstawienia na rysunku.

Nadawca nadaje segmenty przypisując im kolejne numery *SEQ* = 1, 2, ..., 8, ale kolejność odbioru jest zaburzona i w oknie pojawiają się dwie luki: pierwsza w miejscu segmentu *SEQ* = 2 a druga dla *SEQ* = 4.

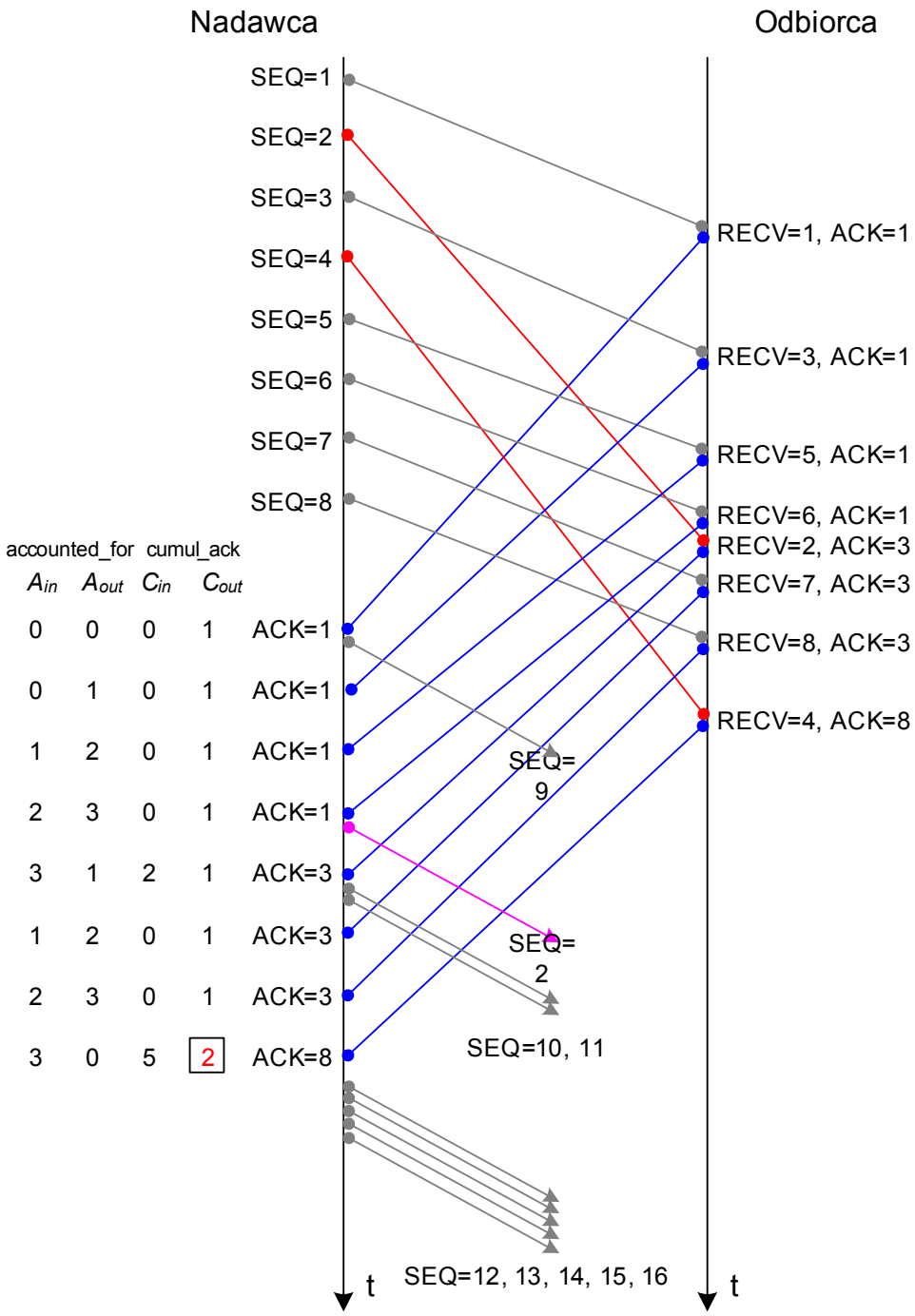
Pierwszy segment przybywa w kolejności, zatem $A_{\text{out}} = 0$ oraz $C_{\text{out}} = 1$. W tym momencie nadawca wysyła kolejny segment *SEQ* = 9. Ponieważ trzeci segment przybywa przed drugim, nadawca wysyła zduplikowane potwierdzenie (*ACK* = 1), u nadawcy A_{out} zwiększa się do 1, zaś C_{out} wynosi 1. Sytuacja powtarza się, aż do otrzymania skumulowanego potwierdzenia lub wejścia w procedurę szybkiej retransmisji [1]. W rozpatrywanym scenariuszu najpierw występuje szybka retransmisja drugiego segmentu. W tym kroku wartość A_{out} zwiększa się do 3.

Powodem chwilowego braku potwierdzenia dla segmentu drugiego jest jego opóźnienie, a nie utrata. Zatem w momencie odbioru potwierdzenia w następnym kroku zajdzie $C_{\text{in}} = 2$ oraz $A_{\text{in}} = 3$, więc $C_{\text{out}} = 1$, zaś A_{out} zmniejszy się do wartości $A_{\text{in}} - C_{\text{in}} = 1$. Mogłoby to oznaczać, że tylko jeden segment został wliczony do szacowania, podczas gdy w rzeczywistości były to dwa segmenty. Dla transmisji zwolnienie bufora nadawcy (segmenty *SEQ* = 2 and *SEQ* = 3) pozwala wysłać nowe segmenty. Chociaż okno przeciążeniowe może zostać zmniejszone i w rezultacie nowe transmisje mogą nie być dopuszczone, to jednak nie wpływa to na ogólność rozważań na temat rozpatrywanego algorytmu zliczania danych.

Podczas gdy odbierane są kolejne potwierdzenia (dla segmentów *SEQ* = 7 and *SEQ* = 8), A_{out} ponownie osiąga wartość 3. Ostatecznie czwarty segment również zostaje odebrany i przybywa jego potwierdzenie. Otrzymujemy, zatem $C_{\text{in}} = 5$ oraz $A_{\text{in}} = 3$, co daje w efekcie wyzerowanie A_{out} i $C_{\text{out}} = 2$. Najistotniejszy jest efekt końcowy: oznacza on, że potwierdzenie *ACK* = 8 zostaje wliczone do szacowania jako dwa segmenty. Zauważmy, że w rzeczywistości jest to potwierdzenie tylko jednego segmentu (*SEQ* = 4). Wliczenie dwóch segmentów może być przyczyną przeszacowania dostępnej przepustowości dla sieci, w której wystąpi znaczna liczba przemieszanych segmentów. W takim przypadku żadna filtracja próbek nie będzie dawała poprawnych wyników.

Po zakończeniu trybu retransmisji, a w naszym przypadku przesłania całego początkowego okna ośmiu segmentów (*SEQ* = 1, 2, ..., 8), kolejne segmenty mogą być przesłane o ile pozwala na to okno przeciążeniowe. Nie wpływa to na zachowanie algorytmu zliczania.

[1] Wymaga to wprawdzie spełnienia pewnych dodatkowych wymagań, co zostało szerzej opisane w pracy [4].



Rys. 1. Zliczanie danych TCP Westwood dla przemieszanych segmentów

Jak pokazano na przykładzie, końcowa wartość C_{out} jest błędna dla algorytmu TCP Westwood. W efekcie działania, aż dziewięć segmentów zostało wliczonych do algorytmu szacowania dostępnej przepustowości, podczas gdy tylko osiem zostało wysłanych. Oznacza to, że przeszacowanie próbek przepustowości przekazane do dalszego szacowania w okresie transmisji całego okna wynosi 9 / 8 – 1 = 12,5%. Jeśli jednak przyjąć, iż dokładne próbkowanie ma największe znaczenie w chwili odbioru potwierdzenia *ACK* = 8, wówczas skoro wliczone zostały dwa segmenty zamiast jednego to błąd wyniesie aż 2 / 1 – 1 = 100%. To oznacza, że znaczne chwilowe przeszacowanie dostępnej przepustowości może doprowadzić do ustawienia zbyt dużej wartości okna przeciążeniowego, a w konsekwencji do przeciążenia i niepotrzebnych strat segmentów. Dodatkowo, mogą się pojawić trudności podczas retransmisji, ponieważ już raz błędnie obliczone parametry zostaną ponownie użyte do obliczenia okna przeciążeniowego, co w szczególnym przypadku może powodować kolejne straty segmentów.

Rozwiązaniem problemu jest ustawienie parametru *accounted_for* zgodnie z formułą

(**) accounted_for = accounted_for - cumul_ack + 1;

w sytuacji, gdy odebrane zostanie potwierdzenie opisane zależnością

$$1 < cumul\_ack \leq accounted\_for.$$

3.4. Przypadek znacznego przemieszania segmentów

Ostatni rozpatrywany przypadek zachodzi, gdy wartość wyjściowa *cumul_ack*, a więc C_{out}, będzie większa niż 2. Odpowiada to sytuacji, gdy kilka przesłanych segmentów zostało wliczonych do szacowania, ale ich potwierdzenia nie zostały jeszcze odebrane, choć znajdują się w drodze. Wówczas, może się zdarzyć odebranie potwierdzenia za więcej niż dwa przesłane segmenty.

Dla TCP Westwood wartość *cumul_ack* jest ograniczona algorytmem zliczania. W rzeczywistości założenie to jest błędne, a jego źródeł można doszukać się w algorytmie szacowania dostępnej przepustowości. Mianowicie, szacowanie oparte na parach sąsiednich potwierdzeń TCP Westwood powodowałoby znaczne błędy dla większych wahań wartości *cumul_ack* nawet, gdy wartości *cumul_ack* będą obliczane w poprawny sposób, całkowicie zgodny z przebiegiem transmisji. Ograniczenie *cumul_ack* pozwala zmniejszyć znaczenie tego typu błędów szacowania, jednak nie jest optymalne, bo informacja o części próbek zostaje w ten sposób utracona. W celu zmniejszenia błędu wprowadzonego przez ograniczenie *cumul_ack*, dla TCP Westwood konieczna staje się dalsza modyfikacja filtru dolnoprzepustowego tak, aby filtrowanie było silniejsze, co uzyskuje się przez zwiększenie stałej czasowej filtru. W efekcie tego, czas odpowiedzi filtru wydłuża się, zatem protokół nie może zareagować na zmianę zasobów odpowiednio szybko.

Dla TCP Cracow przyjmujemy, że poprawnie obliczona wartość *cumul_ack* nie może być ograniczana dodatkowymi warunkami, gdyż informacja o przepustowości byłaby tracona. Zamiast tego szacowanie przepustowości powinno się odbywać w oparciu o dłuższy interwał czasowy, w którym zliczane są próbki przepustowości. Dla TCP Cracow za podstawę czasową przyjmowany jest czas SRTT [4]. Pozwala to na niezależność procedury filtracji od metody próbkowania, zatem umożliwia jednoczesną filtrację z małą stałą czasową. Taka filtracja próbek daje możliwość szybkiej reakcji protokołu TCP Cracow na zmiany dostępnej przepustowości, a w konsekwencji większe możliwości jego adaptacji.

Uwzględniając problemy występujące dla TCP Westwood w szczególnych przypadkach zastosowań warunków *(*)* i *(**)*, można zrozumieć motywację przy wprowadzeniu ograniczenia *(***)*. Dla TCP Cracow problemy opisane powyżej zostały wyeliminowane, zatem naturalne jest wyłączenie warunku *(***)*. Całość proponowanej procedury zliczania przesłanych danych dla TCP Cracow została przedstawiona w tabeli 2.

Tabela 2
Proponowana procedura zliczania przesłanych danych protokołu TCP Cracow

```
PROCEDURE AckedCount
cumul_ack = current_ack_seqno – last_ack_seqno;
if (cumul_ack < 0) return (0);
if (cumul_ack == 0)
    accounted_for = accounted_for + 1;
    cumul_ack = 1;
endif
if (cumul_ack > 1)
    if (accounted_for ≥ cumul_ack)
        accounted_for = accounted_for – cumul_ack + 1;
        cumul_ack = 1;
    elseif (accounted_for < cumul_ack)
        cumul_ack = cumul_ack – accounted_for;
        accounted_for = 0;
    endif
endif
last_ack_seqno = current_ack_seqno;
return(cumul_ack);
END PROCEDURE
```

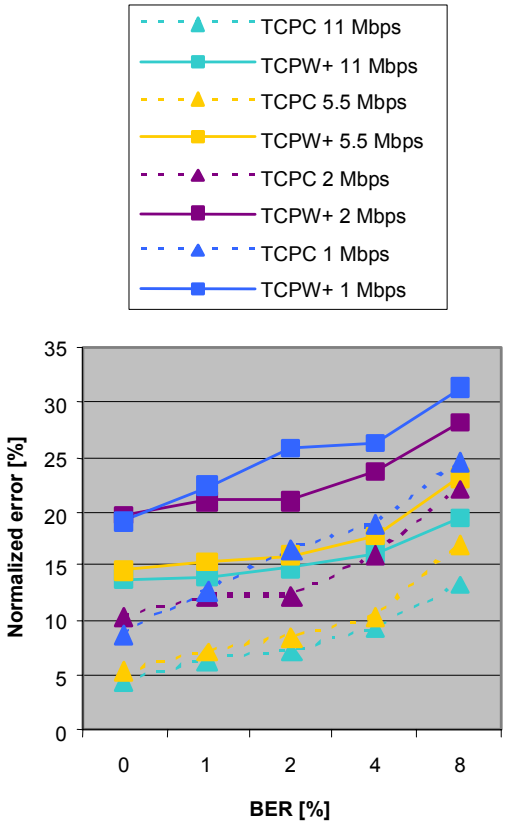
4. Wyniki symulacyjne

Przedstawienie wyników ilościowych dla opisanego zjawiska jest trudne. Wynika to z faktu, że mechanizmy sterowania protokołu TCP są ze sobą mocno powiązane i muszą być badane całościowo. Możliwe i miarodajne jest natomiast przedstawienie dokładności szacowania protokołu TCP Cracow w porównaniu do TCP Westwood.

Przyjmując w scenariuszu *ns-2* sieć ad-hoc IEEE 802.11g, składającą się z 20 węzłów, w której dokonywane jest 10 równoczesnych 200-sekundowych sesji FTP, możliwe jest określenie różnicy rzeczywistej przepustowości *BW* i przepustowości oszacowanej *BWE* dla każdego z protokołów. Przyjęto że jest to bezwzględny błąd szacowania *D*, który może zostać przyrównany do rzeczywistej przepustowości (*BW*), osiągniętej przez każdy z protokołów, dając znormalizowany błąd szacowania przepustowości *NE* (ang. *Normalized Error*). Błędy znormalizowane dla różnych protokołów mogą już być ze sobą porównywane.

W scenariuszu przeprowadzono serię symulacji dla różnych prędkości sieci ad-hoc *R* = {1; 2; 5,5; 11} Mbps oraz różnych bitowych stóp błędów *BER* = {0;

1; 2; 4; 8}%. W ten sposób możliwe jest określenie właściwości protokołów TCP w różnych warunkach. Wyniki zestawiono na rysunku 2.

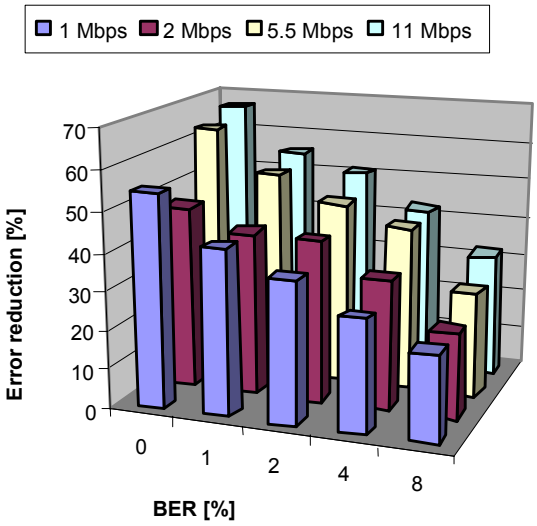


Rys. 2. Znormalizowany błąd szacowania w bezprzewodowej sieci ad-hoc

Generalnie, wzrost stopy błędów powoduje także spadek rzeczywistej przepustowości. Jak można jednak zauważyć na rysunku 2, znormalizowane błędy szacowania rosną wraz ze wzrostem bitowej stopy błędów i mogą osiągać 4–24% dla TCP Cracow oraz 14–32% dla TCP Westwood w badanym scenariuszu. Z drugiej strony, znormalizowane błędy szacowania maleją wraz ze wzrostem prędkości sieci. Wprowadźmy parametr „redukcja błędu" (ang. *Error Reduction*) *ER*, gdzie $ER = 100\ (1 - NE_{TCPC} / NE_{TCPW})$ określający, o ile został zmniejszony błąd znormalizowany NE_{TCPW} mechanizmów szacowania TCP Westwood, dzięki przejściu na zastosowanie mechanizmów TCP Cracow (błąd NE_{TCPC}).

Porównując ze sobą odpowiednie błędy znormalizowane dla TCP Cracow i TCP Westwood otrzymujemy wyniki zestawione na rysunku 3.

Redukcja błędów wynosi 20–65%, przy czym maleje ona wraz ze wzrostem bitowej stopy błędów oraz w przybliżeniu rośnie wraz ze wzrostem prędkości sieci. Zastosowanie mechanizmów TCP Cracow daje ogólnie zwiększoną precyzję szacowania dostępnej przepustowości, ponieważ redukcja jest w każdym przypadku większa od 0%. Dodatkowo, redukcja błędu jest tym większa, im szybsza jest sieć, w której zastosujemy TCP Cracow (zauważmy, że spadek bitowej stopy błędów daje również wzrost rzeczywistej przepustowości). A zatem, korzyść ze stosowania TCP Cracow będzie z czasem rosła, z uwagi na ciągły wzrost prędkości technik transmisji bezprzewodowej. Wnioskiem ogólnym z pracy jest stwierdzenie, że nowe sieci z pewnością wymagają coraz lepszych i dokładniejszych mechanizmów sterowania, a TCP Cracow jest w stanie takie mechanizmy zapewnić.



Rys. 3. Redukcja błędu szacowania dostępnej przepustowości TCP w sieci ad-hoc

5. Podsumowanie

Na podstawie przedstawionych w niniejszym artykule rozważań można stwierdzić, że odpowiednie zliczanie danych TCP może mieć duże znaczenie w przypadku, gdy wysłane segmenty docierają w przemieszanej kolejności. Działanie algorytmu zliczania danych ma wpływ na działanie algorytmu szacowania dostępnej przepustowości, a ten z kolei na mechanizm sterowania oknem przeciążeniowym.

Na przykładach: nieaktualnego potwierdzenia, przemieszanych segmentów oraz znacznego przemieszania segmentów przedstawiono problemy, jakie występują w procedurze zliczania przesłanych danych protokołu TCP Westwood. W szczególnych sytuacjach pojawiają się tutaj błędy próbkowania. Błędy próbkowania zostały wyeliminowane w protokole TCP Cracow, co pozwala także na poprawę mechanizmu szacowania.

Literatura

[1] Allman M., Paxson V., *Stevens W.: TCP Congestion Control.* Network Working Group RFC 2581. Internet Enginnering Task Force, www.ietf.org, 1999

[2] ANSI/IEEE. 802.11g Standard, 802.11 supplement: *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band*. IEEE Standards Association, standards.ieee.org, 2003

[3] Ferrorelli L., Grieco L. A., Mascolo S., Piscitelli G., Camarda P.: *Live Internet Measurements Using Westwood+ TCP Congestion Control*. Proc. IEEE Globecom, Taipei, Taiwan 2002

[4] Głowacz A.: *TCP Bandwidth Estimation for a Heterogeneous Wireless Network*. Rozprawa doktorska, AGH, Kraków, 2006

[5] Mascolo S., Casetti C., Gerla M., Sanadidi M.Y., Wang R.: *TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links*. Proc. ACM/IEEE International Conference on Mobile Computing and Networking, Rome, Italy, 2001, 287-297

[6] McCanne S., Floyd S.: *Network Simulator version 2.29*. USC School of Engineering., Information Sciences Institute, www.isi.edu/nsnam, 2006

[7] Wang R., Valla M., Sanadidi M.Y., Gerla M.: *Adaptive Bandwidth Share Estimation in TCP Westwood*. Proc. IEEE Globecom, Taipei, Taiwan 2002

Andrzej Głowacz – urodzony 7 kwietnia 1978 roku w Krakowie. W 2002 roku uzyskał tytuł magistra inżyniera w Katedrze Telekomunikacji na Wydziale Elektrotechniki, Automatyki, Informatyki i Elektroniki Akademii Górniczo-Hutniczej i otrzymał dyplom z wyróżnieniem. Obecnie uczestnik Studiów Doktoranckich na kierunku Informatyka Wydziału EAIiE. Badania dotyczące najnowszych wersji protokołu TCP stanowią kontynuację pracy magisterskiej wyróżnionej w czterech konkursach, w tym II nagrodą w XIX Ogólnopolskim Konkursie na najlepsze prace magisterskie z Informatyki oraz najlepszą teoretyczną pracę na Wydziale EAIiE w IV Edycji konkursu „Diamenty" AGH.