Michał KARBOWAŃCZYK

ZAPOBIEGANIE PRZECIĄŻENIOM W SIECI IP POPRZEZ STEROWANIE SĄSIEDNIMI WĘZŁAMI

STRESZCZENIE *Jako że IPv4 jest powszechnie stosowanym protokołem sieciowym, poprawa efektywności jego działania poprzez stosowanie kontroli przepływu stanowi interesujący kierunek prac badawczych. Ogólnym celem prac przedstawionych w niniejszym rozdziale jest zapobieganie występowaniu zjawiska przeciążeń w sięciach IPv4 przy założeniu, że działanie samego protokołu IPv4, jak i protokołów warstw wyższych, nie może być modyfikowane. Zadanie to jest o tyle trudne, że w protokole IPv4 nie przewidziano mechanizmu umożliwiającego zestawianie pętli zwrotnej, przy pomocy której węzły sieci mogłyby dostarczać do nadawcy informacje o swoim stanie i zdolnościach przesyłania. W rozdziale zaproponowano dwa mechanizmy rozszerzające działanie stosu sieciowego jądra Linux, umożliwiające zestawianie pętli zwrotnych pomiędzy danym węzłem sieci a węzłami sąsiednimi (tzw. pętle węzeł – węzeł) oraz kontrolowanie przesyłania danych przez węzły sąsiednie. Uzasadniono także zastosowanie mechanizmu sterującego w warstwie IP oraz wybór sterowania węzeł – węzeł.*

Słowa kluczowe: *sieci teleinformatyczne, kontrola przepływu, zapobieganie przeciążeniom.*

---

dr inż. Michał KARBOWAŃCZYK
e-mail: michal.karbowanczyk@p.lodz.pl

Politechnika Łódzka,
Instytut Informatyki, Zakład Sieci Komputerowych

PRACE INSTYTUTU ELEKTROTECHNIKI, zeszyt 249, 2011

1. WSTĘP

Wraz z ciągle poszerzającym się obszarem zastosowań sieci Internet zarówno w działalności biznesowej jak i w rozrywce, zaobserwować można ciągły wzrost ilości transmitowanych danych, a co za tym idzie – rosnące wymagania odnośnie prędkości i niezawodności transmisji. Aby spełnić te wymagania, stosuje się rozmaite mechanizmy kontroli przepływu (ang. traffic control, flow control), pozwalające na dostosowanie prędkości wysyłania danych przez nadawcę do aktualnego stanu i możliwości węzłów sieci biorących udział w transmisji (węzłów pośredniczących). Istotnym aspektem kontroli przepływu jest zapobieganie występowaniu przeciążeń w węzłach pośredniczących. Przez przeciążenie rozumie się tu sytuację, w której bufor służący węzłowi do przechowywania pakietów zostaje przepełniony i węzeł nie ma możliwości odebrania kolejnego pakietu. Bezpośrednim skutkiem przeciążenia jest utrata pakietów, która z reguły pociąga za sobą konieczność ich retransmisji. Retransmisja z kolei skutkuje spadkiem efektywności wykorzystania łącz oraz zjawiskiem samosynchronizacji nadawców, które może prowadzić do eskalacji całego problemu.

Celem pracy przedstawionej w niniejszym rozdziale jest stworzenie takiego rozwiązania kontroli przepływu w sieci IPv4, które zapewniałoby eliminację przeciążeń w węzłach sieci, i mogłoby być zastosowane w wybranym obszarze sieci *bez konieczności modyfikowania sposobu działania istniejących protokołów sieciowych oraz aplikacji z nich korzystających*. Założenie o niemodyfikowaniu sposobu działania istniejących protokołów i aplikacji, zwane w dalszej części rozdziału *założeniem głównym*, w znaczącym stopniu ogranicza wybór sposobu rozwiązania problemu.

Dalsza część rozdziału jest zorganizowana w sposób następujący:

- w podrozdziale 2. uzasadniono wybór warstwy, w której zastosowana została kontrola przepływu (IPv4) oraz wybór podejścia do zestawiania pętli zwrotnych (węzeł-węzeł);
- w podrozdziale 3. przedstawiono zaproponowane rozwiązania mechanizmu pętli zwrotnej oraz algorytmu sterowania przepływem;
- w podrozdziale 4. przedstawiono wyniki eksperymentów przeprowadzonych w sieci rzeczywistej.

2. PRZYJĘTE ZAŁOŻENIA

2.1. Wybór protokołu IPv4

Wybór warstwy (w rozumieniu modelu warstwowego ISO/OSI), w której zastosowana zostanie kontrola przepływu, ma istotne znaczenie dla zasięgu i skuteczności jej działania. W pierwszej kolejności należy zauważyć, że warstwy: sesji, prezentacji i aplikacji są wykorzystywane przez własne protokoły samych aplikacji i próba zaproponowania rozwiązania dla którejkolwiek z tych warstw stałaby w sprzeczności z założeniem o niemodyfikowaniu działania aplikacji. W obszarze naszych zainteresowań pozostaje zatem warstwa transportowa i warstwy niższe.

Rozważając zastosowanie kontroli przepływu w warstwie transportowej należy stwierdzić, iż powszechnie używanymi protokołami w tejże warstwie są: TCP i UDP.

Protokół UDP nie został wyposażony w żaden mechanizm kontroli przepływu. Istnieją wprawdzie propozycje takowych mechanizmów, jednakże bazują one na wprowadzeniu komunikacji pomiędzy końcami konwersacji (nadawcą i odbiorcą), która musi być zrealizowana albo poprzez modyfikację działania protokołu UDP, albo poprzez modyfikację działania aplikacji korzystających z tego protokołu; w każdym z przypadków oznacza to naruszenie przyjętego założenia głównego.

Protokół TCP wyposażony został w mechanizm okna, dzięki któremu prędkość, z jaką nadawca transmituje dane, jest dostosowywana do stanu i możliwości transmisji danych przez węzły pośredniczące. Mechanizm ten jednak bezpośrednio opiera się na doprowadzaniu do przeciążeń i ich obserwowaniu, a zatem jest podatny na występowanie wzmiankowanych we wstępie niekorzystnych zjawisk. Protokół TCP z racji na powszechność jego wykorzystania jest obiektem licznych badań, które zaowocowały szeregiem modyfikacji, przyczyniających się do poprawy jego efektywności, jednakże zasada działania i jej konsekwencje pozostają niezmienne [6]. Zaproponowano także wiele rozwiązań polegających na modyfikacji nie protokołu TCP, lecz sposobu zarządzania kolejkami pakietów (AQM, ang. Active Queue Management) w buforach węzłów pośredniczących tak, aby zminimalizować negatywne skutki wystąpienia przeciążeń [3].

Istnieje oczywiście szereg propozycji innych niż TCP i UDP protokołów warstwy transportowej, często charakteryzujących się znakomitymi właściwościami,

jak chociażby SCTP (ang. Stream Control Transmission Protocol) [4]. Rozwiązania takie jednakże wymagają wsparcia ze strony aplikacji z nich korzystających, co stoi w sprzeczności z przyjętym założeniem głównym.

Kończąc rozważania dotyczące warstwy transportowej należy zauważyć, że zastosowanie nawet doskonałego mechanizmu kontroli przepływu w jednym z protokołów tej warstwy nie daje efektu w odniesieniu do innych protokołów tej warstwy. Biorąc pod uwagę ten problem, atrakcyjną alternatywą wydaje się być zastosowanie kontroli przepływu w warstwie łącza danych i fizycznej. Istotnie, zastosowanie kontroli przepływu w tej warstwie przynosi efekt dla wszystkich protokołów warstw wyższych bez konieczności ich modyfikacji. Mechanizmy oparte o protokoły warstwy transportowej posiadają jednak wadę w postaci ograniczenia ich efektywności do homogenicznego segmentu sieci, podczas gdy w sieci Internet typową sytuacją jest przenoszenie pakietu od nadawcy do odbiorcy poprzez szereg segmentów zbudowanych w oparciu o różne technologie warstwy transportowej.

Podsumowując powyższe informacje należy stwierdzić, iż jedyną warstwą, w której można zaproponować mechanizm oddziałujący na wszystkie protokoły warstwy transportowej i jednocześnie dający się zastosować w wybranym obszarze sieci Internet, niekoniecznie ograniczonym do pojedynczego segmentu fizycznego, jest warstwa sieciowa. Spowodowało to wybranie powszechnie stosowanego w tej warstwie protokołu IPv4, jako obszaru działania zaproponowanego w tym rozdziale mechanizmu kontroli przepływu.

2.2. Wybór zasięgu pętli zwrotnej

Rozważmy przepływ danych przez sieć przełączającą pakiety, jaką jest sieć IPv4. W transmisji danych od nadawcy do odbiorcy bierze udział zestaw węzłów pośredniczących. W konfiguracji tej możliwe są dwa zasadnicze podejścia do zestawienia pętli zwrotnej dla celu sterowania przepływem: koniec-koniec oraz węzeł-węzeł (ang. end-to-end, hop-by-hop).

W podejściu koniec-koniec pętla zwrotna jest zestawiona pomiędzy odbiorcą a nadawcą, zaś węzły pośredniczące nie biorą bezpośredniego udziału w procesie sterowania przepływem.

Wariantem podejścia koniec-koniec jest często stosowany w pracach teoretycznych model wąskiego gardła (ang. bottleneck node). W modelu tym pętla zwrotna jest zestawiona pomiędzy odbiorcą a nadawcą, jednakże zakłada się w nim istnienie węzła, którego zdolność do transmisji danych jest najgorsza z całego przepływu, inaczej mówiąc, stanowi on wąskie gardło dla przepływu. Węzeł ten uczestniczy w procesie kontroli przepływu poprzez dostarczanie informacji o swoim stanie.

Alternatywą dla podejścia koniec-koniec jest podejście węzeł-węzeł, w którym pętla zwrotna jest zestawiona pomiędzy każdą parą węzłów sąsiednich. Jako węzły sąsiednie rozumiemy tu węzły połączone ze sobą bezpośrednio łączem warstwy niższej, bez pośrednictwa innych węzłów tej samej warstwy.

Wybór podejścia do zestawiania pętli zwrotnych dla sterowania przepływem ma kluczowe znaczenie dla działania algorytmu sterującego. Jeżeli bowiem pętla zwrotna jest zrealizowana poprzez wysyłanie danych w sieci teleinformatycznej, to należy uwzględnić opóźnienie w przekazywaniu danych sterujących pomiędzy końcami pętli. Opóźnienie to w rozważanym przypadku jest określane jako czas krążenia pakietu (*RTT*, ang. Round Trip Time). Nie ulega wątpliwości, że lepsze właściwości algorytmu sterującego uzyskuje się dla niższych opóźnień w pętli zwrotnej; zostało to przedstawione w [1, 2, 5] na przykładzie algorytmów opartych o predyktor Smitha. Jeżeli rozważyć pętle zwrotne koniec-koniec oraz węzeł-węzeł dla tego samego przepływu, to oczywiście opóźnienie w każdej z pętli węzeł-węzeł będzie niższe niż w przypadku pętli koniec-koniec. Ponadto należy zauważyć, że w przypadku sieci przełączającej pakiety ścieżka przepływu, czyli zestaw węzłów, przez które pakiety są przesyłane od nadawcy do odbiorcy, może zmienić się bez konieczności zakończenia tego przepływu, przez co może dojść do sytuacji, gdy dane sterujące dostarczone do nadajnika mogą być już nieaktualne. Powyższe argumenty zdecydowały o wyborze pętli zwrotnej węzeł-węzeł dla proponowanego rozwiązania sterowania przepływem.

3. PROJEKT I IMPLEMENTACJA

3.1. Pętla zwrotna

Realizacja pętli zwrotnej pomiędzy pewnym węzłem sieci IPv4 a węzłami sąsiednimi wymaga co najmniej: zidentyfikowania przepływów, które aktualnie są transmitowane za pośrednictwem rozpatrywanego węzła oraz zidentyfikowania (ustalenia adresów) węzłów sąsiednich, które transmitują do rozpatrywanego węzła zidentyfikowane przepływy

Należy zauważyć, iż dostarczanie wymienionych informacji nie należy do funkcji węzła sieci IP (rutera), zatem wymagane jest zaimplementowanie dodatkowego mechanizmu. W przypadku jądra Linux mechanizm śledzenia przepływów zaimplementowany jest od wersji 2.4. Mechanizm ten nie rejestruje

jednak adresów węzłów sąsiednich biorących udział w transmisji danych dla przepływu, dlatego konieczne było jego rozszerzenie. Ponieważ w protokole IPv4 nie przewidziano żadnej formy rejestrowania trasy w pakiecie, niezbędne było dokonanie inspekcji nagłówka warstwy łącza danych i zarejestrowanie adresu źródłowego w tej warstwie jako adresu identyfikującego węzeł sąsiedni.

3.2. Algorytm sterowania

Algorytm sterowania przedstawiony w [5] został zaimplementowany w postaci dyscypliny kolejkowania w stosie TCP/IP jądra Linux. Algorytm ten działa w krokach wykonywanych w czasie dyskretnym z konfigurowalnym okresem T. Wartością sterującą jest ilość danych które węzeł sterowany powinien przesłać, mierzona w bajtach. Ważnym parametrem algorytmu jest wartość referencyjna X^d ; jak podano w [5], długość kolejki w buforze węzła nigdy nie przekracza tej wartości. Wartość X^d powinna być więc jednocześnie wielkością bufora w węźle.

Działanie algorytmu opiera się o regułę przedstawioną w poniższym wzorze [5]:

$$x(k) + a(k) + w(k) = X^4 \tag{1}$$

gdzie $x(k)$ – długość kolejki w buforze węzła, $a(k)$ – wartość sterująca, $w(k)$ – suma wartości sterujących, które zostały wygenerowane wcześniej, lecz nie są jeszcze zrealizowane przez kontrolowane węzły; wszystkie wartości wyliczane są w chwili $t = k*T$.

Wyliczona wartość $a(k)$ jest rozdzielana pomiędzy węzły sąsiednie proporcjonalnie do ilości pochodzących od nich przepływów. Wartość sterująca dla danego węzła jest do niego wysyłana z użyciem pakietu IPv4 o rozgłoszeniowym adresie docelowym i zarejestrowanym dla tego węzła docelowym adresie warstwy łącza danych.

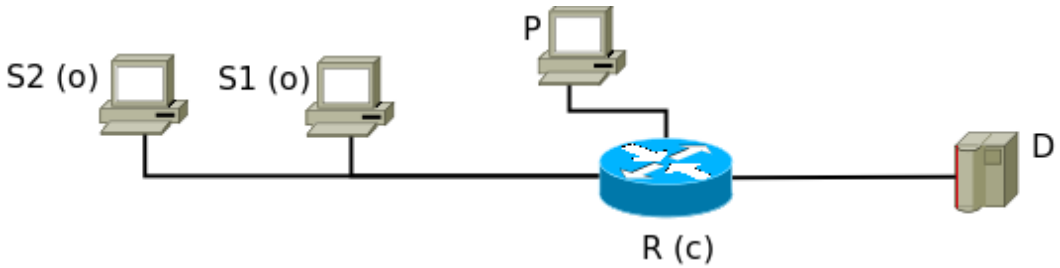
Zadaniem węzła kontrolowanego jest wyłącznie wysyłanie danych w ilości nie przekraczającej otrzymywanych wartości sterujących. Jeżeli węzeł w danym momencie nie może transmitować danych, otrzymywane wartości sterujące kumulują się.

Należy zauważyć, że węzeł, w którym wdrożono proponowane rozwiązanie, może działać w roli obiektu sterującego (kontrolera), obiektu sterowanego lub w obu rolach jednocześnie. W ostatnim przypadku należy podkreślić, że związek pomiędzy wartością sterującą generowaną w roli obiektu sterującego a wartościami otrzymy-

wanymi od innego węzła w roli obiektu sterowanego zachodzi wyłącznie pośrednio poprzez zmiany długości kolejki w buforze rozpatrywanego węzła.

4. WYNIKI EKSPERYMENTÓW

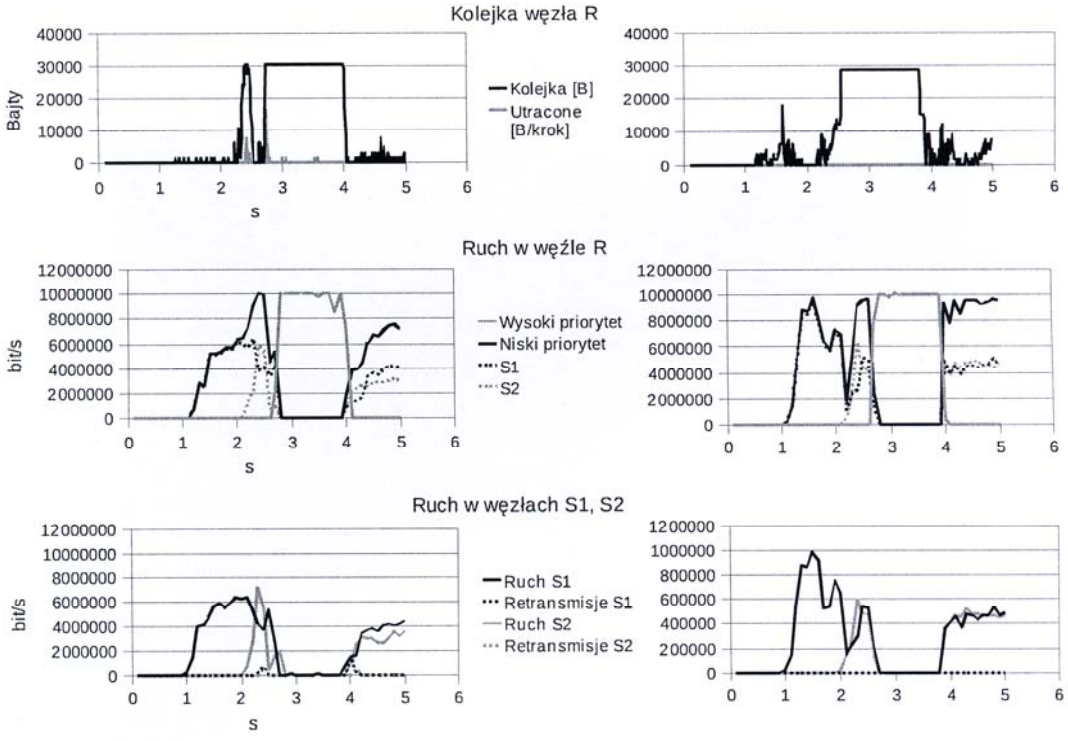
Proponowany mechanizm został zbadany eksperymentalnie z wykorzystaniem komputerów działających pod kontrolą systemu operacyjnego Fedora i jądra Linux w wersji 2.6.27. Konfigurację eksperymentalną przedstawia rysunek 1.



Rys. 1. Konfiguracja eksperymentalna

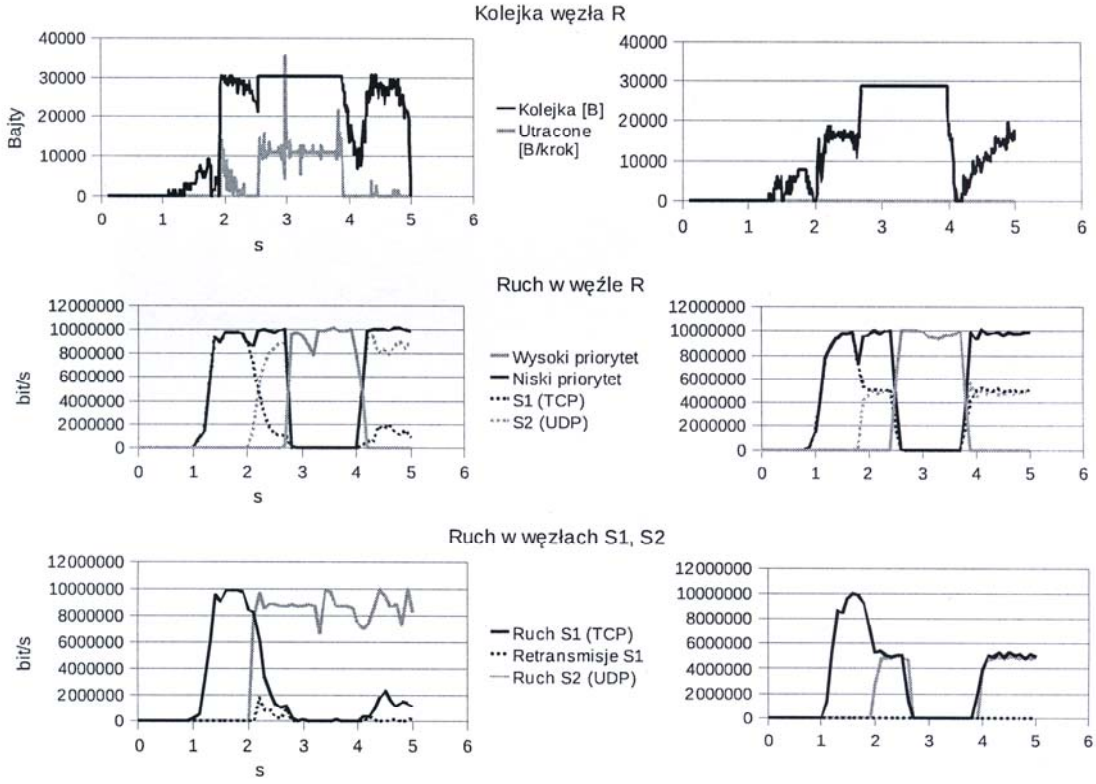
Maksymalna przepustowość łącz pomiędzy węzłami wynosi 10 Mbit/s. Mechanizm w postaci dyscypliny kolejkowania został wdrożony w węzłach S1, S2 i R. Węzeł R posiada dodatkowo hierarchię dyscyplin kolejkowania, dzięki której ruch generowany przez węzeł P ma wyższy priorytet w dostępie do łącza od ruchu generowanego przez węzły S1 i S2. Węzeł D służy jako odbiorca danych od wszystkich przepływów. Wartość referencyjna X^d i zarazem wielkość kolejki w węźle R została ustalona na 30000 bajtów zgodnie z zasadą podaną w [5], przy czasie RTT równym 2 ms i okresie sterowania T równym 10 ms.

W przedstawionej konfiguracji przeprowadzono eksperymenty w dwóch scenariuszach. W scenariuszu 1 węzeł S1 rozpoczyna nadawanie w połączeniu TCP w chwili t = 1s, węzeł S2 rozpoczyna nadawanie w połączeniu TCP w chwili t = 2s, zaś węzeł P rozpoczyna nadawanie w transmisji UDP w chwili t = 2,5s. Scenariusz 2 przebiega identycznie z tym, że węzeł S2 transmituje dane korzystając z protokołu UDP. Transmitowany ruch został zapisany przy użyciu programu *tcpdump* i poddany analizie przy użyciu programu *wireshark*, zaś informacje o wielkości kolejki w buforze węzła i wystąpieniu utraty danych zapisywane są przez zastosowaną dyscyplinę kolejkowania (także wtedy, gdy sam mechanizm sterujący jest wyłączony).



Rys. 2. Wyniki eksperymentu dla scenariusza 1
Lewa kolumna: mechanizm wyłączony, prawa kolumna: mechanizm włączony

W przypadku scenariusza 1. podstawowym problemem jest reakcja połączeń protokołu TCP (S1, S2), na wysycenie przepustowości łącza pomiędzy węzłami R i D. Oczywiście mechanizm okna w połączeniu TCP zapewnia dostosowanie się nadawcy do ograniczenia dostępnej przepustowości, jednakże ma to miejsce dopiero po wystąpieniu utraty danych na skutek przeciążenia węzła; retransmisja danych następuje po zakończeniu transmisji ruchu o wysokim priorytecie. Podobne problemy widoczne są dla połączenia generowanego przez węzeł S1 w chwili, gdy w tym samym łączu zaczyna nadawać węzeł S2. Po włączeniu proponowanego mechanizmu zarówno współdzielenie przepustowości pomiędzy węzłami S1 i S2, jak i reakcja na wysycenie przepustowości łącza pomiędzy węzłami R i D przebiega bez dopuszczenia do przeciążenia węzła R. Ciekawą konsekwencją tej właściwości jest brak redukcji okna TCP w okresie, gdy ruch w połączeniach TCP jest wstrzymany; dzięki temu można zaobserwować znacznie szybsze przywrócenie maksymalnej możliwej prędkości nadawania w tych połączeniach. Dzięki działaniu proponowanego mechanizmu ilość danych przesłanych połączeniami TCP, mierzona na łączu między węzłami R i D, wyniosła 1968738 bajtów, podczas gdy bez włączonego mechanizmu liczba ta wyniosła 1368634 bajty (w tym 59046 bajtów jako retransmisje).



Rys. 3. Wyniki eksperymentu dla scenariusza 2
Lewa kolumna: mechanizm wyłączony, prawa kolumna: mechanizm włączony

W przypadku scenariusza 2. podstawowym problemem jest współdzielenie dostępnej przepustowości przez ruch TCP generowany przez nadawcę S1 oraz ruch UDP generowany przez nadawcę S2. Wykres obrazujący ruch w obu węzłach ilustruje zawłaszczenie wspólnego łącza przez ruch UDP i związane z tym utraty danych i retransmisje w ruchu TCP. Po włączeniu ruchu o wysokim priorytecie widoczny jest całkowity brak kontroli przepływu i reakcji na utratę danych protokołu UDP na skutek przeciążenia w węźle R. Włączenie proponowanego mechanizmu doprowadza do równomiernego podziału przepustowości pomiędzy przepływy generowane przez węzły S1 i S2 oraz wstrzymanie tych przepływów na czas, gdy przepustowość łącza pomiędzy węzłami R i D zostaje wysycona przez ruch o wysokim priorytecie. W szczególności nie odnotowano utraty jakichkolwiek danych w węźle R oraz wystąpienia retransmisji w połączeniu TCP.

Podsumowując wyniki eksperymentalne należy stwierdzić, że zaproponowane rozwiązanie spełniło przyjęte założenia, tj. pozwoliło na eliminację zjawiska przeciążenia węzła sieci IP bez uwzględniania (w szczególności modyfikowania) właściwości protokołów warstw wyższych oraz aplikacji.

LITERATURA

1. Bartoszewicz A.: Nonlinear flow control strategies for connection-oriented communication networks. IEE Proceedings on Control Theory and Applications, 153(1), ss. 21-28, 2006.
2. Bartoszewicz A., Karbowańczyk M.: Sampled Time Flow Control Algorithm For Fast Connection Oriented Communication Networks. Journal of Applied Computer Science, 11(1), ss. 5-16, 2003.
3. Chatranon G. et al.: A survey of TCP-friendly router-based AQM schemes. Computer Communications, Vol. 27, Iss. 15,, ss. 1424-1440, 2004.
4. IETF Network Working Group: Stream Control Transmission Protocol. RFC 2960, 2000.
5. Karbowańczyk M.: Dyskretny algorytm sterowania ruchem w pojedynczym połączeniu połączeniowej sieci telekomunikacyjnej. XV Konferencja Sieci i Systemy Informatyczne – teoria, projekty, wdrożenia, aplikacje. Łódź, 2007.
6. Qureshi B., Othman M., Hamid N.A.W.: Progress in various TCP variants. 2nd International Conference on Computer, Control and Communication, s. 1, 2009.

Rękopis dostarczono dnia 19.10.2010 r.

AVOIDING CONGESTIONS IN IP NETWORKS USING HOP-BY-HOP TRAFFIC CONTROL

Michał KARBOWAŃCZYK

ABSTRACT *As IPv4 is the most widely used protocol, improving its efficience by applying a traffic control solution on it is still worth considering. The overall goal of the work presented in this chapter is to avoid congestions in IPv4 network nodes, with no modifications made to IPv4 nor higher network layers. Unfortunately, the IPv4 original design does not include any facilities provided to establish any kind of closed loop that could be used to feed the source (the sending host) with the information on the state and capabilities of the nodes that take their part in delivering data from the source to the destination. In this chapter two mechanisms extending the standard TCP/IP network stack of Linux kernel are proposed, so that it is possible to establish closed loops between considered node and its neighbour nodes (so called hop-by-hop closed loops), and then to control the way that neighbour nodes send the packets to the considered node. Furthermore, the motivations to choose IPv4 as the layer that traffic control was applied for, and hop-by-hop strategy for a closed loop mechanism, are discussed.*

Dr inż. Michał KARBOWAŃCZYK jest pracownikiem Zakładu Sieci Komputerowych Instytutu Informatyki Politechniki Łódzkiej. Jego prace badawcze koncentrują się na sterowaniem ruchem w sieciach teleinformatycznych, a także na różnych aspektach zastosowania systemów operacyjnych z rodziny GNU/Linux w infrastrukturze sieciowej. Współtwórca programu i współautor podręczników dla prowadzonych w Zakładzie Sieci Komputerowych studiów podyplomowych „Administracja Systemami GNU/Linux”.