STUDIA INFORMATICA 2003
Volume 24 Number 2A (53)

Ryszard MACELUCH, Piotr KASPRZYK, Adam DOMAŃSKI
Politechnika Śląska, Instytut Informatyki

DYNAMICZNY PRZYDZIAŁ PASMA UŻYTKOWNIKA SIECI Z WYKORZYSTANIEM USŁUGI QoS W SYSTEMIE LINUX

Streszczenie. Niniejszy artykuł opisuje mechanizmy zapewniania jakości usług dostępne w systemie operacyjnym Linux. Opisane zostały architektury Integrated Services oraz Differentiated Services oraz mechanizmy kolejkowania pakietów w jądrze systemu.

Słowa kluczowe: sieci komputerowe, kształtowanie ruchu w sieci, jakość usług.

DYNAMICAL BANDWIDTH ALLOCATION FOR NETWORK USER WITH QUALITY OF SERVICE IN OPERATING SYSTEM LINUX

Summary. This paper describes mechanism of Quality of Service in operating system Linux. Two architectures: Integrated Services and Differentiated Services and various packet queueings were described in article.

Keywords: computer network, shapping, quality of service.

1. Wstęp

Prędkość transferu danych w sieciach komputerowych zależy od aktualnego obciążenia sieci (ilości przesyłanych przez sieć danych). Podejście takie jest związane z historią powstawania sieci. Przed laty nikt nie przewidywał, że sieci komputerowe będą wykorzystywane do transmisji danych multimedialnych. Nikt nie spodziewał się tak gwałtownego rozwoju Internetu (zarówno pod względem ilości użytkowników, jak i zapotrzebowania na coraz większe przepustowości). Stąd przyjęto wiele uproszczeń konstrukcyjnych. Jednym z nich jest równouprawnienie wszystkich transmisji sieciowych. Internet jaki znamy przekazuje pakiety w myśl zasady „najlepszego wysiłku" (ang. Best

effort). Oznacza to, że węzły sieci są zaprojektowane tak, aby jak najszybciej przekazywać dalej otrzymane pakiety. W razie wystąpienia przeciążenia, pakiety są odrzucane bez faworyzowania jakichkolwiek. Reasumując, przy przekazywaniu pakietów węzły biorą tylko pod uwagę jego przeznaczenie („gdzie"). Jeśli chcielibyśmy zwiększyć elastyczność dotychczasowego modelu sieci i umożliwić faworyzowanie niektórych transmisji, to dochodzimy do wniosku, że oprócz przeznaczenia „gdzie", należy również przesyłać pakiety w odpowiednim momencie („kiedy").

Jedynym typem sieci, gdzie realne mechanizmy regulacji ruchu w sieci w zależności od potrzeb użytkownika zostały zaimplementowane, jest sieć ATM. Ze względu na popularność protokołu IP czyni się starania stworzenia identycznych możliwości w Internecie. Niniejszy artykuł opisuje problem dynamicznego przydziału pasma użytkownikom sieci opartych na protokole IP w wersji 4. Jako obiekt badań wybrano system operacyjny LINUX, gdzie dynamiczny przydział pasma jest realizowany poprzez „gwarancję jakości usług" (ang. Quality of Service")

Przez QoS rozumie się stałość w ściśle wyznaczonych granicach następujących parametrów transmisji:

- przepustowości: maksymalnej ilości danych (bitów) przesyłanych pomiędzy stacjami,
- opóźnienia: zwłoki w transmisji danych. W węźle opóźnieniem jest czas upływający między przejściem pakietu a jego przesłaniem dalej,
- zmienności opóźnienia: transmisji pakietów w nieregularnych odstępach.

2. Architektury QoS dla sieci IP z jakością usług

Zgodnie z klasyfikacją IETF (ang. Internet Engineering Task Force), zaleca się implementować QoS na jeden z dwóch sposobów:

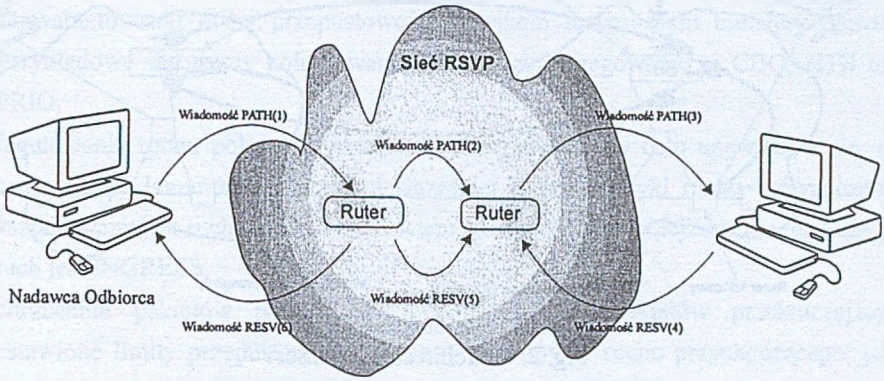
- Integrated Services,
- Differentiated Services.

Poniżej zostaną omówione różnice pomiędzy tymi dwoma sposobami.

2.1. Architektura Integrated Services

W architekturze tej zasoby są przydzielane aplikacjom na żądanie. W tym celu opracowano specjalny protokół RSVP (*ReSerVation Protocol*). Jest to protokół bazujący na strumieniu. Umożliwia on aplikacji realizację żądania rezerwacji zasobów w sieci dotyczących QOS, wysyłając komunikaty sterujące przez sieć. Rezerwacją zasobów i

realizacją zasobów dla strumieni aplikacja zajmują się rutery IP (w każdym ruterze IP wymagana jest implementacja protokołu RSVP).



Rys. 1. Protokół RVSP
Fig. 1. RVSP protocol

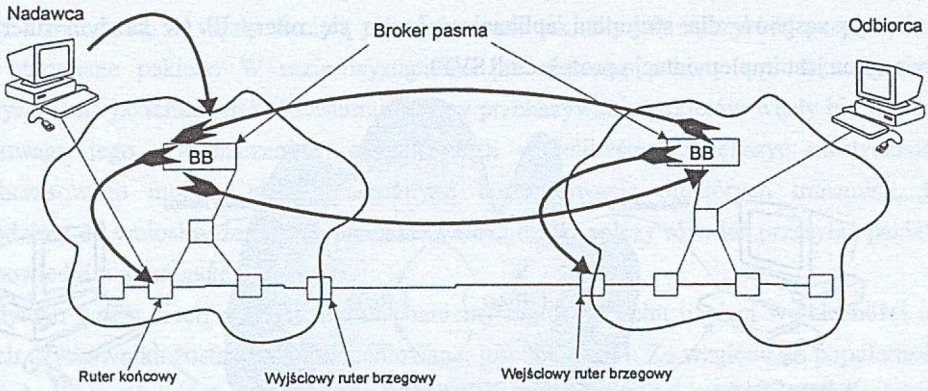
Zasada działania protokołu wygląda następująco (rys. 1):

- nadawca wysyła żądanie rezerwacji zasobów *(PATH message)*,
- przesyłanie wiadomości do miejsca przeznaczenia,
- odbiornik dostaje wiadomość,
- odbiornik wysyła wiadomość o rezerwacji zasobów w sieci *(RESV message)*,
- wiadomość wraca do nadawcy.

Każdy router po drodze, jeśli może dokonać żądanej rezerwacji, to przekazuje dalej wiadomość PATH, w przeciwnym wypadku odsyła do nadawcy informację o odrzuceniu zgłoszenia.

2.2. Architektura Differentiated Services

W architekturze DiffServ definiuje się zestaw usług sieciowych jedynie opierając się na mechanizmach nadawania priorytetów strumieniom w ruterach. W celu ustalenia priorytetów, należy przeprowadzić klasyfikację pakietów. Wykorzystuje się do tego bajt typu usługi ToS (*Type of Service*) - pole to występuje w nagłówku pakietu Ipv4 lub Ipv6. Wymaga ona, niestety, definicji klas ruchu na poziomie konfiguracji sieci. Pakiety są przypisywane do poszczególnych klas przez routery brzegowe, a znajdujące się w obrębie domeny QoS routery rdzeniowe zajmują się wyłącznie przekazywaniem pakietów. Architektura ta w stosunku do poprzedniej rozwiązuje problem skalowalności sieci.



Rys. 2. Architektura DiffServ
Fig. 2. Architecture DiffServ

3. Stan aktualny implementacji QoS w systemie Linux

Opis stanu aktualnego dotyczy jądra systemu dostępnego w czasie realizacji badań, czyli w wersji 2.4.18. Linux może służyć jako router w sieci IntServ. Większy nacisk w systemie położono jednak na realizację sieci typu DiffServ. Implementacja w systemie opiera się na pewnych elementach podstawowych:

- algorytm kolejkowania – pośredniczy w transmisji strumieni TCP między jądrem a interfejsem sieciowym. Pakiety wychodzące są kolejkowane wg pewnych zasad, a następnie pobierane przez jądro i przekazywane sterownikowi karty sieciowej. Algorytmy kolejkowania dzielą się na: zawierające klasy (np. CBQ) i bezklasowe (np. TBF),
- klasa – kolejka, w której przechowywane są pakiety wg zasad determinowanych przez algorytm kolejkowania,
- filtr - jest przyporządkowany do konkretnego algorytmu kolejkowania. Decyduje on, w której kolejce (klasie) zostanie umieszczony pakiet TCP. Pakiet jest sprawdzany sekwencyjnie przez wszystkie filtry, aż do pierwszego dopasowania (analogicznie do reguły firewalla).

Sterowanie ruchem polega na wykonaniu jednej lub kilku z poniższych operacji:

- kształtowanie ruchu polega na regulowaniu przepustowości strumienia wychodzących pakietów. Poza ograniczaniem dostępnej dla danej klasy ruchu przepustowości pozwala również na wygładzanie chwilowych przeciążeń ruchu. Przykładowymi algorytmami kolejkowania realizującymi formowanie ruchu są TBF oraz HBT,

- szeregowanie transmitowanych pakietów jako sposób na poprawę interaktywności ruchu pochodzącego z wymagających tego kolejek przy jednoczesnym zagwarantowaniu dużej przepustowości kolejkom realizującym transfery masowe. Przykładowe algorytmy kolejkowania realizujące szeregowanie to CBQ, HTB oraz PRIO,
- regulowanie ruchu polega na monitorowaniu pakietów w celu upewnienia się, czy nadawca nie łamie zasad ustalonej wcześniej charakterystyki ruchu. Równocześnie kształtowany jest ruch wchodzący. Dostępnym algorytmem kolejkowanie regulującym ruch jest INGRESS,
- odrzucanie pakietów polega na nieobsługiwaniu pakietów przekraczających ustawione limity przepustowości. Dotyczy to zarówno ruchu przychodzącego, jak i wychodzącego. Przykładowym algorytmem działania jest RED.

3.1. Algorytmy kolejkowania pakietów

Aktualnie dostępnych jest kilka algorytmów kolejkowanie. W razie potrzeby można napisać własny moduł realizujący inny algorytm kolejkowania. Poniżej zostaną skrótowo opisane najbardziej popularne algorytmy kolejkowania.

3.1.1. Algorytmy kolejkowania FIFO

Najprostszy z dostępnych algorytmów. W jądrze systemu Linux zaimplementowano trzy różne wersje tego algorytmu:

- pfifo-fast – do trzech podkolejek są kierowane pakiety o różnych priorytetach. Dana kolejka jest obsługiwana, jeżeli nie ma pakietów w kolejce o priorytecie wyższym,
- pfifo oraz bfifo – proste mechanizmy kolejkowania bez ulepszeń. Rozmiar tych kolejek podaje się w pakietach (pfifo) lub bajtach (bfifo).

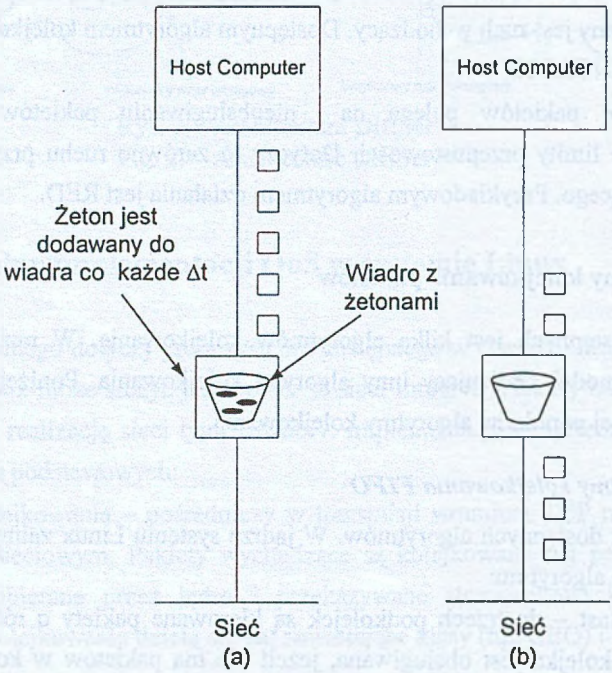
3.1.2. Algorytm kolejkowania PRIO

Jest to uogólniona odmiana algorytmu pfifo-fast. Pewnym ulepszeniem jest tutaj traktowanie podkolejek jako niezależnych klas, z których każda może być skojarzona z różnym algorytmem kolejkowania.

3.1.3. Algorytm kolejkowania TBF („cieknącego wiadra" z żetonami)

Algorytm ten należy do klasy algorytmów kształtujących ruch. To prosta kolejka wypuszczająca wyłącznie pakiety poniżej ustalonego natężenia przepływu (pakiety są wypuszczane co kwant czasu), z możliwością buforowania chwilowych przeciążeń. Ruch

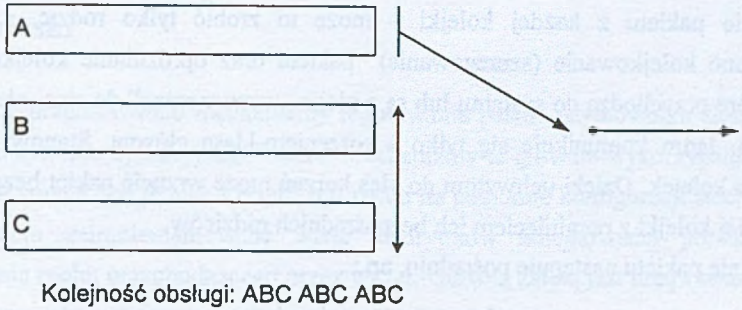
pakietów regulowany jest za pomocą żetonów. Implementacja TBF posiada bufor (kubełek), do którego wpadają żetony z natężeniem określonym parametrem. Rozmiar kubełka (ilość żetonów, które może pomieścić) jest również określany parametrem. Jeżeli w kubełku są żetony, to następuje wysłanie pakietu. Rozmiar pakietu w bajtach musi odpowiadać liczbie żetonów w kubełku – gdy jest większy, pakiet jest umieszczany w kolejce i czeka na odpowiednią ilość żetonów. Wraz z wysłaniem pakietu do sieci, zmniejszana jest liczba żetonów w kubełku.



Rys. 3. Algorytm „cieknącego wiadra" z żetonami
Fig. 3. Token Bucket Filter algorithm

*3.1.4. Algorytm kolejkowania SFQ (*Stochastic Fairness Queueing*)*

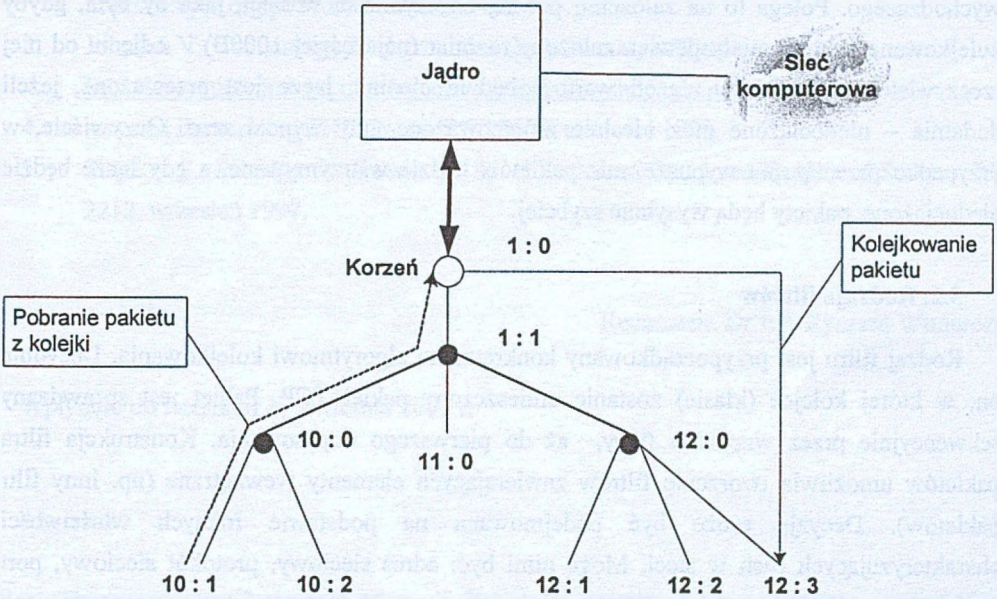
Jest to prosty algorytm, posiadający niewielkie wymagania obliczeniowe, wykonujący szeregowanie pakietów wychodzących, uwzględniający „sprawiedliwe kolejkowanie". Algorytm jest realizowany poprzez opóźnienie transmisji wybranych strumieni pakietów. SFQ to kilka kolejek, w których umieszczane są pakiety. Wysyłanie pakietów z kolejek w sieć następuje przy użyciu algorytmu „*round-robin*" – czyli po jednym pakiecie z każdej kolejki.



Rys. 4. Algorytm sprawiedliwego kolejkowania
Fig. 4. Stochastic Fairness Queueing algorithm

3.1.5. Algorytm kolejkowania CBQ (Class Based Queueing)

Algorytm CBQ jest bardzo elastycznym algorytmem kolejkowania. Hierarchia klas może być tworzona w zależności od potrzeb i jest układana w drzewo. Z każdą klasą jest skojarzony algorytm kolejkowania – domyślnie jest to *pfifo*_fast, można jednak ustawić dowolną dyscyplinę. Typowa hierarchia może wyglądać tak jak na rysunku 5.



Rys. 5. Algorytm kolejkowania CBQ
Fig. 5. CBQ Queueing algorithm

Hierarchia klas układa się w strukturę drzewiastą, w której klasy są korzeniem (ang. *root*), węzłami lub liśćmi. Każda klasa ma swój uchwyt (ang. handle), dzięki któremu można bezpośrednio dostać się do kolejki. Jest to potrzebne do natychmiastowego wkładania pakietu do odpowiedniej klasy. W drugą stronę nie jest to możliwe, tzn. nie można pobrać

bezpośrednio pakietu z każdej kolejki – może to zrobić tylko rodzic. Na rysunku przedstawiono kolejkowanie (szeregowanie) pakietu oraz opróżnianie kolejki. Wszystkie pakiety, które przychodzą do systemu lub są z niego „wypuszczane" do sieci obsługiwane są przez jądro. Jądro komunikuje się tylko z korzeniem-klasą główną. Stanowi ona pewien interfejs dla kolejek. Dzięki uchwytom do klas korzeń może wrzucić pakiet bezpośrednio do odpowiedniej kolejki z pominięciem ich bezpośrednich rodziców.

Pobieranie pakietu następuje pośrednio, np.:

10:1 -> 10:0 -> 1:1 -> 1:0

Pobieranie pakietów z klas następuje za pomocą algorytmu „*round-robin*" (a dokładniej wg ważonego RR. Algorytm WRR jest stosowany, aby pobrać z każdej kolejki po jednym pakiecie w cyklu. Kolejność kolejek, z których obiekt jest pobierany, ustalana jest za pomocą wag.

Poza kolejkowaniem pakietów algorytm CBQ zajmuje się również kształtowaniem ruchu wychodzącego. Polega to na założeniu pewnej bezczynności w sieci, jaka by była, gdyby kolejkowane pakiety miały pewien założony rozmiar (najczęściej 1000B) i odjęciu od niej rzeczywistej bezczynności. Jeżeli wartość będzie ujemna, łącze jest przeciążone, jeżeli dodatnia – niedociążone oraz idealnie zrównoważone, gdy wynosi zero. Oczywiście, w przypadku przeciążenia wypuszczanie pakietów będzie wstrzymywane, a gdy łącze będzie niedociążone, pakiety będą wysyłane szybciej.

3.2. Rodzaje filtrów

Rodzaj filtru jest przyporządkowany konkretnemu algorytmowi kolejkowania. Decyduje on, w której kolejce (klasie) zostanie umieszczony pakiet TCP. Pakiet jest sprawdzany sekwencyjnie przez wszystkie filtry, aż do pierwszego dopasowania. Konstrukcja filtra pakietów umożliwia tworzenie filtrów zawierających elementy wewnętrzne (np. inny filtr pakietów). Decyzja może być podejmowana na podstawie różnych właściwości charakteryzujących ruch w sieci. Mogą nimi być: adres sieciowy, protokół sieciowy, port protokołu transportowego, wejściowy interfejs itp. W rzeczywistych zastosowaniach konstruuje się hierarchie klas korzystając z różnych algorytmów kolejkowania i różnych rodzajów filtrów.

4. Wnioski

W pracy przedstawiono mechanizmy regulowania pasma użytkownika sieci dostępne w środowisku systemu operacyjnego Linux. Mechanizmy te głównie wykorzystują architekturę DiffServ. Wymaga ona jednak klasyfikacji ruchu na poziomie konfiguracji sieci. W systemie Linux zostało zaimplementowane wiele algorytmów kolejkowania pozwalających na kształtowanie ruchu przechodzącego przez węzeł. Główną zaletą jest tutaj otwartość systemu Linux, co pozwala na łatwe wprowadzanie nowych rozwiązań.

LITERATURA

1. Burakowski W., Bąk A, Kopertowski Z., Wykroka A.: Mechanizmy QoS w sieci IP. II Ogólnopolska Konferencja „Internet", Wrocław 2000.
2. Krawczyk P.: Sterowanie przepływem danych w Linuxie. Repozytorium PLD CSV http://cvs.pld.org.pl/PLD-doc/NET4/NET4_tc.pdf.
3. Johnson V.: Technology Backgrounder-QoS-Glossary of Terms. http://www.ucer.nurcad.ufsc.br/docs/johnson 99technology.pdf, 1999.
4. Assured Forwording PHB Group. RFC 2597, czerwiec 1999.
5. Shenker S., Patridge C., Guerin R.: Specification of Guaranted Quality of Service. RFC 2212, wrzesień 1997.

Recenzent: Dr inż. Ryszard Winiarczyk

Wpłynęło do Redakcji 17 kwietnia 2003 r.

Abstract

This paper describes mechanism of Quality of Service in Linux operating system. Two architectures: Integrated Services and Differentiated Services were described in article. Figure 1 ilustrates Integrated Services architecture and RVSP protocol. Figure 2 ilustrates Differentiated Services architecture. Linux supports both architectures, but it prefers Diffserv. Article describes next implementation of QoS in Linux: packet queues, queueing algorithms and filters. Paper presents following queueings algorithms: FIFO, TBF (Fig. 3), Stochastic Fairness Queueing (Fig. 4), Class Based Queueing (Fig. 5).

Adresy

Ryszard MACELUCH: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, rmac@top.iinf.polsl.gliwice.pl .
Piotr KASPRZYK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, kwadrat@top.iinf.polsl.gliwice.pl .
Adam DOMAŃSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, adamd@top.iinf.polsl.gliwice.pl .