ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1997
Seria: INFORMATYKA z. 32 Nr kol. 1356

Tadeusz CZACHÓRSKI, Piotr PECKA
Instytut Informatyki Teoretycznej i Stosowanej PAN

Tülin ATMACA, Salima HAMMA, Badii JOUABER
Institut National des Télécommunictions, Evry, France

MODELE MARKOWA W OCENIE PRACY WĘZŁA SIECI – WYZNACZANIE STRAT W FUNKCJI ROZMIARU PAKIETU

Streszczenie. Artykuł przedstawia model pracy bufora kolejkującego pakiety w węźle sieci przy założeniu mechanizmu progowego: bufor jest podzielony na dwie części, jedna jest dostępna wszystkim pakietom, druga tylko pakietom priorytetowym. Model ma postać łańcucha Markowa, oblicza straty pakietów różnej wielkości, wynikłe z zapełnienia bufora, rezultaty uzyskiwane są numerycznie dla stanu ustalonego i stanów przejściowych.

MARKOVIAN MODELS OF VARIABLE-SIZE FRAMES ADMISSION IN FRAME-RELAY NETWORKS

Summary. The article presents the performance evaluation of a shared buffer for a network switch with threshold priority mechanism. A Markovian model is proposed and solved numerically. The results, both for transient and steady states, estimate the losses for this scheme.

1. Wstęp

Współczesne szybkie sieci z komutacją pakietów stoją przed trudnym zadaniem zapewnienia usług o zróżnicowanej jakości połączeniom korzystających z tych samych fizycznych zasobów sieci, np. łącz i buforów[1]. Dlatego, jak widać to z aktualnej literatury i propozycji standardów, obiektem oceny i modelowania są algorytmy różnicujące dostęp do zasobów ruchowi poszczególnych klas pakietów.

[1] Niniejszy tekst został opracowany częściowo w ramach Projektu Badawczego KBN nr 8 T11C 032 08.

Artykuł bada własności polityki przydziału pamięci w buforze przełącznika sieci typu ATM lub Frame Relay, która wymusza różne prawdopodobieństwa strat przez selektywne niszczenie pakietów w momencie zatłoczenia. Rozważa się bufor o ograniczonej pojemności dzielony pomiędzy pakiety o wyższym (klasa 1) i niższym (klasa 2) priorytecie. Mechanizm progowy, zwany mechanizmem podziału bufora *Partial Buffer Sharing*, dzieli cały bufor na dwie części: jedna jest dostępna dla obu klas pakietów, druga tylko dla klasy o wyższym priorytecie.

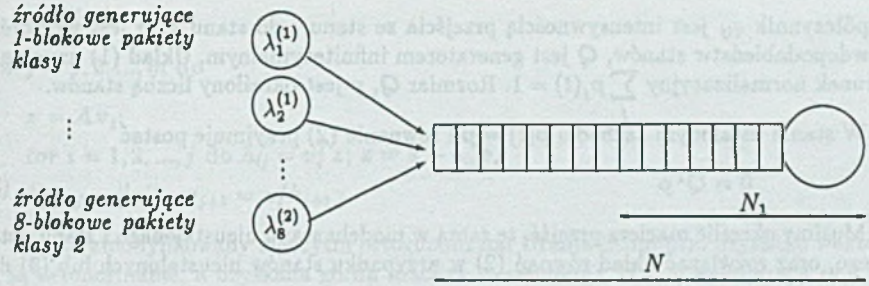
Model analityczny wykorzystuje proces (łańcuch) Markowa o skończonej liczbie stanów. Markowowskie modele cechuje duża elastyczność, mogą one opisać złożone modele kolejkowe, w tym synchronizację i ograniczenia wynikające z niedostępności zasobów. Wykorzystanie rozkładów Erlanga, hiperwykładniczych i Coxa, złożonych z faz o wykładniczym czasie trwania, pozwala na uwzględnienie prawie dowolnych rozkładów czasu obsługi i strumieni wejściowych. Jednakże ich implementacja jest związana z dwoma problemami: jak wygenerować macierz przejść między stanami procesu i jak rozwiązać powstały układ równań, biorąc pod uwagę, że liczba stanów (a więc wymiar macierzy przejść i liczba równań) rośnie bardzo szybko wraz z rozbudową modelu, osiągając często wartość kilkudziesięciu czy kilkuset tysięcy. Wymaga to opracowania efektywnego algorytmu generacji macierzy przejść i metod, które są w stanie pokonać problemy numeryczne związane z rozwiązywaniem tak dużych układów równań.

Problem nie jest nowy. Automatyczną generację macierzy przejścia, znajdując stany sąsiednie i sprawdzając, czy nie ma ich jeszcze na liście stanów, opracowano już 30 lat temu [16, 17]. Opracowano też inne podejście, oparte na pojęciu automatów stochastycznych [9]. Zbadano użyteczność metod numerycznych, np. [8], jest też dostępne bardzo dobre opracowanie monograficzne [15], istnieje pakiet programowy XMARCA [15] tworzenia i rozwiązywania modeli markowowskich. Metoda Arnoldiego, która została oceniona jako najbardziej przydatna [1] w rozwiązywaniu dużych modeli markowowskich, została dokładnie przestudiowana [13, 12, 11], a jej implementacja zoptymalizowana [14].

Jednakże wzrost mocy obliczeniowych i rozwój języków i technik programowania niosą ze sobą nowe możliwości. Przedstawiony tu model wygenerowany został za pomocą przygotowanego wcześniej zestawu obiektów zapisanych w języku C++. Jak dotąd, C++ wykorzystywany był w symulacji, np. w systemie SMURPH [4]. Tu znalazł zastosowanie w narzędziu programowym generującym modele markowowskie, takie jak przedstawiony poniżej model progowego regulaminu szeregowania pakietów o zmiennym rozmiarze w kolejce komutatora. W punkcie 2 omówiono zasady tego regulaminu, punkt 3 zawiera krótkie omówienie opracowanego narzędzia dla tworzenia modeli markowowskich. Punkt 4 omawia metodę Arnoldiego wykorzystywaną do rozwiązywania tych modeli dla stanów nieustalonych i ustalonych. Sekcje 5 i 6 przedstawiają bardziej szczegółowo opracowany model i dyskutują uzyskane wyniki numeryczne.

2. Kolejkowy model mechanizmu podziału bufora

Model kolejki z podziałem bufora przedstawia rys. 1; zakładamy Poissonowskie strumienie wejściowe, pakiety klasy k, $k = 1, 2$, o rozmiarze i bloków, $i = 1, 2, \ldots, 8$, nadcho-



Rys. 1. Bufor z progiem
Fig. 1. Partial buffer sharing

dzą z intensywnością $\lambda_i^{(k)}$. Rozmiar bufora wynosi N, wartość progu N_1. Pakiety niższej klasy ($k = 2$), o rozmiarze i bloków mają dostęp do bufora, jeżeli jest w nim zajętych mniej niż $N_1 - i$ bloków. Pakiety o wyższym priorytecie (klasa 1), o rozmiarze i bloków mają zapewniony dostęp do bufora, gdy jest w nim zajętych mniej niż $N - i$ bloków.

Wprowadźmy następujące oznaczenia. S jest średnim czasem obsługi, tzn. czasem transmisji jednego bloku; $\rho^{(1)} = \sum_{i=1}^{nmax} \lambda_i^{(1)} S$ i $\rho^{(2)} = \sum_{i=1}^{nmax} \lambda_i^{(2)} S$ wyrażają obciążenie serwera ze względu na obsługę bloków należących odpowiednio do pakietów klasy 1 i klasy 2; całkowite obciążenie to $\rho = \rho^{(1)} + \rho^{(2)}$.
$L^{(1)}$ oznacza prawdopodobieństwo strat pakietów klasy 1, a $L^{(2)}$ prawdopodobieństwo strat pakietów klasy 2.
$p_i^{(1)}$ oraz $p_i^{(2)}$ $(0 \le i \le N)$ to rozkłady prawdopodobieństwa liczby bloków należących do pakietów klasy 1 i 2 obecnych w buforze, p_i $(0 \le i \le N)$ jest rozkładem liczby wszystkich bloków w buforze.

Prawdopodobieństwo strat pakietów różnych klas i rozmiarów obliczamy jako

$$L_i^{(1)} = \sum_{j=N-i+1}^{N} p_j \qquad L_i^{(2)} = \sum_{j=N_1-i+1}^{N} p_j$$

3. Podstawowe pojęcia związane z łańcuchami Markowa

Rozważmy łańcuch Markowa z ciągłym czasem, o przestrzeni stanów $\{x_1, x_2, ..., x_n\}$. Jeżeli $X(t) = x_i$, proces jest w stanie i. Oznaczmy $P_i(t) = P[X(t) = x_i]$; dla jednorodnych łańcuchów Markowa prawdopodobieństwa $P_i(t)$ są zdefiniowane układem równań różniczkowych

$$\frac{dP_i(t)}{dt} = \sum_j P_j(t) q_{ji} \tag{1}$$

lub, w notacji macierzowej:

$$\frac{dp(t)}{dt} = Q^T p(t) \tag{2}$$

Współczynnik q_{ij} jest intensywnością przejścia ze stanu i do stanu j, p jest wektorem prawdopodobieństw stanów, Q jest generatorem infinitezymalnym. Układ (1) uzupełnia warunek normalizacyjny $\sum_j p_j(t) = 1$. Rozmiar Q, p jest określony liczbą stanów.

W stanie ustalonym zachodzi $p(t) = p$ i równanie (2) przyjmuje postać

$$0 = Q^T p \tag{3}$$

Musimy określić macierz przejść, tę samą w modelu stanu nieustalonego i stanu ustalonego, oraz rozwiązać układ równań (2) w przypadku stanów nieustalonych lub (3) dla stanu ustalonego.

Macierz Q jest zwykle bardzo duża (liczba stanów, a więc i rozmiar macierzy Q jest często rzędu setek tysięcy), rzadka i źle uwarunkowana; wykorzystanie popularnych algorytmów numerycznych dla układów równań algebraicznych i różniczkowych jest utrudnione.

Gdy znamy macierz Q, wektor prawdopodobieństw stanu $p(t)$ dla określonego czasu t jest rozwiązaniem układu (2). Teoretycznie rzecz biorąc, rozwiązanie można napisać natychmiast:

$$p(t) = p(0)e^{Q^T t}, \quad \text{gdzie} \quad e^{Q^T t} = \sum_{k=0}^{\infty} (Q^T t)^k / k!$$

Numeryczne kłopoty związane z obliczaniem wyrażeń zawierających dużą macierz w eksponencie, por. [6], utrudniają jednak otrzymanie liczbowych wyników. Jeżeli wartość normy $||Q^T||$ macierzy Q^T w chwili czasowej t, dla której szukamy rozwiązania, jest duża, rezultaty dla $e^{Q^T t}$ mogą okazać się niezadowalające. Musimy wówczas podzielić przedział czasu $[t_0, t]$ na mniejsze przedziały $[t_0, t_1, t_2, ..., t]$ i określić rozwiązanie dla wszystkich punktów pośrednich t_j używając jako warunku początkowego rozwiązania dla t_{j-1}.

Wyniki testów numerycznych stosowalności różnych metod do rozwiązania równań (2) przeprowadzonych w [14] wskazują na specjalnie dużą przydatność metody Arnoldiego, wykorzystującej przestrzenie Kryłowa [15]. Poniżej przedstawiony jest jej zarys, oparty na [15]. Dla uproszczenia notacji będziemy oznaczać $A = Q^T$, $v = p(t_i)$, $w = p(t_{i+1})$.

Poszukiwane rozwiązanie ma postać $w = e^A v$ (w wyprowadzeniach pośrednich będziemy opuszczać zapis stałej $\tau_i = t_{i+1} - t_i$).

Przestrzeń Kryłowa jest rozpięta na wektorach $v, Av, ..., A^{m-1}v$:

$$K_m(A, v) \equiv span\{v, Av, ..., A^{m-1}v\}$$

Metoda poszukuje takiego elementu tej przestrzeni, który najlepiej przybliża wektor w.

Oznaczmy zespół wektorów bazowych jako $V_m = [v_1, v_2, ..., v_m]$; $v_1 = v/\beta$, gdzie $\beta = ||v||_2$, $v = \beta V_m e_1$ oraz

$$\begin{aligned} w \approx V_m V_m^{-1} V_m^{-T} V_m^T e^A v &= V_m[(V_m^T V_m)^{-1} V_m^T e^A v] \\ &= V_m[(V_m^T V_m)^{-1} V_m^T e^A V_m]\beta e_1. \end{aligned} \tag{4}$$

Wektor e_i ma i-ty element o wartości 1, a pozostałe zerowe.

Zbiór wektorów bazowych V_m jest wyznaczany za pomocą procedury Arnoldiego, zapisanej tu w notacji zbliżonej do języka Pascal:

1. $v_1 = v/||v||_2$
2. for $j = 1, 2, ..., m$ do

   $z = Av_j$;

   for $i = 1, 2, ..., j$ do $h_{ij} = v_i^T z$; $z = z - h_{ij} v_i$;

   $h_{j+1,j} = ||z||_2$; $v_{j+1} = z/h_{j+1,j}$

Jest to zmodyfikowany algorytm ortogonalizacji Grama-Schmidta, uzyskane wektory v_j są ortonormalne, a uzyskana górna macierz Hessenberga H_m, o rozmiarach $m \times m$, złożona ze współczynników h_{ij} spełnia równanie

$$AH_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T \tag{5}$$

Ponieważ zbiór wektorów V_m jest ortonormalny, możemy uprościć równanie (4):

$$w \approx \beta V_m (V_m^T e^A V_m) e_1$$

i przybliżyć $V_m^T e^A V_m$ przez $e^{V_m^T A V_m}$, a wektor w przyjmie postać:

$$w \approx \beta V_m e^{V_m^T A V_m} e_1$$

Wykorzystując ponownie ortonormalność zbioru wektorów V_m możemy zapisać równanie (5) jako

$$H_m = V_m^T A V_m \tag{6}$$

i wyrazić rozwiązanie w postaci:

$$w \approx \beta V_m e^{H_m} e_1$$

To przybliżone rozwiązanie również wymaga obliczenia eksponenty z macierzą w wykładniku, lecz rozmiar m macierzy H_m jest znacznie mniejszy: w prezentowanych poniżej przykładach numerycznych wybrano $m = 10$. Do obliczeń można więc wykorzystać dowolną metodę, w naszym przypadku była to aproksymacja Padégo.

Wprowadzając na powrót do zapisu stałą τ można zapisać $V_m^T (A\tau) V_m = H_m$. Przestrzenie Kryłowa związane z A i $A\tau$ są identyczne. Ich wykorzystanie przy wyznaczaniu rozwiązania w stanie nieustalonym polega na:

— zastosowaniu procedury Arnoldiego do uzyskania ortonormalnej bazy wektorów V_m i macierzy Hassenberga H_m,

— zastosowaniu aproksymacji Padégo do wyznaczenia $e^{H_m \tau}$,

— obliczeniu wektora prawdopodobieństw stanów w, przybliżanego przez $\beta V_m e^{H_m \tau} e_1$.

Jedyną operacją wykonywaną na dużej macierzy A (Q^T) jest mnożenie macierzy przez wektor w pierwszej linii drugiego kroku procedury Arnoldiego. Szczegóły metody są dyskutowane w [14].

W stanie ustalonym zaleca się [15] przy analizie markowowskich modeli kolejkowych zastąpić wyjściowy problem rozwiązywania układu równań liniowych (3), których macierz Q jest duża i rzadka, co powoduje, że metody iteracyjne, np. Gaussa-Seidela, Jacobiego, nadrelaksacji, mają problemy ze zbieżnością, problemem równoważnym, polegającym na

poszukiwaniu lewego wektora własnego p związanego z największą wartością własną macierzy W^T:

$$W^T p = p, \quad \text{gdzie} \quad W^T = Q^T \Delta t + I. \tag{7}$$

Macierz I jest jednostkowa, a wartość $\Delta t \leq (max_i |q_{ii}|)^{-1}$ jest tak dobrana, by W^T była macierzą stochastyczną, a więc posiadającą taką największą wartość własną, że jej moduł jest równy 1.

Znów wyznaczamy za pomocą procedury Arnoldiego małą macierz H_m, o wektorze własnym w, który przybliża wektor własny p wektora W^T. Stosując równanie (6) możemy zapisać $H_m w = w$ jako

$$\begin{aligned} V_m^T W^T V_m w &= w \\ W^T V_m w &= V_m w \end{aligned} \tag{8}$$

a więc $p = V_m w$.

Znajdujemy wektor własny w macierzy H_m za pomocą ilorazu Rayleigha.

4. Oprogramowanie użyte do generacji macierzy przejść oraz rozwiązywania modeli Markowa

Oprogramowanie, jakiego użyliśmy dla generacji macierzy przejść między stanami łańcucha Markowa odpowiadającego pracy przełącznika sieciowego, zostało napisane w języku C++ i ma postać biblioteki klas. Model sieci stanowisk powstaje przez wybranie odpowiednich obiektów z ich biblioteki i wstawienie ich do głównego programu, który jest kompilowany za pomocą kompilatora C++ z dołączeniem odpowiednich bibliotek.

Generacja macierzy dla modelu odbywa się poprzez dołączanie stanów do listy stanów (tak by zachować porządek generowania) i do słownika funkcji mieszającej (celem szybkiego przeszukiwania listy). Każdy wygenerowany stan ma przyporządkowany sobie jednoznacznie numer; nowy wektor stanu otrzymuje go poprzez inkrementację swego wewnętrznego licznika, numer istniejącego stanu jest podawany przez słownik funkcji mieszającej. Funkcja `GetNextState()` zwraca numer wektora stanu, tranzycje są określane przez obiekt `Dispatcher`, który przesyła dane do każdego elementu sieci za pośrednictwem funkcji `SendTransitions`. Użytkownik może wybrać jeden z kilku typów funkcji `StateSearcher` dołączając ją do programu rozwiązywania układu równań. Typy te wykorzystują:

— *słownik funkcji mieszającej*, zapożyczony z biblioteki `rwtools.h++` (stosunkowo szybki, wymagający dużo pamięci),

— funkcję `ndbm` dostępną w systemach SunOs i Solaris (wolną, lecz wymagającą mało pamięci),

— algorytm N-tree (szybki, wymagający dużo pamięci),

— koncepcję *funkcji odwrotnej*: numer danego wektora stanu jest obliczany przez specjalnie dobraną funkcję, dającą wzajemnie jednoznaczne przyporządkowanie (sposób bardzo szybki, wymagający mało pamięci, lecz możliwy tylko dla niektórych, określonych z góry struktur sieci).

Otrzymany system równań jest rozwiązywany numerycznie przy użyciu opisanego wcześniej algorytmu Arnoldiego.

Poniżej pokazano w tabeli czas obliczeń dla modelu, w którym bufor mieści 150 bloków, a wartość progu wynosi 100 bloków. Rozmiar macierzy przejść wynosi 102 001 × 102 001, macierz zawiera 1370087 niezerowych elementów, rozwiązanie było określone dla czasu $t = 100$ jednostek czasu.

Komputer	Czas użytkownika (s)	Czas rzeczywisty (min)	%cpu
Ultra Sparc 1	1418.0	24:12	97
Convex C3820 (scalar)	2493.49	1:20:29	26
Convex C3820 (vector)	988.469	33:13.30	25
Convex C3820 (vector+parallel)	993.016	24:00	34.8

Specjalna klasa została napisana dla przechowywania rzadkich macierzy. Inna klasa przedstawia aproksymację Padégo. Procedury numeryczne zostały napisane w Fortranie, a część zajmującą najwięcej czasu, mnożenie macierzy przez wektor, napisano w assemblerze wykonującej obliczenia maszyny Convex C3.

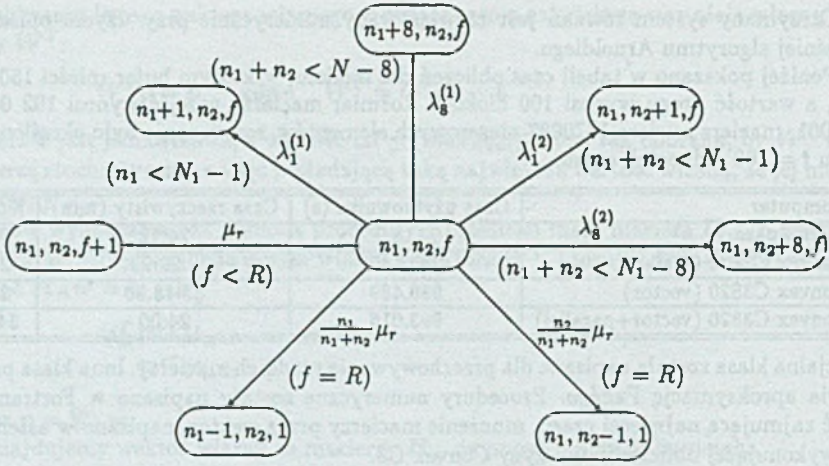
Prawdopodobieństwa stanu służą następnie do obliczenia rozkładów brzegowych: liczby bloków należących do wyższej i niższej klasy pakietów.

Tworzone oprogramowanie obejmuje też inne obok markowowskiego moduły obliczeniowe do rozwiązywania modeli kolejkowych (wykorzystujące aproksymację dyfuzyjną i analizę wartości średnich). Wykorzystanie języka C++ i jego kompilatora zapewnia temu systemowi większą elastyczność, niż ma to miejsce w przypadku tworzenia własnego języka opisu modelu, np. AMOK [3], QNAP [20]. Narzędzie jest przeznaczone dla stacji unixowych i może wykorzystywać mechanizmy wątkowe systemu Solaris, bibliotekę Nexus oraz cechy Compositional C++.

5. Analizowany model i jego parametry

Niech N oznacza rozmiar bufora, mierzony liczbą bloków, które może pomieścić, a N_1 jest wartością progu. Stały czas obsługi jest przybliżony rozkładem Erlanga rzędu R. Stan systemu zdefiniowany jest wektorem (n_1, n_2, f), gdzie n_1 jest liczbą bloków należących do pakietów priorytetowych, n_2 liczbą bloków należących do pakietów o niższyn priorytecie, f jest fazą rozkładu Erlanga, w której znajduje się aktualnie wykonywana transmisja bloku. Wiadomość zawarta w opisie stanu nie jest wystarczająca do opisu ewolucji systemu: brak informacji o klasie aktualnie przesyłanego bloku, a więc nie można stwierdzić, czy po zakończeniu ostatniej fazy transmisji zmniejszy się wartość n_1 czy n_2. Jednakże pamiętanie klasy transmitowanego bloku, a więc także klasy każdego bloku znajdującego się w kolejce, bardzo zwiększyłoby liczbę rozpatrywanych stanów systemu. Dlatego założono w przybliżeniu, że przejście ze stanu (n_1, n_2, R) następuje z intensywnością $\mu_R n_1/(n_1 + n_2)$ do stanu $(n_1 - 1, n_2, 1)$ oraz z intensywnością $\mu_r n_2/(n_1 + n_2)$ do stanu $(n_1, n_2 - 1, 1)$. Rysunek 2 pokazuje schemat przejść między stanami łańcucha odpowiadającego modelowi.

Dane liczbowe modelu. Załóżmy, że linia wyjściowa ma przepustowość 2 Mbajtów/s, że bloki liczą 1024 bajtów długości, a więc czas transmisji jednego bloku wynosi



Rys. 2. Przejścia między stanami modelu
Fig. 2. State-transition-rate diagram

$\frac{1024}{2\times10^6} \approx 0.5$ ms. Wybierzmy 1 ms jako jednostkę czasu, czyli intensywność obsługi wynosi $\mu = 2$. Stały czas przesyłu aproksymujemy rozkładem Erlanga 10 rzędu ($R = 10$); każda jego faza ma rozkład wykładniczy o parametrze $\mu_r = 20$, $r = 1, 2, \ldots 10$. Intensywność poszczególnych źródeł wynosi 64 kbytes/s, a więc ich intensywność mierzona w blokach wynosi $\lambda = 0.064$ bloków/ms. W klasie o wyższym i niższym priorytecie są po 3 źródła generujące 1- i 2-blokowe pakiety, po 2 źródła generujące 3- i 4-blokowe pakiety oraz po 1 źródle generującym pakiety 5-, 6-, 7- i 8-blokowe. Parametry strumieni wejściowych modelu mają więc wartość:

$$\lambda_1^{(1)} = \lambda_1^{(2)} = 0.192, \quad \lambda_5^{(1)} = \lambda_5^{(2)} = 0.013,$$

$$\lambda_2^{(1)} = \lambda_2^{(2)} = 0.096, \quad \lambda_6^{(1)} = \lambda_6^{(2)} = 0.010,$$

$$\lambda_3^{(1)} = \lambda_3^{(2)} = 0.043, \quad \lambda_7^{(1)} = \lambda_7^{(2)} = 0.009,$$

$$\lambda_4^{(1)} = \lambda_4^{(2)} = 0.032, \quad \lambda_8^{(1)} = \lambda_8^{(2)} = 0.008.$$

Długość bufora wynosi $N = 50$, wartość progowa $N_1 = 25$, liczba stanów wynosi w tym przypadku 10 001 (38 751 dla $N = 100$, $N_1 = 50$ oraz 152 501 stanów dla $N = 200$, $N_1 = 100$).

Rysunki 3, 4 przedstawiają przykład wyników w stanie nieustalonym: rozkład liczby bloków obu klas w funkcji czasu; w chwili początkowej, $t = 0$, bufor był pusty.

6. Rezultaty obliczeń w stanie ustalonym

Jak stwierdzono powyżej, przedstawiony model markowowski jest przybliżony, ponieważ nie ma dostatecznej informacji, by określić dokładne wartości przejść ze stanów (n_1, n_2, R), a stały czas transmisji bloku został zastąpiony rozkładem Erlanga E_{10}.

Dlatego rezultaty obliczeń modelu markowowskiego są, dla sprawdzenia jego poprawności i oceny popełnionych błędów, wynikających tak z poczynionych uproszczeń, jak i błędów numerycznych, wyrywkowo porównane na rys. 5 - 7, 9 z rezultatami modelu symulacyjnego. Porównanie wskazuje na wystarczającą dokładność modelu analitycznego.

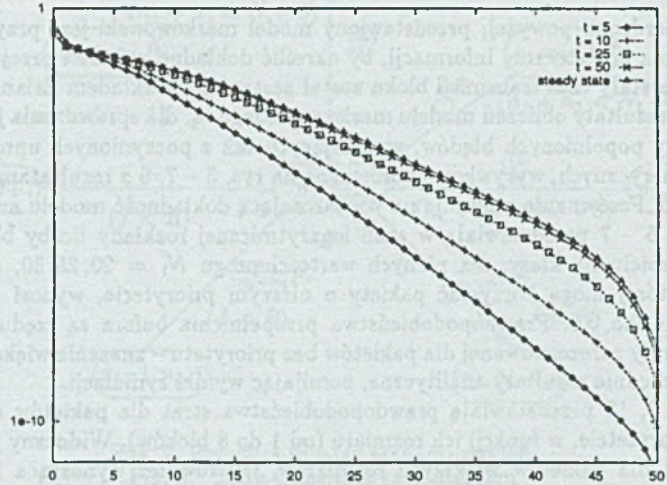
Rysunki 5 - 7 przedstawiają w skali logarytmicznej rozkłady liczby bloków w buforze, w rozbiciu na klasy, dla różnych wartości progu $N_1 = 20, 25, 30$, a więc część bufora, z której mogą korzystać pakiety o niższym priorytecie, wynosi odpowiednio $N_1/N = 0.4, 0.5, 0.6$. Prawdopodobieństwa przepełnienia bufora są rzędu 10^{-8}, przepełnienia strefy zarezerwowanej dla pakietów bez priorytetu - znacznie większe. Rysunek 8 łączy poprzednie rezultaty analityczne, pomijając wyniki symulacji.

Rysunki 9, 10 przedstawiają prawdopodobieństwa strat dla pakietów o wyższym i niższym priorytetcie, w funkcji ich rozmiaru (od 1 do 8 bloków). Widoczny jest wyraźny wzrost strat dla pakietów większych rozmiarów, jak również wynosząca kilka rzędów wielkości różnica pomiędzy stratami pakietów o wyższym i niższym priorytecie. Obserwujemy też wpływ poziomu progowego na straty: wraz z podnoszeniem wartości progu maleją straty pakietów o niższym priorytecie oraz rosną (znacznie szybciej) straty pakietów priorytetowych.

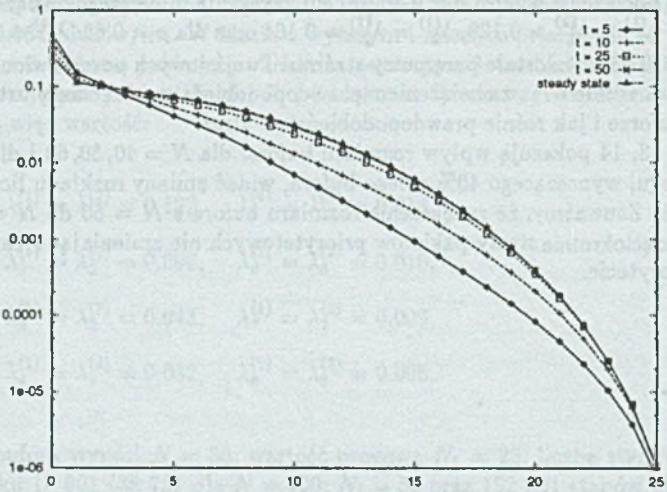
Rysunki 11, 12 pokazują zmiany rozkładu kolejki i zmiany strat, gdy zmienia się obciążenie systemu $\rho = 0.63, 0.75, 0.86$. Przy założonych pierwotnie danych co do strumieni wejściowych obciążenie wynosi $\rho = 0.86$, w celu uzyskania mniejszego obciążenia przyjęto dla $\rho = 0.75$ $\lambda_1^{(1)} = \lambda_1^{(2)} = 0.128$, $\lambda_2^{(1)} = \lambda_2^{(2)} = 0.164$ oraz dla $\rho = 0.63$: $\lambda_1^{(1)} = \lambda_1^{(2)} = 0.064$, $\lambda_2^{(1)} = \lambda_2^{(2)} = 0.032$. Pozostałe parametry strumieni wejściowych pozostawiono bez zmian.

Widać, jak rośnie wraz z obciążeniem prawdopodobieństwo obecności ustalonej liczby bloków w buforze i jak rośnie prawdopodobieństwo strat.

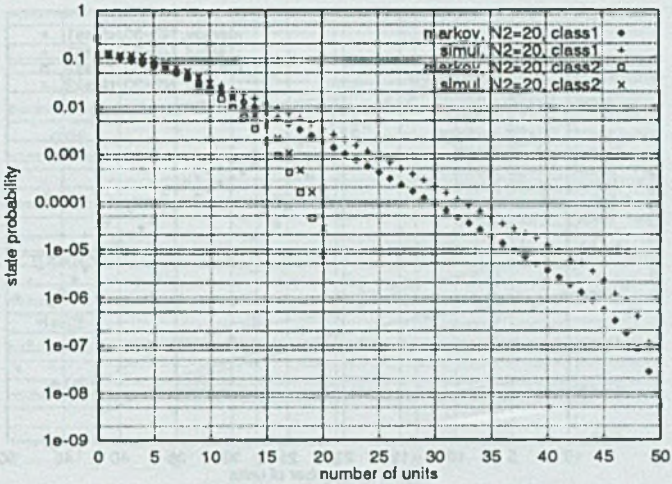
Rysunki 13, 14 pokazują wpływ rozmiaru bufora: dla $N = 40, 50, 60$ i dla ustalonego poziomu progu, wynoszącego 40% całego bufora, widać zmiany rozkładu liczby bloków i zmiany strat. Zauważmy, że zwiększenie rozmiaru bufora z $N = 50$ do $N = 60$ zmniejsza kilkudziesięciokrotnie straty pakietów priorytetowych nie zmieniając strat pakietów o niższym priorytecie.



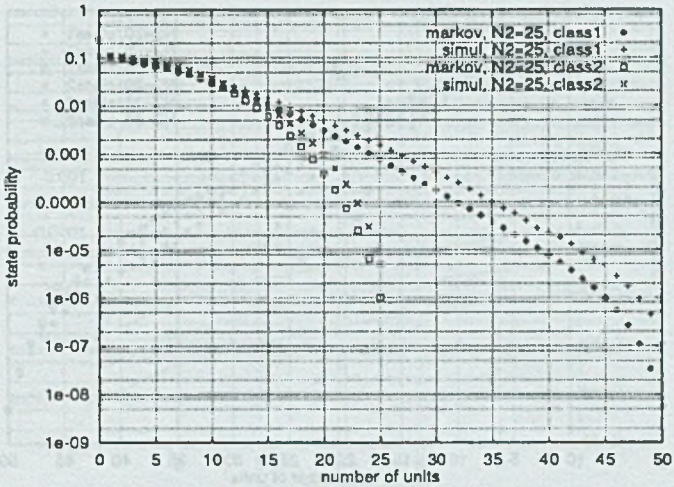
Rys. 3. Zależny od czasu rozkład liczby bloków pakietów priorytetowych
Fig. 3. Priority queue distribution as a function of time



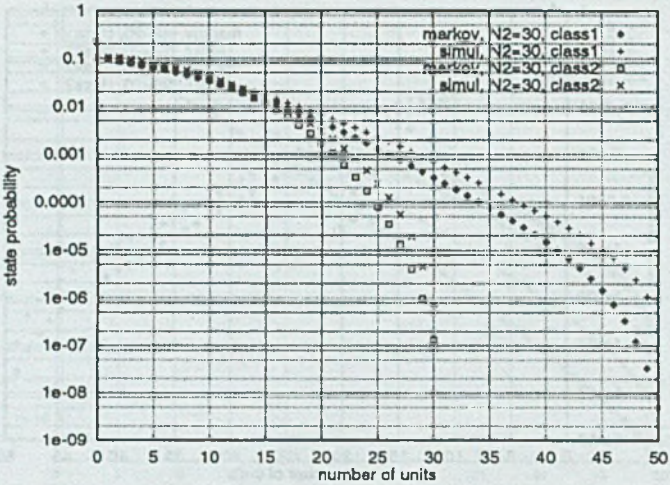
Rys. 4. Zależny od czasu rozkład liczby bloków pakietów o niższym priorytecie
Fig. 4. Second class queue distribution as a function of time



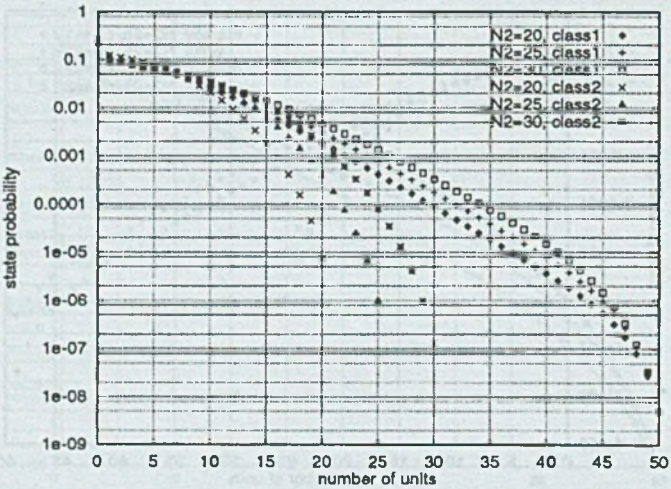
Rys. 5. Rozkład liczby bloków, $N = 50$, $N_1 = 20$
Fig. 5. Steady state probability versus number of blocks $N = 50$, $N_1 = 20$



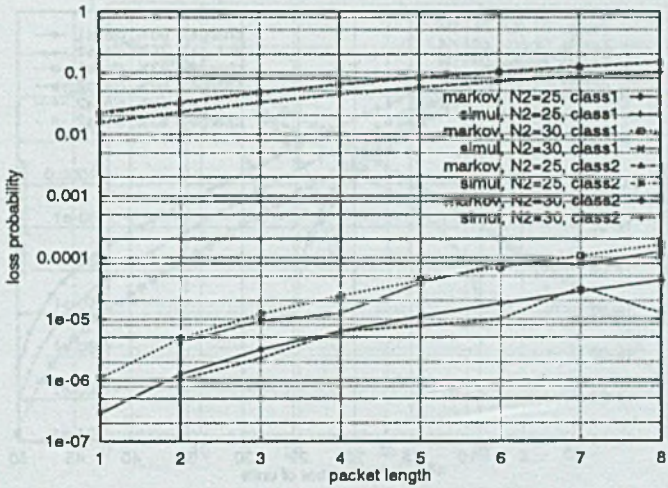
Rys. 6. Rozkład liczby bloków, $N = 50$, $N_1 = 25$
Fig. 6. Steady state probability versus number of blocks $N = 50$, $N_1 = 25$



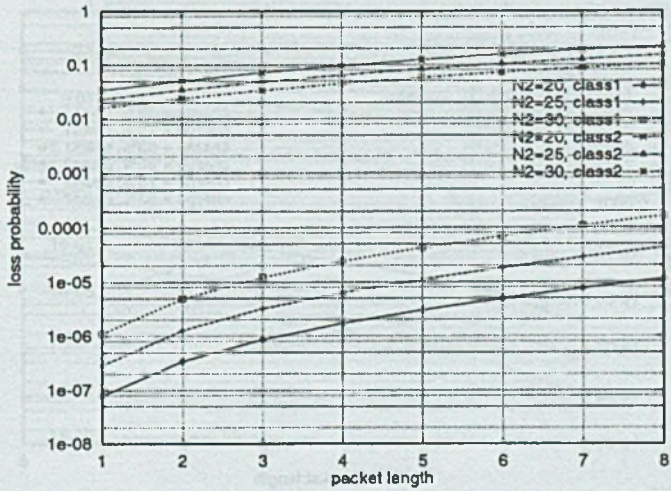
Rys. 7. Rozkład liczby bloków, $N = 50$, $N_1 = 30$
Fig. 7. Steady state probability versus number of blocks, $N = 50$, $N_1 = 30$



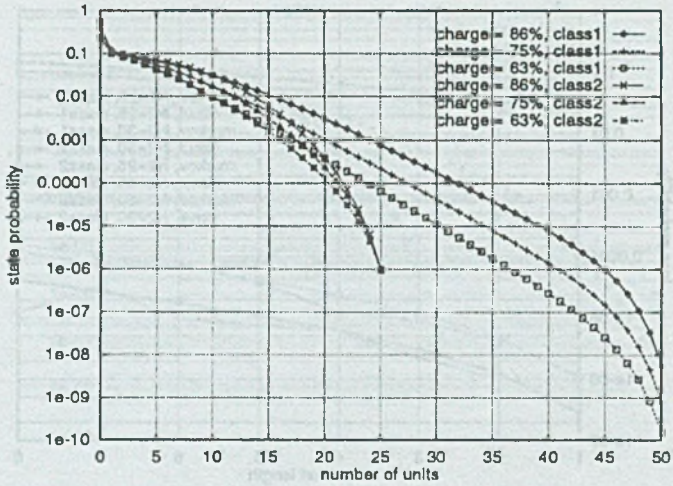
Rys. 8. Rozkład liczby bloków zależny od wartości progu, $N = 50$
Fig. 8. Steady state probability as function of threshold, $N = 50$



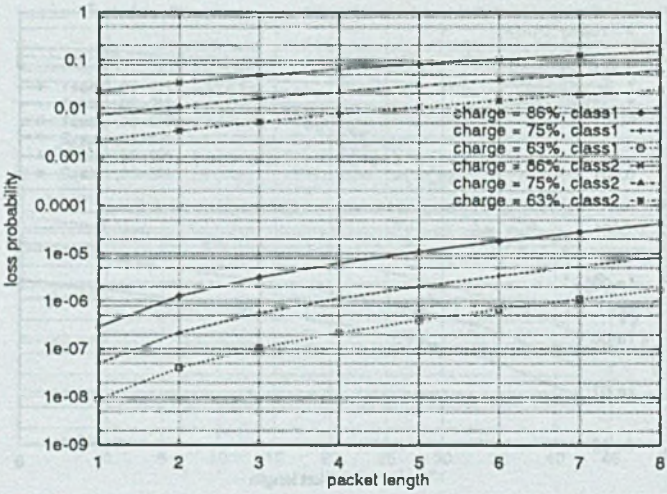
Rys. 9. Prawdopodobieństwo strat w funkcji wartości progu, $N = 50$
Fig. 9. Loss probability as function of threshold, $N = 50$



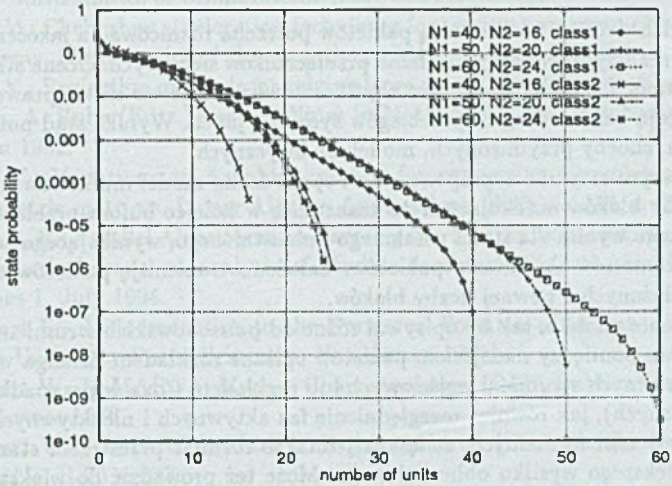
Rys. 10. Prawdopodobieństwo strat w funkcji wartości progu, $N = 50$
Fig. 10. Loss probability as function of threshold, $N = 50$, analytic results



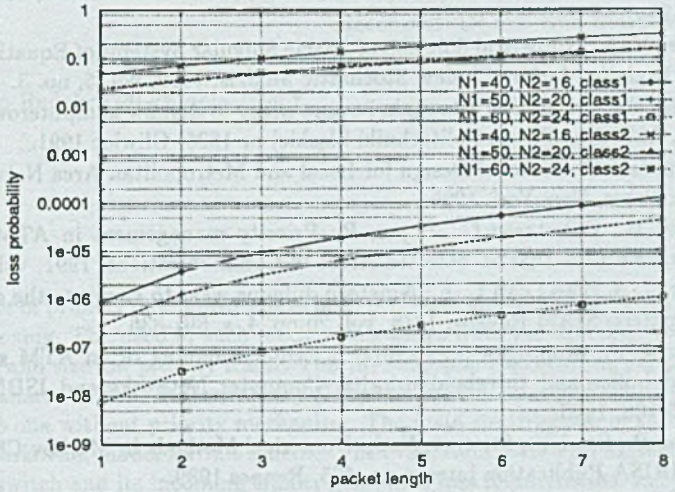
Rys. 11. Rozkład liczby bloków w zależności od obciążenia, $N = 50$, $N_1 = 25$
Fig. 11. Steady state probability as function of charge, $N = 50$, $N_1 = 25$



Rys. 12. Prawdopodobieństwo strat w funkcji obciążenia, $N = 50$, $N_1 = 25$
Fig. 12. Loss probability as function of charge, $N = 50$, $N_1 = 25$



Rys. 13. Rozkład liczby bloków w zależności od rozmiaru bufora
Fig. 13. Steady state probability as function of buffer size



Rys. 14. Straty w zależności od rozmiaru bufora
Fig. 14. Loss probability as function of buffer size

7. Wnioski

W szybkich sieciach z komutacją pakietów potrzeba różnicowania jakości usług rodzi konieczność realistycznej oceny działania przełączników sieciowych. Ocena strat pakietów za pomocą modeli symulacyjnych jest zbyt kosztowna, ponieważ niskie prawdopodobieństwa wymagają bardzo długich przebiegów symulacyjnych. Wynika stąd potrzeba mniej kosztownych, choćby przybliżonych, modeli analitycznych.

W niniejszym artykule zaproponowano i sprawdzono model markowowski oceniający rozkład liczby bloków oczekujących na transmisję w kolejce bufora przełącznika sieciowego. Uzyskano wyniki dla stanu ustalonego i nieustalonego, wynikającego ze zmiennych w czasie parametrów strumienia pakietów. Założono transmisję pakietów o zmiennym rozmiarze, złożonych z pewnej liczby bloków.

Rozszerzenie modelu, tak by opisywał różne od poissonowskich strumienie wejściowe (odstępy czasu pomiędzy nadejściem pakietów opisane rozkładem Erlanga w przypadku bardziej regularnych strumieni wejściowych lub rozkładem Coxa w przypadku strumieni mniej regularnych), jak również uwzględnienie faz aktywnych i nieaktywnych źródeł nie nastręcza trudności formalnych, zwiększa jednakże rozmiar przestrzeni stanów modelu i wymaga większego wysiłku obliczeniowego. Może też prowadzić do większych błędów numerycznych.

LITERATURA

[1] Arnoldi W. E.: The Principle of Minimized Iterations in the Solutions of the Matrix Eigenvalue Problems. Quaterly of Applied Mathematics, 1951, vol. 19, s. 17-29.

[2] Barker V. A.: Numerical Solution of Sparse Singular Systems of Equations Arising from Ergodic Markov Chains. Stochastic Models, 1989, vol. 5, no. 3.

[3] Czachórski T. i in.: Modelowanie i ocena pracy systemów komputerowych za pomocą AMOKu. Skrypt Politechniki Śląskiej nr 1626, Gliwice 1991.

[4] Gburzyński P.: Protocol Design for Local and Metropolitan Area Networks, Academic Press, New York 1995.

[5] Kröner H., Hébuterne G., Boyer, P.: Priority management in ATM switching nodes. IEEE Journal on Selected Areas in Communications, 1991, vol. 9, no. 3.

[6] Moler C. B., Van Loan C. F.: Nineteen dubious ways to compute the exponential of a matrix. SIAM Review, 1978, vol. 20 no. 4, s. 801-836.

[7] Meyer J. F., Montagna S., Paglino R.: Dimensionning of an ATM switch with shared buffer and threshold priority, Comupter Networks and ISDN Systems, pp. 95-108, 1993.

[8] Philipe B., Saad Y., Stewart W. J.: Numerical Methods in Markov Chain Modeling, IRISA Publication interne no. 495, Rennes 1989.

[9] Plateau B., Fournau J-M.: For solving Markov models of parallel systems, RR 819-I-, IMAG, Laboratoire de Génie Informatique, Grenoble, Juin 1990, and Journal of Parallel and Distributed Computing, vol. 12, no. 4, pp. 370-387, August 1991.

[10] Saad Y.: Krylov Subspace Methods for Solving Large Unsymmetric Linear Systems. Mathematics of computation, 1981, vol. 37, no. 155.
[11] Saad Y.: Chebyshev acceleration techniques for solving non-symmetric eigenvalue problems. Mathematics of Computation, 1984, vol. 42, s. 567-588.
[12] Saad Y.: Projection methods for solving large sparse eigenvalue problems. B. Kagstrom, A. Ruhe (Edts.) Lecture Notes in Math. Series, No. 973, Springer Verlag, Berlin 1982.
[13] Saad Y.: Variations on Arnoldi's method for computing eigenelements of large unsymetric matrices. Linear Algebra Applications, 1980, vol. 34, s. 269-295.
[14] Sidje R. B.: Parallel Algorithms for Large Sparse Matrix Exponentials: application to numerical transient analysis of Markov processes. PhD thesis, University of Rennes 1, July 1994.
[15] Stewart W. J.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton, New Jersey 1994.
[16] Stewart W. J.: MARCA: Markov Chain Analyzer. IRISA Publication interne no. 45, Rennes 1976.
[17] Stewart W. J.: MARCA: Markov Chain Analyzer. IEEE Computer Repository, 1976, No. R 76 s. 323.
[18] SunOS 5.3, Guide to Multithread Programming, 1994.
[19] Page T., Berson B., Cheng W., Muntz R.: An Object-Oriented Modeling Environment. OOPSLA, 1989, vol. 24 no. 10.
[20] QNAP2 guide de l'utilisateur. SIMULOG, St-Quentin sur Yvelines, 1992.

Recenzent: Dr inż. Ewa Starzewska-Karwan

Wpłynęło do Redakcji 20 grudnia 1996 r.

Abstract

A number of priority mechanisms have been proposed and analysed for ATM switching and multiplexing. The need of such mechanisms arises from the difficulty to accuraretly characterize and size low priority traffic. This investigation concerns the performance evaluation of a shared buffer for Frame Relay (FR) switch with threshold priority mechanism compared to one without priority mechanism. The main contribution of the paper is the proposed Markovian model for this scheme. The evaluation, based on finite-state Markov model of a switch and its incoming traffic, guarantee loss requirements, i.e. probabilities of 10^{-9} and 10^{-6} for high and low priority frames. The Markovian model was generated with the use of a more general tool, the principles of which are also presented in the article. This tool is prepared to construct various Markovian models of computer networks and their control mechanisms. It contains a set of objects written in C++ which may

be assembled into a network. Both steady-state and transient analysis are performed. In addition, comparative study shows that analytical results appears to be satisfactory with those of simulations.