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

Adam DOMAŃSKI, Jarosław FLAK, Piotr KASPRZYK
Politechnika Śląska, Instytut Informatyki

OCENA PROTOKOŁÓW KOMUNIKACYJNYCH W SIECIACH KOMPUTEROWYCH

Streszczenie. W artykule omówiono zagadnienia oceny protokołów komunikacyjnych. Naświetlono aspekty, jakie należy poruszyć w czasie dokonywania analiz oraz opisano niektóre metody rozwiązywania wynikających problemów przy użyciu reguł matematycznych.

COMMUNICATION PROTOCOLS EVALUATION IN COMPUTER NETWORKS

Summary. This article is about communication protocols evaluation. Some aspects that should be talked about while during making such analyzes were discussed. A few examples of solving problems using mathematical formulas were described.

1. Ocena protokołów komunikacyjnych

Protokół komunikacyjny to zbiór reguł, w jakim dwie jednostki porozumiewają się wzajemnie ze sobą. Gdy mówimy o porozumiewaniu się komputerów, to reguły te muszą być zdefiniowane tak, aby umożliwiały one bezbłędny przesył danych pomiędzy komputerami. Sprawę oceny protokołów komplikuje fakt, że do pełnej komunikacji pomiędzy węzłami potrzeba wiele reguł. Wzajemne ich uzależnienie nazywamy hierarchią protokołów. Sieć komputerową rozumiemy wtedy jako układ warstw, gdzie:

— niższe poziomy dostarczają usług warstwom wyższym,
— zachodzi komunikacja wirtualna pomiędzy równorzędnymi poziomami.

Dlatego też do analizy sytuacji w sieci nie powinno się podchodzić globalnie. W danej

sieci może pracować wiele protokołów naraz, również logicznie należących do tej samej warstwy. Aby zrozumieć to, co dzieje się w sieci, należy przede wszystkim zidentyfikować, z jakich protokołów korzystają użytkownicy danej sieci, a następnie zanalizować indywidualnie każdy protokół.

Przy ocenie protokołów musimy brać pod uwagę trzy aspekty:
— efektywność protokołu,
— przezroczystość informacyjną i transportową,
— weryfikację poprawności protokołu.

1.1. Efektywność protokołu

Efektywność protokołu możemy opisać wzorem:

$$U=\frac{D}{N} \tag{1}$$

gdzie:

U — efektywność protokołu,

D — suma bitów informacyjnych,

N — suma wszystkich wysłanych bitów (zależy od rodzaju protokołu i kanału transmisji).

Aby uzmysłowić sobie, jak wiele zależy tutaj od definicji protokołu, prześledźmy następujące przypadki:

a) protokół jednokierunkowy:

Nadajnik — nadaje ciągi bitów.

Odbiornik — odbiera wszystko bez błędów.

U = 1

b) protokół wyślij i czekaj

Nadajnik — nadaje na przemian ramkę o numerze 0 i 1, oraz czeka na potwierdzenie odebrania ramki przez odbiornik.

Odbiornik — odbiera ramki 0 i 1 oraz wysyła ramkę potwierdzającą otrzymanie ramki.

Założenie: linia doskonała nie wnosząca błędów.

— czas wysłania ramki wynosi F/C, $F = H + D$, gdzie:

F — liczba bitów w ramce,

H — liczba bitów nagłówka,

C — przepustowość kanału [b/s].

— w odbiorniku pojawi się ona po czasie $F/C + I$, gdzie:

I — opóźnienie związane z obsługą ramki i czasem propagacji sygnału.

— wysłanie ramki potwierdzenia $F/C + I + A/C$, gdzie:

A — liczba bitów w ramce potwierdzającej.

— przyjęcie potwierdzenia przez nadajnik zachodzi po czasie $F/C + I + A/C$.

Z powyższych przykładów można wysnuć wniosek, że stopień wykorzystania kanału, możliwy teoretycznie do uzyskania, zależy od:

— długości nagłówka,

— długości ramki potwierdzenia,

— czasu propagacji sygnału i obsługi ramki w węźle.

W razie występowania kanału transmisji, w którym mogą występować przekłamania, mamy do czynienia z retransmisjami, które jeszcze bardziej komplikują problemy oceny efektywności.

1.2. Przezroczystość informacyjna i transportowa

— **Informacyjna:**
Nie istnieją ciągi bitów nie mogące wystąpić w polu danych, np. HDLC.

— **Transportowa:**
Mówimy o niej, gdy protokół może być używany bez ograniczeń w kanałach synchronicznych, asynchronicznych, dupleksowych i półdupleksowych.

Dla tych cech nie można tutaj mówić o żadnej analizie, albowiem dane protokoły albo tę cechę posiadają, albo nie.

1.3. Weryfikacja protokołu

Istnieją modele matematyczne pozwalające na weryfikację protokołów. Poniżej zostaną omówione dwa główne: maszyna o skończonej liczbie stanów (automat skończony) i sieci Petriego.

1.3.1. Maszyna o skończonej liczbie stanów

Na rys. 1 pokazany jest przykład maszyny stanowej dla protokołu wyślij i czekaj. Transmisja odbywa się linią transmisyjną wnoszącą błędy. Każdy stan jest opatrzony etykietą, którą tworzą trzy znaki: N, O, K.

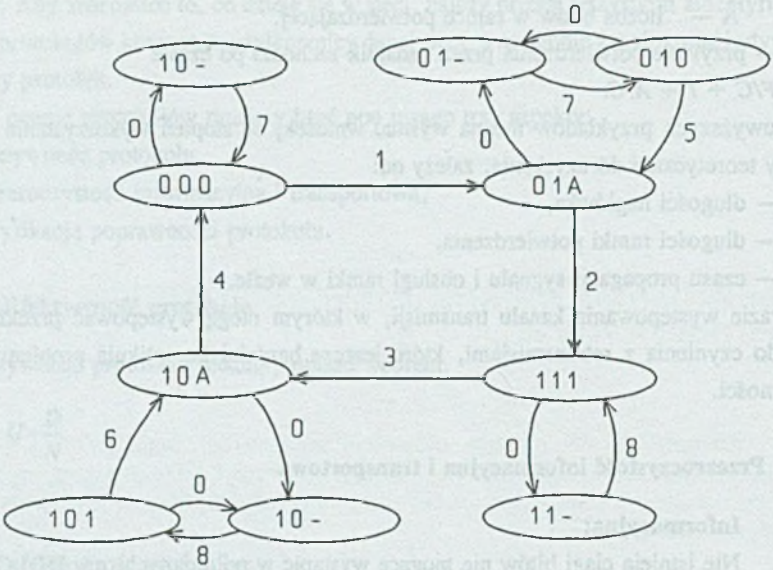
N — ma wartość 1 lub 0, co odpowiada ramce, którą nadajnik próbuje wysłać.

O — ma wartość 1 lub 0, co odpowiada ramce, na którą czeka odbiornik.

K — to stan kanału, może przybierać wartości 0, 1 lub A (ramka odpowiedzi).

Kanał może być także pusty, co jest zaznaczone za pomocą "-".

Można wysnuć następujące wnioski:



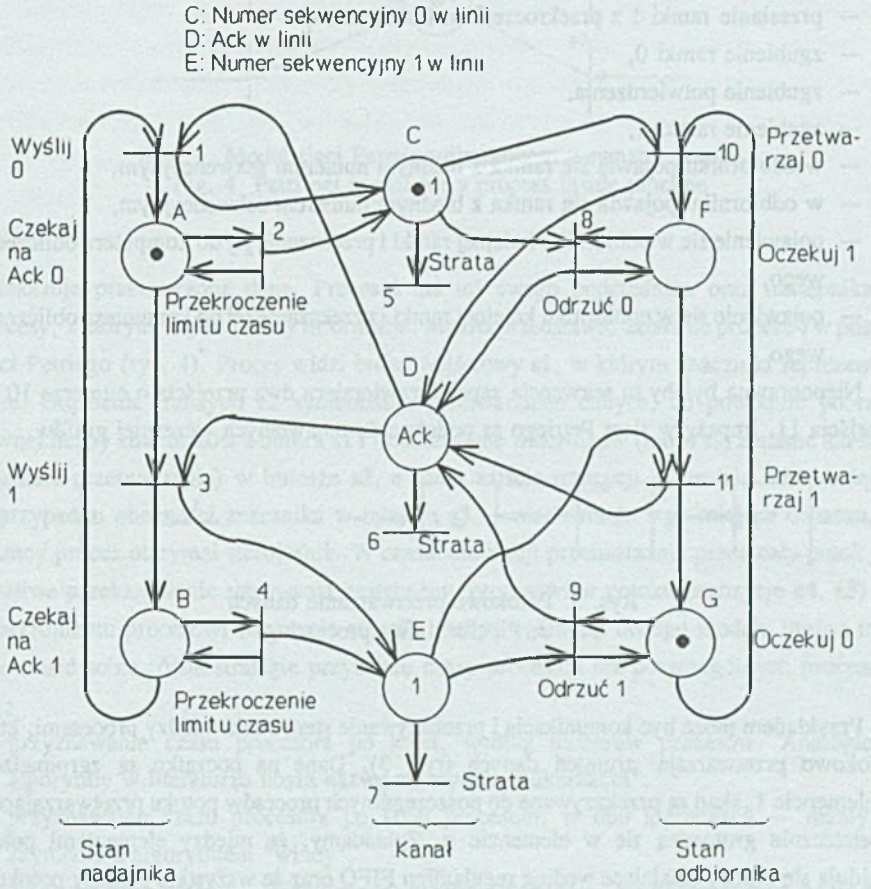
Rys. 1. Przykład grafu maszyny stanowej
Fig. 1. Example of state machine diagram

— Niezależnie od ciągu zdarzeń nie może wystąpić sytuacja, że odbiornik otrzyma po kolei dwie nieparzyste ramki.
— Nie może wystąpić taka sekwencja zdarzeń, w której nadajnik zmieniałby stan z 0 na 1, a potem z 1 na 0, a stan odbiornika nie ulegałby zmianie. Ciąg wiadomości zawierałby nie wykrytą przerwę złożoną z dwóch wiadomości.
— Brak impasów (deadlock), które w modelu grafowym przedstawiałyby się jako niepusty podzbiór stanów osiągalnych ze stanu początkowego o następujących właściwościach:
  - nie istnieje wyprowadzenie z danego podzbioru,
  - nie istnieją w podzbiorze wyjścia powodujące posunięcie do przodu.

1.3.2. Weryfikacja protokołów a sieci Petriego

Pojęcia teorii automatów są bardzo użyteczne dla opisu procesów sekwencyjnych. Jednakże dla opisu modeli procesów równoległych musiały powstać modele koncepcyjnie różne do powyższych. Jednym z nich może być model zaproponowany przez C. A. Petriego. Doszedł on do wniosku, że pojęcie stanu układu jest nieodpowiednie, gdy mamy do czynienia z synchronizacją czasową. Trudno wtedy modelować np. oczekiwanie na obliczenie generowane przez procesy, wymianę danych między komputerami itp.

W sieciach komputerowych mamy do czynienia z dwoma rodzajami komunikacji: poprzez wymianę komunikatów oraz poprzez wspólną pamięć. Wymiana komunikatów jest stosowana między jednakowymi warstwami na dwóch różnych komputerach, podczas gdy wspólna pamięć służy do wymiany informacji między sąsiednimi warstwami protokołu sieciowego na danym komputerze. Za pomocą sieci Petriego można symulować i badać oba mechanizmy.



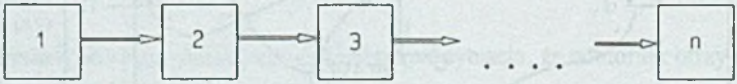
Rys. 2. Model sieci Petriego dla przykładowego protokołu komunikacyjnego
Fig. 2. Petri net model for an example of communication protocol

Podstawowe pojęcia z teorii sieci Petriego, takie jak definicje sieci, sieci Petriego, markowania, tranzycji (i ich odpalania), zagadnienia żywości i bezpieczności sieci zostały opisane w publikacji [2].

Sieć Petriego dla protokołu *wyślij i czekaj* przedstawiono na rysunku 2. Transmisja odbywa się w kanale dopuszczającym błędy. W odróżnieniu od poprzedniej interpretacji nie występują tutaj stany złożone. Stan nadajnika, odbiornika i kanału pokazano oddzielnie. Tranzycje:

1 — przesłanie ramki 0 normalnie,
2 — przesłanie ramki 0 z przekroczeniem limitu czasu,
3 — przesłanie ramki 1 normalnie,
4 — przesłanie ramki 1 z przekroczeniem limitu czasu,
5 — zgubienie ramki 0,
6 — zgubienie potwierdzenia,
7 — zgubienie ramki 1,
8 — w odbiorniku pojawia się ramka z błędnym numerem sekwencyjnym,
9 — w odbiorniku pojawia się ramka z błędnym numerem sekwencyjnym,
10 — pojawienie się w odbiorniku kolejnej ramki i przekazanie jej do komputera obliczeniowego,
11 — pojawienie się w odbiorniku kolejnej ramki i przekazanie jej do komputera obliczeniowego.

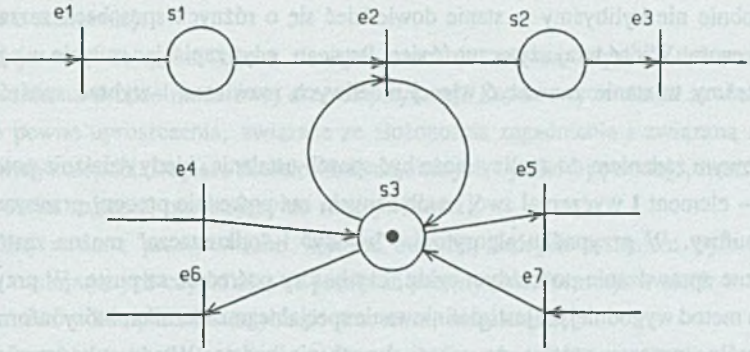
Niepoprawna byłaby tu sekwencja zapaleń zawierająca dwa przejścia o numerze 10 bez przejścia 11. Impasy w sieci Petriego są podobne do omawianych wcześniej grafów.



Rys. 3. Potokowe przetwarzanie danych
Fig. 3. Pipelined data processing

Przykładem może być komunikacja i przekazywanie sterowania między procesami, które potokowo przetwarzają strumień danych (rys. 3). Dane na początku są zgromadzone w elemencie **1**, skąd są przekazywane do poszczególnych procesów potoku przetwarzającego i ostatecznie gromadzą się w elemencie **n**. Zakładamy, że między elementami potoku znajdują się bufory działające według regulaminu FIFO oraz że wszystkie procesy potoku są realizowane przez jeden procesor. W taki sposób są wykonywane polecenia powłoki systemu UNIX (potoki są realizowane w pamięci) oraz potoki w systemie MS-DOS (tutaj rolę buforów spełnia dysk — pliki tymczasowe są przechowywane w katalogu TEMP).

Można się zastanowić, w jaki sposób jeden procesor powinien się przełączać między kolejnymi procesami potoku. Z punktu widzenia pojedynczego procesu **i**, postrzega on swe środowisko jako dwa bufory: wejściowy, z którego pobiera dane, oraz wyjściowy, do którego



Rys. 4. Model sieci Petriego dla procesu wewnątrz potoku
Fig. 4. Petri net model for a process inside pipeline

przekazuje przetworzone dane. Proces i ma też swego poprzednika oraz następnika — procesy, z którymi wymieniamy informacje. Można przedstawić działanie procesu i w postaci sieci Petriego (rys. 4). Proces widzi bufor wejściowy s1, w którym znaczniki reprezentują dane. Odpalenie tranzycji e2 symbolizuje przetwarzanie danych, co powoduje pobranie pewnej liczby znaczników z bufora s1 i umieszczenie znaczników (może się zmienić ich ilość w czasie przetwarzania) w buforze s2, a samo zajście tranzycji e2 będzie miało miejsce w przypadku obecności znacznika w miejscu s3 — markowanie tego miejsca oznacza, że bieżący proces otrzymał sterowanie. W czasie realizacji przetwarzania przez cały potok jest możliwe przekazywanie sterowania następnemu procesowi w potoku (tranzycje e4, e5) lub poprzedniemu procesowi (tranzycje e6, e7). Dzięki ogólności takiego modelu można teraz wyobrazić sobie różne strategie przydziału czasu procesora dla poszczególnych procesów, np.:

— przyznawanie czasu procesora po kolei, według numerów procesów. Analogiczne algorytmy w literaturze noszą nazwę algorytmu "odkurzacza",
— przyznawanie czasu procesora po kolei procesom, w obu kierunkach — mamy do czynienia z algorytmem "windy",
— algorytm adaptacyjny — kiedy proces zapisuje do bufora, ktory jest pełny, sterowanie zostaje przekazane następnemu procesowi (jego działanie opróżni nasz bufor wyjściowy). W momencie, gdy procesowi zabraknie danych wejściowych, sterowanie zostaje przekazane do poprzedniego procesu (ten zapełni nasz bufor wejściowy),
— algorytm dowolny — jest on przedstawiony na rys. 4. Znacznik aktywności danego procesu (markowanie miejsca s3) może swobodnie poruszać się wzdłuż potoku.

Gdyby próbować od razu zaimplementować program realizujący taki potok procesów, prawdopodobnie nie bylibyśmy w stanie dowiedzieć się o różnych sposobach zarządzania czasem procesora. Widać tutaj użyteczność sieci Petriego, gdyż zapisując zadanie w ogólnym modelu jesteśmy w stanie zauważyć więcej możliwych rozwiązań i wybrać spośród nich najlepsze.

Dodatkowym zadaniem do analizy może być sposób ustalenia, kiedy działanie potoku się skończyło — element 1 wyczerpał swój zasób danych, zaś pośrednie procesy przetwarzające opróżniły bufory. W przypadku algorytmów "windy" i "odkurzacza" można zastosować systematyczne sprawdzanie po każdym cyklu, czy bufory pośrednie są puste. W przypadku pozostałych metod wygodniejsze jest zdefiniowanie specjalnego znacznika, który informowałby poszczególne procesy potoku, że więcej danych nie będzie. Wtedy zakończenie przetwarzania nastąpiłoby po dotarciu tego znacznika do elementu n.

Przedstawione powyżej przykłady pozwalają wnioskować, że sieci Petriego są odpowiednie do oceny protokołów - pozwalają analizować wiele problemów z elementami współbieżności i równoległości, z jakimi mamy do czynienia w środowiskach wielozadaniowych i sieciowych. Szeroka gama odmian sieci Petriego (kolorowane, z zależnościami czasowymi, obiektowo zorientowane) pozwalają na rozwiązywanie problemów, z którymi inne modele, np. maszyny o skończonej liczbie stanów, nie dają sobie rady. Szukanie rozwiązań niebanalnych zadań wiąże się zwykle z symulacją bardzo skomplikowanej sieci Petriego, ale dzięki temu można znaleźć błędy w pozornie poprawnych systemach.

2. Testowanie pracy protokołów w warunkach rzeczywistych

Matematyczne lub symulacyjne próby badania efektywności, jak i weryfikacji protokołów, pozwalają na wstępne oszacowanie poprawności danego protokołu. Jednakże złożoność konstrukcji logicznej oraz fizycznej sieci, a w związku z tym mnogość nieprzewidywalnych sytuacji mogących w niej wystąpić, powoduje, że wszystkie modele teoretyczne powinny zostać zweryfikowane na drodze empirycznej.

Do problemu tego można podejść w dwojaki sposób. Pierwszy to badanie rzeczywistej sieci. Jej stan chwilowy jest uzależniony od tego, jaka liczba użytkowników na niej pracuje i z jakiego oprogramowania oni korzystają. Metoda ta ma pewną wadę. Aby uzyskać jakieś konkretne wyniki, testowanie musiałoby trwać pewien określony cykl pracy, w którym wystąpiłyby wszystkie stany sieci. Wydaje się, że oszacowanie, jak długo miałby trwać taki cykl badawczy, jest zadaniem bardzo trudnym, a dla sieci rozległych wręcz niemożliwym. Dlatego też należy zaopatrzyć się w narzędzia generujące sztuczny ruch w sieci. Z ich pomocą można spróbować wprowadzić sieć w stany interesujące dla problemów oceny

(przeciążenia, duży stopień uszkodzonych danych, stany awaryjne stacji roboczych oraz inne nietypowe zdarzenia).

Wydaje się, że pełną ocenę danego protokołu można uzyskać dokonując badań jednocześnie na drodze modelowej oraz empirycznej. Każda z tych metod zawiera bowiem w sobie pewne uproszczenia, związane ze złożonością zagadnienia i związaną z nią dozą nieprzewidywalności. Dopiero skorelowanie uzyskanych wyników powoduje, że otrzymujemy pewien obraz najlepiej przystający do rzeczywistości.

Poniżej zostaną przedstawione wstępne badania, których zadaniem było określenie wpływu następujących czynników na prędkość przepływu informacji w sieci:

— parametrów komputerów (szybkość procesora, magistrala),
— rozwiązań karty kontrolera sieci (magistrala, bufory, sposób komunikacji z procesorem),
— efektywności programów obsługi (driverów),
— zastosowanego protokołu komunikacyjnego.

Badań dokonano za pomocą specjalnie stworzonego w tym celu oprogramowania.

2.1. Badanie sieci Ethernet

Teoretyczna przepustowość sieci Ethernet jest znana i wynosi 10 Mb/s = 1.25 MB/s. Jednak rzeczywista szybkość przesyłania informacji na pewno będzie mniejsza.

Odpowiedź na pytanie o wpływ wymienionych wyżej czynników uzyskamy badając maksymalną przepustowość kanału. W tym celu badania przeprowadzono dla przesyłu pomiędzy dwoma stacjami, uniemożliwiając w sposób administracyjny dostęp do łącza innym stacjom — były one wyłączone. W badaniach wyeliminowano również kanał dyskowy komputera, którego wpływ na transmisje informacji w sieci jest czynnikiem na tyle dominującym, że mógłby on spowodować nieczytelność uzyskanych wyników.

Programy do pomiaru wydajności mechanizmów sieciowych (kart, komputerów, protokołów, driverów) dokonują transmisji ramek z maksymalną prędkością, na adres sieciowy odbiorcy. Zostaje zmierzony czas transmisji wszystkich nadanych ramek. Odbiorca odbiera ramki z maksymalną prędkością, każdą kopiując do bufora w pamięci.

Przepustowość P mierzona jest według wzoru:

$$P = \frac{l_p * r_p}{t * 1024} \quad \text{[kb/s]},$$

gdzie

l_p — liczba nadanych pakietów (w bajtach),
r_p — rozmiar pakietu,
t — czas transmisji.

Przepustowość mierzona jest dla stacji nadającej ramki. Zdarza się, że stacja odbiorcza nie nadąża z odbiorem szybko wysyłanych ramek.

Oprogramowanie zostało napisane w oparciu o funkcje packet drivera. Przeprowadzony został pomiar pomiędzy dwoma komputerami:

— komputer P: Pentium 75, karta ISA EtherExpress 10/PRO+,

— komputer T: 486 DX 25, karta ISA EtherExpress 16.

Program S1 wysyła 10000 ramek o zadanej długości. Program R1 odbiera ramki kopiując je do bufora. Program R2 odbiera ramki bez ich kopiowania do pamięci (anuluje je).

Została obliczona średnia przepustowość dla kilku pomiarów.

Wyniki:

komputer P -> komputer T:

dł.ramki	przepustowość	R1 odebrał ramek	R2 odebrał ramek
1514	1197	7083	10000
700	1178	5003	10000
60	862	2	3

komputer T -> komputer P:

dł.ramki	przepustowość	R1 odebrał ramek	R2 odebrał ramek
1514	445	10000	10000
700	418	10000	10000
60	187	3446	3446

W czasie badań uzyskano transfer 1.17 MB/s, co niewiele odbiega od teoretycznej przepustowości Ethernetu (1.25 MB/s). Widać jednak, że odbiorca odebrał mniej ramek, niż do niego nadano — transmisja była dla niego za szybka. Można też zauważyć, że kopiowanie do bufora zajmuje też dużo czasu — program R2 odebrał wszystkie ramki.

2.2. Badanie protokołów warstwy sieciowej

Wysyłając pakiet warstwy sieciowej trzeba się liczyć z faktem, że transmisja będzie mniej efektywna. Trzeba zapewnić miejsce dla nagłówka protokołu sieciowego, a dane muszą być przetworzone przez oprogramowanie tej warstwy. Stąd szybkość transmisji powinna być niższa. Dla wykonanych badań nośnikiem pakietów była ramka 802.3 sieci Ethernet.

2.2.1. Protokół IPX

Jest to protokół warstwy sieciowej używany w produktach firmy Novell. Badania przeprowadzono na komputerach:

— komputer A: Pentium 75, karta ISA NE2000,

— komputer P: Pentium 75, karta ISA EtherExpress 10/PRO+,

— komputer T: 486 DX 25, karta ISA EtherExpress 16.

Program nadający wysyła 10000 ramek o długości 576 bajtów (łącznie z 30-bajtowym nagłówkiem IPX). Odwołuje się do sieci asynchronicznie.

W trakcie badań okazało się, że driver o nazwie pdipx, przedstawiający się jako:

Novell IPX/SPX v3.10 (911121)

(C) Copyright 1985, 1991 Novell Inc. All Rights Reserved.

LAN Option: Packet Driver V1.03EC (920928) Intel (c) 1992, No Warrenty

Hardware Configuration: First Packet Driver, Find Interrupt

nie zachowuje się poprawnie. Bardzo spowalnia odbiór i uniemożliwia wysyłanie ramek. Jest to driver sieciowy implementujący protokół IPX, działający na driverze pakietowym. Pochodzi z pakietu dystrybucyjnego karty sieciowej SMC. Nazwijmy go D1. Został on wykorzystany na komputerze P.

Dla porównania zamieszczono również wyniki dla standardowego drivera o nazwie ipxpdi.exe, dostępnego z oprogramowaniem firmy Novell.

Novell IPX/SPX v3.01 Rev. A (900507)

(C) Copyright 1985, 1990 Novell Inc. All Rights Reserved.

LAN Option: Packet Driver V2.01 (890427) BYU Copyright 1989

Hardware Configuration: Packet Driver Interface. Find Interrupt.

Nazwijmy go D2.

Wyniki:

transfer	przepustowość	odebrano ramek z D1	odebrano ramek z D2
A->P	736	140	10000
A->T	736	4606	-
T->A	354	10000	-
T->P	354	290	10000
P->T	1136 (tylko z D2)	-	16
P->A	1136 (tylko z D2)	-	1507

Komputer P z driverem D1 zawieszał się przy próbie nadania pakietu, natomiast z driverem D2 działał bez zarzutu. Można zauważyć, że najszybsza karta sieciowa jest w komputerze P, potem w A; w komputerze T jest najwolniejsza. Widać też, że przepustowość protokołu IPX na packet driverze jest mniejsza od przepustowości na poziomie warstwy liniowej.

Przeprowadzono też analogiczne badania protokołu IPX na driverach ODI.

Wyniki:

transfer	przepustowość	odebrano ramek
A->P	1125	10000
A->T	1125	4284
T->A	1055	5704
T->P	1055	10000
P->T	1090	4207
P->A	1090	5386

Wyraźnie widać, że przepustowość na driverach ODI jest szybsza nawet od samego packet drivera. Komputery T i A nie nadążają jednak z odbiorem ramek. Przepustowość gwarantująca dostarczenie danych byłaby więc dla nich ponad dwukrotnie mniejsza.

2.2.2. Protokół NetBIOS

NetBIOS jest to protokół warstwy sesji, działający w sieci lokalnej. Programy korzystające z usług komunikacyjnych zgodnie ze stykiem NetBIOS mogą być uruchamiane bez żadnych adaptacji w bardzo różnorodnych konfiguracjach sprzętowo-programowych. NetBIOS jest traktowany jako wspólna platforma oprogramowania aplikacyjnego, gdzie funkcje NetBIOS realizują zadania najniższych warstw modelu ISO-OSI (warstwa fizyczna, łącza danych, sieciowa i transportowa).

Badania przeprowadzono na komputerach:

— komputer A: Pentium 75, karta ISA NE2000,
— komputer P: Pentium 75, karta ISA EtherExpress 10/PRO+,
— komputer T: 486 DX 25, karta ISA EtherExpress 16.

Program nadający wysyła 10000 ramek o długości 512 bajtów. Program używa interfejsu packet driver. Ramki odbierane są w sposób asynchroniczny z testowaniem pola ncb_cmd_cplt.

Wyniki:

transfer	przepustowość	odebrano ramek
A->P	930	10000
A->T	930	243
T->A	565	10000
T->P	565	10000
P->T	969	86
P->A	969	3774

Przeprowadzono analogiczne badania również na driverach ODI, ale nie stwierdzono większych różnic w wynikach. Dla komputerów T i A operacje NetBIOS są nieco szybsze niż dla protokołu IPX. Potwierdza się poprzednie oszacowanie wydajności badanych komputerów.

Protokół NetBIOS, a właściwie jego rozszerzona wersja NetBEUI, jest stosowany także jako podstawowy protokół w sieci Windows For Workgroups (WFW) firmy Microsoft. Na jednym segmencie sieci Ethernet znajdował się server Novell NetWare, server Windows NT 3.51 oraz komputer będący DOSowym klientem sieci WFW. Przy transmisji plików z serwera Windows NT do klienta DOSowego okazało się, że ramki są wysyłane pośrednio, przez server NetWare, co znacznie spowolniło transmisję oraz niepotrzebnie obciążało server NetWare. Ten przykład pokazuje, że nawet dobrze działający protokół może pracować mało wydajnie w szczególnych warunkach.

2.3. Wnioski

Podczas badań udowodniono, że na rzeczywiste zachowanie protokołów, mają dość znaczny wpływ warunki sprzętowe i programowe stacji roboczej.

Dla krótkich ramek spada wyraźnie przepustowość — przesył jest mało efektywny, dużo czasu zajmuje przygotowanie ramki do wysłania. Odbiorcy odebrali bardzo mało ramek. Wniosek jest oczywisty: raczej należy nadawać długie ramki, nie obawiając się o ich uszkodzenie. W sieciach lokalnych nie zdarzają się praktycznie błędy transmisji, co zostało potwierdzone w czasie przeprowadzanych eksperymentów.

W czasie badań zaobserwowano ciekawą własność sieci Ethernet. Jeśli jedna stacja nadaje z pełną prędkością, to innym stacjom, chociaż algorytm CSMA/CD teoretycznie zapewnia równe szanse w dostępie (publikacja [3]), trudno dostać się do łącza. Podczas testu niemożliwe było skorzystanie z jakiejkolwiek usługi sieciowej na innych stacjach.

Ocena protokołów komunikacyjnych w sieciach komputerowych pozwala na sprawdzenie poprawności i wydajności danego protokołu w konkretnych warunkach. Przez zastosowanie zarówno narzędzi teoretycznych, jak i empirycznego testowania protokolu mamy możliwość dokonania tej oceny w sposób wszechstronny.

LITERATURA

[1] Tanenbaum A.: Computer Networks. Prentice/Hall International, Inc., Englewood Cliffs 1981.

[2] Starke P.: Sieci Petri. PWN, Warszawa 1987.

[3] Sheldon T.: Wielka Encyklopedia Sieci Komputerowych. McGrow-Hill, Berkeley 1994.

[4] Jensen K.: Coloured Petri Nets: A High-level Language for System Design and Analysis. LNCS vol. 483, Springer Verlag 1990.

Recenzent: Dr inż. Ryszard Winiarczyk

Wpłynęło do Redakcji 2 grudnia 1996 r.

Abstract

This article presents problems related to the research of communication protocols that are used in computer networks. The layered model was chosen as a reference point for protocol classification. The aspects that should be taken into consideration during protocol evaluation: efficiency, transparency, protocol verification were discussed. The Stop-and-Wait simplex protocol was an example of a protocol to estimate. Petri net model was used for theoretical verification of the protocol. The suitability for implementation of pipelines and other parallel mechanisms as Petri nets was noticed. The final thesis suggests that theoretical research of communication protocols should be completed with empirical reseach.