ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994

Seria: INFORMATYKA z. 27 Nr kol. 1259

Marcin SKOWRONEK

WYKORZYSTANIE SIECI NETWARE DO RÓWNOLEGŁEJ REALIZACJI ZADAŃ MODELOWANIA

Streszczenie. W artykule przedstawiono koncepcję równoległej realizacji sekwencji eksperymentów, w ramach badań modelowych, w środowisku sieci NetWare. Przedstawiono przykładowe algorytmy *Nadawcy* i *Wykonawcy* uwzględniające cechy środków modelowania. Umieszczono wyniki wykonanych badań oceny efektywności implementacji zaproponowanych algorytmów.

USAGE OF THE NETWARE NETWORK FOR PARALLEL PROCESSING OF THE MODELLING TASKS

Summary. The idea of parallel processing of the sequence of experiments (occurring in the modelling tasks) in the NetWare network environment is presented in this paper. As the examples are described *The Sender* and *The Executor* algorithms, that were built with restrictions resulting from modelling means. The results of the experiments connected with the efficiency estimate of the algorithms are also included.

AUSNUTZUNG DES NETWARE-NETZES ZUR PARALLELEN AUSFÜHRUNG DER MODELLIERUNGSAUFGABEN

Zusammenfassung. Im Artikel wurde das Konzept einer parallelen Ausführung der Experimentenfolge im Rahmen der Modellierungsforschung in der Umgebung des NetWare-Netzes dargestellt. Es wurden die Musteralgorithmen *Der Absender* und

Pracę wykonano w ramach grantu KBN 3 P406 011 04

Der Ausführer vorgeschlagen, welche die Merkmale der Modellierungsmittel berücksichtigen. Der Artikel beinhaltet auch Ergebnisse durchgeführter Forschung der Effizienz von Implementierung o.g. Algorithmen.

1. Wprowadzenie

Poznawanie właściwości układu dynamicznego drogą modelowania jest w wielu przypadkach zadaniem czasochłonnym zarówno z uwagi na złożoność niektórych modeli, jak i z uwagi na potrzebę wykonania dużej liczby (sekwencji) eksperymentów w ramach badań modelowych. Ogólnym trendem jest więc realizacja tego typu zadań na coraz szybszych i wydajniejszych procesorach. Drugim, zazwyczaj nie docenianym, sposobem skrócenia czasu badań modelowych może być pełniejsze wykorzystanie możliwości, jakie udostępniają popularne sieci NetWare firmy Novell.

Koncepcję wykorzystania sieci NetWare do realizacji przetwarzania równoległego przedstawiono w pracy [1]. Istotą tej koncepcji jest to, że pamięć dyskowa serwera sieci, udostępniana stacjom, może być również wykorzystana do celów komunikacji między procesami realizowanymi na poszczególnych stacjach. Do realizacji tej komunikacji wykorzystywane są operacje dyskowe:

- sprawdzenia istnienia pliku o określonej nazwie,
- przemianowania nazwy lub usunięcia pliku o określonej nazwie,
- odczytania zawartości wskazanego pliku,
- zapisu informacji do wskazanego pliku.

Równoległa realizacja algorytmu w sieci polega ogólnie na tym, że niektóre operacje algorytmu wykonywane są współbieżnie na różnych stacjach. Na jednej wybranej stacji wykonywany jest tzw. proces *Nadawcy* – spełniający funkcje inicjujące i sterujące, a na pozostałych stacjach wykonywane są procesy *Wykonawców* – spełniające funkcje usługowe dla procesu *Nadawcy* lub przygotowujące dane dla innych procesów *Wykonawców*. Realizacja kolejnych operacji algorytmu przez procesy *Wykonawców* i proces *Nadawcy* może być przykładem systemu sterowanego przepływem argumentów [2]. Poszczególne operacje algorytmu realizowane są dopiero wtedy, gdy pojawią się wymagane do ich realizacji argumenty. Przez argument będziemy tu rozumieli dane dla procesu *Wykonawcy*, wynik jednostkowej operacji uzyskany w procesie *Wykonawcy* lub informacje o stanie *Wykonawcy*.

2. Charakterystyka badań modelowych

Badania modelowe sprowadzają się zazwyczaj do wykonania sekwencji eksperymentów z wykorzystaniem modelu badanego układu dynamicznego. *Eksperymentem* nazywane jest umownie pewne zadanie jednostkowe, generujące określone wyniki dla zadanych parametrów modelu. Przykładem eksperymentu może być rozwiązanie zagadnienia początkowego, to jest rozwiązanie układu równań różniczkowych w zadanym przedziale czasu, przy zadanych warunkach początkowych i zadanych parametrach modelu. Innym przykładem eksperymentu może być wyznaczanie określonej statystyki modelu układu zdarzeń dyskretnych, dla zadanych wartości parametrów modelu poprzez wykonanie określonej liczby obserwacji zmiennych opisujących stan modelu.

Wynikiem eksperymentu może być pojedyncza wartość liczbowa, jak również ciąg wartości liczbowych. Przykładem wyniku eksperymentu może być wartość wyznaczanej statystyki, końcowa wartość zmiennej opisu modelu, wektor statystyk, wektor końcowych wartości zmiennych opisu modelu lub ciąg wektorów prezentujących przebiegi zmiennych opisu modelu.

Konieczność realizacji sekwencji eksperymentów zachodzi wtedy, gdy celem modelowania jest wyznaczanie charakterystyk lub rodziny rozwiązań ilustrujących wpływ parametrów modelu na określone wyniki eksperymentu. W tego typu zadaniach ciąg wartości parametrów, dla których chcemy wyznaczyć wyniki eksperymentów, znany jest zazwyczaj z góry. Podobny problem występuje podczas rozwiązywania liniowego zagadnienia dwugranicznego. W tym przypadku, w pierwszej kolejności, wykonuje się sekwencje eksperymentów dla zadanych warunków początkowych, w celu wyznaczenia macierzy tranzycji, a następnie korzystając z macierzy tranzycji wyznacza się brakujący warunek początkowy i poszukiwane rozwiązanie zagadnienia dwugranicznego.

Innym przykładem realizacji sekwencji eksperymentów jest rozwiązywanie zadań optymalizacji parametrycznej. Znajdowanie rozwiązania tego typu zadań wymaga wykonania stosunkowo dużej liczby eksperymentów, przy czym ciąg wartości parametrów, dla których mają być wykonane eksperymenty nie jest z góry znany. Analizując przykładowe algorytmy stosowane do rozwiązywania tego typu zadań [3, 4], możemy stwierdzić, że rozwiązanie uzyskiwane jest drogą wykonania serii podzadań, przy czym dla każdego podzadania znany jest z góry ciąg parametrów, dla których mamy wykonać eksperymenty. Przykładowo rozwiązanie zadań optymalizacji parametrycznej sprowadza się do cyklicznego wykonywania podzadań

a) wyznaczania kierunku poszukiwań,

b) poszukiwania wzdłuż kierunku poszukiwań.

Realizacja podzadania a) może polegać na wykonaniu sekwencji eksperymentów w celu wyznaczenia gradientu wskaźnika jakości lub też na wyborze najlepszego kierunku poszukiwań ze zbioru losowo wygenerowanych kierunków poszukiwań dla zadanego wektora początkowych wartości parametrów modelu. Realizacja podzadania b), to wykonanie sekwencji eksperymentów dla parametrów modelu należących do kierunku poszukiwań, to jest należących do rozmaitości liniowej wyznaczonej przez początkowy wektor wartości parametrów modelu i kierunek poszukiwań wyznaczony w podzadaniu a). W wielu algorytmach optymalizacji w ramach podzadania b) znajdowane jest minimum wzdłuż kierunku poszukiwań.

3. Charakterystyka środków modelowania

Budując model układu dynamicznego korzystamy zazwyczaj z dostępnych środków modelowania w postaci gotowych programów, pakietów symulacyjnych lub języków modelowania. Wykorzystanie tych środków pozwala zbudować model układu dynamicznego przy niewielkim nakładzie czasowym. Uzyskane tą drogą modele można podzielić na tzw. *modele interakcyjne* i *modele wsadowe*. Pierwsze z tych modeli wymagają bezpośredniej ingerencji operatora i trudno je wykorzystać do przetwarzania równoległego. Drugie z tych modeli umożliwiają wykonanie pojedynczego eksperymentu lub sekwencji eksperymentów na podstawie zapisanego kodu programu, danych odczytywanych z pliku wejściowego lub przekazywanych jako parametry wywołania. Wyniki eksperymentu zapisywane są zazwyczaj w pliku wyjściowym. Model wsadowy, nazywany dalej *Modelem*, będzie podstawowym elementem programu realizowanego jako proces *Wykonawca* w równoległej realizacji sekwencji eksperymentów. Mogą tu wystąpić trzy przypadki:

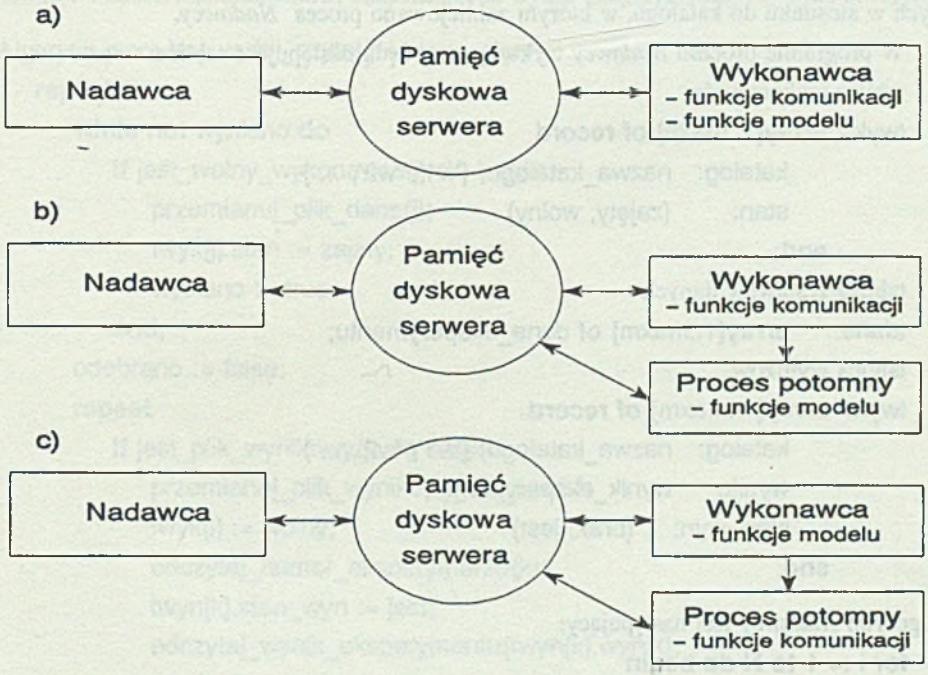
1. Program procesu *Wykonawca* jest pojedynczym programem. Zawarto w nim instrukcje opisujące funkcje komunikacji i funkcje modelu.
2. Program procesu *Wykonawca* bezpośrednio zawiera tylko instrukcje opisujące funkcje komunikacji. Realizację funkcji modelu uzyskuje się poprzez wywołanie potomnego procesu (programu *Model*).
3. Program procesu *Wykonawca* bezpośrednio zawiera tylko instrukcje opisujące funkcje modelu. Realizację funkcji komunikacji uzyskuje się poprzez wywołanie potomnego procesu (programu *Komunikacja*).

Rozwiązanie 1. może być zastosowane wtedy, gdy model zbudowany jest za pomocą pakietu symulacyjnego opartego na uniwersalnym języku algorytmicznym, np. Turbo Pasca-

lu, C lub wprost za pomocą uniwersalnego języka algorytmicznego. Opracowanie procedur komunikacji nie stanowi tu problemu.

Rozwiązanie 2. może być zastosowane prawie zawsze. Od *Modelu* wymaga się tylko, by posiadał wymienione wcześniej cechy modelu wsadowego, to znaczy: odczytywał dane z pliku wejściowego i przekazywał wyniki do pliku wyjściowego. Program procesu *Wykonawca* może być opracowany w dowolnym języku algorytmicznym lub też stanowić zbiór poleceń systemu operacyjnego (plik typu .BAT).

Rozwiązanie 3. może być zastosowane wtedy, gdy środki modelowania użyte do zbudowania modelu udostępniają w trybie wsadowym mechanizm inicjacji potomnego procesu, w którym realizowane byłyby funkcje komunikacji. Program procesu *Komunikacja* może być opracowany w dowolnym języku algorytmicznym lub też tak, jak w rozwiązaniu 2. stanowić zbiór poleceń systemu operacyjnego.



Rys. 1. Realizacja funkcji komunikacji i funkcji modelu
Fig. 1. The realization of communication and model functions

4. Koordynacja równoległej realizacji sekwencji eksperymentów

Zadanie wykonania sekwencji *M* eksperymentów sterowane będzie przez proces zwany *Nadawcą*. Eksperymenty wykonywane będą przez procesy zwane *Wykonawcami*, których liczba *N* jest znana *Nadawcy*. Proces *Nadawcy* i procesy *Wykonawców* realizowane są na różnych stacjach sieci. Przyjmiemy również, że każdy z procesów *Wykonawców* zainicjowany jest z innego katalogu bieżącego. Przyjęcie takiego założenia wynika stąd, że modele zbudowane za pomocą dostępnych środków modelowania często korzystają z plików roboczych o ustalonych nazwach. Inicjowanie procesów *Wykonawców* w oddzielnych katalogach jest tu prostym sposobem uniknięcia kolizji dostępu do plików roboczych modelu. Przyjęto, że procesy *Wykonawców* inicjowane są w katalogach o nazwach W1, W2, ..., W*N* podrzędnych w stosunku do katalogu, w którym zainicjowano proces *Nadawcy*.

W programie procesu *Nadawcy* wykorzystane będą następujące tablice:

- tablica wykonawców

```
twyk: array[1..maxn] of record
          katalog:  nazwa_katalogu; {'w1', 'w2', ... }
          stan:     (zajęty, wolny)
      end;
```

- tablica zestawów danych

```
tdane:  array[1..maxm] of dane_eksperymentu;
```

- tablica wyników

```
twyn: array[1..maxm] of record
          katalog:  nazwa_katalogu; {'w1', 'w2', ... }
          wynik:    wynik_eksperymentu;
          stan_wyn: (brak, jest)
      end;
```

Algorytm *Nadawcy* jest następujący:

```
for i := 1 to N do begin
   twyn[i].katalog := 'w' + napis(i);
   twyn[i].stan := wolny;
end;
ustawienie_tablicy_danych;
```

```
for i := 1 to N do begin
    utwórz_plik_dane(i);
    wysłano := false;
    wyślij_czytaj
end;
While not odczytano_wszystkie_wyniki do begin
    wysłano := true;
    wyślij_czytaj;
end;
```

Procedura utwórz_plik_dane(i) tworzy plik o nazwie DANE zawierający nr eksperymentu (i) oraz dane wejściowe dla tego eksperymentu. Procedura wyślij_czytaj koordynuje proces wysyłania komunikatów, oznaczających żądanie wykonania eksperymentu, do *Wykonawców* i odbioru komunikatów, oznaczających wynik eksperymentu, od *Wykonawców*.

Algorytm procedury wyślij_czytaj jest następujący:

```
repeat
    while not wysłano do
        if jest_wolny_wykonawca(j) do begin
            przemianuj_plik_dane(j);
            twyk[j].stan := zajęty;
            wysłano := true
        end;
    odebrano := false;
    repeat
        if jest_plik_wynikowy(j) do begin
            przemianuj_plik_wynikowy_na_wyniki;
            twyk[j] := wolny;
            odczytaj_numer_eksperymentu(k);
            twyn[k].stan_wyn := jest;
            odczytaj_wynik_eksperymentu(twyn[k].wynik);
            usuń_plik_wyniki;
            odebrano := true;
        end;
    until wysłano or odebrano;
until wysłano;
```

Na podstawie tabeli twyk znajdowany jest wolny wykonawca (j) i następuje przemianowanie pliku DANE na plik Wj\DANE, gdzie Wj jest nazwą katalogu wolnego *Wykonawcy*. Wyniki eksperymentu przekazywane są w plikach o nazwie WYNIKI.Wj, gdzie rozszerzenie Wj wskazuje *Wykonawcę* eksperymentu. Plik wynikowy zawiera numer eksperymentu (k) oraz wynik eksperymentu.

Algorytm znajdowania wolnego wykonawcy może być różnie zrealizowany, na przykład może to być losowe przeszukiwanie wolnego wykonawcy. W praktycznych zastosowaniach, gdy znamy charakterystyki określające szybkość (wydajność) procesorów poszczególnych stacji, możemy uporządkować stacje i realizowane na nich procesy *Wykonawców* od najszybszego do najwolniejszego i poszukiwać wolnego *Wykonawcy* zaczynając od najszybszego. Taki sposób wyboru wolnego *Wykonawcy* zastosowano w implementacji algorytmu *Nadawcy* wykonanej w celu przeprowadzenia badań testowych.

Algorytm *Wykonawcy* zależeć będzie od tego, czy funkcje komunikacji i funkcje modelu zrealizowane będą w jednym lub w dwóch procesach. Przykładowe postacie algorytmu *Wykonawcy* przedstawiono poniżej:

Algorytm *Wykonawcy* (przypadek 1.)

```
kat := nazwa_katalogu_bieżącego;
repeat
    czekaj_na_plik('dane');
    przemianuj_plik('dane', 'rob');
    odczytaj_numer_zestawu('rob', i);
    odczytaj_dane_eksperymentu('rob');
    wykonaj_eksperyment;
    zapisz_numer_zestawu('rob', i);
    zapisz_wyniki_eksperymentu('rob');
    przemianuj_plik('rob', '..\'+'wyniki'+'.'+kat);
until false;
```

Algorytm *Wykonawcy* (przypadek 2.)

```
kat := nazwa_katalogu_bieżącego;
repeat
    czekaj_na_plik('dane');
    przemianuj_plik('dane', 'rob');
    wywołaj_program('..\model', 'rob', 'wyniki');
    usuń_plik('rob');
    przemianuj_plik('wyniki', '..\'+'wyniki'+'.'+kat);
until false;
```

Przyjęto tu, że program **model**, dostępny w katalogu nadrzędnym, czyta dane z pliku ROB, wykonuje eksperyment i zapisuje wyniki eksperymentu do pliku WYNIKI.

Algorytm *Wykonawcy* (przypadek 3.)

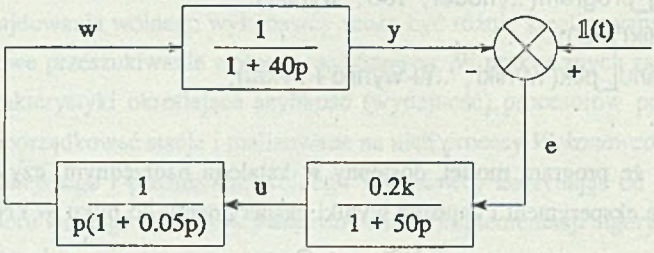
```
repeat
    wywołaj_program('..\komunikacja', 'dane', 'rob', 'wyniki');
    odczytaj_numer_zestawu('rob', i);
    odczytaj_dane_eksperymentu('rob');
    wykonaj_eksperyment;
    zapisz_numer_zestawu('wyniki', i);
    zapisz_wyniki_eksperymentu('wyniki');
until false;
```

gdzie algorytm programu *komunikacja* jest następujący:

```
kat := nazwa_katalogu_bieżącego;
if jest_plik('rob') then
    usuń('rob');
if jest_plik('wyniki') then
    przemianuj_plik('wyniki', '..\'+'wyniki'+'.'+kat);
czekaj_na_plik('dane');
przemianuj_plik('dane', 'rob');
```

5. Wyniki badań

Do oceny efektów, jakie przynosi równoległa realizacja eksperymentów, wykonano zestaw badań na modelu układu dynamicznego o schemacie blokowym przedstawionym na rys. 2.



Rys. 2. Schemat blokowy układu dynamicznego
Fig. 2. The block diagram of the dynamical system

Celem modelowania było wyznaczenie charakterystyki

$$Q(k) = \int_0^{1000} e^2\,dt$$

dla k = 0.02, 0.04, ..., 0.32.

Model układu dynamicznego zbudowano za pomocą pakietu symulacyjnego DarP [5]. Pakiet DarP oparty na języku algorytmicznym Turbo Pascal umożliwia zapis każdego z przedstawionych przypadków realizacji algorytmu *Wykonawcy*. Algorytm *Nadawcy* opracowano w języku Turbo Pascal. Stacje, na których wykonywano badania, były wyposażone w koprocesor arytmetyczny. Charakterystykę stacji wykorzystywanych w badaniach ilustruje tabela 1.

Ocenę rozpatrywanych trzech przypadków realizacji algorytmu *Wykonawcy*, w układzie *Nadawca* (stacja I) i jeden *Wykonawca* (stacja H), przedstawiono w tabeli 2.

Z uzyskanych wyników badań wynika, że dla przypadku 1. narzut czasu związany z komunikacją procesów *Nadawcy* i *Wykonawcy* nieznacznie przekracza 1[s]. Wyniki uzyskane dla przypadków 2. i 3. ilustrują, jak zwiększa się czas badań wskutek wczytywania programu procesu potomnego. Objętości tych programów wynosiły:

161840 [kB] (przypadek 2.)
2080 [kB] (przypadek 3.)

Ostatnia kolumna tabeli 2 dotyczy przypadku 3., gdzie program procesu potomnego stanowi zbiór poleceń systemu operacyjnego.

Tabela 1

Charakterystyka stacji wykorzystywanych w badaniach

Oznaczenie stacji	Procesor	CPU speed (SI z NU 6.01)	Czas badań w [s]
A	486DX2-66	131.9	38.78
B	486DX2-66	131.9	39.87
C	486DX-50	108.3	52.34
D	486DX-50	107.5	73.60
E	486DX-33	72.0	74.26
F	486DX-33	71.3	72.99
G	486DX-33	71.8	76.51
H	386DX-33 + koprocesor	34.2	174.39
I	386SX-25 + koprocesor	12.4	---

Tabela 2

Czasy badań modelowych dla trzech przypadków realizacji algorytmu *Wykonawcy*

Zadania *Nadawcy* i *Wykonawcy* w jednym procesie	*Wykonawca*			
	przypadek 1.	przypadek 2.	przypadek 3.	przypadek 3. (plik .bat)
173.24	174.39	197.68	177.91	179.77

W tabelach 3 i 4 przedstawiono czasy wykonania badania, gdy funkcje *Modelu* realizowane są przez kilku *Wykonawców* (przypadek 1.). Proces *Nadawcy* tak jak w poprzednich przykładach realizowany był na stacji I.

Wyniki przedstawione w tabeli 3 ilustrują rzeczywisty przypadek sieci, w której poszczególne stacje posiadają różną wydajność, co jest często powodem, że przyśpieszenie wykonania badań nie jest proporcjonalne do liczby użytych stacji. Drugą przyczyna tego jest nierównomierne obciążenie poszczególnych stacji. Zwiększanie liczby stacji powoduje generalnie skrócenie czasu wykonania badania. W przeprowadzonych badaniach wyjątkiem jest ostatnie doświadczenie (8 stacji, od A do H), w którym czas wykonania badań jest większy niż dla doświadczeń z sześcioma lub siedmioma stacjami. Ostatnia kolumna tabeli 3 prezentuje liczby eksperymentów wykonane przez procesy *Wykonawców* na poszczególnych stacjach.

Tabela 3

Czasy wykonania badania, gdy funkcje *Modelu* wykonywane są przez kilku *Wykonawców*

Stacje	Czas badania w [s]	Przyśpieszenie	Liczba wykonanych ekperymentów
A	38.78	1	16
A,B	19.90	1.95	8,8
A,B,C	14.99	2.58	6,6,4
A,B,C,D	13.90	2.78	5,4,4,3
A,B,C,D,E	13.90	2.78	4,4,3,3,2
A,B,C,D,E,F	9.89	3.92	4,3,3,2,2,2
A,B,C,D,E,F,G	9.72	3.99	3,3,2,2,2,2,2
A,B,C,D,E,F,G,H	11.10	3.45	3,2,2,2,2,2,2,1

Tabela 4

Czasy wykonania badania, gdy funkcje *Modelu* wykonywane są przez kilku *Wykonawców*

Stacje	Czas badania w [s]	Przyśpieszenie	Liczba wykonanych ekperymentów
D	73.60	1	16
D,E	37.19	1.98	8,8
D,E,F	27.13	2.71	6,5,5
D,E,F,G	19.23	3.82	4,4,4,4

W tabeli 4 przedstawiono czasy wykonania badania, gdy wydajności stacji, na których realizowane są procesy *Wykonawców*, są porównywalne.

6. Uwagi końcowe

Przeprowadzone badania potwierdzają celowość wykorzystania sieci typu NetWare do równoległej realizacji sekwencji eksperymentów podczas badania modeli układów dynamicznych. Eksperymenty w tego typu badaniach są często czasochłonne, a w wielu przypadkach zachodzi konieczność wykonania kilkudziesięciu lub kilkuset eksperymentów w ramach sekwencji lub też zachodzi konieczność wykonania serii sekwencji eksperymentów na przykład podczas rozwiązywania zadań optymalizacji parametrycznej.

Przyjęte tu założenie, że *Nadawca* zna liczbę stacji, na których realizowane są procesy *Wykonawców* i ich uporządkowanie ze względu na wydajność, pozwala w zależności od liczby eksperymentów w sekwencji na wykorzystanie wszystkich lub tylko najszybszych stacji. Przedstawione rozwiązanie nie wymaga również stosowania operacji semaforowych. Porównując czas wykonania badania na jednej stacji z czasem wykonania badania w układzie *Nadawca* – jeden *Wykonawca* (tabela 2), możemy stwierdzić, że czasy transmisji komunikatów są tu ułamkiem czasu wykonania eksperymentów. Zastosowanie oddzielnych procesów do realizacji funkcji modelu i funkcji komunikacji powoduje zazwyczaj zwiększenie czasu wykonania badania.

LITERATURA

[1] Wilk A.: Koncepcja wykorzystania sieci „NetWare" do realizacji przetwarzania równoległego. ZN Pol. Śl. s. Informatyka, z. 26, Gliwice 1994.

[2] Węgrzyn S.: Systemy sterowane przepływem operacji i systemy sterowane przepływem argumentów. ZN Pol. Śl. s. Informatyka, z. 24, Gliwice 1993.

[3] Findeisen W., Szymanowski J., Wierzbicki A.: Teoria i metody obliczeniowe optymalizacji. PWN, Warszawa 1977.

[4] Skowronek M., Starzewska-Karwan E.: Algorytmy optymalizacji w zadaniach modelowania. ZN Pol. Śl. s. Informatyka, z. 27, Gliwice 1994.

[5] Skowronek M.: Modelowanie cyfrowe układów dynamicznych. Wyd. II, Skrypt Pol. Śl. nr 1776, Gliwice 1993.

Recenzent: Dr inż. Andrzej Wilk

Wpłynęło do Redakcji 26 lipca 1994 r.

Abstract

In the paper was discussed the possibility of parallel processing of the sequence of experiments (often occurring in modelling tasks) in the Novell NetWare network environment. It was shown that *The Executor* algorithm depends on the features of the modelling means, that were used for building the digital model. It was assumed, that *The Sender* knows the number of the workstations, in which *The Executor* processes are performed, as well as their order from the point of view of efficiency.

The Executor algorithms proposed in the paper were implemented using the simulation package DarP. The execution time of experiment sequences was measured for different *Executor* algorithms (Table 2) and it was checked how the execution time depends on the number of workstations (Table 3 and 4).

The results of the tests confirm, that the Novell NetWare network can be used for parallel processing of the sequence of experiments.