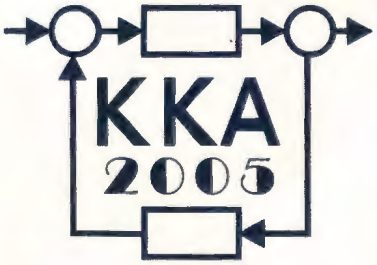
XV Krajowa Konferencja Automatyki

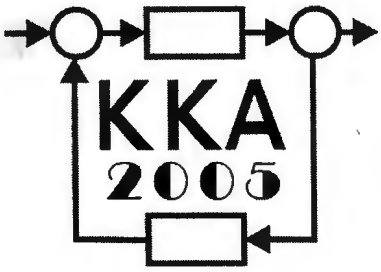
Tom II



Redaktorzy:
Zdzisław Bubnicki
Roman Kulikowski
Janusz Kacprzyk

XV Krajowa Konferencja Automatyki

Tom II



Redaktorzy:
Zdzisław BUBNICKI
Roman KULIKOWSKI
Janusz KACPRZYK

ORGANIZATOR
Komitet Automatyki i Robotyki Polskiej Akademii Nauk
Instytut Badań Systemowych Polskiej Akademii Nauk
WSPÓŁORGANIZATORZY
Politechnika Warszawska
Przemysłowy Instytut Automatyki i Pomiarów
Polskie Stowarzyszenie Pomiarów, Automatyki i Robotyki

ORGANIZATOR

Komitet Automatyki i Robotyki Polskiej Akademii Nauk
Instytut Badań Systemowych Polskiej Akademii Nauk

WSPÓŁORGANIZATORZY

Politechnika Warszawska
Przemysłowy Instytut Automatyki i Pomiarów
Polskie Stowarzyszenie Pomiarów, Automatyki i Robotyki

KOMITET PROGRAMOWY

Przewodniczący	Zdzisław BUBNICKI
Zastępca Przewodniczącego	Roman KULIKOWSKI

CZŁONKOWIE

Stanisław BAŃKA	Michał BIAŁKO
Mikołaj BUSŁOWICZ	Władysław FINDEISEN
Ryszard GESSING	Henryk GÓRECKI
Jakub GUTENBAUM	Jerzy JÓZEFCZYK
Stanisław KACZANOWSKI	Tadeusz KACZOREK
Janusz KACPRZYK	Jerzy KLAMKA
Józef KORBICZ	Zbigniew KOWALSKI
Krzysztof KOZŁOWSKI	Juliusz L. KULIKOWSKI
Krzysztof KUŹMIŃSKI	Kazimierz MALANOWSKI
Krzysztof MALINOWSKI	Wojciech MITKOWSKI
Antoni NIEDERLIŃSKI	Władysław PEŁCZEWSKI
Tadeusz PUCHAŁKA	Leszek RUTKOWSKI
Stanisław SKOCZOWSKI	Roman SŁOWIŃSKI
Jerzy ŚWIĄTEK	Andrzej ŚWIERNIAK
Ryszard TADEUSIEWICZ	Piotr TATJEWSKI
Krzysztof TCHOŃ	Leszek TRYBUS
Jan WĘGLARZ	Andrzej P. WIERZBICKI

KOMITET ORGANIZACYJNY

Przewodniczący	Roman KULIKOWSKI
Zastępcy Przewodniczącego	Janusz KACPRZYK
	Stanisław KACZANOWSKI
	Tadeusz KACZOREK
	Krzysztof MALINOWSKI
Członkowie	Roman OSTROWSKI
	Tadeusz PUCHAŁKA
	Dariusz WAGNER
Sekretarze naukowi	Jan STUDZIŃSKI
	Jan W. OWSIŃSKI

ISBN 83-89475-01-4

Druk: ARGRAF, Warszawa

METODY STOCHASTYCZNE – PROBLEMY NIEDETERMINISTYCZNE

ROZPROSZONA SYMULACJA STOCHASTYCZNYCH SIECI PETRIEGO

Ryszard KONIEWSKI

Politechnika Warszawska, Wydział Elektryczny
Instytut Sterowania i Elektorniki Przemysłowej
ul. Koszykowa 75, 00-662 Warszawa, e-mail:rkoniews@ee.pw.edu.pl

Streszczenie: Sieci Petriego często używane są do symulacji systemów różnego typu. Najprostszą metodą przyspieszenia tego procesu jest jego rozproszenie, ale stworzenie odpowiedniego algorytmu nie jest proste.

Największym problemem jest synchronizacja wszystkich wątków w optymalny sposób. Przedstawiony artykuł prezentuje algorytm, który nie wymaga globalnej synchronizacji, podczas całego procesu symulacyjnego.

Słowa kluczowe: Sieci Petriego, symalacja, obliczenia rozproszone, symulacja rozproszona.

1. WSTĘP

W ciąg ostatnich kilku lat wykorzystanie symulatorów w wielu zagadnieniach naukowych, technicznych, logistycznych jak i wielu innych nabrało szczególnego znaczenia. Większość badań popiera się dodatkowo wynikami symulacyjnymi. Ogranicza to w sposób znaczący koszt prowadzenia badań i analiz.

Wydatki ponoszone na tworzenie symulatorów, jak i czas budowania modeli symulacyjnych są nadal bardzo wysokie, a niektóre symulacje dodatkowo wykonują się w bardzo długim czasie. Za pomocą sieci Petriego można w sposób stosunkowo łatwy i szybki budować symulatory. Szczególnego zaś znaczenia, w procesie budowy symulatorów, nabrały Kolorowe Sieci Petriego [2][7].

Definicja 1 (Kolorowa Sieć Petriego) *Zbiór sześciu elementów* $GN = (P, T, F, C, W, M_0)$ *nazywamy siecią kolorową Petriego, wtedy i tylko wtedy gdy:*

1. P *- zbiór miejsc,*
2. T *- zbiór tanzycji,*
3. P *i* T *- zbiory rozdzielne i skończone,*
4. $F \subseteq (P \times T) \cup (T \times P)$ *jest binarną relacją wewnątrz zbior* N.
5. C *- jest odwzorowaniem mapowania każdego* $x \in P \cup T$ *na niepusty zbiór kolorów* $C(x)$,
6. $W : F \rightarrow N$ *- wartości wag poszczególnych łuków,*
7. M_0 *- początkowy stan sieci.*

Jednym z podstawowych problemów przeprowadzania symulacji jest czas jej trwania. Dla jego ograniczenia wykorzystuje się mechanizmy obliczeń rozproszonych. W niniejszym dokumencie przedstawiony zostanie algorytm przeprowadzania obliczeń rozproszonych w sieciach Petriego. W ciągu ostatnich lat powstało kilka prac na temat tego zagadnienia [4]. Prezentują one metody synchronizacji czasu pomiędzy częściami sieci, jak również mechanizmy predykcji kolejności wykonania zdarzeń, oraz ewentualnego wycofania części kroków procesu symulacyjnego. Przedstawionych zostanie również kilka modyfikacji sieci, które mają na celu jej uproszczenie i dostosowanie dla potrzeb definiowania symulatorów systemów masowej obsługi i nie tylko. Przedstawiona zostanie również metoda rozproszenia obliczeń symulacyjnych dla zaprezentowanego modelu.

2. CZASOWE SIECI PETRIEGO - WPROWADZENIE

Dla stworzenia sieci czasowej logiczne stało się wprowadzenie takich rozwiązań, które nie powodowałyby znaczących zmian w strukturze samej sieci. Dlatego też zdecydowano się na dołaczenie czasu do jednego z jej podstawowych elementów. Powstały zatem trzy podstawowe rozwiązania: czas powiązany z markerami, czas powiązany ze zdarzeniami, czas powiązany z łukami. Poniżej przedstawiono definicję sieci czasowej, gdzie czas przypisany jest do poszczególnych tranzycji.

Definicja 2 *Czasową Siecią Petriego nazywamy parę* (GN, Λ), *gdzie* $GN = (P, T, F, C, W, M_0)$ *jest Kolorową Siecią Petriego, a* Λ *jest odwzorowaniem, przypisującym dodatnią liczbę rzeczywistą do każdej tranzycji* $t \in T_N$

Dla większej atrakcyjności sieci nie zdefiniowano konkretnego podejścia czasowego. Zatem mamy tu do czynienia z pewnym stopniem swobody, co do rozpatrywania odcinka czasowego (np.: stały, wcześniej zdefiniowany, funkcyjny...).

Najbardziej ogólnym i jednocześnie najbardziej złożonym przypadkiem jest definowanie czasów przez

zmienne losowe. W takiej sytuacji mówimy, że mamy do czynienia ze Stochastyczną Siecią Petriego (SPN - ang. Stochastic Petri Nets). Analiza takich sieci jest na ogół bardzo skomplikowana, a proste rozwiązania istnieją tylko przy bardzo dużych obostrzeniach dotyczących typów poszczególnych rozkładów tych zmiennych. Z powyższych powodów dla celów rozwiązania zagadnień definiowanych z pomocą sieci, korzysta się najczęściej z metod symulacyjnych. Są one najprostsze w realizacji i jednocześnie najtańsze.

3. SIECI HIERARCHICZNE

Modele tworzone z wykorzystaniem sieci Petriego bardzo często są bardzo rozbudowane. Dlatego też powstało kilka metod, które mają na celu ich podział na mniejsze części. Podział ten łączony jest w jedną strukturę hierarchczną.

Całą strukturę można podzielić na poziomy. Wyższy poziom sieci nazywany jest wtedy nadsiecią, a niższy podsiecią. Każda nadsieć posiada miejsca określane jako gniazda, a każda podsieć posiada miejsca określane jako porty. Każde gniazdo musi być przypisane do portu, przy czym muszą one mieć zdefiniowany ten sam zbiór kolorów. Nie wszystkie porty muszą mieć przypisane gniazda.

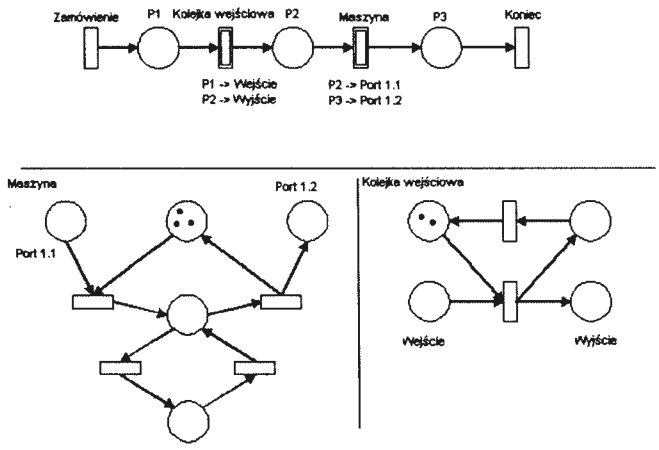
3.1. Zastąpienie tranzycji

W przypadku zastąpienie tranzycji pojedyncze zdarzenie zastępowane jest podsiecią. Oznacza to, że gniazdami i portami mogą być jedynie stany. Składnia zastępowania zdarzenia przedstawia się w sposób następujący. Na początku usuwane jest nadrzędne zdarzenie, wraz z łukami. Wszystkie dołączone do niego stany są gniazdami. Do sieci dołączana jest kopia podsieci, a następnie każde gniazdo jest scalane z przypisanym portem podsieci. W przypadku gdy kilka gniazd przypisanych jest do jednego portu, wtedy następuje ich „sklejenie".

3.2. Zastąpienie stanów

Zastępowanie stanów przebiega w sposób podobny jak w przypadku zastępowania zdarzeń. W tym przypadku gniazdami i portami są zdarzenia. Dopuszcza się aby kilka gniazd było przypisanych do jednego portu. Proces zastępowania stanu przez podsieć przedstawia się następująco. Na początku usuwany jest zastępowany stan, wraz z przyległymi łukami. W jego miejsce wstawiana jest kopia podsieci. W przypadku gdy do jednego port przyłączonych jest kilka gniazd, wtedy port ten jest również powielany. Następnie wszystkie nieprzypisane porty są usuwane. Na zakończenie wszystkie gniazda scalane są z odpowiednimi kopiami portów.

Istnieje możliwość złączenia przedstawionych powyżej metod w jednej sieci. Istnieje jednak jedno poważne ograniczenie. Po operacji zastąpienia tranzycji i miejsc, dwie podsieci nie mogą być bezpośrednio połączone za pomocą łuku. Wymagałoby to łączenia ze



Rys. 1. Przykładowa realizacja hierarchicznej sieci Petriego - zastąpienie tranzycji.

sobą poszczególnych portów podsieci, z pominięciem gniazd nadsieci.

4. MODEL SYMULACYJNY

Istotnym elementem niniejszej pracy jest określenie modelu sieci Petriego za pomocą którego będzie można zbudować symulator. Musi być on na tyle elastyczny, aby można było z jego pomocą budować symulatory o bardzo dużym stopniu złożoności, jak i bardzo proste. Ma ona także umożliwiać tworzenie symulatorów jak największego spektrum systemów. Dodatkowo tworzony model powinien zawierać mechanizmy, które byłyby łatwo implementowane w architekturze rozproszonej.

4.1. Wybór tranzycji

W trakcie działania sieci dochodzi do uruchamiania kolejnych tranzycji, na skutek czego dochodzi do zmian w znakowniu sieci. Urochomiona może zostać tylko tranzycja aktywna, przy danym znakowaniu. Aktywnośc takiej trazycji określa się następująco:

Definicja 3 *Tranzycja $t \in T$ jest aktywna przy znakowaniu M, wtedy i tylko wtedy gdy $\forall s \in \bullet t$:* $M(p) \geq W(p,t)$

Jeżeli tranzycja t jest aktywna przy znakowaniu M i zostaje uruchomiona, wtedy tworzone jest znakownie M' w sposób następujący: $M'(p) =$

$$\begin{cases} M(p) - W_N(p,t) & dla\ p \in \bullet t \setminus t\bullet \\ M(p) + W_N(t,p) & dla\ p \in t\bullet \setminus \bullet t \\ M(p) - W_N(p,t) + W_N(t,p) & dla\ p \in \bullet t \cap t\bullet \\ M(p) & \end{cases}$$

gdzie:

$\bullet t$ *- zbiór miejsc sieci przed uruchomieniem tranzycji t,*

$t\bullet$ *- zbiór miejsc sieci po uruchomieniem tranzycji t,*

W przypadku sieci czasowych do powyższej definicji dodawany jest warunek opisujący wybór tranzycji najwcześniejszej, pod względem czasu symulowanego.

W podstawowych sieciach Petriego, w przypadku gdy można *uruchomić* kilka zdarzeń, dokonuje się wyboru w sposób losowy. Wadą tego rozwiązania jest to, że wszystkie zdarzenia traktowane są w sposób jednakowy, to znaczy mogą zaistnieć z jednakowym prawdopodobieństwem.

Jednym z możliwych rozwiązań jest wprowadzenie priorytetów. W takim przypadku, gdy dochodzi do niejednoznaczności w wyborze zdarzeń, wybiera się tranzycję o najwyższym priorytecie. Jednakże takie rozwiązanie w trakcie symulacji może okazać się bardzo szkodliwe, ponieważ może spowodować izolacje całych fragmentów sieci.

Aby najlepiej oddać prawdopodobieństwo wystąpienia zdarzeń jednoczesnych można wprowadzić wagi zdarzeń. Na ich podstawie budowane jest tzw. koło rulety. Na początku sumuje się wszystkie wagi zdarzeń równoczesnych, zapamiętując kolejność. Następnie losowana jest liczba z przedziału od zera do wielkości sumy. Na zakończenie wylosowana liczba porównywana jest kolejnymi wagami zdarzeń. W momencie, w którym suma cząstkowa wag zdarzeń przekroczy wartość losową, wtedy wybrane zostaje to zdarzenie, którego waga zdecydowała. Podobny mechanizm definiuje się dla algorytmów genetycznych.

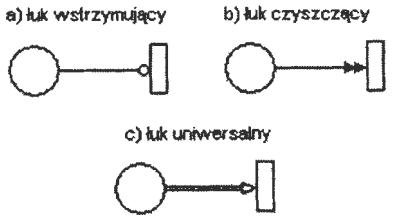
Przedstawiony algorytm oparty jest na generatorze liczb losowych o rozkładzie równomiernym. Można też stosować mechanizmy oparte na innych typach rozkładu, jak również na logice rozmytej.

4.2. Ruch w sieci i jego generacja

W przypadku wykorzystywania sieci Petriego do symulacji procesów logistycznych, jak i systemów kolejkowych istotna jest generacja ruchu w sieci. Można otrzymać ją w sposób bardzo prosty, przez wykorzystanie właściwości sieci stochastycznej (SPN). W tym celu tworzy się tranzycje, które nie posiadają stanów wejściowych. Ustawiając w odpowiedni sposób parametry tranzycji tworzony jest odpowiedni generator ruchu. Istotne jest to, aby możliwości generowania liczb losowych nie były ograniczone do tylko jednego rozkładu (na przykład wykładniczego). Zastosowanie SPN nie ogranicza się tylko do generacji ruchu, ale również stosuje się ją do opisu aparatów obsługi.

Dodatkowym elementem w procesach symulacyjnych, musi być możliwość tworzenia losowej ilości markerów. Jest to bardzo przydatne w przypadku symulatorów systemów masowej obsługi. Dla przykładu w przypadku symulacji portu morskiego [10], do terminala przybywają statki tego samego typu, ale z różną ilością towarów. W trakcie definiowania modelu symulacyjnego jesteśmy w stanie podać jedynie przedział i ewentualnie rozkład ilości tego towaru. Dlatego w przypadku sieci Petriego należy dodać możliwość definiowania wag łuków, przez rozkłady zmiennych losowych. Oczywiście wielkości tych wag powinny być wielkościami ze zbioru liczb naturalnych.

W celu uzyskania większej przejrzystości sieci Petriego wprowadzono pewne modyfikacje, definiowane dla różnych elementów sieci. Jedną z nich są łuki wstrzymujące. Są to łuki wychodzące ze stanów. Na wykresach zaznaczane sa jako łuki zakończone kółkiem. Używane są podczas określania możlwości *odpalenia* zdarzenia. Jeśli w stanie z którego wychodzi znajduje się choć jeden marker, wtedy wskazywne zdarzenie nie może zaistnieć, niezależnie od innych warunków panujących w sieci.



Rys. 2. Dodatkowe typy łuków.

Drugi typ łuków nazywany jest czyszczącym. Może on łączyć elemty w taki sam sposób jak łuk wstrzymujący, a oznaczany jest przez strzałkę z podwójnym grotem. Łuk ten jednak nie wpływa na dostępność zdarzenia, natomiast powoduje, że po wystąpieniu zdarzenia na które wskazuje z wyjściowego stanu usuwane są wszystkie markery, niezależnie od ich ilości.

W przypadku łuków możliwe by było wprowadzenie jeszcze jednej modyfikacji. W niektórych sytuacjach, dla sieci kolorowych, nie jest istotny kolor znacznika. Jego rozróżnienie może mieć natomiast znaczenie w dalszym procesie przetwarzania. Dlatego też przydatne mogą okazać się łuki, które reagują tylko na pojawienie się znacznika, oraz które przekazują go dalej. Łuki te mogą być nazywane łukami uniwersalnymi i mogą mieć definiowaną tylko jedną wagę, która będzie odnosić się do wszystkich kolorów. Powyższa modyfikacja ma na celu uzyskanie większej przejrzystości modelu.

Podczas modelowania niektórych systemów można wyodrębnić zdarzenia niepewne, które nie muszą się pojawić mimo, że wszystkie warunki do ich wykonania są spełnione. Aby zdefiniować tego typu zdarzenie można zdefiniować jego poziom ufności (liczba rzeczywista). Gdy zdarzenie jest możliwe do *odpalenia*, wtedy losowana jest liczba. Gdy przekracza poziom ufności, wtedy zdarzenie nie może zostać *odpalone*. Zdarzenie tego typu powinno być zdarzeniem czasowym. Gdyby nie było, wtedy kolejne uaktywnienie tego zdarzenia może mieć miejsce w przypadku zmiany w stanach wejściowych, lub po zmianie aktywności (zdarzenie ponownie aktywowane, gdy znakowanie sprawia, że przestaje być możliwe do wykonania).

4.2.1 Model - rozszerzona sieć czasowa Petriego

Jedną z podstawowych własności sieci Petriego, jest obrazowanie powiązań i zależności lokalnych. W przypadku sieci czasowych własność ta nie jest już tak widoczna, a co więcej jej znaczenie w procesie symulacji rozproszonej jest trudna do wykorzystania.

Aby wykorzystać tę cechę należy rozpatrzyć możliwość rozproszenia informacji czasowej. Każda z tranzycji sieci, powinna posiadać swój własny znacznik czasowy. Ma on określać moment w jakim znajduje się ten

element. Dla miejsc sieci definiowane są dwa znaczniki: czas wejściowy (powiązany z tranzycjami wchodzącymi do miejsca), oraz czas wyjściowy (powiązany z tranzycjami wychodzącymi z danego miejsca). Czas tego elementu określa czas wyjściowy. Przy takim założeniu ogólny czas przeprowadzonej symulacji równy jest minimum ze wszystkich znaczników czasowych węzłów sieci. Informacje na temat czasu są również związane z każdym z markerów. W tak skonstruowanej sieci każdy marker jest przez cały czas przypisany do miejsca sieci $p \in P_N$. Definicja takiej sieci będzie sie zatem przedstawiać w sposób następujący:

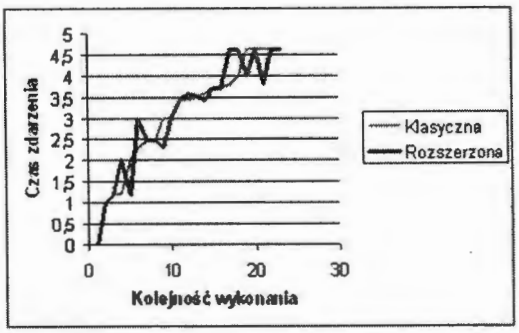
Definicja 4 *Czasową Siecią Petriego nazywamy trókę* $(GN, \Lambda, \lambda_1, \lambda_2)$, *gdzie* $GN = (P, T, F, C, W, M_0)$ *jest Kolorową Siecią Petriego,* Λ *jest odwzorowaniem, przypisującym dodatnią liczbę rzeczywistą do każdej tranzycji* $t \in T_N$, λ_1 *jest odwzorowaniem przypisującym liczbę rzeczywistą do: każdej tranzycji* $t \in T_N$ *i markera* $m \in M$, *a* λ_2 *jest odwzorowaniem przypisującym parę liczb rzeczywistych do każdego stanu* $p \in P_N$.

Aby ustalić wartości poszczególnych znaczników należy stosować następujące zależności:

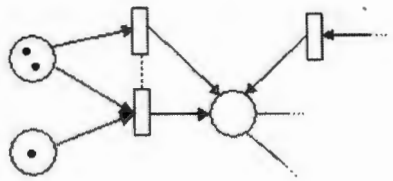
- dla stanów
  1. czas wejściowy stanu równy jest minimum czasowemu ze wszystkich zdarzeń wchodzących do niego,
  2. czas wyjściowy stanu równy jest minimum czasowemu ze wszystkich zdarzeń wychodzących z niego,
  3. jeśli brak jest zdarzeń wchodzących, wtedy wartość czasu wejściowego ustawiana jest na nieskończoność;
- dla zdarzeń
  1. czas zdarzenia równy jest minimum czasowemu ze wszystkich stanów wchodzących do niego (czas zdarzenia zmieniany jest tylko w przypadku gdy nie jest one aktywne),
  2. przy wywołaniu zdarzenia jego czas zwiększany jest o czas jego działania (jako czas wystąpienia zdarzenia rozumie się maksimum ze znaczników czasowych wszystkich markerów je aktywujących);
- dla markerów
  1. marker otrzymuje wartość znacznika równą czasowi zdarzenia które opuścił.

Dodatkowo zmianie ulega definicja aktywności tranzycji. Zdarzenie może zostać *odpalone* wtedy, gdy czasy wyjściowe wszystkich stanów dochodzących do niego są większe, bądź równe czasowi uruchamianego zdarzenia. Oczywiście nadal ma zastowanie definicja 3.

Ostatnie ograniczenie może powodować pewne problemy. Może się okazać, że mimo możliwości uruchomienia zdarzenia (odpowiednia ilość znaczników), nie będzie można go *odpalić*. Można łatwo pokazać, że



Rys. 3. Kolejność wykonanych tranzycji.



Rys. 4. Przykład zdarzeń konkurujących.

dojdzie w niektórych przypadkach do zakleszczenia. W takim przypadku należy wykonać operację odblokowującą. Polega ona na przejrzeniu wszystkich stanów w których znajdują się markery. Następnie wybieramy marker, którego znacznik czasowy jest najmniejszy i większy od znacznika czasu stanu w którym się on znajduje. Na koniec ustawia się wartość czasu wejściowego tego stanu na wielkość wybranego markera. Całą operację powtarzamy do momentu w którym pojawi się możliwość uruchomienia zdarzenia. Gdy nie jest to możliwe, to oznacza to, że sieć nie jest żywa (patrz [5], [6]), a proces symulacji dobiegł końca.

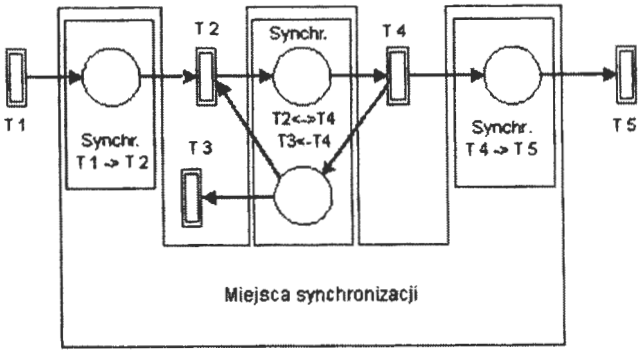
4.3. Kolejność uruchamianych tranzycji

W przypadku klasycznych sieci czasowych, o kolejności wykonania poszczególnych tranzycji decyduje ich uaktywnienie, oraz czas, w jakim są one wykonywane. Jest to wynikiem tego, że następuje synchronizacja poszczególnych kroków procesu z zegarem głównym.

W zaprezentowanym modelu informacja na temat czasu przechowywana jest lokalnie, dlatego też wybór kolejnych tranzycji nie musi się odbywać zgodnie z rzeczywistą kolejnością ich aktywacji w systemie (przykład na rys. 3).

4.4. Zdarzenia wykluczające

Przy tych modyfikacjach można zdefiniować zdarzenia wykluczające się - konkurencyjne. Są to takie zdarzenia, które mogą występują w sposób wykluczający się. Oznacza to, że wystąpienie jednego z tych zdarzeń, powoduje, że wszystkie tranzycje zgrupowane w tym zbiorze mają ustawiane zmienne czasowe na wartość zdarzenia *odpalonego*.



Rys. 5. Synchronizacja w rozszerzonej sieci czasowej Petriego.

5. ROZPROSZONA SYMULACJA STOCHASTYCZNYCH SIECI PETRIEGO

Aby rozproszyć wyżej opisaną sieć należy podzielić je na bloki w sposób podobny jak buduje się sieci hierarchiczne z zastąpieniem zdarzeń, lub z zastąpieniem tranzycji. Ilość bloków, jakie zostaną określone, definiuje ile wątków zostanie stworzonych. Wszystkie elementy poza blokami są wykorzystywane do synchronizacji poszczególnych wątków. Zalecane jest, aby w tej warstwie znalazły się wyłącznie stany, które są gniazdami prowadzącymi do poszczególnych bloków. Wewnątrz danego bloku nie dopuszcza się, aby porty były wejściowe i wyjściowe jednocześnie.

Wszystkie gniazda, oraz pozostałe elementy nadsieci nie posiadające zdefiniowanych struktur podrzędnych, służą do procesu synchronizacji. Wszystkie zmiany w gniazdach powinny być przekazywane do podłączonych portów. Przez zmianę należy rozumieć zarówno pojawienie się nowego markera, jak i zmianę parametru czasowego. Zmiany te można przekazywać na bieżąco, albo pakietami co pewien czas.

We wcześniejszej części tego artykułu zostało stwierdzone, że w opisywanej sieci istnieje ryzyko zakleszczenia. W przypadku obliczeń rozproszonych zjawisko to również istnieje. Jednakże w tym przypadku możemy mówić o dwóch typach zakleszczenia: lokalnym i globalnym.

Każde zakleszczenie będzie na początku traktowane jako zakleszczenie lokalne. Procedura postępowania przebiega w sposób opisany w punkcie 4.2.1. Jedyną modyfikacją jest to, że zmiana wartości znacznika czasowego w danym stanie nie może przekroczyć aktualnej wielkości elementów wejściowych podsieci tj. portów wejściowych. Gdy następuje taka sytuacja, to zachodzi podejrzenie zakleszczenia globalnego. W takim przypadku najmniejsza wartość zmiany (po zdjęciu ograniczeń portów) zapisywana jest w specjalnej strukturze nadrzędnej (na przykład w wątku sterującym). Z zakleszczeniem globalnym mamy do czynienia, jeśli wszystkie wątki zostały zablokowane. Gdy to następuje w wyżej opisanej strukturze odnajdywany jest element o najmniejszej wartości, i w tym miejscu wykonywane są zmiany. Gdy nie osiągają one żadnego skutku, wtedy pobierana jest kolejna wartość.

Jeśli po wykonaniu wszystkich możliwych operacji sieć nadal jest zablokowana, wtedy następuje przedwczesne zakończenie symulacji. Jednocześnie oznacza to, że sieć ta nie jest żywa.

5.1. Efektywne wykorzystanie przedstawionego algorytmu

Trudno jest określić możliwości prezentowanego algorytmu, ponieważ w dużej mierze jest on uzależniony od struktury sieci, oraz jej wielkości. W przypadku małych sieci nie jest opłacalne jego użycie, ze względu na zwiększoną ilość operacji związanych z kontrolą czasu lokalnego poszczególnych elementów sieci. Należy zwrócić uwagę na to, że w przypadku dużych sieci ilość operacji jest zwiększana, ale dzięki możliwości zrównoleglenia obliczeń, otrzymuje się polepszenie wydajności.

Przy procesie zrównoleglania obliczeń istotnym elementem jest odpowiedni podział sieci. Pomiędzy podsieciami musi znajdować się jak najmniejsza liczba połączeń. Zapewnia to niskie nakłady na synchronizację pomiędzy poszczególnymi wątkami. Jeżeli dodatkowo poszczególne podsieci są równomiernie podzielone, to istnieje możliwość pełnego wykorzystania mocy obliczeniowej poszczególnych wątków.

Jeśli chodzi o prędkość wykonania obliczeń jest ona ściśle związana z ilością zakleszczeń, jakie mogą pojawić się podczas obliczeń. To z kolei uzależnione jest od wyglądu sieci. W przypadku sieci jednokierunkowej (np.: taśmociąg) nie będzie dochodzić do zakleszczeń.

Dodatkową cechą prezentowanego algorytmu jest to, że nie istnieje konieczność przewidywania kolejności zdarzeń, oraz w przypadku niewłaściwej predykcji, cofania w czasie kilkudziesięciu operacji tranzycji, co aktualnie jest stosowane w trakcie rozproszonych symulacji sieci stochastycznych.

6. PODSUMOWANIE

Zaprezentowany powyżej model symulacyjny przeznaczony jest przede wszystkim do symulacji systemów zdarzeniowo dyskretnych i sekwencyjnych. Jednak ze względu na elastyczność wykorzystania sieci Petriego istnieje możliwość zaadaptowania go również do innych zastosowań. W ostatnim okresie powstało sporo prac dotyczących wykorzystania sieci Petriego w takich zagadnieniach jak systemy ekspertowe, zagadnienia workflow i inne (na przykład: [8]). Zaprezentowane rozwiązania rozszerzają funkcjonalność sieci Petriego, dlatego też, można uznać że znacznie poszerzą i ułatwią zastosowanie sieci w wielu dziedzinach.

Przedstawiony model czasowej sieci Petriego dostosowany jest głównie do implementacji komputerowej. Właściwe zbudowanie modelu sieci, oraz odpowiedni podział sieci może zapewnić bardzo dobre parametry pod względem synchronizacji i wymiany informacji pomiędzy podsieciami, jak i czasu wykonania obliczeń.

Opisany algorytm rozproszenia informacji czasowej

nie jest przeznaczony tylko i wyłącznie dla sieci kolorowych. Większą przydatność może okazać w innych typach sieci Petriego (zwłaszcza sieci Predykcyjno Tranzycyjnej, czy obiektowej).

DISTRIBUTED SIMULATION OF STOCHASTIC PETRI NETS

Abstract: Petri nets are often used to simulate different types of systems. The simplest way to accelerate this process is threading, but creation of proper algorithm is not so easy. The biggest problem is to synchronize all the threads in optimal way.

This article presents an algorithm which does not require global synchronization, during whole simulation process. This type of synchronization is used rarely in specific situation.

References

[1] Dr Peter Kemper „*Petri-Nets*" - http://www.iai.inf.tu-dresden.de/ms/

[2] K. Jensen „*Coloured Petri Nets*" Vol. I-III Springer ETACS 1992-96

[3] Christos G. Casandras „*Discrete Event Systems: Modeling and Performance Analysis*" Aksen Associates Incorporated Publishers 1993 ISBN: 0-256-11212-6

[4] Alois Ferscha „*Parallel and Distributed Simulation of Discrete Event Systems*" Handbook of Parallel and Distributed Computing, McGraw-Hill 1995

[5] Wolfgang Reisig „*Petri Nets An Introduction*" Springer-Verlag Berlin Heidelberg 1985

[6] P. H. Starke „*Sieci Petriego. Podstawy, zastosowania, teoria.*" PWN Warszawa 1987

[7] „*The Petri Nets World*" - http://www.daimi.au.dk/CPnets/

[8] W. M. P. van der Aalst, K. M. van Hee, G. J. Houben „*Modelling and analysing workflow using a Petri-net based approach*"

[9] „*Stochastic Petri Nets - An introduction in to the Theory*", Vieweg Verlag 2002, ISBN: 3-528-15535-3

[10] A. Dzieliński, K. Amborski, J. Sukiennik, P. Kowalczuk - „*Simulation-based Improvments of Logistic Processes of the Port of Gdansk*" w: „Applications of Simulation and IT Solutions in the Baltic Port Areas of the Associated Candidate Countries" JUMI Ltd. 2003, ISBN: 9984-30-057-9

Instytut Badań Systemowych
Polskiej Akademii Nauk

ISBN 83-89475-01-4