Qatar Univ. Sci. J. (1994), 14 (1): 16-22

THE MODELS OF A GENERAL COMPONENT OF A DISTRIBUTED COMPUTING SYSTEMS BASED ON STOCHASTIC PETRI NETS

By

SAMI SERHAN and ZIAD AL-QADI
College of Applied Engineering, University of Jordan
Amman

نظام حاسوب موزع يعتمد على شبكات بتري العشوائية (الاحتماليـــة)

ســـامي سرحـــان و زيـــاد القـــاضي

نطرح في هذا البحث نماذج لنظام حاسوب موزع تعتمد على شبكات بتري العشوائية (الاحتمالية) العامة . كما نطرح أيضا وسائل لتخفيض عدد الحالات ونوضح أن درجة التعقيد لنظام الحاسوب الموزع لا تؤثر على هيكلية النموذج (يتغير فقط عدد النقاط) .

Key Words: Distributed computing system, Stochastic petri nets, Petri model, Tokens transitions states.

ABSTRACT

In this paper we propose models of a distributed computing system (DCS) based on general stochastic Petri nets (GSPN). We also propose the means to decrease the number of states. We indicate that the complexity of DCS doesn't affect the models structure (a number of tokens is changed only). In this paper we also propose models for investigation of processes transfer, data interchange, and also the priority interchange.

INTRODUCTION

The distribution of tasks, jobs and other objects among the system resources is performed by control programs, which are constructed based on the planning and distribution of resources, algorithms of routing and organization of queues ... etc. Based on the analysis of data processing and transmission algorithms in correspondence with typical tasks and jobs and setting the purpose of increasing the effectiveness and saving the number of used equipment's we can theoretically motivate and develop an effective DCS structure. The most suitable and clear communication between resources, as well as between operators of control program activities is presented with the aid of graphs. This is because Graph theory represents a wide class of mathematical models, which can well describe the components relations of different processors. The wide spreading of Petri nets for computing system modeling is a result of the fact that, Petri nets are suitable for constructing the image of parallel and synchronized processes. With the aid of Petri nets the system behavior is described as a result of the correspondence between events and conditions.

Models based on Petri nets are used for investigation of existence of particular states in the system, e.g. conflicts arising from practical applications, many authors found the required extensions and/or modified basic definitions of Petri nets to obtain more convenient modeling instruments [6-15]. For example timing presentation of nets was included in E-net [1].

Merlin and Farber [2] proposed temporary Petri nets for analysis of data interchange protocols. With each transition the maximum and minimum time are related. Using Petri nets for modeling of algorithms and processes became wide spread since 1966, when the results of the C. petri dissertation were drawn to specify the theory of information systems. There are several approaches of practical employment of Petri nets for system design and analysis. In the first general approach Petri nets are considered as an auxiliary instrument of analysis. For some algorithms and structures the corresponding models are constructed in Petri nets form and all difficulties which appear during the analysis are interpreted for system improvement. After that, the analysis problem on Petri nets models is repeated and, in such way, defined configurations of systems and algorithms are reached, which satisfy the given requirements.

The basic definitions and terms are mentioned in the appendix.

2. Petri models for distributed computing system

For models construction based on GSPN we refer to the architecture of a classical computer system. This system consists of three processors PRI, PR2, PR3, which have their

own interfaces B1, B2, B3 and local memories LM1, LM2, LM3 (Fig. 1). The processors are combined with the aid of two system interfaces I1 and I2, which provide access to three modules of common memory CM1, CM2, CM3. In general, a set of N processors, each of them has its own memory N-LM, exchange messages through a set of N common memories, using connecting nets from B interfaces.

The analysis of computing system models can be achieved with different purposes, among these we indicate: investigation of parallel processes synchronization, definition of conflicts when using global system resources and ... etc.

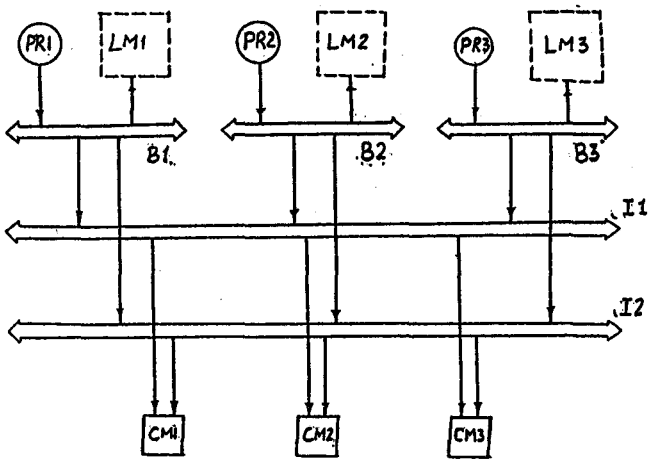

Figure 1: Computer System Architecture.

To transfer to Markov models system functioning, we allow that each processor refers to its own memory with rate λ, and after that the processor sends request to one of the system common memory modules. A request may not be immediately served, if the interface is busy or the corresponding memory is used by another processor or other processors. We will assume that the processor connection with common memory is achieved with rate μ. The control strategy of an interface leads to : if a processor, say A exists in a queue of a common memory requested by another processor, say B, then when the common memory terminates linking with processor B, the utilized interface does not turn off, but immediately would be allocated by processor A. To simplify the models we permit, that the controlling and releasing time of an interface is not significant compared with waiting time in a queue and processing time (if controlling and releasing time are taken into consideration, the vanishing states would be real states, and so the number of real states will increase and consequently the computation overhead will also increase). So that, we can construct a model based on GSPN, using temporary transition to represent the formation characteristic and requests servicing to resources, and instantaneous transitions to represent the controlling and releasing of interfaces. On figure 2, a DCS model is shown. This model consists of 5 processors, three modules of common memory, and two interfaces. In this set up, GSPN contains 21 places, 16 temporary transitions and 15 instantaneous transitions. The correspondence between places and transitions to states and events, which occur in the DCS, is presented in Table 1.

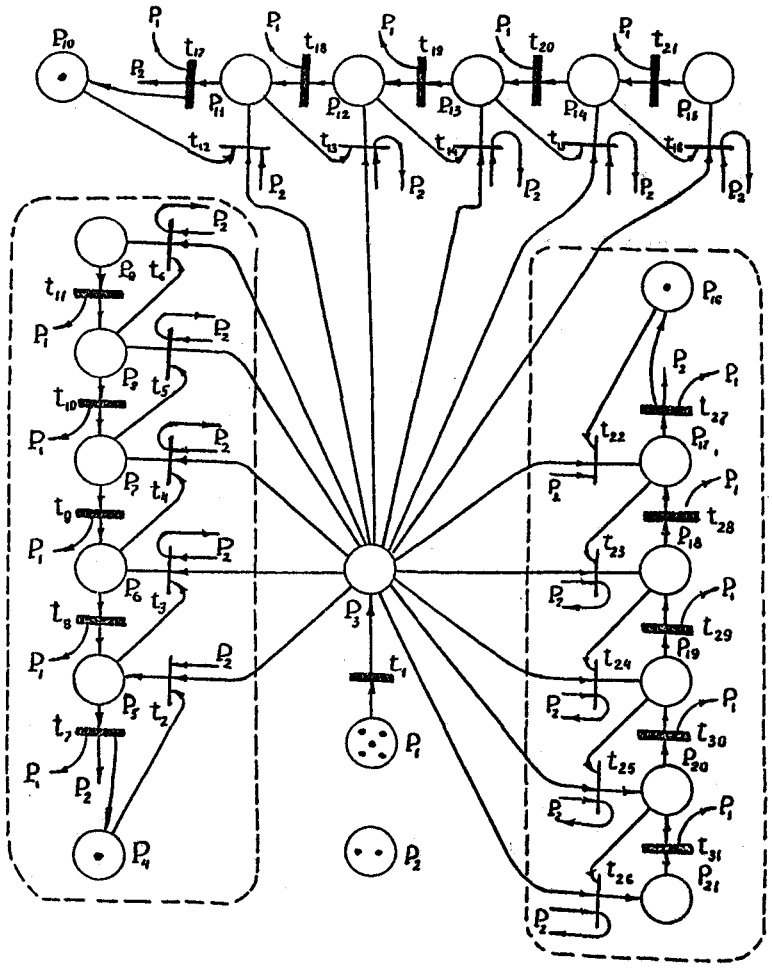

Figure 2: GSPN model complexity of which depends on the number of memory modules in the DCS.

Tokens in P_1 reflect readiness of the corresponding number of the processors to receive or send data, while in P_2 they reflect accessibility of the corresponding number of interfaces for data transfer. Tokens in P_4, P_{16}, P_{10} reflect the status of the corresponding number of common memory modules, free for receiving and sending messages. Each processor forms requests with rate λ, consequently transition t_1 fires with rate $m_1\lambda$. Tokens in P_3 means status of the corresponding number of processors, awaiting the release of interfaces to access the modules of common memory. If an interface is released, then a processor selects memory according to some distribution probability to one of the common memory modules. Proceeding from the symmetry of GSPN we can analyze its characteristics, paying attention only to the packed subnet (P_4, P_5, P_6, P_7, P_8, P_9) which specifies access to a common memory module. We consider a situation, when one of the interfaces is free, and a processor, presented by a token in P_3, refers to the first module of common memory.

Table 1
Description of states and events

Place	States description
P_1	Reflects the processor states
P_2	Reflects the interfaces states.
P_3	Reflects the processors states, waiting the releasing of interfaces to access a common memory module.
P_4, P_{10}, P_{10}	States of common memory modules
P_5, P_{11}, P_{17}	States of a queue to the common memory (No processor in the queue)
P_6 $P_{,12}$, P_{13}	States of a queue to the common memory (1 processor is standing in the queue)
P_7, P_{18}, P_{19}	States of a queue to the common memory (2 processors are standing in the queue)
P_8, P_{14}, P_{20}	States of a queue to the common memory (3 processors are standing in the queue)
P_9, P_{15}, P_{21}	States of a queue to the common memory (4 processors are standing in the queue)
Transition	**Event description**
t_1	End of a processor activity
t_2, t_{12}, t_{22}	A processor selected an interface and started access
t_3, t_{13}, t_{23}	A processor became the first in the queue to the common memory.
t_4, t_{14}, t_{24}	A processor became the second in the queue to the common memory
t_5, t_{15}, t_{25}	A processor became the third in the queue to the common memory
t_6, t_{16}, t_{26}	A processor became the fourth in the queue to the common memory
t_7, t_{17}, t_{27}	End of access to the common memory, an interface is released.
t_8, t_{18}, t_{28}	End of access to the common memory, a processor comes out from the queue

Table 1 Contd.

Transition	Event description
t_9, t_{19}, t_{20}	End of access to the common memory, a processor comes out from the queue
t_{10}, t_{20}, t_{30}	End of access to the common memory, a processor comes out from the queue
t_{11}, t_{21}, t_{31}	End of access to the common memory, a processor comes out from the queue

During that two situations are possible: the first is related with access to a released common memory module. The firing of t_2 shifts one token from P_2, P_3, P_4, replacing it in P_5. After that the given common memory module is busy, meanwhile one of the processors has an access to fit, using one of the interfaces, but in its queue there are no processors. Temporary transition t_2 fires instantly, imitating the interface being busy and receiving access to memory. But at the same time a temporary transition t_7 is actuated, which fires with an intensity μ (through some time), imitating the memory releasing process as a consequence of sending-receiving termination. The second situation is related with access to a busy common memory module. Four processors can be in a queue to the corresponding memory module in the formation process of inquiries to exchange. This is shown on figure 2 with the aid of tokens, passing correspondingly to P_6 - one processor in the queue, P_7-two processors in the queue, P_8-three processors in the queue and P_9-four processors in the queue. In the given subnet only one place can contain one token, obviously, that with intensity $m\lambda$ references to memory will appear, and with intensity μ these references will be satisfied. Firing of transitions t_3, t_4, t_5, t_6 provides the replacement of tokens to more "higher' places of the subnet. During that the earlier actuated transitions t_7, t_8, t_9, t_{10} are stopped with counting continuation more with more "upper" transition, for example, t_8 according to t_7, t_9 according to t_8 and so on. With timing flow temporary transition is fired and the corresponding processor returns to operate with its own memory, and the same interface permits another processor, standing in a queue, to access the common memory module. An interface is released after that when the last standing processor in queue terminates exchanging, i.e. after firing of t_7, when a token returns to P_4 again.

A reachability set of the considered model includes 64 real states and 22 vanishing states.

The considered model in the GSPN form can be extended in order to investigate a system with a big number of processors, common memory modules, and interfaces. Changing of the interfaces number requires only the modification of the initial mark, and the extension of a subnet P_4,-P_9, P_{10}-P_{15}, P_{16}-P_{21} by one place on each processor. Changing the common memory modules number requires including an additional subnet in a model. We consider a subnet modification, imitates as access to selected common memory module, which can replace each of the subnets in the model on figure 3. The features of a new net can be defined, bearing in mind the access to the first common memory module.

A reference to the first module of the common memory is imitated by firing of the instantaneous transition t_2, when token

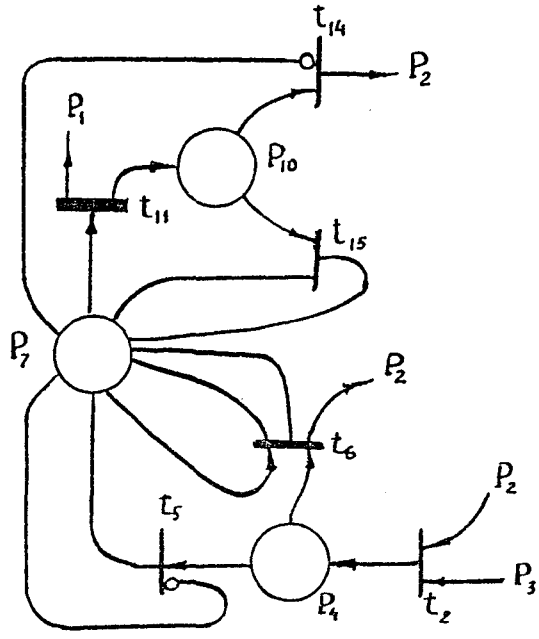

Figure 3: Proposed chain to improve the model in figure 2

from P_2 and P_9 are shifting to P_4, after that one of the instantaneous transition t_5 or t_6 can be fired depending on the situation, which is formed after the reference to the given common memory module. A transition t_4 is fired if in P_7 there are no tokens, i.e. $m_7 = 0$, which means that none of the processors has an access to common memory module, i.e. the corresponding memory module is free. So, a token from P_4 passes to P_7, In the second situation, if a token exists in P_7 ($m_7 \geq 1$), imitating a connection realization of some other processor to the given memory module by means of the earlier selected interface. The instantaneous transition, t_6, is fired with $m_7 \geq 1$ and a token from P_4 passes to P_7, changing the mark $m_7 = m_7+1$ and at the same time a token comes to P_2, imitating the releasing of an interface, obtained by the given processor, which has been referred to the memory module. In such a case the processor will wait in the queue of memory and interface releasing used at this moment by another processor in P_7. At the same time a temporary μ, imitating the satisfaction of processor inquires. During that the processors start again to operate with their own memories, and a token passes to P_1, and another to P_{10}, imitating the vanishing conditions for an interface releasing. An instantaneous transition is fired, replacing a token from P_7 and P_{10} once again to P_7, imitating that the interface is not released, but as a result of t_{11} and t_{15} firing the number of tokens in P_7 is decreased by 1, $m_7 = m_7-1$. The instantaneous transition t_{14} is fired, if replacing a token from P_7, marking of the place P_7 became equal to 0. The last means that the last inquiry to exchange with the given memory module is achieved and there are no other processor inquiries. By firing of t_{14} a token from P_{10} is moved to P_2, imitating the interface releasing. By changing the given model by the earlier used subnets a set of reachability of DCS new models will include 64 real states and 199 vanishing states. Using a new GSPN subnet it is not difficult to change the number of processors and interfaces in the model, which leads only to change the initial mark. At the same time changing the number of memory modules requires adding of similar subnets, considering that the model complexity will increase linearly with increasing the number of common memory modules. The disadvantages of the earlier considered model with relation to the changing of the net configuration and the number of used resources are obvious.

We consider a new convolution of GSPN model, oriented to DCS description with any number of common memory modules. The new GSPN model is represented by figure 4. It includes 9 places, 6 temporary and 5 instantaneous transitions. The description of the system resources states (places) and events, occurring in the system (transitions), is presented in Table 2.

Table 2
Description of states and events

Place	States description
P_1	Reflects the processor activity
P_2	Reflects the interface reachability
P_3	Reflects the states of processors, waiting the releasing of an interfaces for an access to the common memory
P_4	Reflects the states of the common memory modules
P_5	Reflects the fact, that one of the processor has an access to the common memory (the queue is empty)
P_6	Reflects the queue states to the common memory, the queue length is equal 1
P_7	Reflects the queue states to the common memory, the queue length is equal 2
P_8	Reflects the queue states to the common memory, the queue length is equal 3
P_9	Reflects the queue states to the common memory, the queue length is equal 4
Transition	**Events description**
t_1	End of processors activity
t_2	A processor selected an interface and started access to the common memory
t_3	A processor became the first in the queue to the common memory.
t_4	A processor became the second in the queue to the common memory
t_5	A processor became the third in the queue to the common memory
t_6	A processor became the fourth in the queue to the common memory
t_7	End of access, the common memory, and the an interface are released
t_8 - t_{11}	End of access, the processor comes out from the queue

The common memory modules which are not busy are defined by the corresponding number of tokens in P_4, the operating processors of the memory itself by tokens in P_1, and the interfaces, having a connection, by tokens in P_2. The temporary transition t_1 imitates the requests formation for an access to common memory modules. Requests (tokens) are

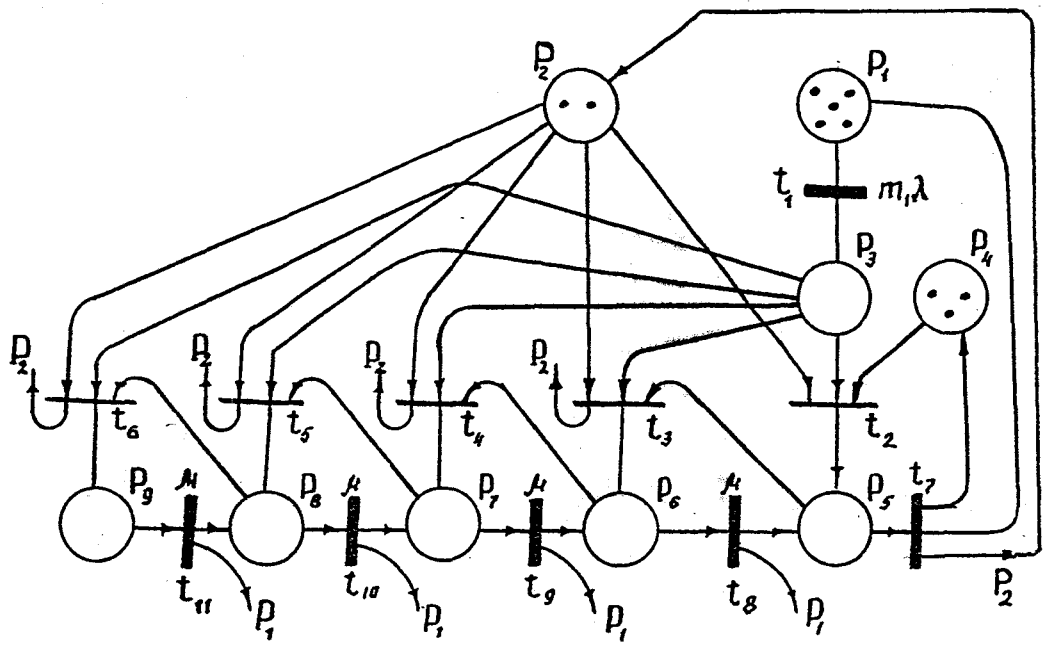

Figure 4: GSPN model complexity of which depends on the number of processors in the DCS.

fixed in P_9, forming the vanishing conditions according to the distribution keys, i.e. to the firing probabilities of the instantaneous transitions. The places P_5-P_9 can include tokens, which means that the transition t_2 is always fired. During that, tokens from P_2, P_3, P_4 are moved to P_5, imitating the busying of one interface and one common memory module for a requested realization of one processor. With the existence of tokens in P_5 two transitions, t_2 and t_2, can be fired with probability Pr (t_2), Pr (t3), where these probabilities are defined as : Pr (t_2) = $m_4/3$, Pr (t_3) = $m_5/3$ (3- the number of common memory modules). These are logical conditions, called the distribution keys, which confirm what have been said earlier. If $m_4 = 3$ corresponds to all the three free common memory modules, then the probability in Pr (t_2) = 3/3=1. If m_5 = 1, then the mark is m_4 = 2. So the distribution keys (transition conditions) are defined by: Pr (t_2) = 2/3, Pr (t_3) = 1/3. By forming the queues to each module of common memory, the marks P_5, P_6, P_7, ... are changed from 0 to 3. During that, the distribution keys will have the following relations : Pr (t_3) = $m_5/3$, Pr (t_4) = $m_6/3$, Pr (t_5) = $m_7/3$, Pr (t_6) = m8/3. Since the states of a queue to one common memory module are defined by locating only one token in P_5-P_9, then the queue state to the common memory is defined by three tokens, and consequently, the summation by $m_4 + m_5 + ... + m_9$ (number of common memory modules). In such a way, the function of all three subnets of the first module is reproduced. The temporary transitions t_7-t_{11} imitate performing the requests of the processors exchange with common memory modules with an intensity μ. The actuation and reactuation of these transitions are carried out by marks of the corresponding places m_5-m_9. Firing of t_7 returns one token to P_2, P_4 and P_1. In other situations a token is moved to the right in the subnet.

The investigation of such a model becomes simpler, since the set of reachability decreases, for example, the number of real states is decreased to 16, and the vanishing states to 8. The utilization of such a "convolution", GSPN model allows significantly the decreasing of the processes analysis time in DCS for various numbers of used resources. By changing the number of interfaces and common memory modules the net configuration does not change, but the initial mark is changed only. Increasing the number of processors requires, in addition to the initial mark changing, the subnet extension P_5-P_9, i.e. addition one processor leads to add one place in the subnet P_{10}. The considered models on figure 4 are suitable enough for modeling of a computing system with a big number of processors and common memory modules. Therefore to investigate the interface organization of a general DCS component, it is necessary to construct such a model in GSPN form, the complexity of which depends only on the number of interfaces. The constructed model, for this purpose, is represented on figure 5, it contains 5 places and 5 transitions:3 temporary and 2 instantaneous. The place P_1 reflects the processor state, exactly their requests to access a common memory module. Place P_2 reflects the interfaces state. The existence of free interfaces is represented by a token in P_2. The temporary transition t_1 with intensity $m_1\lambda$ forms inquiries to access. Place P_3 is used for the formation of the vanishing conditions of instantaneous transitions t_2 and t_3 and in P_3 a queue is formed to occupy the interface Place P_4 defines, that some processor has an access to the common memory through the indicated interface, considering that there are no inquiries to the given memory. Place P_5 defines the state of processors requests to the busy memory modules, i.e. requests, standing in a queue to the same or other memory module. Firing of the temporary transition t_5 means the termination of an access to memory, i.e. requests servicing. Firing of the transition t_4 means the termination of an access, i.e. releasing of the memory, to which there are no requests in a queue. The instantaneous transition t_2 simulates the interface occupation and receiving an access to memory. As a result of its firing a token is moved to P_4 from P_2 and P_3. The selection of the same memory module leads to the instantaneous t_3 firing. During that a token is fixed in P_5, imitating requests, standing in a queue to memory. The firing probability of these transitions,

i.e. distribution Keys, is defined as Pr (t_3) = 1/m where m is the number of available memory modules (1/3 for our situation), Pr (t_2) = 1-1/m (2/3 for our situation). when a token exists in P_3, three situations may appear:

1. All memory modules are free (m_4 =0, m_5 = 0, m_2 = 2, m_3 = 0), during that is appears that t_2 is allowed only. After its firing m_4 = 1;
2. One memory module is busy and one interface is also busy (m_4 = 1, m_3 = 1, m_2 = 1). During that both transitions t_2 and t_3 are allowed, t_3 is firing with probability 1/3 and fixing a token in P_5, i.e. m_5 = 1;
3. Two memory modules are busy (m_4 = 1, m_2 = 1). During that one of the transitions t_2 and t_3 is not allowed, since a system has only two interfaces.

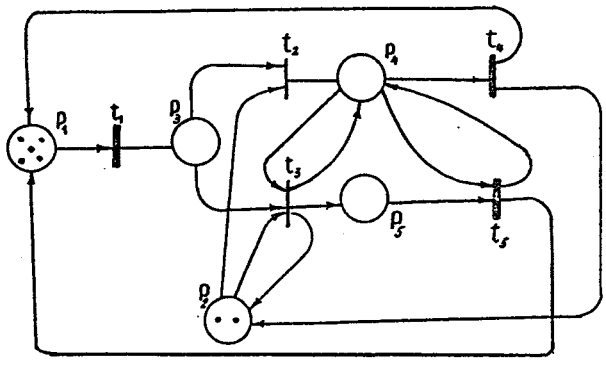

Fig. 5: GSPN model complexity of which depends on the number of interfaces in the DCS.

This model is suitable for system with any number of processors, common memory modules, but with a fixed number of interfaces. In our situation B=2. For a situation, when the number of interfaces is greater than 2 the subnet t_3 P_5 t_5 must be repeated B-1 times.

The summary of the proposed models characteristics is given in Table 3

Table 3
The characteristics of the proposed models

Characteristic Model	Advantages	Disadvantages	The suitable situation for using the model
1. Model which depends on the number of memory modules	Complexity doesn't depend neither on the number of processors nor on the number of interfaces	A great computation overhead (64 real states). The model needs adjustment if the number of memory modules is changed	In systems which may need to increase the number of processors and (or) the number of interfaces
2. Model which depends on the number of processors	Complexity doesn't depend neither on the number of memory modules nor on the number of interfaces. Less computation overhead than model 1 (16 real states).	The model needs adjustment if the number of processors is changed	In systems which may need to increase the number of memory modules and (or) the number of interfaces
3. Model which depends on the number of interfaces	Complexity doesn't depend neither on the number of processors nor on the number of memory modules. Less computation overhead than models 1 and 2 (8 real states).	The model needs adjustment if the number of interfaces is changed	In systems which may need to increase the number of processors and (or) the number of memory modules

Now it is necessary to draw the following important conclusion: the investigation purposes define the models character in GSPN form which can be different for the same system. Each model structure reflects special features of DCS working during the connections organization in the system enabling it to obtain in such a way the required quality characteristics of systems organization.

CONCLUSION

In this paper, three different models are constructed which represent the architecture of a DCS, so that any one of them can be selected to investigated and evaluate a DCS when there is a need to increase the number of memory modules, processors, or interfaces to obtain an efficient architecture of the DCS. For example the first model depends on the number of memory modules which are used in a way that the addition of new processors or interfaces does not affect the model. The complexity of investigation is different from one model to another, and the most complicated is the first model, which has 64 states, while the second model has 16 states, and the last one has only 8 states.

Appendix

Basic definitions and terms:

1. Petri nets:

It is a set, see [3], PN = (P, T, F, H, M_0)

where P is a finite (not null) set of places,

T is a finite (not null) set of transitions,

F is P x T ——> {0,1}, incident functions,

H is T x P ——> {0,1},

M_0 is the initial mark.

Petri nets can be represented by a graph with a set of nodes and transitions. In graph representation, places are denoted by circles, and transitions by entablatures. Curves (arcs) link places with transitions and transitions with places. A place can contain tokens marked by black dots inside the place. From a place into a transition leads a curve if and only if F (P, T) = 1, and from a transition into a place leads a curve if and only if H (P, T) = 1. Petri nets states are defined by the number of

tokens in each place and denotes using M vector in which the i-th component defines the number of tokens in the i-th place.

Petri nets start functioning with the initial mark (M_0). The marks replacement occurs as a result of firing one of the transitions. A transition fires (works) with mark M, if the following condition is satisfied:

$$\forall p \in P, M(p) - F(p, t) \geq 0$$

2. Stochastic Petri nets

Stochastic Petri nets, in which transitions fire during a random time interval, were introduced by Molloy and Shapiro [4,5]. In stochastic Petri nets transitions are related with the exponential distribution time of firing.

The formal definition of stochastic Petri nets:

$$SPN = (P, T, F, H, M_0, R)$$

$$R = (V_1, V_2, \ldots\ldots V_n)$$

R : is a set of firing rates of Petri nets transitions.

Stochastic Petri nets are isomorphic with Markov processes [4]. A mark in stochastic Petri nets corresponds to a state in a Markov process. Practically, κ-bounded stochastic Petri nets are isomorphic with a finite Markov process (graph).

In general stochastic Petri nets (GSPN), a set of transitions, is divided into 2 subsets, temporary and instantaneous transitions. Instantaneous transitions fire in zero time when they are permitted, but temporarily - after a random time. A set of marks as a result of instantaneous transitions firing is called a vanishing state, while for temporary transitions a real state.

The stay time of GSPN in each mark (state) is a variable with exponential random distribution, the average value of which,

$$t_i = \left| \sum_{i \in H} V_i \right|^{-1}$$

where H is a set of permitted transitions in a mark. The rate of a transition from mark M_i into M_j would be defined as:

$$R_{ij} = \sum_{i \in H_{ij}} V_k$$

where H_{ij} is a set of transitions, permitted with M_i, firing of which leads into mar M_j.

REFERENCES

[1] **Noe J.** and **G. Nutt, 1973.** Macro E-nets representation of parallel system, IEEE Transaction on computer, 8.

[2] **Merlin** and **D.J. Farber, 1970.** Recoverability of communication protocol-Implication of theoretical study. IEEE Transaction on communication. 9: 1036-1043.

[3] **Evans, D.J., 1982.** Parallel processing systems, Cambridge University Press, Melbourne Sydney.

[4] **Molly, M., 1982.** Performance analysis using stochastic Petri nets. IEEE Transaction on computers. C-31: 913-917.

[5] **Shapiro, S.D., 1974.** A stochastic Petri nets with applications to modeling occurrency times for concurrent task system-networks. 9: 375-379.

[6] **Lin, C** and **D.C. Marinescu, 1988.** "Stochastic High-level Petri Nets and Applications, "IEEE, Transaction on Computers. 37 (7): 815-825.

[7] **Peng D.** and **K.G. Shin, 1987.** "Modeling concurrent Task execution in a Distributed System for Real-Time Control, "IEEE, Transaction on Computers. C-36: 500-516.

[8] **Balbo, G., S. C. Bruell** and **S. Ghanta, 1988.** "Combining Queuing Networks and Generalized Stochastic Petri Nets for the solution of Complex Models of System Behavior," IEEE, Transaction on Computers. 37 (10): 1251-1269.

[9] **Chiola, G., 1985.** "A software Package for the Analysis of Generalized Stochastic Petri Net Models," in Proc. Int. Workshop Timed Petri Nets, Torino, Italy, July, 136-144.

[10] **Murata, T., 1979.** "Petri Nets: Properties. Analysis, and applications,' Proc. IEEE, 77 (4): 541-580.

[11] Proceedings of the International Workshop on timed Petri Nets, 1985. Torino, Italy, Los Alamos, CA: IEEE Computer Society.

[12] Proceedings of the International Workshop on Petri Nets and Performance Models, Madison, 1987. WI. Los Alamos, CA: IEEE Computer Society.

[13] Proceedings of the International Workshop on Petri Nets and Performance Models, Kyoto, 1989. Japan, Los Alamos, CA: IEEE Computer Society.

[14] **W. Brauer, W. Reisig** and **G. Rozenber, 1987.** Eds. Advances in Petri Nets 1986, Vols. 254 and 255 LNCS. New York: Springer-Verlag.

[15] **Ushio, T., 1990.** "Maximally Permissive Feedback and Modular Control Synthesis in Petri Nets with External Input Places", IEEE, Transaction on Automatic Control, 35 (7): 844-848.