Multilayered petri-nets for distributed decision making*

by A.Z. GHALWASH, P.A. LIGOMENIDES, and R.W. NEWCOMB
University of Maryland
College Park, Maryland

ABSTRACT

Decision making networks, employed for the control of complex cybernetic systems,[6,7,8,9,10] operate on the "Command, Status, and Message Layers" of concurrent decision making activity. Decision making "nodes" function as multitasking operators on all three layers, by executing command decomposition, status reporting, and message exchanging tasks for the concurrent implementation of various control policies. Aspects of real-time concurrency in hierarchical command decomposition over the command layer of the dm-net are, more particularly, analyzed in this paper, using concepts and tools of Petri-net theory.[2,3,4,5,11,12,14]

*This work was partially supported by National Science Foundation Grant Number IST 84-08063.

INTRODUCTION

The control of systems that are significantly more complex than any single decision maker can deal with alone, has motivated investigations of distributed decision making organizations.[13,15] The employment of multiple decision makers, coordinated in their local decision making efforts to regulate complex systems, underlines the approaches followed in these investigations.

In a more general sense, distributed decision making is used in the design of real-time management and control organizations for the regulation of "cybernetic" systems, such as various large scale business, military, and complex engineering systems. Cybernetic systems are characterized by strongly nonlinear interactions, and by regulatory processes designed to counter the homeostatic tendencies of the controlled system and the incoherent (noisy) or regulated forces from the environment, so as to derive the system away from certain intigenous behavioral trajectories and toward preferred "gainful" ones,[6,7,8] as illustrated in Figure 1.

Hierarchical decision making organizations for the control of complex cybernetic systems have been used by military, government, and business establishments for centuries. However, the concept of real-time, computer-based, hierarchical control of complex systems is a recent development.[4,5,6,7,13,15] The adaptive implementation of strategies and policies along a command decomposition hierarchy, in the face of continuous environmental and system perturbations, involves the concurrent and coordinated functioning of many, level-organized, decision making modules. Command decomposition along the behavior generating hierarchy is guided by the incitement of the "best" monotonic attainment of local goals, derived from the goals and constraints contained in the input command statements, and from the options of alternative actions available.

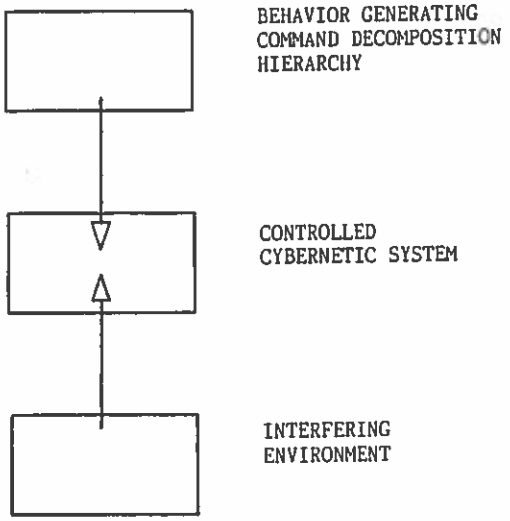

Figure 1—Designed, homeostatic, and incoherent forces on a controlled cybernetic system

In top-down hierarchical decision making networks (denoted "dm-nets"), high level commands are decomposed both spatially and temporally into related temporal sequences and patterns of subcommands, unfolding from top-down. This makes command decomposition a highly dynamic, behavior generating, activity. Decisions at one level of the hierarchy directly affect the decision making environment at other levels, both lower and higher, by exerting influence on the states, conditions, and alternatives available to other decision makers.

Because of the highly concurrent and dynamic character of hierarchical dm-nets, the use of concepts and tools of Petri-net theory[4,5,11,12,14] offer special advantages for performance-analysis and system-specifications. In this paper we will show the use of Petri-net concepts for the analysis of hierarchical multilayered dm-nets, in which decision making modules are allowed to operate concurrently on various policies and on the various layers of the coordinated decision making activity. In the second section we review concepts of dm nodes and nets, and in the third section we derive the equations for concurrent processing of commands along the command decomposition hierarchy, using Petri-net symbolism. We conclude with comments in the last section.

DM NODES AND NETS

Decision making networks with emerging collective goal-seeking capabilities operate like highly asynchronous, real-time, cellular automata. The decision making nodes (denoted "dm-nodes") receive, process, and distribute commands, status reports, and messages from/to other dm-nodes of the network in a highly asynchronous, real-time fashion. As such, a dm-node operates concurrently on three "layers" of activity, namely the command decomposition, the status reporting, and the message exchanging layers, as illustrated in Figure 2.[2] In this paper we will limit our discussion to the role of the dm-node within the command decomposition hierarchy. As illustrated in Figure 3, the dm-node, $P_{uv}(m, n)$ (i.e., the vth node at the uth level), has m input connections and n output connections. Input commands, δ^i, are received over the m input channels in a totally asynchronous manner, and, after some processing delay, output subcommands, δ^o, are transmitted to lower level dm-nodes in the hierarchy. In output transmissions, subcommands are distributed in accordance

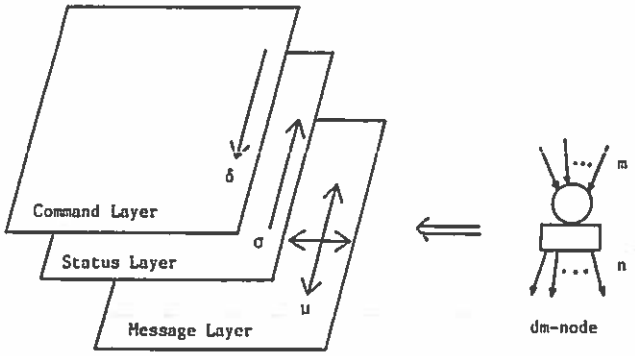

Figure 2—Layered concurrent operation of dm-nodes and nets

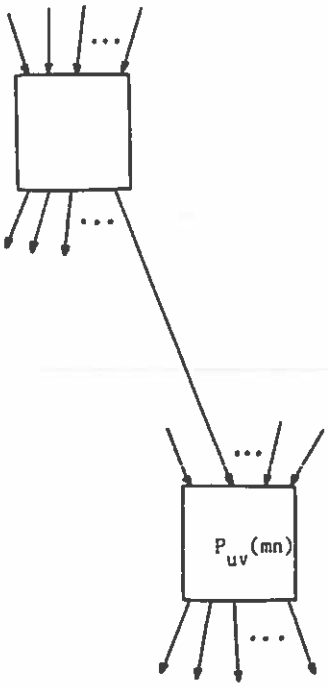

Figure 3—dm-node: $P_{uv}(m, n)$-model

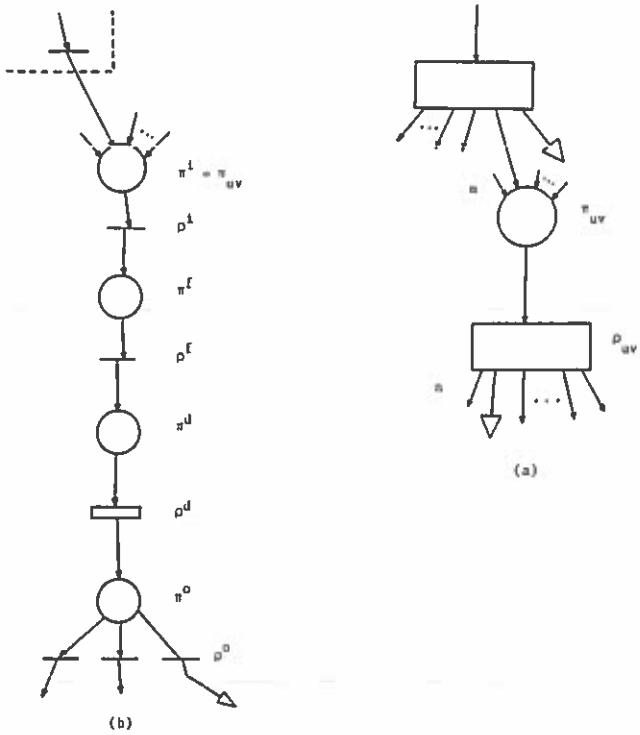

Figure 4—dm-node: (a) π_{uv}/ρ_{uv} model; (b) more detailed model.

with the spatio-temporal distribution programs, which are part of the output plan of the decision maker, functioning like microprograms of subcommand distribution.

Borrowing concepts and symbolism from Petri-net theory, we may represent the dm-node as the combination of an interface place, π_{uv}, and of a decision making transition, ρ_{uv}, where the ordered subscripts uv denote the "level, individual"-number designation of these components, as shown in Figure 4(a). In accordance with the symbolism of "binary" (also called "safe") Petri-nets,[1,2,3,4,5,11,12] places within the dm-nodes designate the presence of commands by a single "set-token." Each dm-transition has only one incident arc (place connection), and it is enabled to "fire" if a set-token is present in the incident place. Firing of a transition is also enabled by the satisfaction of a local condition.

On a more detailed level of description, the transition π_{uv} may be broken down to an input transition ρ^i, a command-fusion place and transition combination, $\pi^f \rho^f$, the actual command decomposition place and transition combination, $\pi^d \rho^d$, (further detailed in A.Z. Ghalwash's Ph.D. dissertation[2]), and an output place, π^o, connected to the output transitions, ρ^o, that distribute the output subcommands, δ^o, as it is illustrated in Figure 4(b).

Each dm-node functions within its own characteristic "decision making worlds" (denoted "dm/w"), each specified by a corresponding "domain of objectives" of the decision making activity, and each composed of domain-related attributes, objects, and events of the decision maker's concern. In response to different input command statements, specifying temporal goals and constraints, the decision maker of the dm-node (ρ^d) determines the corresponding "decision making subworlds" (denoted "dm/sw"), on which the current decision making attention is focused.[8,9,10]

The dm-nodes may operate concurrently on various temporal tasks (within corresponding dm/sw's), specified in the different, concurrently received, input command statements. Decision making tasks deal with the analysis, implementation, and monitoring of different policies specified over the various domains of objectives of the decision maker's concern, as, for example, a manager in a business organization may deal concurrently with various tasks as part of implementation of different policies of production, marketing or finance.

There are various characteristic time-delays in the operation of a dm-node. In order to analyze and determine various temporal aspects in the operation of a dm-net, processing and

propagation delays must be defined and determined. Most critical of such delays are those that must be determined in real-time and are dependent on conditions and data measured only dynamically.[2]

For purposes of demonstration, we derive now the average delay in a dm-node under the following simplifying assumptions: The decision maker, ρ^d, deals with a finite set of objects in his dm/w, each object, Q_i, taking only a finite number of discrete values, q_{ij}. Input command statements contain conditions (IF part) of the type "Q_i is q_{ij}," which are found to be satisfied in the dm/w with a probability (distribution) P_{ij}. We let P_i be the probability that the next input command, δ^i, will address the object Q_i, and $P_{j/i}$ be the probability that the value q_{ij} will be addressed given that Q_i is addressed. Also we let that $P_{i\&j}$ be the probability that both Q_i and q_{ij} are addressed in δ^i.

The average time between two successive input commands is T_0, while the time required to check an IF-condition about object Q_i is T_c^i, and the actual execution time for a command when the IF-condition is satisfied is T_e.

Then,

$$P_{i\&j} = P_{j/i} \cdot P_i$$

and the average firing time delay in the ρ^d-transition is easily derived to be,

$$T_{avg} = \sum_{i=1}^{n} T_a^i$$

$$T_a^i = P_i \cdot T_c^i + \sum_{j=1}^{m_i} P_{i\&j}\left[\left(\frac{1}{P_{ij}} - 1\right)T_0 + T_e\right]$$

where m_i is the number of the discrete values of Q_i, and n is the number of objects in the dm/w. Note that the total delay in the dm-node ρ_{uv} may be determined if the delays in the other transitions of the node (see Figure 4(b)) are estimated and added to the delay in ρ^d. This derivation of T_{avg} demonstrates that temporal aspects in the command decomposition hierarchy may be computed under various statistical assumptions, or by estimations of delays from data collected in real-time.

Within a hierarchical command decomposition organization, each dm-node is appropriately connected and is designated to operate within specified dm/w's, in accordance to the various assigned domains of objectives. As new policies are generated by global commands issued at higher levels of the command decomposition hierarchy, each defining its own global goals and constraints, related decision making activity is generated and ripples-down the fired dm-nodes of the hierarchy. We classify the commands reaching and leaving each multitasking dm-node by color-coding the different policies generated by the global commands. Different color-codes are used to identify the related decision making activity, which evolves over the three concurrent layers of command decomposition, of status reporting, and of message exchanging. We distinguish three types of global commands which regulate the generation, the maintenance, and the cancellation of the active color-coded policies over the dm-net, namely: "new policy-generating," "policy-modifying," and "policy-cancelling" types of global commands.[2]

In the following section we derive functional relationships about the operation of command decomposition hierarchy, using Petri-net concepts.

RELATIONSHIPS AND EXAMPLES OF PETRI-NET ANALYSIS

A hierarchical organization of N decision makers (dm-nodes) consists of transitions and interface places, as illustrated in Figure 5. The decision making transition ρ_{uv} is connected to other such transitions through interface places that hold set-tokens according to their markings. The transitions will "fire" (i.e., will perform decision making activity) if their incident place holds a token *and* a corresponding firing condition is satisfied. The places are represented by circles, the tokens by dots in the places, and the transitions by bars. Each level of the hierarchical organization contains interface places with single outputs incident to corresponding transitions at the same level. Incident upon each place are connections from transitions at higher levels and from sources external from the hierarchy. "External" inputs incident on the place π_{uv} are denoted as x_{uv}.

At the firing time λ, tokens are moved through the fired transitions from the corresponding incident interface places into the places on which the transition is incident, in accordance with an "activity vector" associated with each fired transition. Notice that, in general, the activity vector may be time

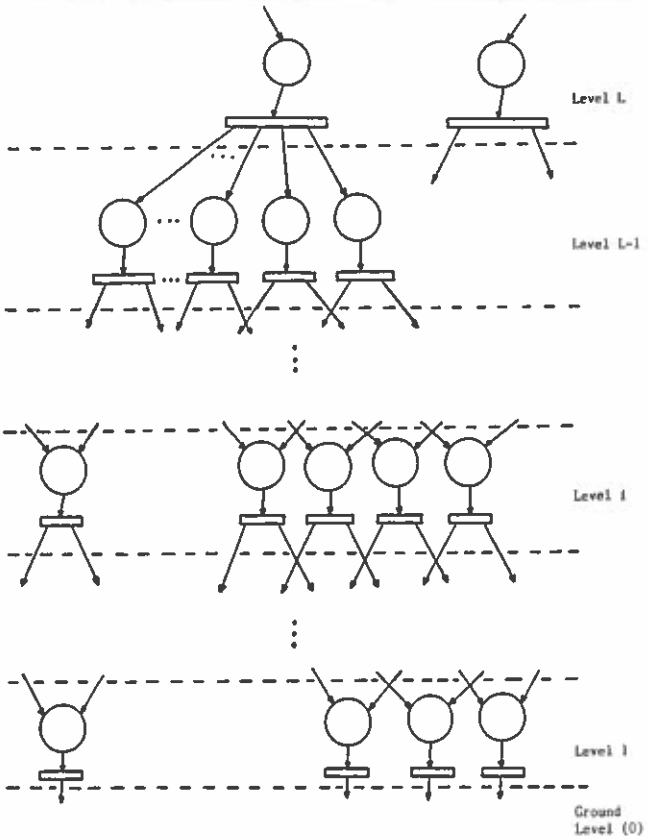

Figure 5—Command decomposition hierarchy

variable. The activity vector helps us to compute the next marking (i.e., the marking at firing time $\lambda + 1$).

If we assume that only one external incident arc x_{uv} may be accepted at most per interface place π_{uv}, and that only one external output (to an actuator) y_{uv} may be transmitted by a transition ρ_{uv}, then each dm-net is characterized by an N-bit external incidence input vector $X = [\ldots, x_{uv}, \ldots]$ and by a N-bit external output vector $Y = [\ldots, y_{uv}, \ldots]$.

Petri-net Description of Operations

For purposes of analysis we consider the hierarchical organization shown in Figure 5. For a Petri-net representation of N places and N transitions, we use the P-vector (N-bit binary) of markings at time λ, $M_0(\lambda)$ and the T-vector (N-bit binary) of firings at time λ, $F(\lambda)$. An entry equal to 1 in the firing vector F denotes that the corresponding transition will fire at time λ. In addition, the P-vector $X(\lambda)$ denotes the external inputs at λ to the interface places of the hierarchy, and the T-vector $Y(\lambda)$ denotes the outputs to external actuators at λ. An N-diagonal matrix is defined to designate whether the corresponding transitions are "ready" (i.e., conditioned) for firing. An entry equal to 1 in the Condition (diagonal) matrix, $S(\lambda)$ denotes that the corresponding transition is conditioned for firing at λ. The N-bit activity vector associated with the transition ρ_{uv} is denoted with $C_{uv}(\lambda)$.

Equations of Operations

A marking of the dm-net at time λ, $M(\lambda)$, designates the distribution of tokens over the interface places at λ. In order to take the input $X(\lambda)$ into account we use the dotted equality[1] to obtain the total binary marking vector at λ, as follows:

$$M(\lambda) \doteq M_0(\lambda) + X(\lambda)$$

For the dot equality we use normal integer arithmetic and we replace any resulting positive number with 1 and all other results by 0.

Since a transition is enabled to fire by both a token in its incident place *and* a satisfied condition, we may calculate the firing vector at $\lambda + 1$ as follows:

$$F(\lambda + 1) = S(\lambda) \cdot M(\lambda)$$

When a ρ_{uv}-transition fires, a token moves from the place incident on ρ_{uv} into those places on which the transition is incident and are designated in the associated activity vector at λ. We use again the dotted equality to compute the marking at time $\lambda + 1$.

$$M(\lambda + 1) \doteq M(\lambda) - F(\lambda + 1) + K(\lambda + 1)$$

where

$$K(\lambda + 1) \doteq \sum_{\rho_{uv} \in \phi} C_{uv}(\lambda)$$

where ϕ is the set of "firable" transitions (enabled by token and condition) at time λ.

We may calculate the Output (external actuation) vector at time $\lambda + 1$, $Y(\lambda + 1)$ as follows:

$$Y(\lambda + 1) = D \cdot F(\lambda + 1)$$

where D is a diagonal matrix, $D = \text{diag}(y_{uv}/\rho_{uv})$ and $y_{uv}/\rho_{uv} = 1$ if ρ_{uv} is connected to an external actuator (denoted with triangular arrows in Figures 4–6). Using the above derived relations we may compute the ripple-effects from an initial marking to the outputs to the external actuators.

If we assume that all transitions cause the same average delay T_{avg}, then the levels will fire synchronously and the total ripple delay from the time of global command input, λ_{input}, to the time of (say, ground level) actuation, λ_{out}, is given by

$$T_{ripple} = (\lambda_{out} - \lambda_{input}) \cdot T_{avg}$$

If the assumption of uniform delay, T_{avg}, is lifted, asynchronous firing will result, which will alter the timings over the various firing paths. An N-diagonal delay matrix, $V = \text{diag}(T_{uv}/\rho_{uv})$, will have to be defined, or determined in real time. Particularly interesting will be the formulation of a solution for the ripple-time, if the delays T_{uv} are time variable and situation (command)-dependent.[2]

Illustrative Example

Let us consider the dm-net shown in Figure 6, where

$$\Pi = [\pi_{31}/\pi_{21}\ \pi_{22}\ \pi_{23}/\pi_{11}\ \pi_{12}\ \pi_{13}\ \pi_{14}]^T$$
$$\tau = [\rho_{31}/\rho_{21}\ \rho_{22}\ \rho_{23}/\rho_{11}\ \rho_{12}\ \rho_{13}\ \rho_{14}]^T$$

We will assume that

$$M_0(0) = [0/000/0000]^T$$
$$X(0) = [1/000/0000]^T$$
$$S(\lambda) = \text{diag}(1/111/1111) \text{ for all } \lambda.$$
$$D(\lambda) = \text{diag}(0/000/1111) \text{ for all } \lambda.$$

and the activity vectors as

$$C_{31}(\lambda) = [0/111/0100]^T$$
$$C_{21}(\lambda) = [0/000/0100]^T$$
$$C_{22}(\lambda) = [0/000/0010]^T$$
$$C_{23}(\lambda) = [0/000/0011]^T \text{ for all } \lambda.$$

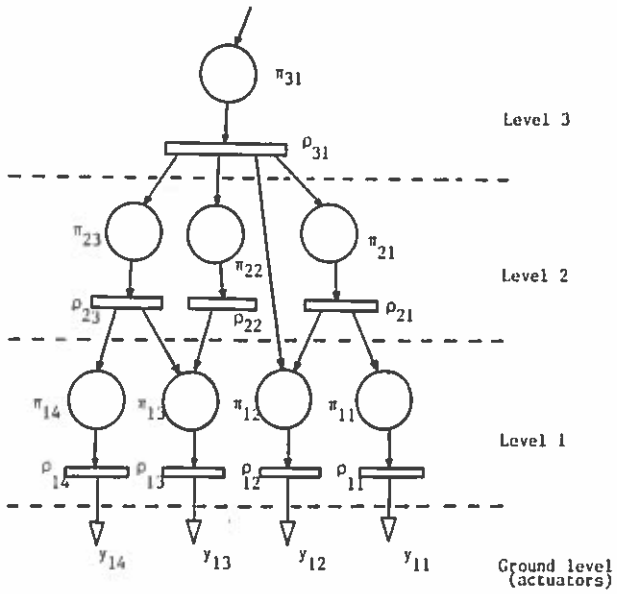

Figure 6—Example of hierarchical dm-net

At λ_{input} there is only one input to the place π_{31}, and at λ_{out} the hierarchy affects the controlled system only through the transitions $\rho_{11}, \rho_{12}, \rho_{13}, \rho_{14}$. If the Condition matrix enables all transitions to fire, then

$$M(0) \doteq M_0(0) + X(0) \doteq [1/000/0000]^T$$
$$F(1) = S(0) \cdot M(0) = \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = [1/000/0000]^T$$

At the next firing time, only ρ_{31} will fire.

$$K(1) \doteq C_{31}(0) = [0/111/0100]^T$$
$$M_0(1) \doteq M(0) - F(1) + K(1)$$
$$\doteq \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \doteq [0/111/0100]^T$$

We have

$Y(1) = D \cdot F(1) = [0/000/0000]^T$, no effect on external actuators.

$M(1) \doteq M_0(1) + X(1) \doteq [0/111/0100]^T$, assuming no external input at $\lambda = 1$

$F(2) = S(1) \cdot M(1) = [0/111/0100]^T$

$K(2) \doteq C_{21} + C_{22} + C_{23} \doteq [0/000/0121]^T \doteq [0/000/0111]^T$

$M_0(2) \doteq M(1) - F(2) + K(2) \doteq [0/000/0111]^T$

The $M_0(2)$ marking shows that at $\lambda = 2$ there are tokens at $\pi_{12}, \pi_{13}, \pi_{14}$. Also,

$Y(2) = D \cdot F(2) =$ $[0/000/0100]^T$, (i.e., there is an output y_{13}).

Assuming again that $X(2) = 0$, we have

$M(2) \doteq M_0(2) + X(2) \doteq [0/000/0111]^T$

$F(3) = S(2) \cdot M(2) = [0/000/0111]^T$

$K(3) \doteq [0/000/0000]^T$

$M_0(3) \doteq M(2) - F(3) + K(3) \doteq [0/000/0000]^T$

$Y(3) = D \cdot F(3) = [0/000/0111]^T$ (i.e., there are outputs at y_{13}, y_{12}, y_{11} to the corresponding external actuators).

We see that the sole initial marking at π_{31} has generated an external output at $\lambda = 2$, (y_{13}), and again at $\lambda = 3$, (y_{13}, y_{12}, y_{11}). Since the hierarchy received no further external inputs, there were no more actuations (no more tokens left in the M_0-marking).

Having assumed uniform delay in the dm-nodes, the external actuations were delayed by $2T_{avg}$ and $3T_{avg}$ respectively.

CONCLUSIONS

This paper has presented some aspects of applying Petri-net symbology and concepts to the analysis of layered dm-nets, and more particularly to policy implementation over command decomposition hierarchies. The work is currently being extended[2] on all three concurrent layers of dm-nets. Problems of reachability, timing, reconfigurability, and stability are especially being investigated, and results will be reported soon.

REFERENCES

1. Alayan, H. and R.W. Newcomb. "Binary Petri-Net Relationships," submitted for publication, 1987.
2. Ghalwash, A.Z. *Petri-Net Modeling of Decision Making Organizations*, PhD Dissertation (in preparation), EE Dept. University of Maryland, 1987.
3. Ghalwash, A.Z., P.A. Ligomenides, and R.W. Newcomb. "Modes and Job Performance Evaluation of Robot Petri-Nets." EE Dept., University of Maryland, Technical Report. (Submitted for conference publication.)
4. Ho, G.S. and C.V. Ramamorthy. "Performance Evaluation of Asynch. Concur. Systems Using Petri-Nets." *IEEE Trans. Soft. Eng.*, SE-6 (1980) 5, pp. 440–449.
5. Johnsonbaugh, R. and T. Murata. "Petri-Nets and Marked Graphs. Math. Models of Concur. Computation." *The Amer. Math. Monthly*, 89 (1977) 8, pp. 552–566.
6. Ligomenides, P.A. "An Engineering Cybernetic Model for Policy Analysis and Implementation." *Int'l J. of PAIS*, 6 (1982) 3, pp. 273–284.
7. Ligomenides, P.A. "Command Decomposition as a Decision Making Problem." in S.K. Chang (ed), *Management and Office Information Systems*, Plenum, 1983.
8. Ligomenides, P.A. "Dynamic Models for Information Quality Enhancement." *Proc. IEEE Workshop on Lang. for Autom.*, Chicago: November 7–9, 1983.
9. Ligomenides, P.A. "Specific. of an Experiential Data Base for Decision Making." *Proc. IEEE Conf. Trends and Applic.* Bethesda, MD, NBS, May 22–24, 1984.
10. Ligomenides, P.A. "The Experiential Knowledge Base as a Cognitive Prosthesis." in S.K. Chang, T. Ichikawa, P.A. Ligomenides (eds), *Visual Languages*, Plenum Press, 1986.
11. Murata, T. "Modeling and Analysis of Concur. Syst." in C.R. Vick and C.V. Ramamoorthy (eds), *Handbook of Software Engineering*, Van Nostrand Reinhold, 1984.
12. Peterson, J.L. "Petri-Nets." *Computer Surveys*, 9 (1977) 3, pp. 223–252.
13. Sandell, N.R., Jr., P. Varaiya, M. Athans, and M. Safonov. "Survey of Decentr. Control Meth. for Large Scale Sys." *IEEE Trans. Aut. Contr.*, AC-23 (1978) 2.
14. Sifakis, J. "Use of Petri-Nets for Perf. Evaluation." in H. Beilner and E. Gelembe (eds), *Measuring, Modeling and Evaluation of Comp. Systems*, North Holland, 1977.
15. Tenney, R.R. and N.R. Sandell, Jr. "Structures for Distributed Decision Making." *IEEE Trans. SMC*, SMC-11 (1981) 8, pp. 517–527.