CALCULATING TIME-RELATED PERFORMANCE MEASURES OF A DISTRIBUTED TACTICAL DECISIONMAKING ORGANIZATION USING STOCHASTIC TIMED PETRI NETS[1]

R.Paul Wiley and Robert R. Tenney

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

I. INTRODUCTION

One of the key categories in which a distributed tactical decision making organization must be evaluated to assess its effectiveness is time-related performance measures. Examples of these measures include: the rates at which different tasks are completed, the probability that shared resources are available when requested, the average number of tasks waiting to be performed, the percentage of tasks successfully completed, etc. These performance measures depend strongly on the delays in decisionmaking due to organizational architecture and associated coordination protocols. If we are able to understand the effects of these architectures and protocols on the performance measures, we will be one step closer to being able to design effective distributed, real-time decisionmaking organizations.

A realistic model of distributed real-time decisionmaking organizations must incorporate at least three features: asynchronous protocols, concurrent operations, and random task-completion times. The asynchronous protocols are necessary since the groups or agents which compose the organization cannot be tightly synchronized; i.e. they do not communicate with each other at prespecified times. The concurrent operations are necessary since different parts of the organization work independently, coordinating their activities regularly to make sure the overall objective is being achieved. Finally, random task-completion times are necessary since most agents or groups perform a wide variety of tasks under many different conditions.

Stochastic Timed Petri Nets (STPNs) naturally incorporate all of these features. The basic objective of this paper is to introduce a methdology that can be used to analyze the dynamic and steady state behavior of these nets and, consequently, enhance our understanding of the time-related issues of the organizations they model.

The methodology presented in this paper can be used to analyze live and save STPNs (terms defined later). This class of STPNs can model organizations with finite queues and interesting protocols such as priorities and/or probabilistic choices. For organizations that can be modeled with this class of STPNs, the time-related performance measures listed in the first paragraph, among others, can be calculated.

So as not to lose sight of applications in the abstraction of mathematics, all steps of the methodology will be illustrated by an example. This example models the use of a shared resource by three groups. The groups could correspond to three naval battle groups, each responsible for a geographical sector, collectively in charge of defending a naval task force against submarine attacks. The shared resource in this situation could correspond to a special purpose helicopter, based on the main carrier of the task force, equipped with bouys and sonar equipment (i.e., a LAMPS helicopter). Periodically, the captain of each battle group will request the use of the helicopter.[2] We assume that the probability distribution for the time between requests for each battle group captain is known, as well as the probability distribution for the time that the helicopters will spend in each sector. In addition, we assume that the protocol or decision rule that decides which among the three captains will get use of the helicopter, in case of a conflict, is assumed (or hypothesized). Given the required information, this shared resource problem can be modeled with a STPN and such questions as the percentage of time the helicopter (the shared resource) is being used and by whom, the average amount of time that each captain must wait after he has requested use of the helicopter, the rates at which each captain will get use of the fleet, etc., can be answered.

An overview of this paper goes as follows. We start by defining STPNs and presenting the relevant concepts. We then show how the operation of a STPN can be viewed in terms of the operation of an infinite collection of Unfolded STPNs. This decomposition is then used to write state equations for the system, which in turn can be used to analyze the STPN (and the organization it models) with respect to time-related performance measures.

II. STOCHASTIC TIMED PETRI NETS

Stochastic Timed Petri Nets are graphs with two types of nodes: places (drawn as a circle and labeled p_i) and transitions (drawn as a line segment and labelled t_i), along with directed arcs going from one type of node to the other (see Figure 1). Besides

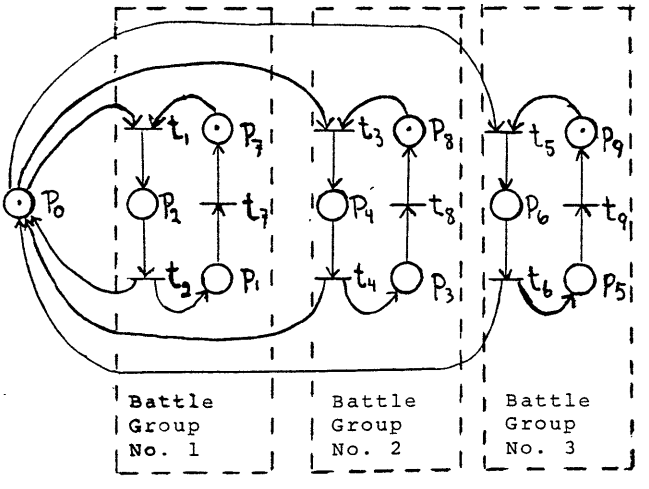

Figure 1. STPN Model of Naval Defense Shared Resource Example

nodes and arcs, a STPN assigns to every place a non-

[1]This research was supported by the Office of Naval Research under grant ONR/N00014-77-C-0532 (NR 041-519) and ONR/N00014-84-K-0519 (NR 649-003).

negative number of tokens, called the marking of the net, and a nonnegative random processing time[3]. When a token arrives at a place, it is defined to be unavailable (following Sifakis [3]). The token remains in this state until the instant when the processing has finished. At this time, the token becomes available. We assume that the processing time probability distributions for every place are given, and that they are stationary and independent from place to place.

In the STPN shown in Figure 1, the shared-resource example, a token arriving at p_1 represents the helicopter being sent to geographical sector No. 1. The token being processed in that place represents the helicopter performing its tasks in that sector. And finally, the token becoming available again represents the helicopter completing its tasks, a time after which it is available to be sent elsewhere.

Tokens move around by transition firings. A transition is enabled, that is, it may fire, only when all of its input places have available tokens. When a transition does fire, it removes a token from all of its input places and adds a token to all of its output places. If transition t_1 in Fig. 1 fires, for example, we remove a token from places p_0 and p_1 and add a token to place p_2.

The firing of the mentioned transition t_1, in terms of the naval example, represents the act of sending the LAMPS to the first geographical sector. Before the LAMPS will be dispatched, however, two facts must be true. The LAMPS (represented by a token in p_0) must be available, and the captain of the first geographical sector must request its use (request represented by a token in p_7). Thus, transition t_1 coordinates the requests from the captain of Battle Group No. 1 with the availability of the shared resource. Moreover, while the LAMPS is performing its tasks in the first geographical sector, the captains of the other geographical sectors cannot use it. In the STPN, this restriction is enforced by the fact that, during this time, no token is available in p_0 to fire transitions t_3 or t_5.

Two or more transitions are said to be in conflict if firing one will disable the others. In Fig. 1, transitions t_1, t_3 and t_5 are in conflict. If an enabled transition is not in conflict, it fires instantly. If several enabled transitions are in conflict, then a decision rule, specified a priori, selects one and that transition fires instantly. These decision rules can depend on the relative firing times of the transitions that feed tokens into the input places of the transitions which are in conflict.

In the naval defense example, this conflict represents the situation where two or more captains request the use of the LAMPS, while it is in service in the area. A decision rule must be provided to settle this conflict. This decision rule can be a function of the relative firing times of t_7, t_8, t_9 and either t_2, t_4 or t_6 (whichever transition fed the token into p_0). Possible decision rules include priorities, and the fleet assigned accordingly, or probabilistic choices, whereby the fleet is assigned according to some probabilistic rule (which might account for varying battle conditions), or some combination of the above.

In the preceding paragraphs, we have formulated the rules of operation for a STPN. In order to implement these rules, it is important to determine what information must be recorded in order to predict the future behavior of the system. That is, we need to determine the "state" of a STPN. One fact we need to know, certainly, is the position of every token (the marking of the net) since this specifies what transitions will potentially be enabled. In addition, we need to know the amount of time that each token has been in each place to determine whether it is available or not. One way of determining the amount of time a particular token has been processed is to remember the time at which the transition that created that token fired, and subtracting that time from real time. In the next section, we will take this last approach. Thus, a state for a STPN consists of the marking of the net plus the time that each token has been in each place. We can further argue that if any of this information is missing, then we do not have enough information to predict the future behavior of the system. Thus, the mentioned state is also minimal.

There is one last item that we must specify in order to follow the evolution of a STPN, and that is the initial conditions. Given the previous discussion on the state of a STPN, we assume that the initial marking and the time that each token has been in each place at time zero are given. With the given initial conditions, the known decision rules, and the processing times determined by their respective probability distributions, the STPN evolves autonomously in time. It is this autonomous evolution what we wish to study, understand, and compute performance measures for.

One can think of the operation of a STPN as a succession of markings, with the changes from one marking to another specified by the transition firings and the rules for moving tokens. If the number of tokens in any particular place can never exceed one, regardless of what processing time probability distributions and decision rules are assigned, the STPN is said to be safe. If the probability that any particular transition firing gets arbitrarily close to one if we wait long enough, regardless of what state the STPN has reached, the STPN is defined to be live. And finally, if there exists a directed path from any node (place or transition) to any other node, the STPN is strongly connected.

Throughout this paper, we will assume that the STPNs we are studying are strongly connected, live, safe, and with a finite number of nodes. The first assumption is a necessary condition for the STPN to be safe (Ramchandani [2]) and also ensures that we are not solving a problem that could be split into two or more independent problems. The second assumption, liveness, says that all parts of the system will operate regularly. If there exists a transition that is not live, then the part of the system associated with that transition should be deleted since it is not performing any activities. Liveness is also a necessary condition for convergence. The third condition, safeness, is necessary to impose ordering. If we allowed multiple tokens in a place, then they would be free to cross each other depending on their variable processing times. In order not to keep track of all possible orderings of tokens, we require safeness. Notice that this still allows us to model (bounded) finite first-in first-out queues by concatenating a series of safe places, as shown in Figure 2.

Now that all the necessary concepts have been defined, we can specify the performance measures that we are interested in obtaining. These performance measures are the average transition firing rates, the average number of tokens in each place, the average amount of time a token spends in each place, and finally, the probability that a token will enable one of several transitions which may be in conflict. The basic methodology actually obtains much more since we also obtain the probability distribution (and not just the mean value) for the time a token spends in each

[2]For simplicity, we assume that there is only one helicopter. Generalization to many is easy.

[3]Many authors, Ramchandani [2], Zuberek [5] and Molloy [1], associate processing times with transitions instead of places. The distribution makes no difference for the class of problems studied in this paper.

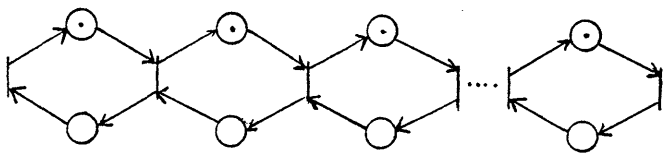

Figure 2. STPN Model of a Finite First-In First-Out Queue

place. From the list of performance measures, most, if not all, of the time-related performance measures of the original organization can be answered.

III. STATE EQUATIONS

III.a Introduction

In the previous section, we established that the state of a STPN consists of the marking of the net and the time each token has spent in each place. We also mentioned that one could deduce the time a token spends in each place by recording the firing time of the transition that fed the token into that place. In this section, we will derive equations which keep track of the transition firing times and from which the marking can be deduced. Thus, in light of the previous discussion, these equations can be used to follow the state of the STPN as it evolves.

The derivation is performed in two steps. In the first step, we show that the evolution of a STPN can be visualized in terms of the evolution of an infinite collection of Unfolded STPNs. This visualization provides an ordered index that can be used to decribe the system mathematically. In the second step, the mentioned index is used to write firing time state equations.

III.b Unfolded STPNs

One of the main difficulties of analyzing STPNs, or any other mathematical model of asynchronous concurrent systems, is obtaining an ordered index which can be used to describe the system mathematically. The usual index, time, is not appropriate since different parts of the system work independently. But STPNs are causal systems, that is, certain transitions must fire and certain tokens must be processed before other transitions can fire. The Unfolded STPNs, which we discuss in this subsection, capture this causality and provide the ordered index that we seek.

An Unfolded STPN is constructed by taking the original STPN, cutting every directed circuit at an appropriate transition, and "unfolding" the net. For example, Figure 3 shows the Unfolded STPN corresponding to the STPN of Fig. 1.

As can be seen from this example, every place and transition is labeled by the superscript k. This variable indicates to which copy of the Unfolded STPN the places and transitions belong. This labelling is important since, as we already mentioned, the operation of the regular STPN can be viewed in terms of the operation of an infinite collection of Unfolded STPNs. Also observe that the Unfolded STPN begins and ends with the same set of transitions, each transition in the first set labeled with the variable k and each transition in the second set labeled with the variable k+1. This fact does not mean that subsequent copies of the Unfolded STPN are connected. Rather, it means that there are two copies of each of these transitions. The reason we need duplicate copies of these transitions will become obvious later.

As mentioned previously, the operation of the regular STPN can be viewed in terms of the operation of

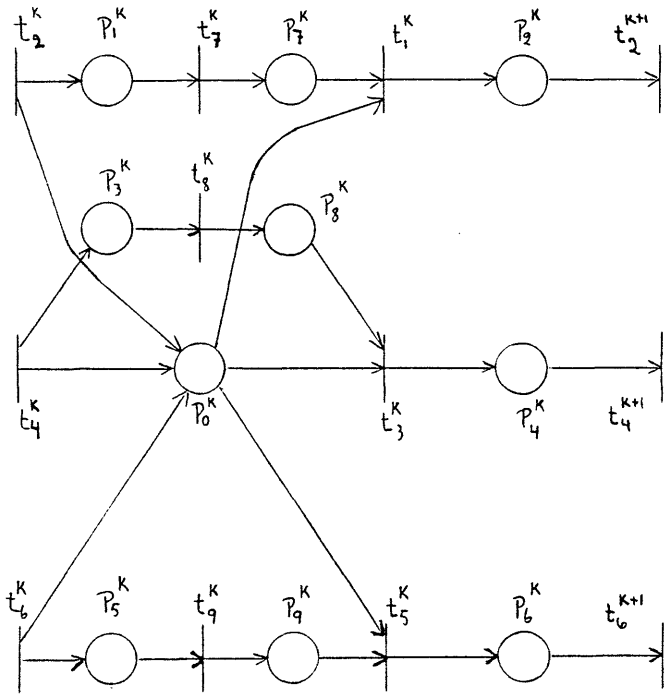

Figure 3. Unfolded STPN Corresponding to Shared Resource STPN of Fig. 1

an infinite collection of Unfolded STPNs. Since the latter form a disjoint collection, we must specify what we mean by the operation of the collection Unfolded STPNs. We do this now.

Imagine having an infinite collection of Unfolded STPNs numbered k=1,2,3,... The k^{th} copy of this collection is referred to as the k^{th} stage. Now suppose we place a token in every place of stage 1 for which the corresponding place in the regular STPN contains a token in the initial marking. Also suppose that the time each of these tokens has been processed at time 0 is initialized to be the same as the initial times specified for the regular STPN. Then we can implement the rules of operation that we describe in the previous section for stage 1. Since the Unfolded STPN contains no directed circuits, two things can happen to the tokens. The first possibility is that they will leave the stage by having some of the transitions labelled with the superscript 2 fire. The other possibility is that the tokens will be deadlocked.

As an illustration, consider the STPN of Fig. 1 along with its corresponding Unfolded STPN in Fig. 3. Initially, stage 1 contains a token in p_0^1, p_7^1, p_8^1 and p_9^1. Also initialized are the firing times for transitions t_2^1, t_7^1, t_8^1, and t_9^1 (this is equivalent to initializing the times each token has been processed at time 0). Now, suppose the given decision rule picks t_1^1 as the transition to fire (since t_1^1, t_3^1 and t_3^1 are in conflict). This firing takes a token from p_0^1 and p_1^1 and places a token into p_2^1, where it is processed. After that token is processed, transition t_2^2 fires, and that token leaves stage 1. Meanwhile, the tokens in p_3^1 and p_5^1 are deadlocked since the output of those transitions will never fire.

So far we have explained how stage 1 of an Unfolded STPN will operate. To initialize stage 2, we must remember which transitions fired at the end of stage 1. A token is placed in the output place of

every transition that fired. Thus, for the example that we have been considering, a token is placed of p_0^2 and p_1^2. We also place a token in every place of stage 2 whose corresponding place in stage 1 contains a deadlocked token. In the example, this corresponds to placing a token in p_8^2 and p_9^2. The transition firing times that originally fed these tokens are also recorded in stage 2. In the example, the firing time of $t_4^2(t_6^2)$ is set to be the same as the firing time of $t_4^1(t_6^1)$. Now, stage 2 has all the necessary state variables initialized and we can implement the rules of operation for it.

Thus, the collection of Unfolded STPNs operates one stage at a time. Each stage evolves according to the rules of operation until every token has left the stage or is deadlocked. Then the following stage is initialized according to the discussion given above, and the collection of Unfolded STPNs keeps operating in this manner forever.

It is not hard to see that the operation of the Unfolded STPNs reflects the operation of the original STPN. In mathematical terms, there exists a one to one correspondence between the sample paths of the Unfolded STPNs and the sample paths of the regular STPN (see [4] for proof). This correspondence allows us to study the evolution of the regular STPN by studying the evolution of the corresponding Unfolded STPNs. This approach is exactly the one we shall take.

III.c Transition Firing Times

The evolution of the Unfolded STPNs can be divided into two separate processes. The first process keeps track of the evolution from one marking to the next which results from transition firings. The second process keeps track of the transition firing times and the processing of tokens. The two processes are connected since the first process, through its markings, determines which transitions will potentially be enabled while the second process actually picks the transition which fires next. The rules for keeping track of markings have already been given in Section II. Therefore, in this section, we will derive equations which will enable us to keep track of the transition firing times.

In order to perform this derivation, consider the STPN shown in Figure 4, which we assume is part of a larger Unfolded STPN. Let $x_i(k)$ be the firing time for

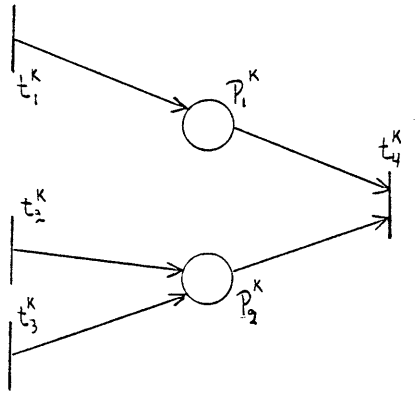

Figure 4: STPN for Derivation of State Equations

transition t_i^k and $z_i(k)$ be the processing time for place p_i^k. Our objective is to obtain an equation which will give us the numerical value of $x_4(k)$ given $x_1(k)$, $x_2(k)$, $x_3(k)$, $z_1(k)$ and $z_2(k)$. In order to do this, we must know at what times the tokens entered p_1^k and p_2^k. The time the token entered p_1^k is given by $x_1(k)$, and the time the token entered p_2^k is given by $\max[x_2(k), x_3(k)]$. We can deduce this last fact since we know only t_2^k or t_3^k fired, and the latest firing is the one which fed the token into p_2^k. After each token enters their respective places, they are processed. When both tokens have finished being processed, transition t_4^k fires. We conclude that

$$x_4(k) = \max\{z_1(k) + x_1(k),$$
$$z_2(k) + \max[x_2(k), x_3(k)]\}$$

If transition t_4^k had contained more than two input places, then the main maximization operator in this last equation would have contained more terms. Similarly, more input transitions to places p_2^k or p_3^k could also be easily handled.

Firing time equations can be written in the manner prescribed above for all transitions that fire in the k^{th} stage. Which transitions fire in each stage depend on the initial marking of that stage and the decision rules. As specified previously, the evolution of the k^{th} stage of the Unfolded STPN will terminate when transitions corresponding to the next stage fire or when the tokens become deadlocked. To initialize the calculations in the next stage, the deadlocked tokens must be "transported" to the corresponding places in the next stage and the firing times of the transitions that fed these tokens must also be initialized.

Let us illustrate this discussion with the shared resource example. Suppose the initial marking of stage k places a token in p_0^k, p_1^k, p_8^k and p_9^k, and that the known firing times are $x_2(k)$, $x_8(k)$, and $x_9(k)$. Further assume that t_2^k is the transition that fed the token into p_0^k. The only transition that can fire from this state is t_7, and

$$x_7(k) = z_1(k) + x_2(k).$$

Now let the token in p_7^k be processed. Once this has been accomplished, the decision rule picks t_1^k, t_3^k or t_5^k to fire. Suppose it picks t_3^k. As a result of that decision t_4^{k+1} will also fire. The equations are

$$x_3(k) = \max\{z_8(k) + x_8(k),\ z_0(k) + x_2(k)\}$$

and

$$x_4(k+1) = z_4(k) + x_3(k).$$

As a result of all these firings, a token is placed in p_3^{k+1} and p_0^{k+1}. Meanwhile, the tokens in p_7^k and p_9^k are deadlocked. Therefore, these tokens must be moved to p_7^{k+1} and p_9^{k+1}, respectively, and the firing times of t_7^{k+1} and t_9^{k+1} initialized as follows:

$$x_7(k+1) = x_7(k)$$

and

$$x_9(k+1) = x_9(k)$$

Thus, we started with a state in stage k and ended up with a state in stage k+1. The state equations can be used in this manner to follow the evolution of the Unfolded STPNs, and of the regular STPN that it is derived from.

III.d Basic Methodology

Similar state equations to the ones just discussed can be derived for any live and safe STPN. Once these equations are obtained, they can be used to analyze the STPN with respect to the time related performance

measures cited previously. The basic methodology uses the state equations to recursively calculate the probability distributions of the states as a function of k, the ordered index that tells us what stage the evolution of the Unfolded STPNs is going through, starting from the initial state. The details, algorithms, and conditions necessary for convergence and uniqueness are given in [4].

The state equations that we derive for a STPN could also be used to simulate the system by using random draws to determine the processing times and the probabilistic decisions. The shared resource example that we discuss in the next section was solved by using both methods: analysis and simulation. As they should, the values of performance measures obtained by both programs agree.

IV. COMPUTER EXAMPLE

Throughout this paper, we have used the naval defense shared resource example to motivate the study of STPNs and explain the steps of the methodology. In this section, we present the results of a computer program that implements this methodology and interpret some of the results in terms of the naval defense example.

As noted in Section I, inputs to the methodology are the probability distributions for the processing times of every place, and the decision rule which determines which battle group will get use of the LAMPS helicopter in case of conflicting demands. The processing time probability distributions are shown in Fig. 5.

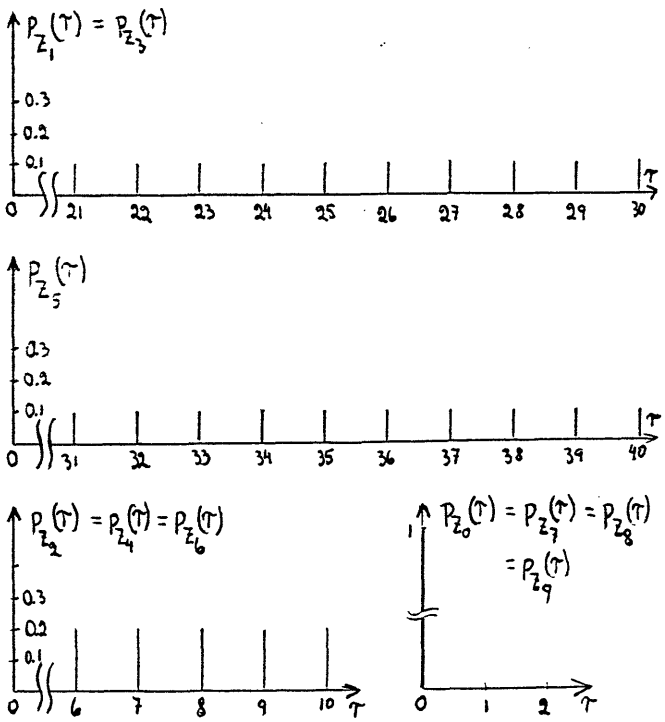

Figure 5. Processing time Probability Distributions.

The decision rule that the program uses is assigning equal priorities to Battle Groups Nos. 1 and 2 and sending the LAMPS helicopter to whichever has been waiting the longest. (In case of a tie, the helicopter is sent to the Battle Group No. 1.) Battle Group No. 3 is given lowest priority. The LAMPS will be sent there only if the captains of the other battle groups have not requested it. The decision rule just described might be reasonable if the submarine attack is most likely to come from the geographical regions protected by Battle Groups Nos. 1 and 2, and is less likely to come from the geographical region protected by Battle Group No. 3.

The results of the program are listed in the following table. From these results, we can deduce

Place Number	Expected Delay	Expected Number of Tokens
0	4.7874e+00	3.7438e-01
1	2.5500e+01	7.2613e-01
2	7.9999e+00	2.2780e-01
3	2.5500e+01	7.2354e-01
4	7.9999e+00	2.2699e-01
5	3.5500e+01	7.5795e-01
6	7.9999e+00	1.7080e-01
7	1.6173e+00	4.6055e-02
8	1.7429e+00	4.9454e-02
9	3.3363e+00	7.1233e-02

Transition Number	Expected Rate	Probability of Decision
1	2.8475e-02	3.6424e-01
2	2.8475e-02	1.0000e+00
3	2.8374e-02	3.6272e-01
4	2.8374e-02	1.0000e+00
5	2.1350e-02	2.7302e-01
6	2.1350e-02	1.0000e+00
7	2.8475e-02	1.0000e+00
8	2.8374e-02	1.0000e+00
9	2.1350e-02	1.0000e+00

Table. Output of Computer Programs

that the LAMPS is idle 37% of the time (average number of tokens in p_0), that the probability of the LAMPS being sent to Battle Group No. 2, once it is available is .36 (probability of decision t_3), that the captain of Battle Group No. 3 must wait, on the average, 3.3 units of time after a request for the LAMPS (average time token spends in p_9), that the LAMPS is sent out at an average rate of .078 (sum of t_1, t_3 and t_5 firing rates), etc. These and similar measures give a complete picture of the situation with respect to time-related performance measures.

V. CONCLUSIONS AND FUTURE RESEARCH

We have shown, how STPNs can be used to model distributed tactical decisionmaking organizations. We then outlined the methodology by which STPNs can be analyzed with respect to time-related performance measures. Finally, the modeling and methodology were applied to a simple, yet interesting, naval defense shared resource example. The results clearly indicate that the study of STPNs will help us understand the dynamic and steady state behavior of the organizations they are capable of modeling.

Future research directions are many. One of these is modeling of more complex organizations using STPNs.

In the naval defense example, for instance, we could have modeled the asynchronous protocols and delays of each battle group's own operations. These models could then be substituted for places p_1, p_3, and p_5, and the resulting analytical program would provide a more accurate assessment of the situation. Another area of research is performing a sensitivity analysis to small changes in processing times. This analysis could help an organization designer understand what changes could be made to improve the organization. And finally, a third area of research is finding alternate, faster ways of using the state equations to find the steady state probability distributions. The methodology outlined in this paper calculates these distributions recursively, which can converge rather slowly.

References

[1] M.K. Molloy, "Performance Analysis Using Stochastic Petri Nets", IEEE Transactions on Computers, Vol. (AS-24, No. 7, (July 1977), pp. 400-405.

[2] C. Ramchandani, "Analysis of Asynchronous Concurrent Systems by Timed Petri Nets", Technical Report 120, Lab. for Computer Science, M.I.T., Cambridge, MA (1974).

[3] J. Sifakis, "Use of Petri Nets for Performance Evaluation". Measuring, Modeling and Evluating Computer Systems, H. Beilner and E. Gelenbe, eds., North-Holland, pp. 75-93.

[4] R.P. Wiley, "Performance Analysis of Stochastic Timed Petri Nets", Ph.D. Thesis, Electrical Engineering and Computer Science Department, M.I.T., Cambridge, MA (expected December 1985).

[5] W.M. Zuberek, "Timed Petri Nets and Preliminary Performance Evaluation", The 7th Annual Symposium on Computer Architecture Conference Proceedings (May 6-8, 1980), pp. 88-96.