7-th Phoenix Int. Conference on Computers and Communications, Scottsdale AZ, 16–18 March 1988, pp.324–329.

TIMED PETRI NET MODELS OF QUEUEING SYSTEMS

W.M. Zuberek

Department of Computer Science, Memorial University of Newfoundland
St. John's, NL, Canada A1C 5S7

Abstract

It is shown that for timed Petri nets with exponentially distributed firing times (M-timed Petri nets), the state space can be generated directly from net specifications, and then the stationary probabilities of states can be obtained by standard methods, developed for analysis of (continuous-time) Markov chains. Numerous performance measures can be derived from stationary probabilities of states. For unbounded nets (models of open network systems are usually unbounded), the state space is infinite, a transformation is thus needed that *folds* this infinite space into a finite representation, used for effective evaluation of probabilities. The paper presents a short theoretical background for performance evaluation using timed Petri nets followed by several examples of closed and open network models of simple computer systems.

1. INTRODUCTION

The widespread acceptance that Petri net models [1,3,12,15,16] are gaining in modelling of distributed, parallel and multiprocessor systems is due to simple representation of concurrency and synchronization of activities in such systems. It appears that the traditional methods, developed for analysis of sequential systems, are inadequate in such applications. Basic Petri nets, however, are not complete enough for the study of systems performance since no assumption is made on the duration of system activities. Several concepts of timed Petri nets have been proposed by assigning firing times to the transitions and/or places of Petri nets [2,7,9,11,14,17,18,19].

This paper is a continuation of the Ramchandani's approach [13] in which firing times are associated with transitions of a net, and tokens are removed from corresponding input places when the firing begins. For timed Petri nets with exponentially distributed firing times (the so called M-timed Petri nets [18]), the state space is a homogeneous continuous-time Markov chain (the firing rates assigned to transitions of a net do not depend upon time). The stationary probabilities of states can thus be obtained by known techniques [5,8], used for analysis of Markov processes, and then many performance measures can be derived by applying rules of operational analysis [4]. The paper shows that the state space (or the set of reachable states) can be (automatically) generated directly from net specifications, and explicit determination of the state space (required by other analytic approaches, e.g., queueing theory) is thus eliminated.

In this paper, the rates of firing transitions as well as free-choice probabilities are assumed to be constant since this is sufficient in most of the models of popular queueing systems. However, a very simple modification of some definitions can also take into account firing rates which depend upon the state of a net. Similarly, another simple extension can deal with state-dependent free-choice probabilities. Several other simplifications are assumed in this paper in order to simplify the state descriptions; a more general presentation is given in [18,22].

It should be noted that there are several similarities between M-timed Petri nets and stochastic (and generalized stochastic) Petri nets [2,6,11] since in both cases the firing times are exponentially distributed random variables. However, the similarities are often misleading since the mechanisms of transition firings are quite different [21]. In stochastic nets the tokens of a *firing* transition actually remain in corresponding (input) places, and after the firing time (sometimes called enabling time [14]) the transition fires instantaneously. Consequently, the state space of stochastic nets is determined by the set of reachable markings which does not take into account timing constraints. In M-timed nets the tokens of firing transitions are removed from corresponding places at the beginning of firing, and they remain in corresponding (firing) transitions for the whole firing period. The state space is usually quite different from the marking space (for example, infinite state space may correspond to finite marking space and vice versa [18]), and this allows to represent the behavior of the modelled systems more realistically than in the stochastic approach. Finally, the *timed* approach can be used for different firing time distributions with minor modifications of the formalism [19] while the stochastic approach is valid only for the exponential distribution of firing times.

This paper is organized in 4 main sections. Section 2 recalls basic concepts for marked Petri nets. M-timed nets are introduced in section 3. Section 4 presents several examples of closed network models network models of simple computer systems, while section 5 shows how to use timed Petri nets for analysis of open network models.

2. MARKED PETRI NETS

A (generalized) Petri net $\mathbf{N}$ (sometimes called net structure) is a 6-tuple $\mathbf{N} = (P, T, A, w, B, C)$ where:

P is a finite, nonempty set of places,

T is a finite, nonempty set of transitions,

A is a set of directed arcs which connect places with transitions and transitions with places, $A \subseteq P \times T \cup T \times P$,

w is a weight function which assigns a positive integer *weight* to each arc of the net, $w : A \to \{1, 2, ...\}$,

B is a (possibly empty) set of inhibitor arcs, $B \subset P \times T$, and A and B are disjoint sets,

C is a (possibly empty) set of interrupt arcs, $C \subseteq B$.

As usual, $Inp(p)$, $Out(p)$, $Inp(t)$, $Out(t)$, $Inh(t)$ and $Int(t)$ denote the sets of input and output transitions of a place p, the sets of input and output places of a transition t, and the sets of inhibitor and interrupting places of t, respectively. The notation is extended in the usual way to sets of places and transitions. (The distinction between inhibitor and interrupt arcs becomes important for timed nets; for nets *without time*

interrupt and inhibitor arcs are exactly the same, however, it is convenient to introduce this two types of arcs from the very beginning, and it should be observed that the set of interrupt arcs is a subset of inhibitor arcs.)

A place p is shared iff it is an input place for more than one transition. A shared place p is guarded iff for each two different transitions sharing p there exists another place p_k which is in the input set of one and in the inhibitor set of the other of these two transitions. A place p if free-choice iff the input sets of transitions sharing p are identical and the weights of corresponding arcs are equal. A net is free-choice iff each shared place is either guarded or free-choice.

The relation of *sharing a place* in a free-choice net is in fact an equivalence relation in the set of transitions T, hence it determines a partition of T into a set of free-choice equivalence classes denoted by $Free(T) = \{T_1, T_2, ..., T_k\}$.

A marked generalized Petri net $\mathbf{M}$ is a pair $\mathbf{M} = (\mathbf{N}, m_0)$ where:

$\mathbf{N}$ is a generalized Petri net, $\mathbf{N} = (P, T, A, w, B, C)$,

m_0 is an initial marking function, $m_0 : P \to \{0, 1, ...\}$.

Let any function $m : P \to \{0, 1, ...\}$ be called a marking of a net $\mathbf{N} = (P, T, A, w, B, C)$.

A transition t is enabled by a marking m iff every t's input place $p \in Inp(t)$ contains at least $w(p, t)$ tokens, i.e., $m(p) \geq w(p, t)$, and every t's inhibitor place $p \in Inh(t)$ contains zero tokens. The set of all transitions enabled by m is denoted by $En(m)$.

Every transition enabled by a marking m can fire. When an enabled transition t fires, tokens are removed from t's input places (but not inhibitor places) in numbers corresponding to the weights of input arcs, and similarly, the weights of t's output arcs determine the numbers of tokens added to output places. A firing of an enabled transition is thus a transformation of the marking function. A marking m_j is directly reachable (or t_k–reachable) from a marking m_i iff there exists a transition $t_k \in En(m_i)$, such that

$$\forall(p \in P)m_j(p) = \begin{cases} m_i(p) - w(p, t_k), \text{if } p \in Inp(t_k) - Out(t_k), \\ m_i(p) + w(t_k p), \text{if } p \in Out(t_k) - Inp(t_k), \\ m_i(p) - w(p, t_k) + w(t_k, p), \\ \qquad \text{if } p \in Inp(t_k) \cap Out(t_k), \\ m_i(p), \text{otherwise.} \end{cases}$$

Since the firings in free-choice equivalence classes are selected in a random way, it is convenient to describe all possibilities of different firings as a function of the marking m. The *selection* function is defined as a function which indicates all possible transition firings that can occur *simultaneously* (and some transitions may fire *several times*).

A selection function of a marking m in a net $\mathbf{N}$ is any function $g : T \to \{0, 1, ...\}$ such that:

(1) there exists a sequence of transitions $v = (t_{i_1}, t_{i_2}, ..., t_{i_n})$ in which $t_{i_j} \in En(m_{i_{j-1}})$ and

$$\forall(p \in P)m_{i_j}(p) = m_{i_{j-1}}(p) - \begin{cases} w(p, t_{i_j}), & \text{if } p \in Inp(t_{i_j}), \\ 0, & \text{otherwise,} \end{cases}$$

for $j = 1, ...n$ and for $m_{i_0} = m$,

(2) the set of transitions enabled by m_{i_n}, $En(m_{i_n})$, is empty,

(3) for each $t \in T$ the number of occurrences of t in the sequence v is equal to $g(t)$.

The set of all selection functions of a marking m is denoted by $Sel(m)$.

3. M-TIMED PETRI NETS

In timed Petri nets, each transition t takes a *real* time to fire. When a transition tt@ is enabled, a firing can be initiated by removing tokens from tt@'s input places. The tokens remain in the transition t for the *firing time*, and then the firing terminates by adding tokens to each of t's output places. Each of the firings is initiated in the same instant of time in which it is enabled. If a transition is enabled while it fires, a new, independent firing can be initiated.

The firing times can be defined in several ways. In D-timed nets [19] they are deterministic (or constant), i.e., there is a *firing time* function which assigns the duration of firings to each transitions of a net. These durations can be state-dependent [7] or state-independent [13,14,19]; in both cased the resulting state graphs are homogeneous discrete-time semi-Markov processes. In M-timed nets [18] and stochastic nets [2], the firing times of transitions are exponentially distributed random variables, and a *firing rate* function determines corresponding (state dependent or state independent) rates of firing transitions. The memoryless property of the exponential distribution is an important factor in analysis of nets with exponentially distributed firing times.

In timed Petri nets with interrupt arcs, a firing of a transition can be discontinued. If, during a firing period of a transition t, at least one of t's interrupting places becomes nonempty (i.e., it receives a token as a result of a termination of another firing), the firing of t ceases and the tokens removed from t's input places at the beginning of firing are *returned* to their original places.

A net $\mathbf{N} =)P, T, A, w, B, C)$ is simple if input sets of transitions with nonempty interrupting sets do not contain interrupting places

$$\forall(t \in T)Int(t) = \emptyset \vee Int(Inp(t)) = \emptyset,$$

where $\emptyset$ denotes the empty set. Simple nets do not allow *propagation* of interrupts when an interrupted transition, through its input places, interrupts another transition. Nonsimple nets require a rather straightforward extension of the description [18], and since nonsimple nets are required in modelling rather infrequently, only simple timed Petri nets are considered in this paper.

An M-timed Petri net $\mathbf{T}$ is a triple $\mathbf{T} = (\mathbf{M}, c, f)$, where:

$\mathbf{M}$ is a free-choice marked Petri net, $\mathbf{M} = (\mathbf{N}, m_0)$, $\mathbf{N} = (P, T, A, w, B, C)$,

c is a choice function which assigns a *free-choice* probability to each transition of a net in such a way that

$$\forall(T_i \in Free(T)) \sum_{t \in T_i} c(t) = 1,$$

r is a firing rate function which assigns the rate of firing $r(t)$ to each transition t of the net, $r : T \to \mathbf{R}^+$, and $\mathbf{R}^+$ denotes the set of positive real numbers; the firing time of a transition t is a random variable $x(t)$ with the distribution function

$$\text{Prob}(x(t) > y) = e^{-y * r(t)}, y > 0.$$

Since in timed nets tokens are distributed in places as well as in (firing) transitions, the *state* of a timed net is defined as a pair of functions, one describing the distribution of tokens in places, and the second in (firing) transitions.

A state s of an M-timed Petri net $\mathbf{T}$ is a pair $s = (m, n)$ where:

mm@ is a marking function, $m : P \to \{0, 1, ...\}$,

n is a firing-rank function which indicates (for each transition of the net) the number of active firings, i.e., the number of firings which have been initiated but are not yet terminated, $n : T \to \{0, 1, ...\}$.

An initial state s_i of a free-choice net $\mathbf{T}$ is a pair $s_i = (m_i, n_i)$ where n_i is a selection function from the set $Sel(m_0)$, $n_i \in Sel(m_0)$, and the marking m_i is defined by

$$\forall (p \in P) m_i(p) = m_0(p) - \sum_{t \in Out(p)} n_i(t) * w(p, t).$$

A free-choice net $\mathbf{T}$ may have several different initial states.

A state $s_j = (m_j, n_j)$ is directly reachable (or (t_k, g_l)-reachable) from the state $s_i = (m_i, n_i)$ iff:

(1) $n_i(t_k) > 0$,

(2) $g_l \in Sel(m_{ikj})$,

(3) $\forall (p \in P) m_j(p) = m_{ikj}(p) = \sum_{t \in Out(p)} g_l(t) * w(p, t)$,

(4) $\forall (t \in T) n_j(t) = n_i(t) - e_i(t) - d_{ij}(t) + g_l(t)$,

(5) $\forall (pinP) m_{ikj}(p) = m_{ik}(p) + \sum_{t \in Out(p)} d_{ij}(t) * w(p, t)$,

(6) $\forall (p \in P) m_{ik}(p) = m_i(p) + \sum_{t \in Inp(p)} e_i(t) * w(t, p)$,

(7) $\forall (t \in T) d_{ij}(t) = \begin{cases} 0, \text{if } \sum_{p \in Int(t)} m_{ik}(p) = 0, \\ n_i(t) - e_i(t), \text{otherwise,} \end{cases}$

(8) $\forall_(t \in T) e_i(t) = \begin{cases} 1, & \text{if } t = t_k, \\ 0, & \text{otherwise.} \end{cases}$

A state s_j is (generally) reachable from a state s_i if there is a sequence of directly reachable states from the state s_i to the state s_j. Also, a set $S(\mathbf{T})$ of reachable states is defined as the set of all those states which are reachable from the initial states of the net $\mathbf{T}$.

A state graph $\mathbf{G}$ of an M-timed Petri net $\mathbf{T}$ is a labeled directed graph $\mathbf{G}(\mathbf{T}) = (V, D, u)$ where:

V is a set of vertices which is equal to the set of reachable states of the net $\mathbf{T}$, $V = S(\mathbf{T})$,

D is a set of directed arcs, D $subseteqV \times V$, such that (s_i, s_j) is in D iff s_j is directly reachable from s_i in $\mathbf{T}$,

u is a labeling function which assigns the rate of transitions from s_i to s_j to each arc (s_i, s_j) in the set D, $u : D \to \mathbf{R}^+$, in such a way that if s_j is (t_k, g_l)-reachable from s_i, then

$$u(s_i, s_j) = r(t_k) * n_i(t_k) \prod_{T_z \in Free(T)} a(T_z, g_l) \prod_{t \in Ts_z} c(t)^{g_l(t)}$$

where the coefficient $a(T_z, g_l)$ describes the number of ways in which the selection function g_l can be realized in a free-choice class $T_z \in Free(T)$. It can be determined as follows. Let an n-argument function ψ be defined recursively:

(1) $\psi(0, 0, ..., 0) = 1$,;

(2) $\psi(k_1, k_2, ..., k_n) =$

$$\sum_{1 \le i \le n} \begin{cases} \psi(k_1, ..., k_{i-1}, k_i - 1, k_{i+1}, ..., k_n), & \text{if } k_i > 0, \\ 0, & \text{if } k_i = 0. \end{cases}$$

Then, for the class $T_z = \{t_{z_1}, t_{z_2}, ..., t_{z_n}\}$:

$$a(T_z, g) = \psi(g(t_{z_1}), g(t_{z_2}), ..., g(t_{z_n})),$$

and, for any marking m

$$\sum_{g \in Sel(m)} \prod_{T_z \in Free(T)} a(T_z, g) \prod_{t \in T_z} c(t)^{g(t)} = 1.$$

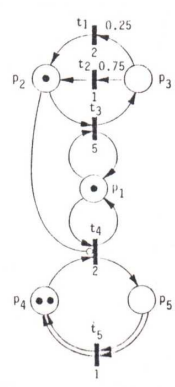
It should be noticed that state graphs of free-choice M-timed Petri nets are continuous-time homogeneous Markov chains. Consequently, the stationary probabilities of states can be obtained by standard techniques used in analysis of Markov processes. Stationary probabilities of states are used in derivations of many performance measures, for example utilization factors, throughput rates, response and waiting times, etc. [4,5,8].

4. CLOSED NETWORK MODELS

In Petri net models of queueing systems places represent system queues, transitions servers, directed arcs model flow of activities in the model as well as synchronization constraints for concurrent activities, inhibitor and interrupt arcs are used to model priorities of simultaneous events and preemptions of servers, and arcs with weights greater than one are sufficient to represents group arrivals and services.

The M-timed Petri net shown in Fig.1 (as usual, places are represented by circles, transitions by bars, inhibitor arcs have small circles instead of arrowheads, the initial marking function is indicated by a number of dots in corresponding places, and the free-choice and firing rate functions are given as additional descriptions of transitions) is a model of an interactive system with 2 classes of users (and jobs) and a nonpreemptive priority scheduling discipline; the system consists of a central server with a queue of waiting jobs, n_1 class-1 terminals and n_2 class-2 terminals. The class-1 users submit one job at a time, while the class-2 users submit two jobs at each arrival instant, and processing of both jobs must be completed to start another terminal (or *thinking*) phase. Terminal times for class-2 jobs are exponentially distributed with the average of 1 time unit, while terminal times of class-1 jobs are hyperexponentially distributed, and the average is equal to 0.5 time units with probability 0.25, and 1 time unit with probability 0.75. Moreover, the class-1 jobs have *higher* priority than the class-2 ones, i.e., they receive service before class-2 jobs. It is also assumed that all service times are exponentially distributed, and that the average service time is equal to 0.2 time units for class-1 jobs, and 0.5 time units for class-2 jobs (the numbers are not *realistic* since this is an example).

The central server is modelled by p_1, t_3 and t_4; the transitions t_3 and t_4 correspond to the central server processing class-1 jobs (t_3) and class-2 jobs (t_4) with the service rates (or the firing rates) equal to 5 and 2, respectively. The place p_1 with its initial number of tokens models the number of server channels, in this case 1. The places p_2 and p_4 represent the waiting queues (for class-1 and class-2 jobs, respectively). The inhibitor arc (p_2, t_4) disables t_4 whenever there is a job waiting in p_2, i.e., it models the priority of class-1 jobs over class-2



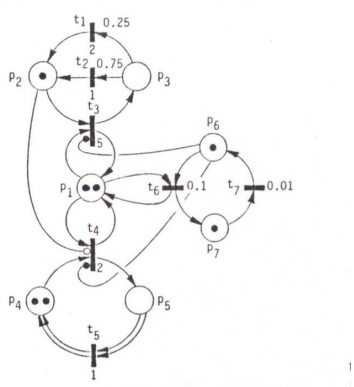


Fig.1. M-timed Petri net $\mathbf{T}_1$. Fig.2. M-timed Petri net $\mathbf{T}_2$.

ones. The transitions t_1 and t_2 model terminal times for class-1 (two-stage hyperexponential distribution), and t_5 - terminal times for class-2. The initial number of tokens in the places p_2 and p_3 represents the number of terminals in the class-1, n_1, and the initial number of tokens assigned to places p_4 and p_5 models the number of terminals in the class-2, n_2.

The derivation of the set $S(\mathbf{T}_1)$ of reachable states in given in Tab.1 which also contains the stationary probabilities $x(s)$ of the states $s \in S(\mathbf{T}_1)$. The stationary probabilities are obtained by solving the following system of *flow* equations:

$$\begin{cases} \sum_{1\le j\le K} u(s_j,s_i) * x(s_j) = x(s_i) \sum_{1\le j\le K} u(s_i,s_j); \\ \qquad\qquad\qquad\qquad\qquad\qquad i = 1,...,K-1; \\ \sum_{1\le i\le K} x(s_i) = 1. \end{cases}$$

where K is the number of states in the set $S(\mathbf{T})$.

		m_i	n_i		g_ℓ		
s_i	$x(s_i)$	1 2 3 4 5	1 2 3 4 5	t_k	1 2 3 4 5	s_j	$u(s_i.s_j)$
1	0.020	0 0 0 2 0	0 0 1 0 0	3	0 1 0 1 0	2	3.75
					1 0 0 1 0	3	1.25
2	0.131	0 0 0 1 0	0 1 0 1 0	2	0 0 0 0 0	4	1.00
				4	0 0 0 1 0	5	2.00
3	0.020	0 0 0 1 0	1 0 0 1 0	1	0 0 0 0 0	4	2.00
				4	0 0 0 1 0	6	2.00
4	0.086	0 1 0 1 0	0 0 0 1 0	4	0 0 1 0 0	7	2.00
5	0.130	0 0 0 0 1	0 1 0 1 0	2	0 0 0 0 0	8	1.00
				4	0 0 0 0 1	9	2.00
6	0.021	0 0 0 0 1	1 0 0 1 0	1	0 0 0 0 0	8	2.00
				4	0 0 0 0 1	10	2.00
7	0.034	0 0 0 1 1	0 0 1 0 0	3	0 1 0 1 0	5	3.75
					1 0 0 1 0	6	1.25
8	0.086	0 1 0 0 1	0 0 0 1 0	4	0 0 1 0 1	11	2.00
9	0.317	1 0 0 0 0	0 1 0 0 1	2	0 0 1 0 0	11	1.00
				5	0 0 0 1 0	2	1.00
10	0.056	1 0 0 0 0	1 0 0 0 1	1	0 0 1 0 0	11	2.00
				5	0 0 0 1 0	3	1.00
11	0.100	0 0 0 0 0	0 0 1 0 1	3	0 1 0 0 0	9	3.75
					1 0 0 0 0	10	1.25
				5	0 0 0 0 0	1	1.00

Tab.1. The set of reachable states for $\mathbf{T}_1$.

Many performance measures [5] can be derived from stationary probabilities of the states. For example, since the server is idle only in the states s_9 and s_{10} ($m_9(p_1) = m_{10}(p_1) = 1$), the stationary probability that the system is idle is equal to the sum $x(s_9)+x(s_{10}) = 0.373$ (Tab.1). Then the utilization of the system is immediately $1 - 0.373 = 0.627$ which is composed of 0.154 for class-1 jobs ($x(s_1)+x(s_7)+x(s_{11})$ since $n_i(t_3) > 0$ for $i = 1, 7, 11$), and 0.473 for class-2 jobs. Since the average service time for class-1 jobs is equal to 0.2 time units, and the server utilization for this class is 0.154, then the average throughput rate for class-1 jobs is equal to $0.154/0.2 = 0.77$ jobs per time unit, and the average turnaround time is equal to $1/0.77 = 1.30$ time units. Similarly, for class-2 jobs, the average throughput rate is equal to $0.473/(2 * 0.5) = 0.473$ *double* jobs per time unit, and the average turnaround time (for a pair of jobs) is equal to 2.114 time units.

Similar calculations can be repeated for different sets of model parameters. For example, to study the influence of the number of class-1 jobs on the turnaround time of class-2 jobs, it is sufficient to repeat the evaluations for different values of $m_0(p_2)$, i.e., the initial marking of the place p_2 (Fig.1). Moreover, different values of $m_0(p_1)$ represent different numbers of server's channels, etc. Fig.3 shows an example of such results, i.e., the average class-2 turnaround time as a function of the number of class-1 jobs with the number of server's channels as a parameter.

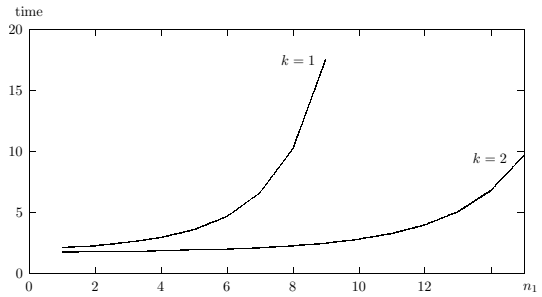


Fig.3. Class-2 turnaround time as a function of n_1.

A similar approach can be used for processors that are not completely reliable. Let each processor go through alternating periods of being operative and broken down, independently of other processors. If the operative and inoperative periods are distributed exponentially with means a and b, respectively, the average probability that a processor is operative is equal to $a/(a + b)$. This means that for the total number of processors equal to N, the effective number of available processors is reduced to $N * a/(a + b)$. It turns out that the operative and inoperative periods of processors can be modelled as a *superprocess* which *preempts* processors when they become inoperative.

A Petri net model of the previous system with unreliable processors is shown in Fig.2 (in fact, $\mathbf{T}_2$ models central server with two unreliable processors, $m_0(p_1) = m_0(p_6)+m_0(p_7) = 2$; interrupt arcs have black dots instead of arrowheads). The circuit p_6, t_6, p_7, t_7 models exponentially distributed operative and inoperative periods of time with the rates $r(t_6) = 0.1$ and $r(t_7) = 0.01$. During inoperative periods (t_6), the tokens representing processors are removed from the central server (i.e., from p_1). If a processor becomes inoperative while it is processing one of jobs, the arcs (p_6, t_3) and (p_6, t_4) interrupt firing transitions t_3 and t_4 (the tokens return to their input places, p_1 - the *processor* token, and p_2 and/or p_4 - the *job* tokens), after which t_6 initiates its firing removing the token from p_1 for an *inoperative* period of time.

Some other models of computer systems are discussed in [18,20,21].

5. OPEN NETWORK MODELS

Slightly different approach must be used for models in which the number of reachable states is infinite (state space of open network models is usually infinite). The M-timed Petri net shown in Fig.4 is a simple model of a batch processing system in which the interarrival times are exponentially distributed and the arrival rate is constant and does not depend upon the number of jobs in the system; the source of arriving jobs is modelled by p_4 and t_4 which *generate* the jobs with rate equal to 2 jobs per time unit. The queue of waiting jobs is represented by p_2, and the transitions t_1, t_2 and t_3 model the (batch) processor; the processing time is either exponentially distributed with the rate equal to 5 (t_1), or is described by a two stage hypoexponential distribution with the rates of 4 and 2 for stages 1 and 2, respectively (t_2 and t_3 with p_3 linking the stages); the corresponding probabilities are equal to 8020represents the number of available server channels, i.e., one channel in this case. It should be noticed that the jobs after termination of service simply *disappear* from the system.

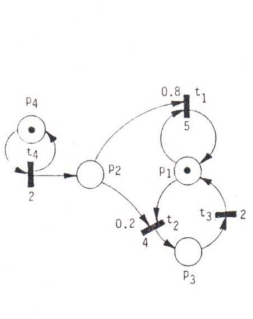


Fig.4. M-timed Petri net $\mathbf{T}_3$.

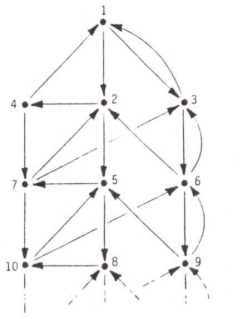


Fig.5. State graph for $\mathbf{T}_3$.

The initial part of the (infinite) set of reachable states is shown in Tab.2, and the corresponding fragment of the state graph is shown in Fig.5. It can be observed that there are 4 *basic* states (s_1, s_2, s_3 and s_4), and that the remaining states are obtained by systematic *replication* of the states s_5, s_6 and s_7.

	m_i	n_i		g_ℓ		
s_i	1 2 3 4	1 2 3 4	t_k	1 2 3 4	s_j	$u(s_i, s_j)$
1	1 0 0 0	0 0 0 1	4	0 1 0 1	2	0.40
				1 0 0 1	3	1.60
2	0 0 0 0	0 1 0 1	2	0 0 1 0	4	4.00
			4	0 0 0 1	5	2.00
3	0 0 0 0	1 0 0 1	1	0 0 0 0	1	5.00
			4	0 0 0 1	6	2.00
4	0 0 0 0	0 0 1 1	3	0 0 0 0	1	2.00
			4	0 0 0 1	7	2.00
5	0 1 0 0	0 1 0 1	2	0 0 1 0	7	4.00
			4	0 0 0 1	8	2.00
6	0 1 0 0	1 0 0 1	1	0 1 0 0	2	1.00
				1 0 0 0	3	4.00
			4	0 0 0 1	9	2.00
7	0 1 0 0	0 0 1 1	3	0 1 0 0	2	0.40
				1 0 0 0	3	1.60
			4	0 0 0 1	10	2.00
8	0 2 0 0	0 1 0 1	2	0 0 1 0	10	4.00
			4	0 0 0 1	11	2.00
...	.....	.....	...	.....	...	.....

Tab.2. Derivation of reachable states for $\mathbf{T}_3$.

This regularity can be used for *folding* of the state space, i.e., for representation of the original infinite space by a finite equivalent reduced space. The balance equations for the states s_i, s_{i+1} and s_{i+2}, for $i = 5, 8, 11, ...$, are as follows (Fig.5 and Tab.2):

$$6 * x_i = 2 * x_{i-3} + x + i + 4 + 0.4 * x_{i+5}$$
$$7 * x_{i+1} = 2 * x_{i-2} + 4 * x_{i+4} + 1.6 * x_{i+5}$$
$$4 * x_{i+2} = 2 * x + i - 1 + 4 * x_i$$

and they are obviously satisfied when x_{j+3} is replaced by $\rho * x_j$, $j = 5, 6, 7, ...$, since ρ can be simply canceled in all equations. This means that the stationary probabilities x_{i+j}, $i = 5, 8, 11, ...$, $j = 0, 1, 2$, have geometric distributions, and the infinite set of equations for $i = 5, 8, 11, ...$ can be replaced by just three equations (nonlinear since ρ is a new unknown):

$$6 * x_5 = 2 * x_2 + \rho * x_6 + 0.4 * \rho * x_7,$$
$$7 * x_6 = 2 * x_3 + 4 * \rho * x_6 + 1.6 * rho * x_7,$$
$$4 * x + 7 = 2 * x_4 + 4 * x_5.$$

Adding the remaining four flow equations for s_1, s_2, s_3 and s_4:

$$2 * x_1 = 5 * x_3 + 2 * x_4,$$
$$6 * x_2 = 0.4 * x_1 + x_6 + 0.4 * x_7,$$
$$7 * x_3 = 1.6 * x_1 + 4 * x_6 + 1.6 * x + 7,$$
$$4 * x_4 = 4 * x_2,$$

and the *normalizing* equation:

$$x_1 + x_2 + x_3 + x_4 + (x_5 + x_6 + x_7)/(1 - \rho) = 1$$

creates a system of eight nonlinear (in fact, quadratic) equations with eight unknowns (for a class of one-place unbounded nets [6] the equations are always quadratic). This system can easily be solved by a series of substitutions and eliminations. The solution is $\rho = 0.6196$, $x_1 = 0.4015$, $x_2 = 0.0419$, $x_3 = 0.1439$, $x_4 = 0.0419$, $x_5 = 0.0234$, $x_6 = 0.0733$ and $x_7 = 0.0444$.

The throughput rate can be used for verification of this derivation. Since in the equilibrium state the throughput rate is obviously equal to the arrival rate, the total throughput rate of this model is equal to 2 jobs per time unit ($r(t_4)$). On the other hand, the throughput rate can be determined from the utilization of the stages within the processor. The transition t_1 fires in s_i, $i = 3, 6, 9, ...$ (Tab.2); the utilization of this stage is thus equal to $x_3 + x_6/(1 - \rho) = 0.3366$, and since the average service time of this stage is equal to 0.2 time units ($r(t_1) = 5$), then the throughput rate of t_1 is $0.3366/0.2 = 1.6829$ jobs per time unit. The second part of the throughput rate can be determined either from t_2 or t_3 (since these two throughput rates must be equal). For t_3 the utilization is equal to $x_4 + x_7/(1-\rho) = 0.1586$, and then the t_3's throughput rate is $0.1586/0.5 = 0.3172$. The total throughput rate is thus 1.6829+0.3172 = 2.0001, which verifies the derivation.

6. CONCLUDING REMARKS

Since the behavior of free-choice M-timed Petri nets, i.e., free-choice Petri nets with exponentially distributed firing times assigned to transitions of a net, is represented by state graphs which are homogeneous continuous-time Markov chains, the stationary probabilities of the states can be obtained by techniques developed for analysis of Markov processes, and then

many performance indices can be evaluated using results of operational analysis [4,5].

The stationary probabilities are obtained by solving a set of simultaneous flow equations which describe the equilibrium conditions of the underlying Markovian process. For open network models the state space is infinite, and then a *folding* of the states is required in order to perform effective evaluations. It has been shown that for a subclass of unbounded M-timed nets such a reduction is rather straightforward, and in fact can be performed automatically by an appropriate computer program. It can be observed, that the same approach can be used for nets with more than one unbounded place (the case of stochastic nets with one unbounded place has been discussed by Florin and Natkin [6]) provided that the state space is a finite *union* of one-place unbounded subspaces.

The formalism presented in this paper may seem rather complicated and thus impractical, however, it should be noticed that it can easily be implemented as a computer program, and then all the detailed state descriptions and state transitions can be completely *invisible* for users. In fact, an interactive program for analysis of timed Petri nets is currently being developed at Memorial University [22], and all examples in this paper have been generated by one of early versions of this program.

Timed Petri nets discussed in this paper are restricted in several ways, however, many restrictions can be removed easily by appropriate modifications of the formalism [18,22]. Moreover, some further *flexibility* is offered by enhanced Petri nets [18,19] in which the set of transitions is subdivided into two classes, timed and immediate transitions, and the firing times are associated with timed transitions only (immediate transitions fire *instantaneously*). This allows not only modelling of arbitrarily complex conditions (e.g., input and output *OR-logic* [12]), but it can also reduce many *intermediate* states which are insignificant for performance analysis.

Acknowledgement

The Natural Sciences and Engineering Research Council of Canada partially supported this research through Operating Grant A8222.

REFERENCES

[1] T. Agerwala, Putting Petri nets to work; IEEE Computer Magazine, vol.12, no.12, pp.85-94, 1979.

[2] M. Ajmone Marsan, G. Conte, G. Balbo, A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems; ACM Trans. on Computer Systems, vol.2, no.2, pp.93-122, 1984.

[3] W. Brauer, W. Reisig, G. Rozenberg (eds.), **Advances in Petri Nets 1986** (vol.1: Petri nets - central models and their properties, vol.2: Petri nets - applications and relationship to other models of concurrency); Proc. of the Advanced Course, Bad Honnef, 1986; Lecture Notes in Computer Science 254 and 255, Springer Verlag 1987.

[4] J.P. Buzen, Fundamental operational laws of computer system performance; Acta Informatica, vol.7, no.2, pp.167-182, 1976.

[5] D. Ferrari, **Computer systems performance evaluation**; Prentice-Hall, Englewood Cliffs NJ, 1978.

[6] G. Florin, S. Natkin, One-place unbounded stochastic Petri nets: ergodic criteria and steady-state solutions; Journal of Systems and Software, vol.1, no.2, pp.103-115, 1986.

[7] M.A. Holliday, M.K. Vernon, A generalized timed Petri net model for performance evaluation; Proc. Int. Workshop on Timed Petri Nets, Torino, Italy, pp.181-190, 1985.

[8] L. Kleinrock, **Queueing systems**, vol.1: Theory, vol.2: Computer applications; J. Wiley & Sons 1975, 1976.

[9] P.M. Merlin, D.J. Farber, Recoverability of communication protocols - implications of a theoretical study; IEEE Trans. on Communications, vol.24, no.9, pp.1036-1049, 1976.

[10] I. Mitrani, Probabilistic modelling of distributed computing systems; in: **Distributed Computing Systems Programme**, D.A. Duce (ed.), pp.139-153, Peregrinus Ltd., London 1984.

[11] M.K. Molloy, Performance analysis using stochastic Petri nets; IEEE Trans. on Computers, vol.31, no.9, pp.913-917, 1982.

[12] J.L. Peterson, **Petri net theory and the modeling of systems**, Prentice-Hall, Englewood Cliffs NJ, 1981.

[13] C. Ramchandani, Analysis of asynchronous concurrent systems by timed Petri nets; Project MAC Technical Report MAC-TR-120, Massachusetts Institute of Technology, Cambridge MA, 1974.

[14] R.R. Razouk, The derivation of performance expressions for communication protocols from timed Petri nets; Computer Communication Review, vol.14, no.2, pp.210-217, 1984.

[15] W. Reisig, **Petri nets - an introduction**; Springer Verlag 1985.

[16] G. Rozenberg (ed.), **Advances in Petri Nets 1987** (Lecture Notes in Computer Science 266); Springer Verlag 1987.

[17] J. Sifakis, Use of Petri nets for performance evaluation; in: **Measuring, modelling and evaluating computer systems**, pp.75-93, North-Holland 1977.

[18] W.M. Zuberek, M-timed Petri nets, priorities, preemptions, and performance evaluation of systems; in: **Advances in Petri Nets 1985** (Lecture Notes in Computer Science 222), G. Rozenberg (ed.), pp.478-498, Springer Verlag 1986.

[19] W.M. Zuberek, Inhibitor D-timed Petri nets and performance analysis of communication protocols; INFOR Journal, vol.24, no.3, pp.231-249, 1986.

[20] W.M. Zuberek, Timed Petri nets in modelling and evaluation of multiprocessor systems; Proc. Int Conf. on Parallel Processing, St. Charles IL, pp.695-698, 1987.

[21] W.M. Zuberek, Stochastic Petri nets and timed Petri nets in modelling and performance evaluation; Proc. APICS Annual Computer Science Conf., Wolfville NS, Canada, pp.66-82, 1987.

[22] W.M. Zuberek, TPNEV, an interactive program for evaluation of timed Petri nets; Technical Report, Department of Computer Science, Memorial University of Newfoundland, St. John's, Canada A1C 5S7 (in preparation).