35th Midwest Symposium on Circuits and Systems; Washington DC, 9-12 August 1992, pp.1576-1580.
Copyright © 1992 IEEE (DOI 10.1109/MWSCAS.1992.271058).

Throughput Analysis in Timed Petri Nets

W.M. Zuberek

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

Abstract

It is shown that the steady–state behaviour of a class of timed Petri nets can be analysed using the concept of throughput and simple rules of operational analysis. Since such analysis is based on the average values of firing times rather than firing time distribution functions, the same approach can be applied to a variety of net models. Throughput analysis uses structural properties of nets only, so it avoids the potential problems of reachability analysis, it can be applied to unbounded nets, however, it does not provide as much information as can be obtained from analysis of the state space. Simple examples of of D–times and M–timed nets are used as an illustration of the proposed approach.

1. INTRODUCTION

For several years Petri nets have been successfully used in modelling [Di82,MF76], validation [BT82] and analysis [Di82,Raz85] of systems of events in which it is possible for some events to occur concurrently, but there are constraints on the occurrence, precedence, or frequency of these occurrences. Multiprocessor and distributed systems, communication networks and data flow architectures are just a few examples of systems for which traditional approaches, developed for analysis of systems with sequential behavior, are simply inconvenient and inadequate. In order to study performance aspects of Petri net models, the duration of activities must also be taken into account and included into model specifications. Timed Petri nets have been introduced by Ramchandani [Ram74] by assigning firing times to the transitions of Petri nets (t–timed nets), and by Sifakis [Si77] by assigning time to places of a net (p–timed nets). It should be noted that the basic difference between these two approaches in not in association of time but in the mechanism of transition firings. In t–timed nets, transition firings are not instantaneous events; a firing occurs in "real-time", i.e., the tokens are removed from input places at the beginning of the firing period, and they are deposited to the output places at the end of this period (sometimes this is called a "three–phase" firing mechanism). In p–timed nets, the "firing time" determines the period of time that tokens must "wait" before firing which is an instantaneous event, as in ordinary nets (so the "firing times" are – in fact – "enabling times"). A simple consequence of this difference is that the "firing" process in a p–timed net can easily be discontinued (using a simple configuration of conflicting transitions), while in a t–timed there in no "access" to tokens once the firing started, so a special type of inhibitor arcs has been proposed to provide this capability. On the other hand, in p–timed nets, the conflict resolution policies cannot be defined independently of timing specifications [C3S89], while in t–timed nets conflict resolution and timing specification are independent aspects. The class of p–timed nets also includes the model proposed by Merlin and Farber [MF76], in which a time threshold and maximum delay were assigned to each transition of a net to allow the incorporation of timeouts into protocol models. Razouk [Raz85] discussed a class of timed Petri nets with enabling as well as firing times (p & t–timed nets), and derived performance expressions for simple communication protocols. Stochastic Petri nets [Mo82,Na80] and generalized stochastic nets [ACB84] are also p–timed nets. And there is a myriad of improved, generalized, augmented, extended and enhanced models [CMT89,Du85,HV85] which belong to one of these basic categories of timed nets.

This paper describes timed Petri nets with inhibitor arcs and "interrupt" arcs which can discontinue initialized firings of transitions, as required in strict modelling of timeouts and preemptions. Similarly as in [Ram74,Zu86,Ho86,Zu88], (deterministic or stochastic) firing times are associated with transitions of a Petri net, and then the "state space" of a net is a discrete–space, discrete–time or continuous–time (depending upon the character of firing times of transitions) homogeneous semi–Markov process. If this process is ergodic, the stationary probabilities of the states can be determined [Ki90], and many performance measures, such as utilization of systems components, average waiting times and turnaround times or average throughput rates, characterizing the steady–state behaviour of the model, can be derived from stationary probabilities of states.

Analysis of net models based on the derivation of the state space is known as the reachability analysis. Although reachability analysis is quite general (e.g., it can easily handle state–dependent routing probabilities as well as state–dependent timing properties), it becomes very inefficient when the state space is large (for some models, the number of states grows exponentially with model parameters, e.g., the token count of the initial marking function, which is known as the "state explosion problem"). Also, reachability analysis is usually restricted to bounded nets. Therefore, other approaches are recently gaining popularity, that are based on structural properties of models, and that avoid the derivation of the state space. Such structural methods of analysis cannot provide as much information as the reachability approach does, quite often, however, all the detailed results of reachability analysis are not really needed, and more synthetic performance measures, that can also be provided by structural approach, are quite satisfactory. Structural approaches can be used to obtain exact or approximate performance measures, e.g., lower and upper performance bounds [BG85,C3S89,CCS89,Mo85].

Structural approach to analysis of net models is similar to operational methods developed for analysis of queueing systems [Bu76,DB78]. It uses only the first moment (the average values) of random variables, and general relationships and laws which do not depend upon probability distribution functions. One of the basic variables of operational analysis is throughput. Many other measures can be obtained from throughputs, for example, the throughput and the maximum service rate of a component determine its utilization factor, which – in turn – is an indicator of systems saturation. The throughput is one of the elements used in the Little formula, etc.

The paper is organized in 3 main sections. Section 2 recalls basic concepts for (extended) free–choice timed Petri nets. Section 3 introduces the concept of throughput and discusses some of its properties. Section 4 shows simple examples of throughput analysis applied to timed nets with deterministic firing times (D–times nets) and exponentially distributed firing times (M–timed nets).

2. TIMED PETRI NETS

This section recalls and formalizes many concepts used in subsequent parts of this paper. It is rather brief since more detailed discussion is provided elsewhere [Zu86,Zu88].

An inhibitor Petri net N is a quadruple N=(P, T, A, B) where:

P is a finite, nonempty set of places,

T is a finite, nonempty set of transitions,

A is a set of directed arcs, $A \subseteq P \times T \cup T \times P$ such that for each transition there is at least one place connected with it,

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

For each place p and each transition t, the input and output sets are defined as follows:

$$Inp(p) = \{t \in T | (t, p) \in A\}, \quad Out(p) = \{t \in T | (p, t) \in A\},$$
$$Inp(t) = \{p \in T | (p, t) \in A\}, \quad Out(t) = \{p \in P | (t, p) \in A\}.$$

and this notation is extended on sets of places and transitions. Moreover, $Inh(t) = \{p \in P | (p, t) \in B\}$ denotes the inhibitor set of t.

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

$\mathcal{N}$ is an inhibitor Petri net, $\mathcal{N} = (P, T, A, B)$,

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 in a net $\mathcal{N} = (P, T, A, B)$.

A transition t is enabled by a marking m iff every input place of this transition contains at least one token and every inhibitor place of t contains zero tokens.

A place p is shared iff it is an input place for more than one transition. In inhibitor nets, a shared place p is guarded iff for each two different transitions t_i and t_j sharing p there exists another place p_k such that p_k is in the input set of one and in the inhibitor set of the other of these two transitions, i.e., no two transitions from the output set of p can be enabled by the same marking.

A shared place p is free–choice (or extended free–choice) iff the input sets and inhibitor sets of all transitions sharing p are identical. An inhibitor net is free–choice iff all its shared places are either free–choice or guarded. Only free–choice nets are considered in this paper since in most cases free–choice nets are sufficient for modelling random events, e.g., random faults in communication networks or any random events described by discrete distributions.

Since the relation of sharing a free–choice place is an equivalence relation in T, it determines a partition of T into a set of free–choice equivalence classes denoted by $Free(T) = \{T_1, T_2, ..., T_k\}$.

Every transition enabled by a marking m can fire. When a transition fires, a token is removed from each of its input places (but not inhibitor places) and a token is added to each of its output places. This determines a new marking in a net, a new set of enabled transitions, and so on.

In timed Petri nets each transition takes a “real time” to fire, i.e., there is a “firing time“ associated with each transition of a net. The firing times can be defined in several ways. In D–timed Petri nets [Zu88] they are deterministic (or constant), i.e., there is a nonnegative number assigned to each transition of a net which determines the duration of transition's firings. In M–timed Petri nets [Zu86] (or stochastic Petri nets [Na80,Mo82,ACB84]), the firing times are exponentially distributed random variables, and the corresponding firing rates are assigned to transitions of a net. In this paper, the firing times associated with transitions of the net are the average values of firing times, so for nets in which firing times are random variables, the distribution function is ignored and only the average (or expected) values of firing times are included into net specifications.

Timed nets discussed in this paper use two types of inhibitor arcs, “proper” inhibitor arcs and “interrupt” arcs [Zu86,Zu88]. “Proper” inhibitor arcs affect the transitions only at the beginning of their firings, as in ordinary nets. Interrupt arcs affect a transition also during its firing; they can “interrupt” firing transitions and preempt the “resources” acquired at the beginning of firing. Interrupt arcs are needed to model preempting scheduling disciplines, to represent properly timeout mechanisms, and to model unreliable processors which can “fail” during processing of user jobs. In some cases such interrupts and preemptions can be represented by inhibitor nets [Zu88], but usually such models (and their behaviour) are unnecessarily complicated. The set of interrupting places of a transition t is denoted $Int(t)$.

A free–choice timed Petri net $\mathcal{T}$ is a triple $\mathcal{T} = (\mathcal{M}, c, f)$ where:

$\mathcal{M}$ is an extended free–choice marked Petri net, $\mathcal{M} = (\mathcal{N}, m_0)$, $\mathcal{N} = (P, T, A, B, C)$, and C is a set of interrupt arcs, $C \subseteq B$,

c is a choice function which assigns a “free–choice” probability to each transition t of the net in such a way that for each free–choice equivalence class $T_i \in Free(T)$ the sum of these probabilities is equal to 1,

f is a firing time function which assigns the nonnegative (average) firing time $f(t)$ to each transition t of the net, $f : T \to \mathbf{R}^{\oplus}$, and $\mathbf{R}^{\oplus}$ denotes the set of nonnegative real numbers.

In ordinary nets (i.e., nets without time), interrupt arcs are equivalent to inhibitor arcs. In extended timed Petri nets, the firing of a transition may be “discontinued” by any one of interrupt arcs associated with this transition. If, during a firing period of a transition t, one of places connected with t by interrupt arcs becomes nonempty (i.e., it receives at least one token), 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.

The behavior of an extended timed Petri net can be represented by a sequence of “states” where each “state” describes the distribution of tokens in places and firing transitions of the net; detailed definitions of states and state transitions for D–timed and M–timed nets are given in [Zu88] and [Zu86], respectively. The states and state transitions can be combined into a graph of reachable states; this graph is a semi–Markov process defined by the timed net $\mathcal{T}$.

A timed net is ergodic iff the semi–Markov process defined by it is ergodic. Only ergodic timed nets are considered in this paper.

Many concepts of structural analysis apply to timed nets as well as their subnets; moreover, there are important relationships between properties of nets and their subnets.

A timed net $\mathcal{T}_i = (((P_i, T_i, A_i, B_i, C_i), m_i), c_i, f_i)$ is a P_i–implied subnet of a net $\mathcal{T} = (((P, T, A, B, C), m_0), c, f)$ iff

$$P_i \subseteq P,$$
$$T_i = \{t \in T \mid \exists (p \in P_i)\ (t, p) \in A \vee (p, t) \in A\},$$
$$A_i = A \cap (P_i \times T_i \cup T_i \times P_i),$$
$$B_i \subseteq B \cap (P_i \times T_i),$$
$$C_i \subseteq C \cap (P_i \times T_i),$$
$$m_i = m_0 \mid P_i,$$
$$c_i = c \mid T_i,$$
$$f_i = f \mid T_i.$$

P_i–implied subnet contains all arcs which are incident with places in P_i in the original net and transitions which are incident with these arcs.

The set of all subnets of the net $\mathcal{T}$ is denoted $Sub(\mathcal{T})$.

3. THROUGHPUT

Intuitively, throughput of a place p in a timed net $\mathcal{T}$, $\theta_{\mathcal{T}}(p)$, is equal to the average number of tokens entering p in a unit time, or leaving p (or t) in a unit time; in the steady–state of the net, the average numbers of tokens entering and leaving p must be equal since no "accumulation" of tokens can occur. Similarly, throughput of a transition t in a net $\mathcal{T}$, $\theta_{\mathcal{T}}(p)$, is equal to the average number of transition's firings in a unit time. It should be noted that the throughput of a transition does not depend upon the number of incoming or outgoing arcs; in the steady–state, the (average) numbers of tokens removed from each of transitions input places and deposited to each of its output places in a unit time are the same.

More formally, the throughput of a timed net $\mathcal{T}$ is defined as a function $\theta : P \cup T \rightarrow \mathbf{R}^{\oplus}$ which assigns a nonegative number to each place and each transition of the net in such a way that:

$$\forall(x \in P \cup T)\ \theta(x) = \lim_{i\to\infty} \frac{i}{\tau_i(x)}$$

where $\tau_i(x)$ denotes the time instant at which the i-th consecutive token enters (or leaves) the place x or at which the transition x initiates (or terminates) its i-th firing.

Property 1: It follows immediately from the definition of throughput that:

- the throughput of a place p is equal to the sum of throughputs of its input transitions as well as the sum of throughputs of its output transitions:
  $\forall(p \in P)\ \theta(p) = \sum_{t_i \in Inp(p)} \theta(t_i) = \sum_{t_j \in Out(p)} \theta(t_j)$,
- for each non–shared place p, the throughput of p's output transition is equal to the throughput of p:
  $\forall(p \in P)\ Out(p) = \{t\} \Rightarrow \theta(t) = \theta(p)$,
- for each free–choice place p, throughputs of p's output transitions are determined by the choice function c:
  $\forall(T_i \in Free(T))\ \forall(p \in Inp(T_i))\ \forall(t \in T_i)\ \theta(t) = c(t)\theta(p)$,
- for each place p that is shared by two transitions t_i and t_j, and which is guarded by a place p_k:
  $p_k \in Inp(t_i) \Rightarrow \theta(t_i) = \theta(p_k)$,
  $p_k \in Int(t_j) \Rightarrow \theta(t_j) = \theta(p) - \theta(p_k)$.

An elementary net is a net in which there is exactly one input place and exactly one output place for each transition of the net, and one input transition and one output transition for each place of the net. In other words, the (directed) graph of an elementary net is a (simple) cycle.

It follows immediately from the property 1 that in elementary nets the throughputs of all transitions and all places are the same.

Property 2: For an elementary net $\mathcal{T}$:

$$\forall(x \in P \cup T)\ \theta(x) = \frac{\sum_{p\in P} m_0(p)}{\sum_{t\in T} f(t)}$$

It should be observed, that in an elementary net, all tokens will traverse the net without any "delays", i.e., any termination of a firing immediately starts a firing of another transition. Consequently, the average time of a complete traversal of the net by a single token is equal to Δ, the sum of the average firing times of all transitions, $\Delta = \sum_{t\in T} f(t)$. For $k = \sum_{p\in P} m_0(p)$ tokens independently traversing the net and for large values of i, $E(\tau_i(t))$ can be approximated by $i\Delta/k$, so:

$$\theta(t) = \lim_{i\to\infty} \frac{i}{\tau_i} = \frac{k}{\Delta} = \frac{\sum_{p\in P} m_0(p)}{\sum_{t\in T} f(t)}$$

Property 3: If a transition t (or a place p) belongs to a subnet $\mathcal{T}_i = (((P_i, T_i, A_i), m_i), c_i, f_i)$ of the net $\mathcal{T}$, the throughput $\theta_{\mathcal{T}_i}(t)$ (or $\theta_{\mathcal{T}_i}(p)$) in the subnet $\mathcal{T}_i$ is greater than or equal to the throughput $\theta_{\mathcal{T}}(t)$ (or $\theta_{\mathcal{T}}(p)$) in the net $\mathcal{T}$

$$\forall(\mathcal{T}_i \in Sub(\mathcal{T}))\ \forall(x \in T_i \cup P_i)\ \theta_{\mathcal{T}_i}(x) \geq \theta_{\mathcal{T}}(x)$$

The property immediately follows from the observation that a transition in the net $\mathcal{T}$ can fire only if all its input places received their tokens, so $\mathcal{T}$ may introduce some "delays" which do not exist in the subnet $\mathcal{T}_i$.

4. EXAMPLES

Example 1: The D–timed net shown in Fig.1 is a model of a very simple protocol in which messages are exchanged between a sender (place p_1) and a receiver (place p_3), and each received message is confirmed by an acknowledgement sent back to the sender (in the loop $p_1, t_1, p_2, t_2, p_3, t_4, p_1$). The subnet (p_7, t_7) models a "source" that generates messages, and p_6 is simply a buffer of messages waiting for transmission.

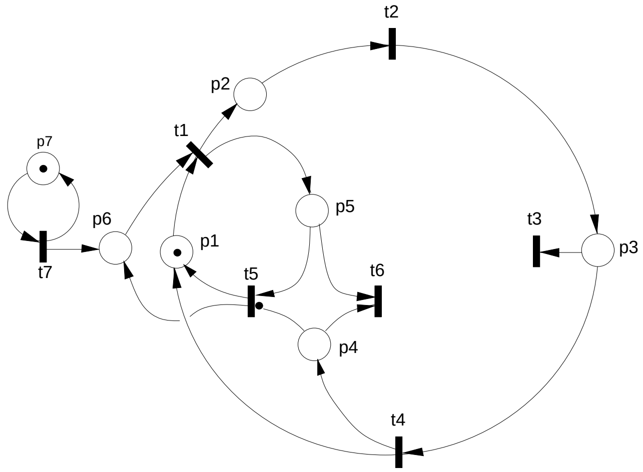

Fig.1. Petri net model of a simple protocol.

There is a nonzero probability that the system can lose (or distort) a message or an acknowledgement; the place p_3 is a free–choice place, and the transition t_3 models a message/acknowledgement "sink"; the probability associated with t_3, $c(t_3)$, represents thus the probability of losing a message or an acknowledgement (or shortly a "token") in the system. A "timeout" is used to recover from lost "tokens". It works in the following way. An event of "sending a message" is modelled by the transition t_1. When it fires, single tokens are deposited in p_2 (a "message") and in p_5 (a "timeout"). A token in p_5 immediately starts a firing of the "timeout" transition t_5 (since p_5 is empty). The firing time associated with t_5 is large enough to allow the transfer of a message and an acknowledgement. If there is no loss of tokens, i.e., if t_4 is selected for firing (according to its probability), the transition t_4 will finish its firing before t_5, and then a token in the place p_4 interrupts and cancels the timeout (i.e., the firing of t_5), the "timeout" token is returned to p_5, and then t_6 fires and removes the tokens from p_4 and p_5 (t_6 is another token "sink"). If, however, a message or acknowledgement has been lost (i.e., if t_3 has been selected for firing instead of t_4), the timeout t_5 terminates its firing without interruption, and regenerates the "lost" token in p_1 and the "message" in p_6, so the message can be retransmitted to the receiver.

It should be observed that the place p_6 is potentially unbounded; it is unbounded in the ordinary net, so boundedness of p_6 in the timed net depends upon the functions m_0, c and f.

For the steady–state, the throughputs can be determined by simple structural relations based on property 1:

$$\begin{array}{ll}
\theta(t_1) = \theta(p_1) = \theta(p_6) & \theta(p_1) = \theta(t_4) + \theta(t_5) \\
\theta(t_2) = \theta(p_2) & \theta(p_2) = \theta(t_1) \\
\theta(t_3) = q\ \theta(p_3) & \theta(p_3) = \theta(t_2) \\
\theta(t_4) = (1-q)\ \theta(p_3) & \theta(p_4) = \theta(t_4) \\
\theta(t_5) = \theta(p_5) - \theta(t_6) & \theta(p_5) = \theta(t_1) \\
\theta(t_6) = \theta(p_4) & \theta(p_6) = \theta(t_7) + \theta(t_5) \\
\theta(t_7) = \theta(p_7) & \theta(p_7) = \theta(t_7)
\end{array}$$

It can be observed that there is a simple cycle described by the relations:

$$\begin{array}{l}
\theta(t_7) = \theta(p_7) \\
\theta(p_7) = \theta(t_7)
\end{array}$$

so p_7 implies a simple subnet that includes p_7 and t_7 only, and $\theta(p_7) = \theta(t_7) = m_0(p_7)/f(t_7)$ (property 2), and then the (symbolic) solution is:

$$\begin{array}{l}
\theta(t_1) = \theta(t_2) = \frac{m_0(p_7)}{(1-q) f(t_7)} \\
\theta(t_3) = \theta(t_5) = \frac{q m_0(p_7)}{(1-q) f(t_7)} \\
\theta(t_4) = \theta(t_6) = \theta(t_7) = \frac{m_0(p_7)}{f(t_7)} \\
\theta(p_1) = \theta(p_2) = \theta(p_3) = \theta(p_5) = \theta(p_6) = \frac{m_0(p_7)}{(1-q) f(t_7)} \\
\theta(p_4) = \theta(p_7) = \frac{m_0(p_7)}{f(t_7)}
\end{array}$$

It should be noted (again) that this solution is meaningless when the net is not in the steady–state, i.e., is unbounded. Validity of the solution can be checked by comparing throughputs with the maximum throughputs that correspond to the "boundary" of the boundedness region; these "boundary" values can be obtained by analysing the net with the source subnet (p_6, p_7, t_7 and all incident arcs) removed.

Example 2: The M–timed net shown in Fig.2 is a typical model of a memory–constrained multiprogramming system [La84]. p_1 and t_1 model the source of jobs which arrive with the average rate of $1/f(t_1)$ jobs per time unit. p_2 is the memory queue for jobs that must wait for entering the execution stage; the system limits the number of jobs that receive allocation of memory and can execute concurrently (this limit is also called the level of multiprogramming), so only certain number of jobs can enter into the central server.

The memory constraint is represented by the place p_6 with its initial marking $m_0(p_6)$; each job entering the central system reduces the number of tokens in p_6, and each job leaving the system (transition t_6) increases the number of tokens in p_6. p_3 and t_3 represent the central server with its waiting queue p_8, while p_4 and t_4 model a disk server with its queue p_5; the number of (identical) processors in the central server is determined by the initial marking of the place p_3, and the number of (identical) disk drives by the initial marking of the place p_4. The place p_7 is a free–choice place with two choices, termination of job execution (transition t_6 with probability q) or continuation of execution (transition t_5 with probability $(1-q)$). The probability q can be determined on the basis of the average number of disks requests per job execution; if each job requests K disk operations on average, $q = 1/(1+K)$.

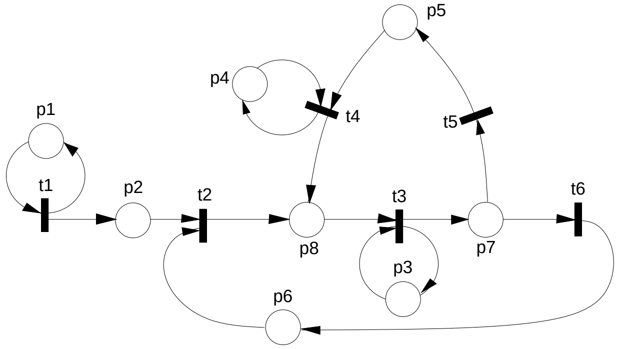

Fig.2. Petri net model of a multiprogramming system.

It should be observed that the place p_2 is unbounded because, for stochastic firing times, there is no limit on the number of firings of t_1 before termination of any other firing in the net. Consequently, the net is unbounded.

For the steady–state of this net, the throughput relations are as follows:

$$\begin{array}{ll}
\theta(t_1) = \theta(p_1) & \theta(p_1) = \theta(t_1) \\
\theta(t_2) = \theta(p_2) = \theta(p_6) & \theta(p_2) = \theta(t_1) \\
\theta(t_3) = \theta(p_3) = \theta(p_8) & \theta(p_3) = \theta(t_3) \\
\theta(t_4) = \theta(p_4) = \theta(p_5) & \theta(p_4) = \theta(t_4) \\
\theta(t_5) = (1-q)\ \theta(p_7) & \theta(p_5) = \theta(t_5) \\
\theta(t_6) = q\ \theta(p_7) & \theta(p_6) = \theta(t_6) \\
 & \theta(p_7) = \theta(t_3) \\
 & \theta(p_8) = \theta(t_2) + \theta(t_4)
\end{array}$$

Because of the relations:

$$\begin{array}{l}
\theta(t_1) = \theta(p_1) \\
\theta(p_1) = \theta(t_1)
\end{array}$$

p_1 implies a simple subnet that includes p_1 and t_1 only, so $\theta(p_1) = \theta(t_1) = m_0(p_1)/f(t_1)$, and then the solution is:

$$\begin{array}{l}
\theta(t_1) = \theta(t_2) = \theta(t_6) = \frac{m_0(p_1)}{f(t_1)} \\
\theta(t_3) = \frac{m_0(p_1)}{q\ f(t_1)} \\
\theta(t_4) = \theta(t_5) = \frac{(1-q)\ m_0(p_1)}{q\ f(t_1)} \\
\theta(p_1) = \theta(p_2) = \theta(p_6) = \frac{m_0(p_1)}{f(t_1)} \\
\theta(p_3) = \theta(p_7) = \theta(p_8) = \frac{m_0(p_1)}{q\ f(t_1)} \\
\theta(p_4) = \theta(p_5) = \frac{m_0(p_7)}{f(t_7)}
\end{array}$$

As before, this solution is valid only for the steady–state of the net, which can be verified by checking the utilization factors of transitions. The utilization factor $u(t)$ is equal to the ratio of throughput of t to the maximum firing rate, and the maximum firing rate can be determined from the loops on transitions t_1, t_3 and t_4 (property 3), so

$$\begin{array}{l}
u(t_1) = \theta(t_1) \frac{f(t_1)}{m_0(p_1)} = 1 \\
u(t_3) = \frac{m_0(p_1)}{q\ f(t_1)} \frac{f(t_3)}{m_0(p_3)} \\
u(t_4) = \frac{(1-q)\ m_0(p_1)}{q\ f(t_1)} \frac{f(t_4)}{m_0(p_4)}
\end{array}$$

and the remaining transitions have unlimited maximum firing rates.

The values of utilization factors beyond the interval [0,1] indicate that the requirement of steady–state behaviour is not satisfied, so the solution is not valid. For example, for $m_0(p_1) = m_0(p_3) = 1$,

$f(t_1) = 1$, $f(t_3) = 0.1$ and $q = 0.01$, $u(t_3) = 10$ which clearly indicates that the net is not ergodic; actually, it indicates that the processor represented by t_3 should be 10 times faster to handle the service demand. The ergodicity condition for this example is

$$\max(u(t_3), u(t_4)) \leq 1$$

so the maximum arrival rate, or the source throughput $\theta_{\max}(t_1)$, that this central server can handle is:

$$\max\left(\theta_{\max}(t_1)\frac{1}{q}\frac{f(t_3)}{m_0(p_3)}, \theta_{\max}(t_1)\frac{1-q}{q}\frac{f(t_4)}{m_0(p_4)}\right) = 1$$

or equivalently:

$$\theta_{\max}(t_1) = \min\left(\frac{m_0(p_3)}{q\ f(t_3)}, \frac{q\ m_0(p_4)}{(1-q)f(t_4)}\right)$$

Many other results can be obtained in a similar way.

5. CONCLUDING REMARKS

It has been shown that, for a class of timed Petri nets, steady–state throughput analysis can be performed on the basis of structural properties of the net (and – of course – the m_0, c and f functions). For stochastic firing times, throughput analysis uses the average values of firing times only, so the exact firing time distribution function may not even be known, actually. Furthermore, the approach can be used to nets with different distribution functions associated with different transitions of the same nets; for example, some transitions may have deterministic firing times while other transitions may use stochastic firing times.

Timed nets used in the examples correspond to open networks of the queueing theory [Ki90]. It is believed that the proposed approach can be extended to closed network models as well; for example, marked graphs [Mu89] need only a rather straightforward extension of some concepts, but other classes of nets may be more difficult to deal with.

The proposed approach can be used for analysis of unbounded nets, but the ergodicity condition is strictly required. As shown in the examples, quite often the solution can be "verified" by checking additional requirements (e.g., utilization factors or performance bounds).

An attractive aspect of throughput analysis is the possibility of obtaining the solution in a symbolic form rather than as a numerical value. Different combinations of parameter values, sensitivity of the results with respect to different parameters or functional dependencies can be investigated very conveniently when symbolic solutions are available.

The proposed throughput analysis can be automated quite easily and, in fact, can be incorporated into more general tools for analysis Petri net models as one of alternative analysis methods.

Acknowledgement

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

References

[ACB84] 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.

[BT82] G. Berthelot, R. Terrat, "Petri net theory for the correctness of protocols"; IEEE Trans. on Communications, vol.30, no.12, pp.2497–2505, 1982.

[BG85] S.C. Bruell, S. Ghanta, "Throughput bounds for generalized stochastic Petri net models"; Proc. Int. Workshop on Timed Petri Nets, Torino, Italy, pp.250–261, 1985.

[Bu76] J.P. Buzen, "Fundamental operational laws of computer system performance"; Acta Informatica, vol.7, no.2, pp.167–182, 1976.

[C3S89] J. Campos, G. Chiola, J.M. Colom, M. Silva, "Tight polynomial bounds for steady–state performance of marked graphs"; Proc. Int. Workshop on Petri Nets and Performance Models, Kyoto, Japan, pp.200–209, 1989.

[CCS89] J. Campos, G. Chiola, M. Silva, "Properties and steady–state performance bounds for Petri nets with unique repetitive firing count vector"; Proc. Int. Workshop on Petri Nets and Performance Models, Kyoto, Japan, pp.210–220, 1989.

[CMT89] G. Ciardo, J. Muppala, K. Trivedi, "SPNP - stochastic Petri net package"; Proc. Int. Workshop on Petri Nets and Performance Models, Kyoto, Japan, pp.142–151, 1989;

[DB78] P.J. Denning, J.P. Buzen, "The operational analysis of queueing network models"; ACM Computing Surveys, vol.10, no.3, pp.225-261, 1978.

[Di82] M. Diaz, "Modeling and analysis of communication and cooperation protocols using Petri net based models"; Computer Networks, vol.6, no.6, pp.419–441, 1982.

[Du85] J.B. Dugan, A. Bobbio, G. Ciardo, K. Trivedi, "The design of a unified package for the solution of stochastic Petri net models"; Proc. Int. Workshop on Timed Petri Nets, Torino, Italy, pp.6–13, 1985.

[Ho86] M.A. Holliday, "Deterministic time and analytical models of parallel architectures"; Ph.D. Thesis, Computer Science Department, University of Wisconsin - Madison, Technical Report #652, 1986.

[HV85] 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.

[Ki90] P.J.B. King, "Computer and communication systems performance modelling"; Prentice–Hall 1990.

[La84] E.D. Lazowska, J. Zahorjan, G.S. Graham, K.C. Sevcik, "Quantitative system performance"; Prentice-Hall 1984.

[MF76] 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.

[Mo82] M.K. Molloy, "Performance analysis using stochastic Petri nets"; IEEE Trans. on Computers, vol.31, no.9, pp.913–917, 1982.

[Mo85] K. Molloy, "Fast bounds for stochastic Petri nets"; Proc. Int. Workshop on Timed Petri Nets, Torino, Italy, pp.244–249, 1985.

[Mu89] T. Murata, "Petri nets: properties, analysis and applications"; Proceedings of IEEE, vol.77, no.4, pp.541–580, 1989.

[Na80] S. Natkin, "Les réseaux de Petri stochastique"; Thèse de Docteur Ingenieur, CNAM, Paris, France, 1980.

[Ram74] 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.

[Raz85] 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.

[Si77] J. Sifakis, "Use of Petri nets for performance evaluation"; in: "Measuring, modelling and evaluating computer systems", pp.75–93, North–Holland 1977.

[Zu86] 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.

[Zu88] W.M. Zuberek, "D–timed Petri nets and modelling of timeouts and protocols"; Transactions of the Society for Computer Simulation, vol.4, no.4, pp.331–357, 1988.