The derivation of performance expressions for communication protocols from timed petri net models — Source link

Rami R. Razouk

Institutions: University of California, Irvine

Published on: 01 Jun 1984 - ACM Special Interest Group on Data Communication

Topics: Stochastic Petri net, Petri net, Process architecture, Communications protocol and Correctness

The Derivation of Performance Expressions for Communication Protocols from Timed Petri Net Models

by

Rami R. Razouk

TR# 211

ABSTRACT

Petri Net models have been extended in a variety of ways and have been used to prove the correctness and evaluate the performance of communication protocols. Several extensions have been proposed to model time. This work uses a form of Timed Petri Nets and presents a technique for symbolically deriving expressions which describe system performance. Unlike past work on performance evaluation of Petri Nets which assumes a priori knowledge of specific time delays, the technique presented here applies to a wide range of time delays so long as the delays satisfy a set of timing constraints. The technique is demonstrated using a simple communication protocol.

Technical Report #211

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717

October 31, 1983

The Derivation of Performance Expressions for Communication Protocols from Timed Petri Net Models

Rami R. Razouk

ABSTRACT

Petri Net models have been extended in a variety of ways and have been used to prove the correctness and evaluate the performance of communication protocols. Several extensions have been proposed to model time. This work uses a form of Timed Petri Nets and presents a technique for symbolically deriving expressions which describe system performance. Unlike past work on performance evaluation of Petri Nets which assumes a priori knowledge of specific time delays, the technique presented here applies to a wide range of time delays so long as the delays satisfy a set of timing constraints. The technique is demonstrated using a simple communication protocol.

Introduction

The issue of specification and verification of communication protocols has drawn a great deal of attention. Much of the work in this area has focused on "correctness" issues and has ignored important "performance" issues. With few exceptions, correctness proofs for communication protocols have either ignored timing or have attempted to prove that correctness is independent of time. The first approach raises serious questions about the usefulness of the proofs, while the latter approach generally makes proofs more difficult. In some cases where timing is critical to the correct operation of a system, time-independent proofs may be impossible. The search continues for a specification method which can bridge the "gap" between correctness and performance.

Several approaches are currently being used to specify communication protocols. Semi-formal approaches based on structured English text are quite popular but are ambiguous and do not lend themselves well to formal analysis. Formal approaches based on axiomatic specifications [YGH 82], temporal logic [HaOw 80, SaSc 82], extended finite state machines [BocG 80, SiKa 82], and transmission grammars [TeLi 80] are useful in proving correctness issues fail to address timing and performance issues. Petri-Nets, on the other hand, have been used to specify and prove the correctness of protocols [BeMe 83, BeTe 82, RaEs 80, SymF 80] and have also been used to analyze performance [MolM 81, RaHo 80, RamC 74, RaPh 83, SifK 77, VSE 83, ZubW80]. It is this important difference which motivates our work on Petri

This paper presents an approach to performance analysis which derives performance expressions from Timed Petri Nets. The technique is based on Timed Reachability Graphs first proposed by Zuberek [ZubW 80] and extended in [RaPh 83]. While this paper focuses on performance we expect that these graphs can be used to prove correctness much as the work by Berthomieu [BeMe 83] suggests.

Section 1 of the paper reviews the version of Timed Petri Nets we use in this work, and discusses how these nets differ from Time Petri Nets (introduced by Merlin and Farber [MeFa 76] and used in [BeMe 83]) and other time-extensions to Petri Nets. Section 2 of the paper discusses a technique first proposed by Zuberek for analyzing the performance of Timed Petri Net models . Section 3 presents a more generalized approach which can be used to symbolically derive performance expressions in nets where specific time delays are not known, but where a set of timing constraints is known. Section 4 demonstrates the technique by applying it to a simple communication protocol.

1. Timed Petri Nets

In the description of Timed Petri Nets which follows, we use some notation which requires clarification. In particular:

$\mu(p)$ is used to indicate the number of tokens in place p in marking μ.

$\#(p, I(t))$ is used to indicate the number of occurrences of place p in the input bag of place t.

A Timed Petri Net Γ is defined as follows:

$$\Gamma = (P, T, I, O, E, F, \mu_0)$$

Where P is the set of *places* in the net.

T is the set of *transitions* in the net.

$I : T \rightarrow P^*$ is the *input* function for transitions

$O : T \rightarrow P^*$ is the *output* function for transitions

$E : T \rightarrow R$ is the *enabling time* function for transitions (we assume non-negative real numbers).

$F : T \rightarrow R$ is the *firing time* function for transitions (we assume non-negative real numbers).

$\mu_0 : P \rightarrow N$ is the *initial marking* for the net (non-negative intergers).

The description of a Petri Net only specifies the static relationship between places and transitions. The dynamic behavior of the net depends of the definition of the semantics of the underlying model. Below is an informal definition of the semantics of Timed Petri Nets.

Enabling Rules

We assume the normal enabling rules for Petri Nets:

A transition $t_j \in T$ in a marked Petri Net Γ with marking μ is enabled if and only if $\forall p_i \in P$,

$$\mu(p_i) \geq \#(p_i, I(t_j))$$

Firing Rules

A transition $t_j \in T$ in a marked Petri Net Γ with marking μ is *firable* at time τ if and only if it is continuously enabled during the interval $\tau - E(t_j)$ to τ. In our version of Timed Petri Nets we assume that when a transition becomes *firable*, it *must* begin firing at that instant of time (unless disabled by the firing of a conflicting transition). When a transition begins firing, it absorbs tokens from its input places.

If a transition begins firing at time τ it is said to be *firing* during the interval τ to $\tau + F(t_j)$. At time $\tau + F(t_j)$ the transition finishes firing and produces tokens on its output places.

Conflict Sets

Petri Net models use places to model flow of control as well as contention for resources. Contention for resources is usually modeled using places which are in the input bag of two or more *conflicting* transitions. With only a single token in such a place, only one of a set of conflicting transitions can fire. In the basic Petri Net model, the choice is non-deterministic. In order to evaluate the performance of a Timed Petri Net it is necessary to determine the probability that each of a set of conflicting transitions will fire. In this version of Timed Petri Nets, each net must be partitioned into disjoint sets of conflicting transitions. These sets are referred to as *conflict sets* and are formally defined as follows. Every transition t_i belongs to exactly one conflict set C such that:

$$C = \{ t_j \mid I(t_i) \cap I(t_j) \neq \emptyset \}$$

The above definition implies that conflict sets cannot overlap. With each transition t_i in a conflict set, the user must define a relative firing frequency f_i. A firing frequency of zero indicates that other transitions in the same conflict set, if firable, always have priority. When a state is reached where one or more transitions in a conflict set is firable (referred to as a *decision* state, or decision node), the probability of firing a firable transition t_i is calculated as follows:

$$\frac{f_i}{\sum_{j \mid t_j \in C \wedge t_j \text{ is firable}} f_j}$$

If only *one* transition is firable, then the probability of firing it is 1, regardless of firing frequency.

In order to use conflict sets to calculate firing probabilities as shown above, limitations must be placed on the Petri Net models. In particular, we require that firing a transition disable all conflicting transitions. In other words, only one transition can fire from any conflict set at any instant of time. Since we consider a transition to be in conflict with itself, this requirement eliminates from consideration nets which allow multiple *firings* (at the same instant) of a transition. This is a property which is slightly less restrictive that the T-safeness assumption in [BeTe 82] (T-safeness does not admit nets with multiply- *enabled* transitions).

Figure 1 shows an example of a Timed Petri Net. For the sake of readability we show the graphical representation of the net. This particular net models a simple protocol based on unnumbered messages and acknowledgements. In this protocol the sender sends a packet (transition t_2) and waits for an acknowledgement. A timeout (transition t_3) is used to recover from lost packets. The receiver waits for a message and sends an acknowledgement immediately (transition t_6). The medium can lose packets (transition t_4) and acknowledgements (transition t_9). In its design, this protocol assumes that timeouts are only triggered if the packet or its acknowledgement have been lost. We assume that the receiver can detect a duplicate message, but that the sender cannot detect a duplicate acknowledgement. This is a trivial protocol, which can be easily extended to be more robust by using alternating bits for message and acknowledgement sequencing. For the sake of brevity, we have opted for the simpler, less robust, protocol.

The net contains 3 conflict sets containing more than one transition:

1. $\{t_4: 0.05, t_5: 0.95\}$, modeling a 5% chance of losing packets.

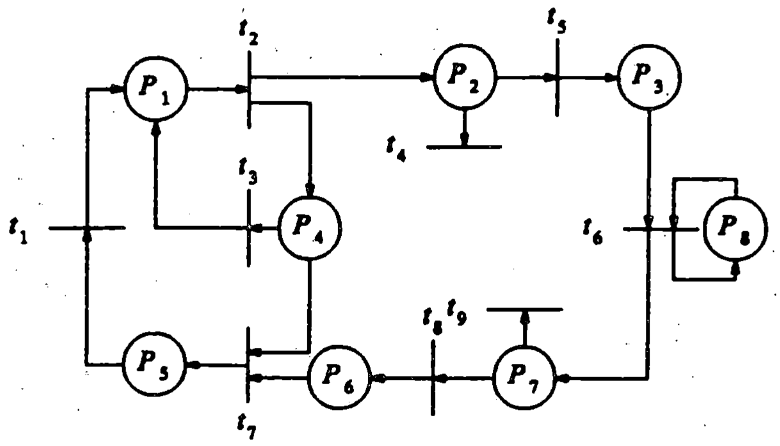


Figure 1a. Model of Simple Protocol

Transition	Enable Time (milliseconds)	Firing Time (milliseconds)
t_1	0	1
t_2	0	1
t_3	1000	1
t_4	0	106.7
t_5	0	106.7
t_6	0	13.5
t_7	0	13.5
t_8	0	106.7
t_9	0	106.7

Figure 1b. Enabling and Firing Times

2. **$\{t_3$: 0, t_7: 1$\}$, modeling the fact that t_7 has priority over t_3 whenever they are both firable.**

3. **$\{t_8$: 0.95, t_9: 0.05$\}$, modeling a 5% chance of losing acknowledgments.**

Other Time Extensions

Many other time extensions to the basic Petri Net model have been proposed. Ramchandani [RamC 74] first proposed the use of transition delays. These delays are equivalent to firing times in the model described above. Sifakis [SifJ 77] proposed associating delays with places instead of transitions. Coolahan and Roussopoulos [CoRo 83] recently used a similar approach in their work on "time-driven" systems. Associating delays with places instead of transitions does not increase the power of the model, but adheres to the instantaneous firing rules of the basic Petri Net model. In fact, transition delays and place delays are equivalent since a simple procedure can be used to translate one into the other. Molloy [MolM 81] proposed the use of transition delays with exponential distributions in order to use Markov Chain analysis. Vernon [VSE 83] used delays similar to transition delays in work using the UCLA Graph Model of Behavior, a form of extended Petri Nets.

Merlin and Farber [MeFa 76] proposed a time extension which is significantly different than other time extensions. In their *Time Petri Net* model two values (Min/Max times) are used to define a *range* of delays for each transition. The model retains the instantaneous firing rules of the basic Petri Net model by allowing tokens to remain on the input places during the transition delay. This approach proved particularly well suited for modeling timeouts in communication protocols. This model of time has been used by Berthomieu [BeMe 83] in developing techniques to prove the correctness of protocols. On the surface, Min times appear similar to our enabling times, and Max times appear similar to firing times. The similarity arises from the use of Min times to model timeouts. The only use of enabling times in our model is also to model timeouts. The similarity is, however, superficial. Min/Max times can be used to model ranges of delays, while firing times can only model one specific delay value. As a result, there is more flexibility in Merlin and Farber's model than in the model used in this paper. In their model, after Min times have elapsed, transitions *may* fire. When they do fire, they fire instantaneously. On the other hand, after enabling times have elapsed, transitions *must* begin firing. In Merlin and Farber's model the enabling tokens remain on the input places, while in our model they are absorbed as soon as the transition begins firing and do not reappear until the transition finishes firing. The added flexibility of the Merlin and Farber model does, however, make performance analysis more difficult.

The difference is illustrated in Figure 2. Figure 2a shows a Timed Petri Net with enabling and firing times. Figure 2b shows an equivalent Time Petri Net. Since Timed Petri Nets require that a transition fire as soon as it becomes firable, an equivalent Time Petri Net must contain added transitions where the Min time and the Max time are both equal to the enabling time. These transitions ensure that tokens on the inputs of the firable transitions are absorbed immediately. The fixed delays in the Timed Petri Nets can also be modeled with transitions with equal Min and Max times.

Should Figure 2a be interpreted as a Time Petri Net (Min times as enabling times and Max times as firing times), it would be possible for a token arriving at place P_2 at time 2 to cause transition t_2 to fire instantaneously. It is therefore possible for transition t_1 to be prevented from firing even after its Min time has expired. So long as activities which are being modeled by transitions are instantaneous, this view of time is reasonable. If, however, transitions model activities which consume time (they have a beginning and an end) then our Timed Petri Nets are more suitable. It is possible to model activities which consume time using Time Petri Nets by using two transitions to model the beginning and the end of each activi-

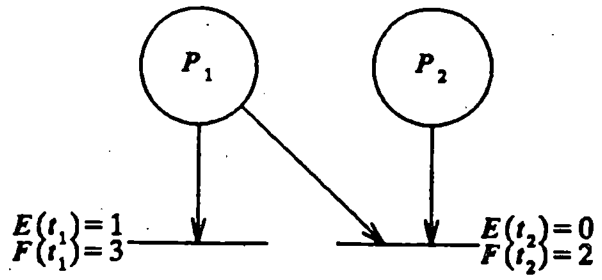


Figure 2a. Timed Petri Net

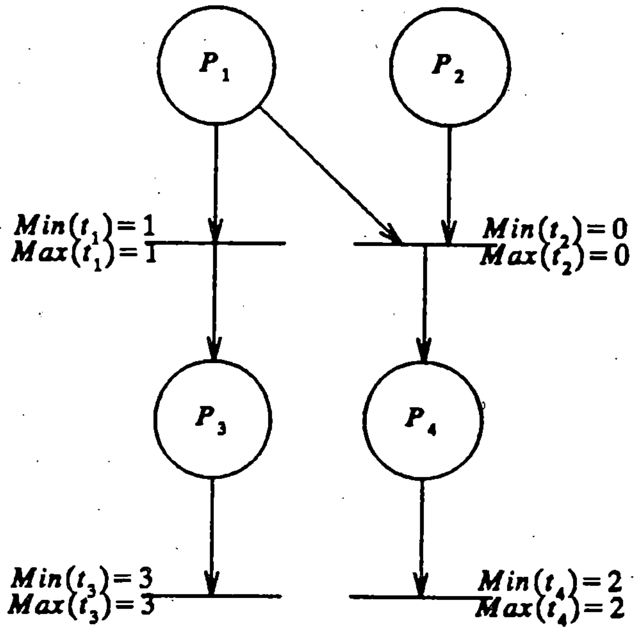


Figure 2b. Equivalent Time Petri Net

ty. While the event is occurring, the token remains on a place between the two transitions.

In the final analysis, the difference between the two models appears to be one of *intended use*. Timed Petri Nets are used to model fixed delays, and have been extended to model timeouts. Time Petri Nets are used to model ranges of delays and use the same mechanism to model timeouts. It is this dual usage of Min times which troubles us. Models such as the one in figure 2a (interpreted as a Time Petri Nets), do not arise often in practical situations. It can therefore be argued that the dual usage of Min times is reasonable. We prefer to view the two mechanisms as distinct. If ranges of delays must be modeled, our approach would be to extend firing times to include time ranges, but to retain enabling times to model timeouts.

2. Analysis of Timed Petri Nets

The basis of our technique for deriving performance expressions is the use of *Timed Reachability Graphs*. Reachability Graphs are cyclic directed graphs which enumerate all the reachable states of a system, and which describe all possible transitions between states. Reachability graphs for un-timed Petri Nets have been used extensively to prove properties related to correctness (such as deadlock-freeness). Each node in such reachability graphs consists of a marking describing the state of the net. In [ZubW 80], Zuberek extended reachability graphs to include time. This was accomplished by adding a time component to each state. A node in such Timed Reachability Graphs includes a marking and a list of *remaining firing times* (RFT) of transitions. This additional component accounts for transitions which are in the process of firing. In order to accommodate the version of Timed Petri Nets presented above, Timed Reachability Graphs were extended in [RaPh 83] to include the notion of enabling times. In such graphs, the state of a system is characterized by:

1. A marking, indicating the distribution of tokens on places.

2. A vector of remaining enabling times (RET), indicating the amount of time each enabled transition must remain enabled before it becomes firable.

3. A vector of remaining firing times (RFT), indicating the amount of time each firing transition must continue to fire before it terminates and places tokens on its outputs.

The reachability graph is constructed by recursively calculating the successors of every reachable state, starting from some initial state. Figure 3 shows a detailed procedure for calculating successors of a state. This procedure is explained in detail in [RaPh 83] and is only briefly outlined below.

There are two possibilities which arise when calculating the immediate successor(s) of a particular state S. In the first case, one or more transitions are firable. Since the act of beginning to fire is instantaneous, the delay between S and its successor(s) is zero (no time elapses). Such a state can have more than one successors if several conflicting transitions are firable.

In the second case, there are no firable transitions. The state is terminal (has no successors) if no transitions are enabled (have a non-zero RET) or are currently firing (have a non-zer RFT). In most cases a successor exists. Calculating the successor in such cases requires a calculation of the minimum time which must elapse before an action of interest occurs. This minimum time can be found by calculating the minimum non-zero RET or RFT. The *only* successor to S is then calculated by allowing that minimum time to elapse (subtracting

```
Given State S
Let $\Lambda$ be the set of firable transitions
if $\Lambda \neq \emptyset$
        Partition $\Lambda$ into firable conflict sets
        Let the set of selectors Sel = cross product of firable conflict sets
        Calculate the probability of using each selector s in Sel
        For every selector s in Sel
            generate a successor state S' from S
                Remove tokens from input places of transitions in s.
                Set the RFT of each transition $t_i$ in s to $F(t_i)$.
                For every transition which becomes disabled, in S' reset its RET to 0.
                Assign a zero time delay to the edge from S to S'
Else
        Let Tmin = smallest non-zero RET or RFT in S
        Generate state S' from S by subtracting Tmin from all
                non-zero RET and RFT in S
        For all transitions $t_i$ whose RFT > 0 in S and RFT = 0 in S'
                add tokens to output places of $t_i$
        For all transitions $t_j$ which become enabled in S'
                set RET of $t_j$ to $E(t_j)$ in S'
        Assign Tmin as the time delay for the edge between S and S'
endif
```

Figure 3. Procedure for Generating Successors of a State S

the minimum time from all the non-zero RET and RFT entries). The existence of only one successor to each such state plays an important role in the analysis of timed reachability graphs. Figure 4 shows the timed reachability graph for the net in figure 1.

Zuberek's approach to performance evaluation relies on deriving a *Decision Graph* where only the decision nodes of the reachability graph remain (e.g. in Figure 4a, nodes 3 and 11 are the only decision nodes). All other nodes are eliminated and paths through them are collapsed. Time delays along collapsed paths are accumulated. Figure 5 shows the Decision Graph derived for Figure 4. Decision graphs can be used to measure system throughput and resource utilization.

3. Symbolic Timed Reachability Graphs

In this section we generalize the concept of timed reachability graphs in order to accommodate cases where specific time delays are not known, and where specific firing frequencies are also not known. Our goal is to construct timed reachability graphs where arcs are labeled with expressions representing both time delays and branching probabilities. To accomplish this goal it is necessary to determine the conditions under which symbolic time delays can be introduced into the net without changing the properties of timed reachability graphs which

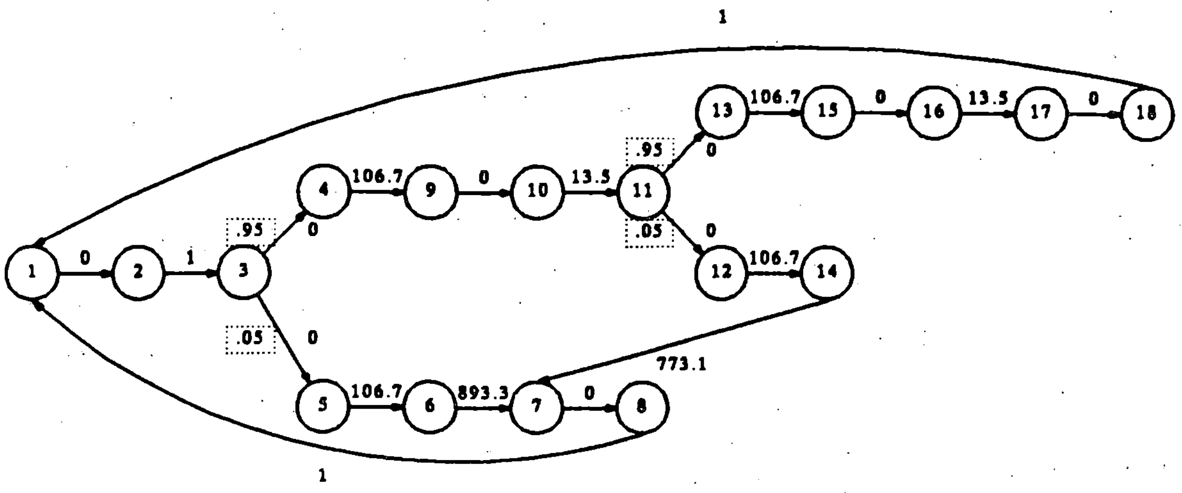
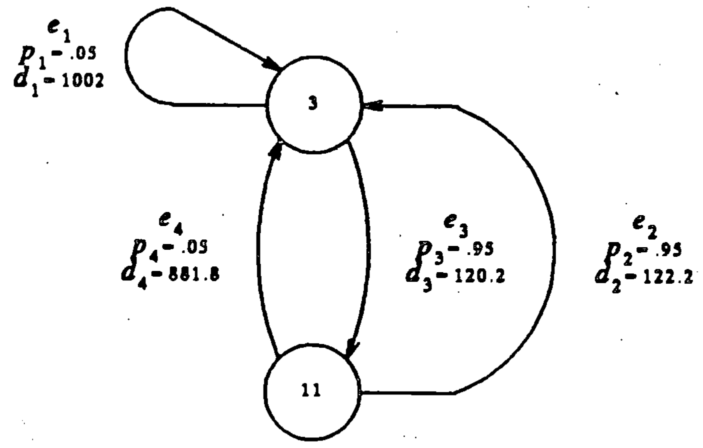


Figure 4a. Timed Reachability Graph of Simple Protocol

State	Marking								RET	RFT								
	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	t_3	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
3	0	1	0	1	0	0	0	1	1000	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	1	1000	0	0	0	0	106.7	0	0	0	0
5	0	0	0	1	0	0	0	1	1000	0	0	0	106.7	0	0	0	0	0
6	0	0	0	1	0	0	0	1	893.3	0	0	0	0	0	0	0	0	0
7	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
9	0	0	1	1	0	0	0	1	893.3	0	0	0	0	0	0	0	0	0
10	0	0	0	1	0	0	0	0	893.3	0	0	0	0	0	13.5	0	0	0
11	0	0	0	1	0	0	1	1	879.8	0	0	0	0	0	0	0	0	0
12	0	0	0	1	0	0	0	1	879.8	0	0	0	0	0	0	0	0	106.7
13	0	0	0	1	0	0	0	1	879.8	0	0	0	0	0	0	0	106.7	0
14	0	0	0	1	0	0	0	1	773.1	0	0	0	0	0	0	0	0	0
15	0	0	0	1	0	1	0	1	773.1	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	13.5	0	0
17	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

Figure 4b. Description of States in Timed Reachability Graph



e_i refers to edge i
p_i refers to the probability of chosing edge i from its source node
d_i refers to the delay for edge i

Figure 5. Decision Graph

make them analyzable.

The technique described above relies heavily on the fact that each transition has a single know value for its enabling time and firing time. As a result, timed reachability graphs consist of only two types of nodes:

1. Decision nodes with many successors and a probability associated with each successor, and

2. Non-decision nodes with only a single successor each.

If it is the case that a specific transition delay is not known, then timed reachability graphs become more complex. The complexity can be seen even in a simple case where two transitions (t_i and t_j) begin firing at the same time. When the two transitions begin firing, the state of the net can be characterized by the remaining firing times of the two transitions. These remaining firing times are exactly the firing times of the transitions ($F(t_i)$, and $F(t_j)$). In calculating successor states we must identify the smaller of the two values. If we do not know the relationship between the two firing times, three cases must be taken into consideration.

1. $F(t_i) > F(t_j)$. In this case transition t_j will finish firing first and the successor state will have a remaining firing time for t_i equal to

$$F(t_i) - F(t_j)$$

The edge from the current state to its successor can be assigned a delay of $F(t_j)$.

2. $F(t_i) < F(t_j)$. In this case transition t_i will finish firing first and the successor state will have a remaining firing time for t_j equal to

$$F(t_j) - F(t_i)$$

The edge from the current state to its successor can be assigned a delay of $F(t_i)$.

3. $F(t_i) = F(t_j)$. In this case both transitions will finish firing at the same time. The edge from the current state to its successor can be assigned a delay of $F(t_i)$.

Since we assume fixed delays, only one of the above relationships can hold, but the reachability graph must include all three possibilities. The above example is quite simple and assumes only two simultaneously active transitions. If more transitions can fire in parallel, the size of the graph grows rapidly. In the worst case, if *nothing* is known a priori about the enabling and firing times of the transitions, the reachability graph may become unmanageably large. In order for the reachability graph to be analyzable, the model must include sufficient timing *constraints* to guarantee that all vertices which do not involve decisions have at most one successor each. This is the case when timing constraint are sufficiently specific to identify the smallest non-zero RET and RFT for every state in the graph. An automated tool could be designed to prompt designers for timing constraints at the necessary points.

The procedure for constructing timed reachability graphs with symbols in place of specific times remains the same as figure 3. The differences are

1. Evaluating the smallest non-zero values is replaced by a procedure for evaluating the smallest value in a set of expressions, given a set of timing constraints.

2. Subtractions must also be done symbolically and expressions must be simplified algebraically.

3. Calculations of firing probabilities are done symbolically whenever firing frequencies are not known.

The resulting reachability graph has timing expressions associated with each edge. Decision graphs can then be constructed and analyzed in order to obtain expressions which describe the performance of the system. These expressions apply for all enabling times and firing times which are consistent with the timing constraints of the net.

In the next section we demonstrate the above approach on the communication protocol described earlier.

4. Example: A Simple Communication Protocol

To demonstrate the approach described above, we assume that specific values of enabling and firing times are not known for the protocol in figure 1. Instead, we assume that the following constraints apply:

(1) $E(t_3) > F(t_5) + F(t_6) + F(t_8)$

(2) $E(t_i) = 0 \quad \forall i, i \neq 3$

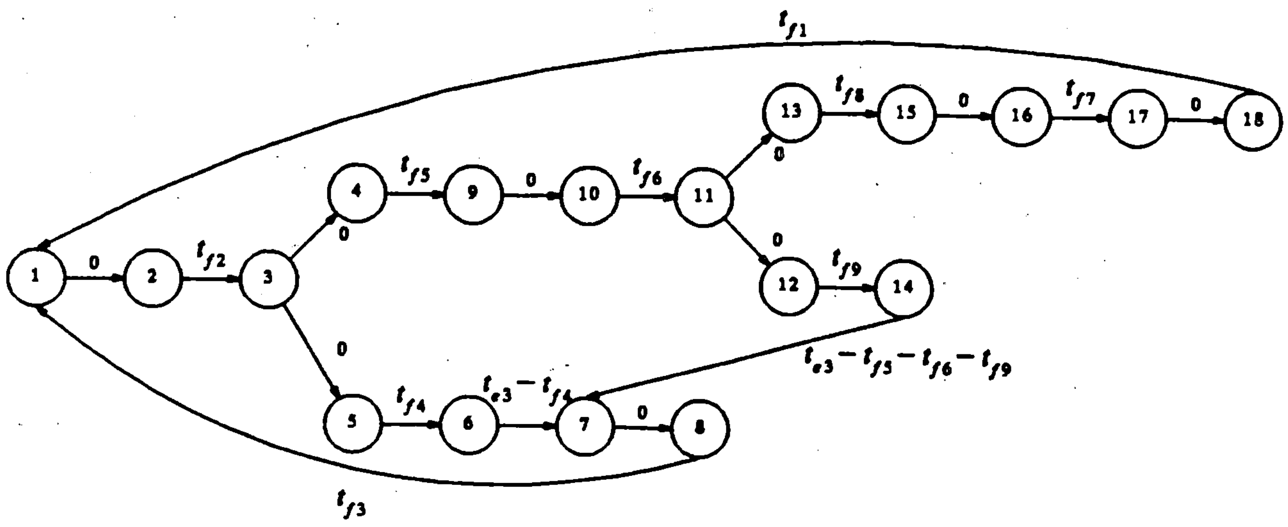
(3) $F(t_4) = F(t_5)$

(4) $F(t_9) = F(t_8)$

Constraint 1 simply states that the timeout period must be greater that the round-trip delay for a message and its acknowledgement. Constraint 2 indicates that only the timeout transition t_3 has a non-zero enabling time. This constraint is only added to simplify the example. Constraints 3 and 4 specify that the loss of a message requires no more time than its successful transmission. Without this constraint it would be possible, under some circumstances, for the safeness assumption to be violated.

The Timed Reachability Graph which satisfies the above constraints is shown in figure 6. In the description below we use the symbols t_{e_i} and t_{f_i} in place of the more cumbersome E(t_i) and F(t_i).

State 1 in the graph is the initial state and has only one possible successor. That successor (state 2) is reached when transition 2 begins to fire. State 2 can only lead to state 3 after a delay of t_{f_2}. In state 3, transition t_3 is enabled, and transitions t_4 and t_5 are both firable. Transition t_4 fires with a probability of $\frac{f_4}{f_4+f_5}$, and leads to state 4. Transition t_5 fires with a probability of $\frac{f_5}{f_4+f_5}$, and leads to state 5. State 4 has three possible successors. The calculation of the correct successor depends on the values of t_{e_3} and t_{f_5}. Constraint 1 indicates that $t_{e_3} > t_{f_5}$. Therefore, the only successor to state 4 is reached after a delay of t_{f_5}. The remaining enabling time of t_3 in state 9 is therefore $t_{e_3} - t_{f_5}$.



Probability for 3 → 4 = $f_4/(f_4+f_5)$
Probability for 3 → 5 = $f_5/(f_4+f_5)$
Probability for 11 → 13 = $f_8/(f_8+f_9)$
Probability for 11 → 12 = $f_9/(f_8+f_9)$

Figure 6a. Symbolic Timed Reachability Graph of Simple Protocol

State	Marking								RET	RFT								
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	t_3	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	t_{f_2}	0	0	0	0	0	0	0
3	0	1	0	1	0	0	0	1	t_{e_3}	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	1	t_{e_3}	0	0	0	0	t_{f_5}	0	0	0	0
5	0	0	0	1	0	0	0	1	t_{e_3}	0	0	0	t_{f_4}	0	0	0	0	0
6	0	0	0	1	0	0	0	1	$t_{e_3}-t_{f_4}$	0	0	0	0	0	0	0	0	0
7	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	t_{f_3}	0	0	0	0	0	0
9	0	0	1	1	0	0	0	1	$t_{e_3}-t_{f_5}$	0	0	0	0	0	0	0	0	0
10	0	0	0	1	0	0	0	0	$t_{e_3}-t_{f_5}$	0	0	0	0	0	t_{f_6}	0	0	0
11	0	0	0	1	0	0	1	1	$t_{e_3}-t_{f_5}-t_{f_6}$	0	0	0	0	0	0	0	0	0
12	0	0	0	1	0	0	0	1	$t_{e_3}-t_{f_5}-t_{f_6}$	0	0	0	0	0	0	0	0	t_{f_9}
13	0	0	0	1	0	0	0	1	$t_{e_3}-t_{f_5}-t_{f_6}$	0	0	0	0	0	0	0	t_{f_8}	0
14	0	0	0	1	0	0	0	1	$t_{e_3}-t_{f_5}-t_{f_6}-t_{f_9}$	0	0	0	0	0	0	0	0	0
15	0	0	0	1	0	1	0	1	$t_{e_3}-t_{f_5}-t_{f_6}-t_{f_8}$	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	t_{f_7}	0	0
17	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	1	0	t_{f_1}	0	0	0	0	0	0	0	0

Figure 6b. Description of States in Symbolic Timed Reachability Graph

The constraints shown above only influence the construction of the graph in states 4, 5, 10, 12 and 13 since these are the only states containing more than one non-zero RET and RFT. Figure 7 summarizes the constraints which are used in each of these states.

Transition	Constraint Used	Derived from:
$4 \rightarrow 9$	$t_{e_3} > t_{f_5}$	1
$5 \rightarrow 6$	$t_{e_3} > t_{f_4}$	1,3
$10 \rightarrow 11$	$t_{e_3} - t_{f_5} > t_{f_6}$	1
$12 \rightarrow 14$	$t_{e_3} - t_{f_5} - t_{f_6} > t_{f_9}$	1,4
$13 \rightarrow 15$	$t_{e_3} - t_{f_5} - t_{f_6} > t_{f_8}$	1

Figure 7. Timing Constraints Used in Reachability Graph

The derivation of performance expressions from this Symbolic Timed Reachability Graph follows the same procedure proposed by Zuberek and elaborated in [RaPh 83]. First, the reachability graph is collapsed into a *Decision Graph* which contains only decision nodes. All other nodes are eliminated and the transition delays are summed (here we do it symbolically). Figure 8 shows the decision graph. Edge 1 corresponds to the path 3-5-6-7-8-1-2-3. Edge 2 corresponds to path 11-13-15-16-17-18-1-2-3. Edge 3 corresponds to path 3-4-9-10-11. Edge 4 corresponds to path 11-12-14-7-8-1-2-3. From the decision graph we derive expressions for the rate of traversal (r_i)of each edge i. The relative amount of time spent on each transition can then be calculated as:

$$w_i = r_i d_i$$

To derive the traversal rates we note that the rate at which an outgoing edge is traversed is a function of the branching probability for that edge and of the rate at which the incoming edges are traversed. Therefore, a set of equations can be derived for each r_i as shown below:

$$r_1 = \left(\frac{f_4}{f_4 + f_5} \right) (r_1 + r_2 + r_4)$$

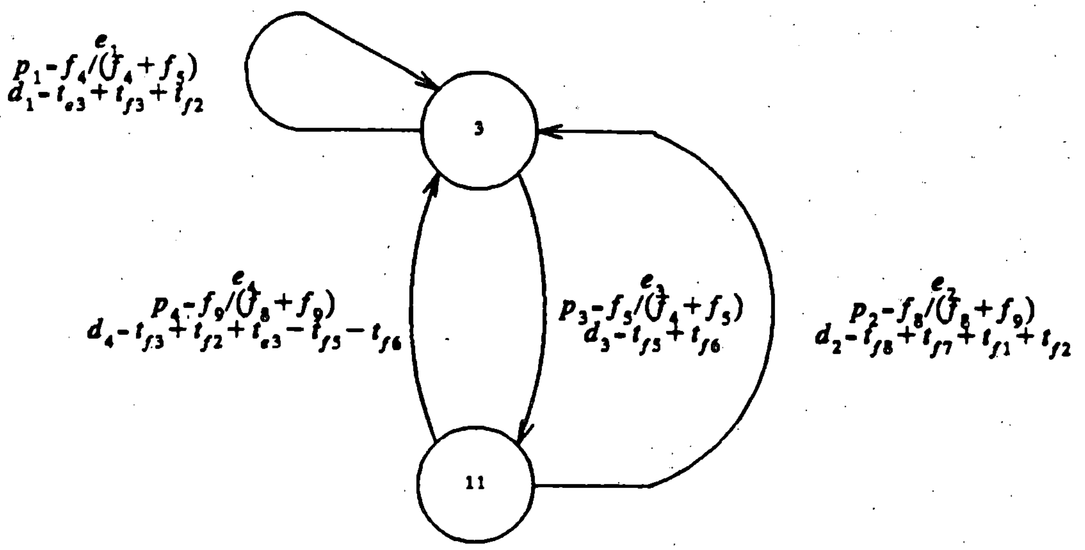


Figure 8. Decision Graph

$$r_2 = \left(\frac{f_8}{f_8+f_9}\right) r_3$$

$$r_3 = \left(\frac{f_5}{f_4+f_5}\right)(r_1 + r_2 + r_4)$$

$$r_4 = \left(\frac{f_9}{f_8+f_9}\right) r_3$$

By assuming a particular value for any of the r_i's we can solve for the remaining variables and obtain a relative traversal rate. Assuming $r_1 = 1$, we get:

$$r_1 = 1, \qquad r_2 = \left(\frac{f_8}{f_8+f_9}\right)\left(\frac{f_5}{f_4}\right)$$

$$r_3 = \frac{f_5}{f_4}, \qquad r_4 = \left(\frac{f_9}{f_8+f_9}\right)\left(\frac{f_5}{f_4}\right)$$

The relative amount of time spent on each edge is therefore:

$$w_1 = r_1 d_1 = t_{e_3} + t_{f_3} + t_{f_2}$$

$$w_2 = r_2 d_2 = \left(\frac{f_8}{f_8+f_9}\right)\left(\frac{f_5}{f_4}\right)(t_{f_8} + t_{f_7} + t_{f_1} + t_{f_2})$$

$$w_3 = r_3 d_3 = \left(\frac{f_5}{f_4}\right)(t_{f_5} + t_{f_6})$$

$$w_4 = r_4 d_4 = \left(\frac{f_9}{f_8+f_9}\right)\left(\frac{f_5}{f_4}\right)(t_{f_3} + t_{f_2} + t_{e_3} - t_{f_5} - t_{f_6})$$

While several performance measures can be derived from this graph, we focus on throughput. To calculate the throughput of the protocol we note that the traversal of edge 2 corresponds to a successfully received and acknowledged message. Therefore, the throughput of the protocol is simply:

$$\frac{r_2}{\sum_i w_i}$$

If we assume a loss probability of 5% for both packets and acknowledgements, the throughput expression simplifies to:

$$\frac{18.05}{1.95(t_{e_3} + t_{f_3}) + 20t_{f_2} + 18.05(t_{f_1} + t_{f_5} + t_{f_6} + t_{f_7} + t_{f_8})}$$

Conclusion

A procedure for symbolically evaluating the performance of systems has been presented. The approach is based on generalizing a method first presented by Zuberek for analyzing Timed Petri Nets. The conditions under which the analysis can be performed have been defined in terms of a set of timing constraints to be specified in place of transition enabling and firing times.

This approach to performance analysis makes Petri Nets the only approach to modeling communication protocols which can be used with the objective of proving correctness and deriving accurate performance estimates. Work by Berthomieu has already shown that correctness proofs can be carried out on Time Petri Nets. We expect that Timed Reachability Graphs can accomplish the same objective since they reveal all the allowed state transitions, given a set of timing constraints.

The work on Timed Petri Nets continues. We are currently exploring techniques for constructing and analyzing Timed Reachability Graphs for nets which allow ranges of firing times. The use of Timed Reachability Graphs for correctness proofs is being explored. There is also a great deal of interest in developing tools which automate the techniques discussed in this paper.

References

[BeMe 83] Berthomieu, B., and M. Menasche, "An Enumerative Approach for Analyzing Time Petri Nets" *Proceedings of the 1983 IFIP Congress*, Paris, (Sept. 1983).

[BeTe 82] Berthelot, G. and Richard Terrat, "Petri Net Theory for the Correctness of Protocols," *Protocol Specification, Testing and Verification, C. Sunshine (ed.)*, North-Holland Pub. Co., (1982).

[BocG 80] Bochmann, G.V., "A General Transition Model of Protocols and Communication Services," *IEEE Transactions on Communications COM-28,4 (April 1980)*, pp. 643-650.

[CoRo 83] Coolahan, J.E., and N. Roussopoulos, "Timing Requirements for Time-Driven Systems Using Augmented Petri Nets" *IEEE Transactions on Software Engineering*, SE-9,5 (Sept. 1983), pp. 603-616.

[HaOw 80] Hailpern, B. and S. Owicki, "Verifying Network Protocols Using Temporal Logic", *Proceedings of Trends and Applications Symposium, 1980, Computer Network Protocols*, National Bureau of Standards, Maryland, (May 1980).

[MeFa 76] Merlin, P. and D. Farber, "A Methodology for the Design and Implementation of Communications Protocols," *IEEE Transactions on Communications*, COM-24, 6 (June 1976).

[MolM 81] Molloy, M. "On the Integration of Delay and Throughput Measures in Distributed Processing Models", Computer Science Dept., University of California, Los Angeles, Report No. CSD-810921, September 1981.

[RaEs 80] Razouk, R.R., and G. Estrin "Modeling and Verification of Communication Protocols: The X.21 Interface" *IEEE Transactions on Computers*, C-29,12 (December 1980), pp. 1038-1052.

[RaHo 80] Ramamoorthy C.V. and G.S. Ho, "Performance Evaluation of Asynchronous Concurrency Systems using Petri Nets," *IEEE Transaction on Software Engineering*, SE-6, 5 (September 1980), 440-449.

[RamC 74] Ramchandani, C. "Analysis of Asynchronous Concurrent Systems by Timed Petri Nets," Ph.D. Thesis, MIT 1974, Project Mac Report No. MAC-TR-120.

[RaPh 83] Razouk, R.R., and C. Phelps "Performance Analysis Using Timed Petri Nets", Tech. Rept. 206, University of California, Irvine, (1983). (Submitted to 4th Int. Conf. on Distributed Computing Systems, May 1984).

[SaSc 82] Sabnani, K. and M. Schwartz; "Verification of a Multidestination Protocol Using Temporal Logic", *Protocol Specification, Testing and Verification, C. Sunshine (ed.)* North-Holland Pub. Co., (1982).

[SifJ 77] Sifakis, J. "Petri Nets for Performance Evaluation," *Measuring, Modelling and Evaluating Computer Systems*, Proceedings of the 3rd International Symposium, IFIP Working Group 7.3, H. Beilner and E. Gelenbe (eds.), North-Holland Pub. Co. (1977), pp. 75-93.

[SiKa 82] Simon, G. and D. Kaufman, "An Extended Finite State Machine Approach to Protocol Specification," *Protocol Specification, Testing and Verification, C. Sunshine (ed.)*, North-Holland Pub. Co. (1982).

[SymF 80] Symons, F.J.W., "Verification of Communication Protocols using Numerical Petri Nets," *Australian Telecommunication Research*, 14,1 (1980) 34-38.

[TeLi 80] Teng, A.Y., and M.T. Liu, "A Formal Approach to the Design and Implementation of Network Communication Protocol," *Proceedings of COMPSAC, (May 1980)*, pp. 722-727.

[VSE 83] Vernon, M.K., E. de Souza e Silva, and G. Estrin "Performance Evaluation of Asynchronous Concurrent Systems: The UCLA Graph Model of Behavior" *9th International Symposium on Computer Performance Modelling, Measurement and Evaluation*, College Park, Maryland, May 25-27, 1983.

[YGH 82[Yelowitz, L., S. Gerhart, and G. Hilborn, "Modeling a Network Protocol in Affirm and Ada", *Protocol Specification, Testing and Verification, C. Sunshine (ed.)*, North-Holland Pub. Co., (1982).

[ZubW 80] Zuberek, W.M., "Timed Petri Nets and Preliminary Performance Evaluation," *7th Annual Symposium on Computer Architecture*, (1980), pp. 88-96.