MATLAB BASED PETRI NET ANALYSIS

Zdeněk Hanzálek
Center for Applied Cybernetics, DCE FEE
Czech Technical University in Prague

1. Introduction

Petri Nets make it possible to model and visualize behavior comprising concurrency, synchronization and resource sharing. They are convenient tools to model parallel algorithms and communication protocols.
Petri Nets offer profound mathematical background originating namely from linear algebra and graph theory. Murata [5] provides a good survey on properties, analysis and applications of Petri Nets.
Various tools listed on http://www.daimi.au.dk/~petrinet/ perform analysis and simulation of various Petri Net classes. Very often they offer convenient graphical environment and sometimes they tend to be too complex. On the other hand these tools have very limited possibility of extensions to problems specifically needed for given application. Some of them are accessible in source code, but these projects are relatively large and difficult to modify.
Approach adopted in this article is based on the three steps:

- Petri Net modeling in convenient graphical design tool (e.g. Pmaker)
- export to matrix representation in Matlab
- Matlab based Petri Net analysis and visualization of simulation results.

This approach allows to organize a work in a modular way, to use standard libraries and to build own tools. In other words we are no more using a 'universal' tool but we are programming our own tool with support in a modeling and visualization stage. This is quite more convenient because no tool is universal enough.
This article is organized in two principal sections. One is based on Petri Net (PN) structural properties and one on Stochastic Timed Petri Net (STPN) token player. Various examples used in laboratory exercises of the subject Distributed Control Systems are shown through the text.

2. Analysis based on structural properties

The structural properties are those that depend on the topological structure of Petri Nets. They are independent of the initial marking M_0 in the sense that these properties hold for any initial marking or are concerned with the existence of certain firing sequences from some initial marking. Thus these properties can often be characterized in terms of the incidence matrix C and its associated homogeneous equations or inequalities.

2.1. Linear invariants

Structural properties of Petri Nets are given by linear invariants introduced by Lautenbach [10]. In practice we are usually interested in a positive version of linear invariants defined as follows:
Let a finite PN system with $P=\{P1,P2, \dots , Pm\}$ and $T=\{T1,T2, \dots ,Tn\}$ be given then a vector $f \in Z^m$ is called a positive P-invariant of the given PN, iff: $C^T \bullet f = 0 \; \wedge \; f_i \geq 0 \; \forall \, i=1,\dots,m.$
In this article, furthermore, only P-invariants will be considered. To obtain T-invariants, it is needed to transpose the matrix C and the use the same method.
More generally a given invariant f can be written as a composition of invariants called generators
In general there are two approaches to find positive P-invariants (Matlab algorithms source code for both approaches is available from the author upon request):
1) Find a basis of a certain type and then construct the generators by varying the basis vectors [12].
2) Find a set of positive P-invariant generators by combinations of all input and all output places. This algorithm was published in [11]. It generalizes in some sense the Jensen's rules, offering a systematic way to finding all invariants.

2.2. Implicit place

Let a finite PN system with $P=\{P1,P2, \dots , Pm\}$ and $T=\{T1,T2, \dots ,Tn\}$ be given. A place Px is called an implicit place iff the two following conditions hold:
1) any reachable marking $M(Px)$ is equal to a positive linear combination of markings of places from P
2) $M_0(Px)$ does not impose any additional condition to fire its output transitions

2.3. Deadlock avoidance

Figure 1 shows a Petri Net model of two communicating computers in deadlock situation. The core of the problem is that both computers are ready to receive but none is ready to send.
When analyzing PN from Figure 1 in Matlab we can obtain positive P-invariant generators specifying conservative components (subnet preserving number of tokens):

- {P1,P2}
- {P3,P4}
- {P3,P5,P2,P6}

As a consequence the following equations hold for marked PN:
M(P1) + M(P2) = 1
M(P3) + M(P4) = 1
M(P3) + M(P5)+M(P2)+M(P6) = 0
PN given in Figure 1 is a marked graph, so the third equation proofs that the system is not live because this conservative component does not contain any token.

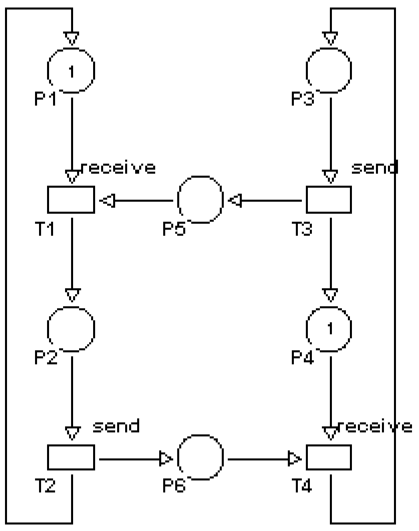

Figure 1: Two communicating computers in deadlock

2.4. Algorithm parallelisation

Consider an example algorithm given by the following pseudo-code:

```
for k=1 to n
      A(k)=fce(G(k-1))
      B(k)=fce(A(k),E(k-1))
      C(k)=fce(A(k))
      D(k)=fce(D(k-1),C(k))
      E(k)=fce(B(k),D(k))
      F(k)=fce(C(k))
      G(k)=fce(E(k),F(k),C(k))
      H(k)=fce(I(k-1))
      I(k)=fce(H(k))
endfor
```

The code is represented by the the PN model given in Figure 2. Places correspond to the data and transitions correspond to the code.

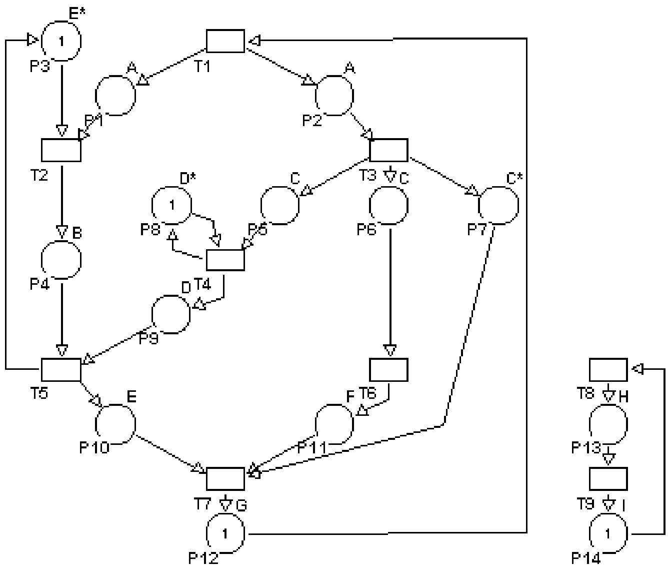

Figure 2: Algorithm data dependencies

Markings correspond to the presence of valid data computed by the previous transition or assigned during algorithm initialization. The cyclic algorithm can be scheduled to unlimited (but possibly minimal) number of processors while analyzing PN from Figure 2 in Matlab:

- reduce implicit places (e.g. C* in Figure 2)
- reduce self-loop places (e.g. D* in Figure 2)
- find positive P-invariant generators
- find optimal schedule by assigning processors to positive P-invariant generators (number of processors correspond to the number of tokens in the positive P-invariant)

2.5. Token Ring

Figure 3 shows two nodes accessing the media with the use of token ring access method. The media activity is represented by P9 and by P10 (Bus_Idle and Bus_busy). These two places are in fact implicit - they do not contribute to the system behavior. This is proven by the following reasoning:

- from positive P-invariant generators

$M(P1)+M(P5)+M(P10) = 1$
$M(P1)+M(P3)+M(P5)+M(P7) = 1$

- by substitution the following holds

$M(P10) \geq M(P3)$
$1-M(P1)-M(P5) \geq 1-M(P1)-M(P5)-M(P7)$
$M(P7) \geq 0$

- in similar way

$M(P10) \geq M(P7)$
$1-M(P1)-M(P5) \geq 1-M(P1)-M(P3)-M(P5)$
$M(P3) \geq 0$

- as consequence P10 is implicit
- in similar way P9 is implicit too

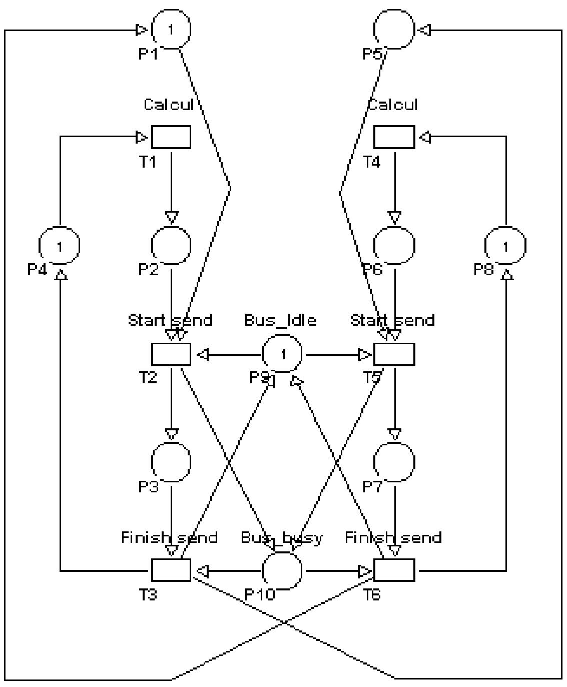

Figure 3: Token Ring access method

3. Simulations based on STPN token player

Stochastic Timed Petri nets used in this article contain either zero, timed or stochastic timed transitions with uniform distribution. There are two types of non-zero time transition behavior given in literature. The definition used in the article, considering that transition does not reserve the tokens in input places, is described e.g. in [6]. The functionality of the model under consideration is fully specified by the interpretation of the token player given in Matlab like syntax by Algorithm 1.

Algorithm 1:

```
[Seq,M] = PlaySTPN(Pre, Post, M0, TimeT, TypeT, ticks)
%Pre - Matrix of preconditions, Post - Matrix of postconditions, M0 - Vector of initial markings
%TimeT - Vector of transition time, TypeT - Vector of transitions types (zero / timed / stochastic)
%ticks  - Number of simulation ticks
%Seq - Firing sequence, M - Markings at the end of simulation

%1-initialization
M=M0;
for j=1:number of transitions
        CountT(j)=initial counter value
end
%Main Cycle
while t<=ticks
        %2-generate vectors x-enabled transitions and y-firable transitions
        for j=1:number of transitions
                if transition j is enabled
                        then    x(j)=1
                                if CountT(j)=0   then y(j)=1
                                                 else y(j)=0
                                end
                        else    x(j)=0
                                CountT(j)=initial counter value
                end
        end
        %3-decrement counters or change markings
        if y=zeros
                then    %decrement counters for enabled transitions in x
                        CountT=CountT-1
                        t=t+1
                else    %change marking by firing one transition
                        f = randomly chosen firable transition in y
                        M = M + Post(:,f) - Pre(:,f)
                        %add fired transisiton to firing sequence
                        Seq=[Seq,f]
        end
end
```

With respect to the Algorithm 1 the following drawbacks should be mentioned:
In the case of effective conflict, no token reservation is assumed. It means that the first fireable transition (enabled & counter=0) wins. When “reservation” behavior is needed, then zero time transition should precede timed or stochastic transition.
In the case of the actual conflict representing the system non-determinism (two and more fireable transitions in conflict) the winning transition is chosen in random manner. It means that one possible firing sequence is chosen among several ones.
Initial transition counter CountT(j) is either 0 (in the case of zero-time transition), or TimeT(j) (in the case of timed transition) or integer number from interval <1,TimeT(j)> (in the case of the stochastic transition with uniform distribution).

3.1. Predictive p-persistent CSMA model

The STPN model of one node MAC layer using predictive p-persistent CSMA is shown in Figure 4. Zero time transitions are not labelled (e.g. T_1, T_2, T_3), timed transitions are labelled by “t” with corresponding time (e.g. 5 ticks in the case of T_4) and stochastic transitions are labelled by “s” with corresponding upper margin of uniform distribution interval. Figure 4 consists of the following parts:
Idle Channel Detection Model is situated on the top of the picture (P_3 - P_5, T_2 - T_4).
Left side of the figure represents the backlog estimator (P_{17} - P_{27}, $T_{18} – T_{36}$).
Physical layer model is on the right side ($P_8 – P_{12}$, $T_8 – T_{11}$).
Medium Access Model is on the bottom of the figure (P_{13} - P_{16}, $T_{12} – T_{17}$).

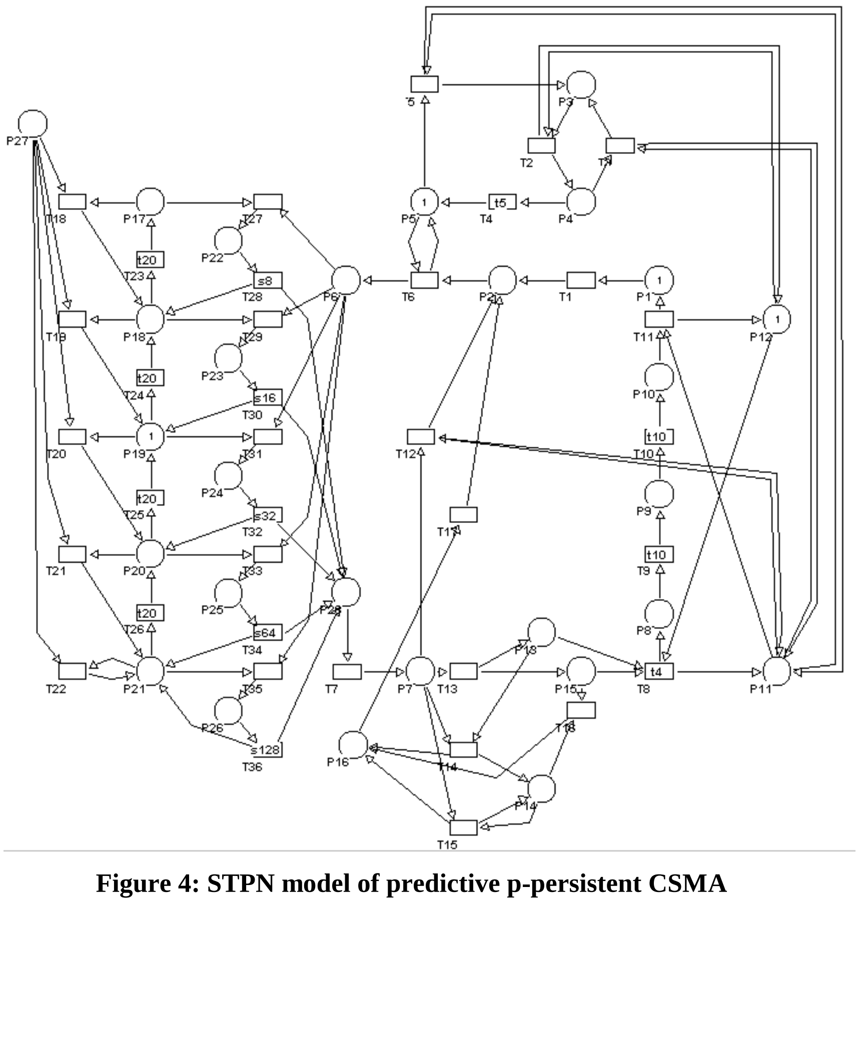

Figure 4: STPN model of predictive p-persistent CSMA

The channel backlog (BL) corresponds to mean value of the predicated channel load. All nodes increment the estimated backlog by one when receiving request frame (owing to the broadcast capability of the bus topology, a particular node is aware of this frame even if it is not destination node) or sending the request frame. The estimated channel backlog is decremented by one at the end of each packet cycle (request and response frame). Our model assumes that all frames have the same length (10 ticks) and that all packets are acknowledged (T_9 – request, T_{10} - response).
Because of lack of space only 5-level backlog estimator is modelled. Places P_{17}, P_{18}, P_{19}, P_{20}, and P_{21} represent the state of the backlog estimator and correspond to the current value of channel backlog *BL*. Place P_{17} represents BL=1, P_{18} ~ *BL=2*, and so on. Message *Frame_OK* (place P_{27}), indicates a successfully received request frame as a broadcast message from the physical layer (from the transition T_9 in sending node) and transitions T_{18}, T_{19}, T_{20}, T_{21} increment current value of backlog. Transition T_{23}, T_{24}, T_{25}, and T_{26} are *timed transitions* with deterministic firing time corresponding to the packet cycle. Via T_{28}, T_{30}, T_{32}, T_{34}, *or* T_{36} predictive p-persistent CSMA generates random time delay from random timing interval, which is dependent on the estimated channel backlog. Please notice, that just one of the transitions T_{28}, T_{30}, T_{32}, T_{34}, T_{36} is fired at a given time instant.

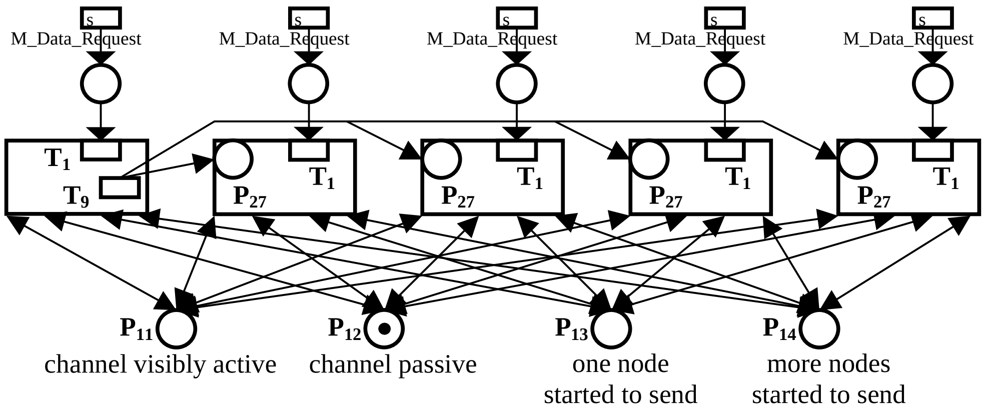

Figure 5: Interconnection of five nodes

Simulated network consisting of five nodes is shown in Figure 5. Broadcast communication of request frame is realized by arcs going form T_9 in each particular node to P_{27} in all other nodes (for simplicity reasons Figure 5 shows just broadcasting from the first node to other nodes).

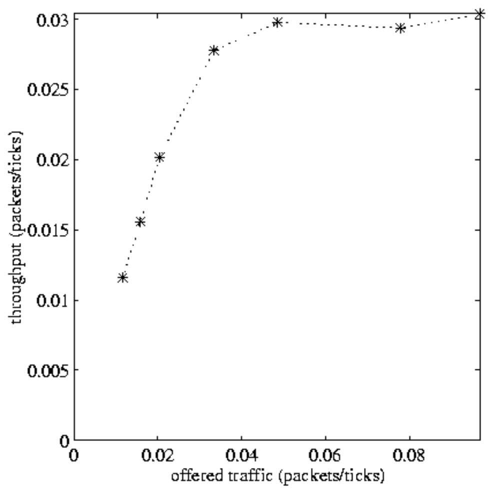

Figure 6: Simulated offered traffic

Characteristic in Fig.6 show that even in the case of a big traffic load network throughput is very closed to simple saturation without communication bottleneck. This is achieved due to approximation of the channel backlog (higher traffic ⇒ higher backlog ⇒ longer average random delay ⇒ less collisions) without charging low traffic by long random delay.

Acknowledgement: This work was supported by the Ministry of Education of the Czech Republic under Project LN00B096.

References

[1] Blum, I., Juanole, G.: Formal Specification and Analysis of a Control System Based on Computer Networks, WFCS'97 – IEEE workshop on Factory Communication Systems, October 1-3, 1997, Grafismar Barcelona

[2] Hilmer, H., Kochs, H.D., Dittmar, E.: A Fault Tolerant Communication Architecture for Real-time Control Systems, WFCS'97 – IEEE workshop on Factory Communication Systems, October 1-3, 1997, Grafismar Barcelona

[3] Tovar E., Vasques F., Guaranteeing Real-Time Message Deadlines in Profibus Networks, 10th Euromicro Workshop on R-T Systems, 1998, Berlin

[4] ECHELON® CORPORATION, Document No.19550, „LonTalk® Protocol Specification" -Version3.0. Palo Alto, 1994.

[5] Tadao Murata: "Petri Nets: Properties, Analysis and Applications" Proceedings of the IEEE, vol. 77, No. 4, April 1989.

[6] M. Menasche and B. Bartolomieu: „Time Petri nets for analyzing and verifying time dependent communication protocols". Proc. Of 3rd Int. Workshop on Protocol Specification, Testing and Verification, Elsevier Science, 1983, pp. 161-172

[7] Zhen Liu, "Performance Analysis of Stochastic Timed Petri Nets using Linear Programing Approach, INRIA 1997 ISSN 0249-6399

[8] G. Chiola, A. Ferscha, Distributed Simulation of Petri Nets, University of Torino, Italy, 1993

[9] Enhanced Media Access Control with *LonTalk*® Protocol, LonWorks® engineering bulletin, January 1995, Part Number 005-0001-01 Rev.C

[10] K. Lautenbach, H.A. Schmid: Use of Petri Nets for Proving Correctness of Concurrent Process Systems, IFIP 74, North Holland Pub. Co., (1974) 187-191.

[11] J. Martinez, M. Silva: A Simple and Fast Algorithm to Obtain All Invariants of a Generalised Petri Net, in: C. Girault, W. Reisig (eds): Application and Theory of Petri Nets, Informatik Fachberichte No.52, Springer (1982), 301-310.

[12] F. Kruckeberg, M. Jaxy: Mathematical Methods for Calculating Invariants in Petri Nets, in: G. Rozenberg (ed): Advances in Petri Nets, LNCS 266, Springer (1987) 104-131.

Contact address:
Karlovo nám. 13, Czech Technical University in Prague
121 35, Prague 2, Czech Republic
hanzalek@rtime.felk.cvut.cz
http://dce.felk.cvut.cz/hanzalek/
phone: ++420 2 24357434
fax: ++420 2 24357610