Stochastic Petri nets

Stochastic Petri nets

- Markov Chain grows very fast with the dimension of the system

- Petri nets: High-level specification formalism

- Markovian Stochastic Petri nets

  adding temporal and probabilistic information to the model
  the approach aimed at equivalence between SPN and MC
  idea of associating an exponentially distributed random delay
  with the PN transitions (1980)

- Non Markovian Stochastic Petri nets
  non exponentially distributed random delay

Automated tools supporting SPN for modelling and evaluation

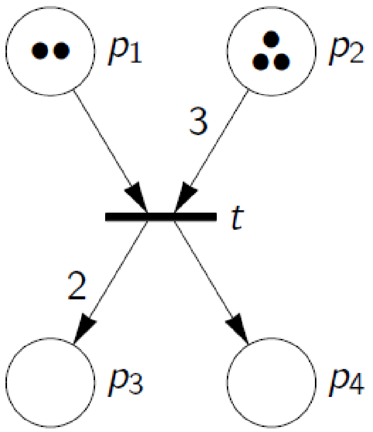
Petri nets

$$N_{P/T} = \langle P, T; F, W, M_0 \rangle$$

- Place ○
- Transition ▬
- Token ●
- Arcs $\longrightarrow$

$$W : F \to \mathbb{N} - \{0\}$$ Weigth of arcs

$$M_0 : P \to \mathbb{N}$$ Initial marking



t, y transitions

$${}^{\bullet}t = \{p \in P \mid \langle p, t \rangle \in F\}$$ preset

$$y^{\bullet} = \{z \in P \mid \langle y, z \rangle \in F\}$$ postset

t enabled if: $\forall_{p \in {}^{\bullet}t}\ M(p) \geq W(\langle p, t \rangle)$

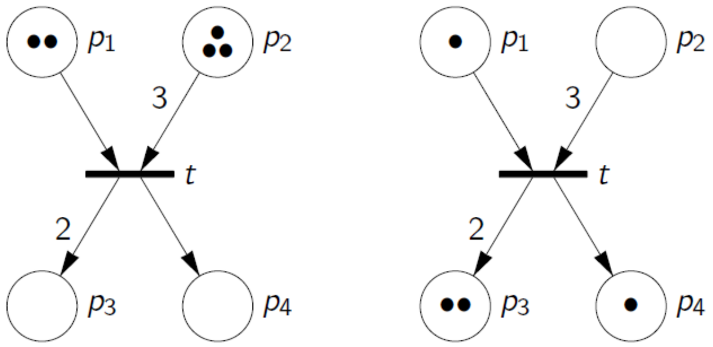
we write $M\,[t\rangle$

Transition firing

Firing rule

$$\begin{array}{ll}
\forall_{p\in({}^\bullet t - t^\bullet)} & M'(p) = M(p) - W(\langle p, t\rangle) \\
\forall_{p\in(t^\bullet - {}^\bullet t)} & M'(p) = M(p) + W(\langle t, p\rangle) \\
\forall_{p\in({}^\bullet t \cap t^\bullet)} & M'(p) = M(p) - W(\langle p, t\rangle) + W(\langle t, p\rangle) \\
\forall_{p\in P-({}^\bullet t \cup t^\bullet)} & M'(p) = M(p)
\end{array}$$

We write $M\,[t\rangle\,M'$



M0[>M1

M0=(2,3,0,0)

M1=(21,0,2,1)

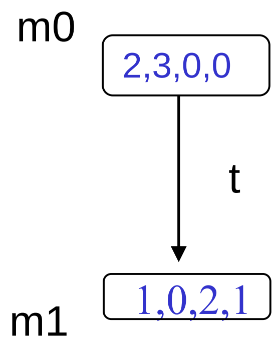
Reachable marking $R_N(M)$:

$$M \in R_N(M)$$

$$M' \in R_N(M) \wedge \exists_{t \in T} M' \, [t\rangle \, M'' \Rightarrow M'' \in R_N(M)$$

We can build the reachability graph

Reachability graph



Transition sequence

Let $M\,[t_1\rangle\,M' \wedge M'\,[t_2\rangle\,M''$ then $M\,[t_1 t_2\rangle\,M''$

If $M\,[t_1 \ldots t_n\rangle\,M^{(n)}$ then $t_1 \ldots t_n$ is a transition sequence

Analysis

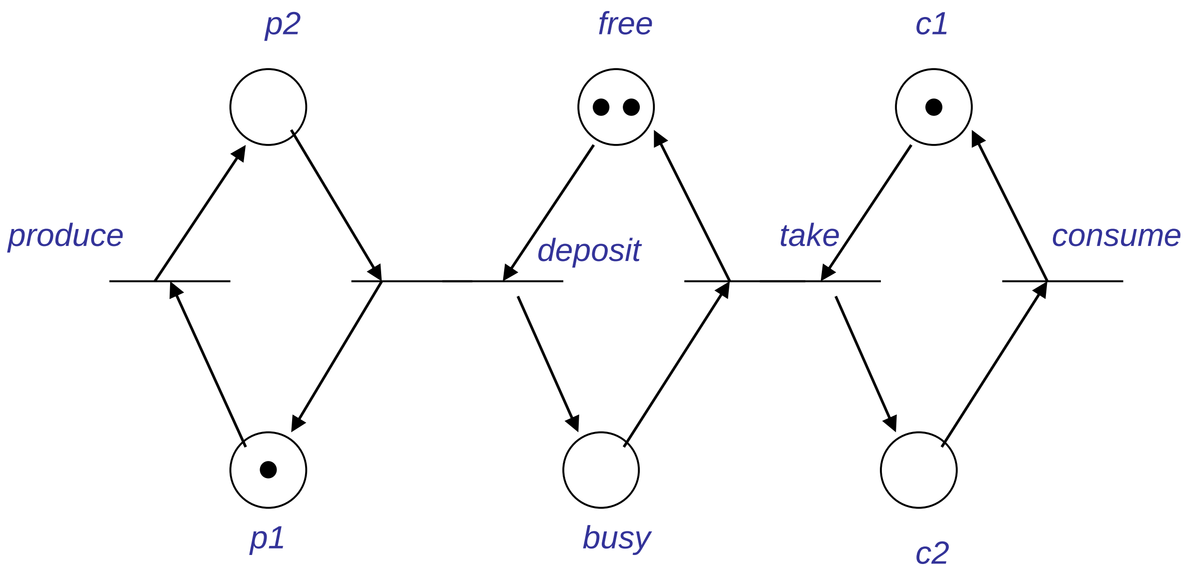
Reachable markings

Transitions never enabled

Conditions on reachable markings

………………………

Producer/Consumer example



Producer

Buffer
(2 slots)

Consumer

System description with Petri nets

- Place:
    - a system component (one for every component)
    - a class of system components (CPU , Memory, ..)
    - components in a given state (CPU, FaultyCPU, ..)
    - …….
- Token:
    - number of components (number of CPUs)
    - Occurrence of an event (fault, ..)
    - …..
- Transitions:
    - Occurrence of an event (Repair, CPUFaulty, …)
    - Execution of a computation step
    - …

Timed transitions

Timed transition: an activity that needs some time to be executed

When a transition is enabled a local timer is set to a delay d;

- the timer is decresed
- when the timer elapses, the transition fires and remove tokens from input places

- if the transitions is desabled before the firing, the timer stops.

Handling of the timer (two alternatives):

Continue:
the timer maintains the value and it will be used when the transition is enabled again

Restart:
the timer is reset

Sequence of timed transitions:

$(\tau k1, Tk1) \ldots (\tau kn, Tkn)$

where

$\tau k1 <= \tau k2 <= \tau kn$

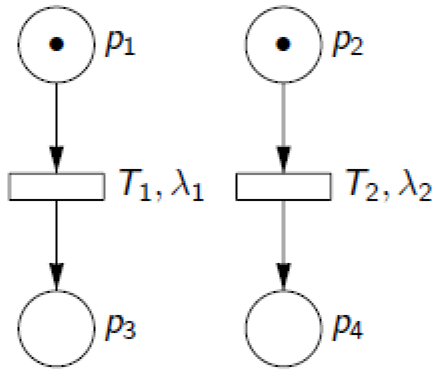
$[\tau ki, \tau ki+1)$ is the period of time between the firing of two transitions

period of time the net stay in a marking

STOCHASTIC PETRI NET:

when the delay d of a timed transition is a random variable

Stochastic Petri nets (SPN) – reachability graph



$$\begin{tikzcd}
 & \overset{m_0}{\boxed{1,1,0,0}} \arrow[dl, "T_1"'] \arrow[dr, "T_2"] & \\
\overset{m_1}{\boxed{0,1,1,0}} \arrow[dr, "T_2"'] & & \overset{m_2}{\boxed{1,0,0,1}} \arrow[dl, "T_1"] \\
 & \overset{m_3}{\boxed{0,0,1,1}} &
\end{tikzcd}$$

A timed transition T enabled at time t, with d the random value for the transition delay, fires at time t+d if it remains enabled in the interval [t, t+d)

Markov chain

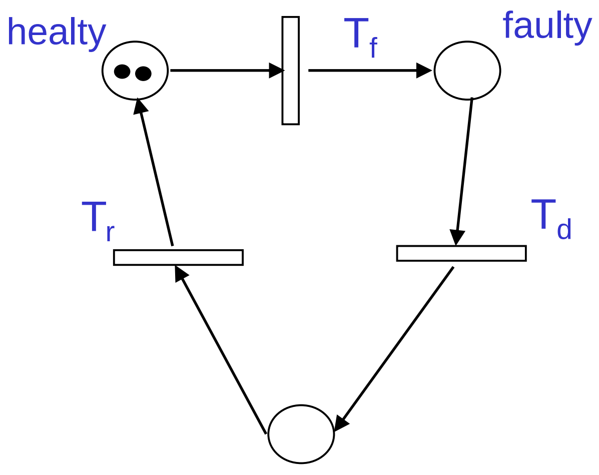
Random process {M(t), t>=0}

with M(0) = M0 and M(t) the marking at time t

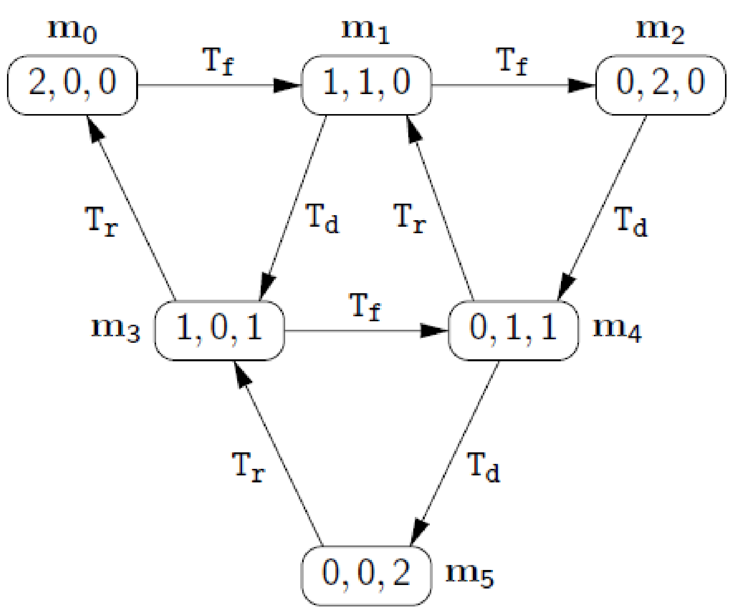
{M(t), t>=0} is a CTMC (memoriless property of exponential distributions)

A redundant system with repair

- Two identical CPUs
- Failure of the CPU: exponentially distributed with parameter λ
- Fault detection: exponentially distributed with parameter δ
- CPU repair: exponentially distributed with parameter ⌠



SPN



Reachability graph

Markov chain

$$\begin{tikzcd}
2,0,0 \arrow[rr, "2\lambda"] & & 1,1,0 \arrow[rr, "\lambda"] \arrow[dl, "\delta"] & & 0,2,0 \arrow[dl, "2\delta"] \\
& 1,0,1 \arrow[ul, "\mu"] \arrow[rr, "\lambda"] & & 0,1,1 \arrow[ul, "\mu"] \arrow[dl, "\delta"] & \\
& & 0,0,2 \arrow[ul, "2\mu"] & &
\end{tikzcd}$$

Properties

- Steady-state probability that both processors behave correctly
- Steady-state probability of one undetected faulty processor
- Steady state probability that both processors must be repaired
- ....

M. Ajmone Marsan. Stochastic Petri nets: An elementary introduction. In G. Rozenberg, editor, *Advances in Petri Nets 1989*, volume 424 of *LNCS*, pages 1–29. Springer Verlag, 1990.

Stochastic Activity Networks (SANs)

A system is described with SANs through four disjoint sets of nodes:

- places
- input gates
- output gates
- activities

activity:

instantaneous or
timed (the duration is expressed via a time distribution function)

input gate:

enabling predicate (boolean function on the marking) and an input function

output gate:

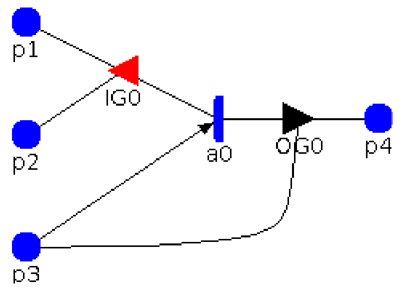
output function

Cases and case distribution:

instantaneous or timed activity may have mutually exclusive outcomes,
called cases, chosen probabilistically according to the case distribution
of the activity. Cases can be used to model probabilistic behaviors.

Output gates allow the definition of different next marking rules for the cases of the activity

Example



Gate	Definition

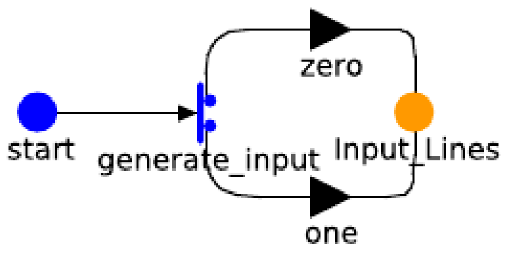
IG0	Enabling predicate
	if ((p1 $\rightarrow$ Mark() $> 0 \wedge$ p2 $\rightarrow$ Mark() ==0) ||
	(p1$\rightarrow$ Mark()==0 $\wedge$ p2 $\rightarrow$ Mark()$>$ 0))
	return 1;
	else return 0;

OG0	Output function
	p4 $\rightarrow$ Mark()++;
	if (p3 $\rightarrow$ Mark() == 0) p3 $\rightarrow$ Mark()=1;

Enabling predicates, and input and output gate functions are usually expressed using pseudo-C code.

Graphically, places are drawn as circles, input (output) gates as left-pointing (right-pointing) triangles, instantaneous activities as narrow vertical bars, and timed activities as thick vertical bars. Cases are drawn as small circles on the right side of activities.

Case activity: generate_input



SAN evolution starting from a given marking M

(i) the instantaneous activities enabled in M complete in some unspecified order;
(ii) if no instantaneous activities are enabled in M, the enabled (timed) activities become active;
(iii) the completion times of each active (timed) activity are computed stochastically, according to the respective time distributions;
the activity with the earliest completion time is selected for completion;
(iv) when an activity (timed or not) completes, the next marking M' is computed by evaluating the input and output functions;
(v) if an activity that was active in M is no longer enabled in M', it is removed from the set of active activities.

Mobius tool

Mobius tool provides a comprehensive framework for evaluation of system dependability and performance.

The main features of the tool include:

(i) a hierarchical modeling paradigm, allowing one to build complex models by specifying the behavior of individual components and combining them to create a complete system model;

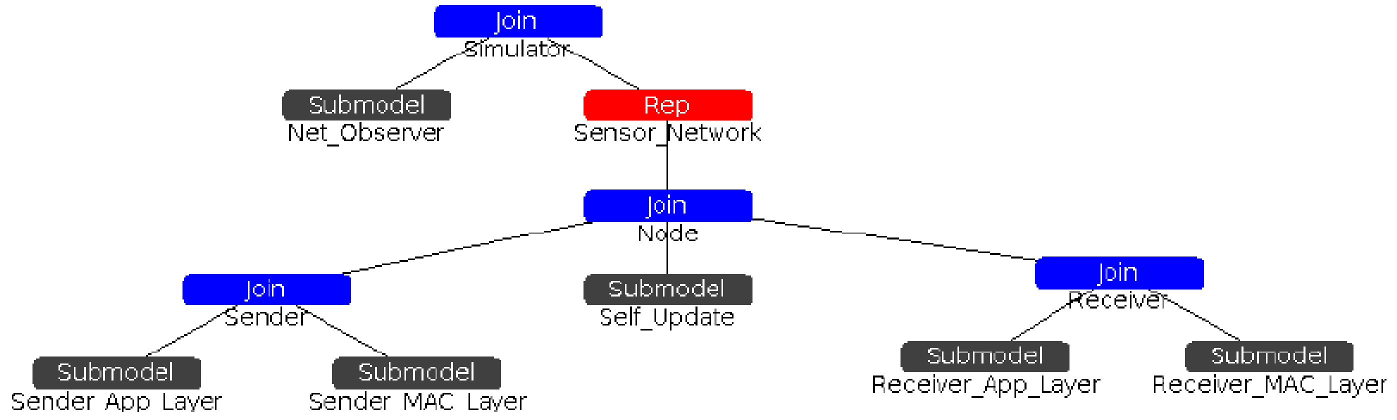
(ii) customised measures of system properties;

(iii) distributed discrete-event simulation;

(iv) numerical solution of Markov models.

Example
Simulator of a sensor network protocol

Hierarchical model



Rep operator:

constructs a number of replicas of an individual component

Join operator:

groups individual and/or replicated components.

Other extension to PN:

Shared variables
(global objects that can be used to exchange information among modules)

Extended places
(places whose marking is a complex data structure instead of a non-negative integer)

Activities

Activity	Duration	Description
Sender MAC Layer		
`start`	instantaneous	transmission start
`test`	instantaneous	listening the channel
`backoff`	exponential	backoff period
`tx_chunk`	$c(\texttt{chunk\_length} \rightarrow Mark())$	transmit the preamble
`tx_data`	constant	transmit the data packet
`ready_to_send`	instantaneous	check interval end
Receiver MAC Layer		
`duty_on`	constant	wake up and channel check
`rx_chunck`	exponential	reception of chunck
`sleep`	$c'(\texttt{sleep\_length} \rightarrow Mark())$	sleep period between chunck and data packet
`rx_data`	constant	reception of a data packet
`duty_off`	$c''(\texttt{duty\_off\_length} \rightarrow Mark())$	duty off period
`restart`	instantaneous	check interval end/restart
Self_Update		
`time`	constant	periodic update of timers
`update`	instantaneous	update timers

Simulation and Reward functions

Mobius allows system simulation and allows properties of interest to be specified with **reward functions**

Reward function:
specifies how to measure a property on the basis of the SAN marking.

Measurements can be made at specific time instants, over periods of time, or when the system reaches a steady state.

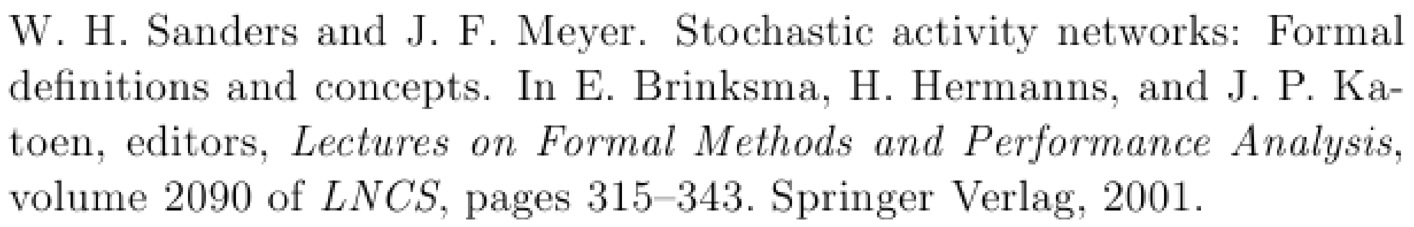
A desired confidence level is associated to each reward function.

At the end of a simulation the Mobius tool is able to evaluate for each reward function whether the desired confidence level has been attained or not thus ensuring high accuracy of measurements.

Solutions on measures

Numerical solver
model is the reachability graph.
Only for models with deterministic and exponentially distributed actions
Finite state, small states-space description

Simulation
model is a discrete event simulator.
For all models
Arbitrary large state-space description
Provide statistically accurate solutions within some user specifiable confidence interval



W. H. Sanders and J. F. Meyer. Stochastic activity networks: Formal definitions and concepts. In E. Brinksma, H. Hermanns, and J. P. Katoen, editors, *Lectures on Formal Methods and Performance Analysis*, volume 2090 of *LNCS*, pages 315–343. Springer Verlag, 2001.