Stochastic Petri nets

Stochastic Petri nets

- Petri nets: High-level specification formalism
- Markov Chain grows very fast with the dimension of the system
- 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

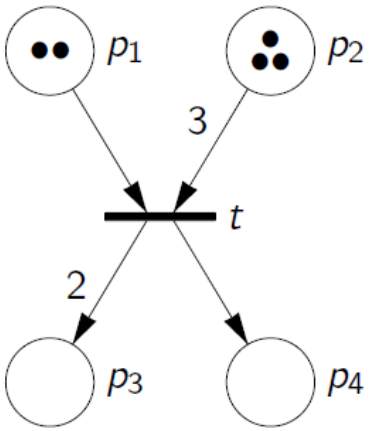
Petri nets

- Place
- Transition
- Token
- Arcs

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

$W : F \to \mathbb{N} - \{0\}$ 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: $\forall_{p \in {}^\bullet t}\ M(p) \geq W(\langle p, t \rangle)$

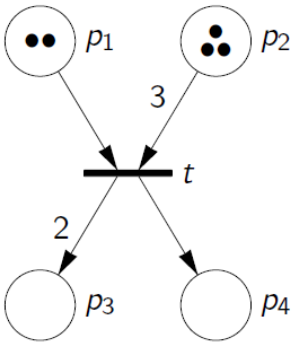
we write $M\ [t\rangle$

Transition firing

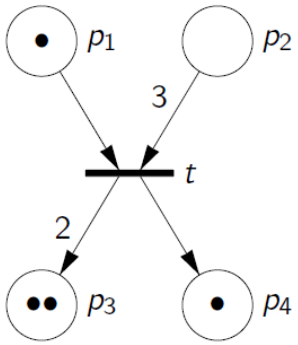
Firing rule

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

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



M0=(2,3,0,0)



M1=(21,0,2,1)

M0[>M1

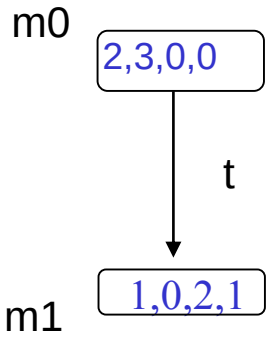
Reachable marking $R_N(M)$:

$$M \in R_N(M)$$

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

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

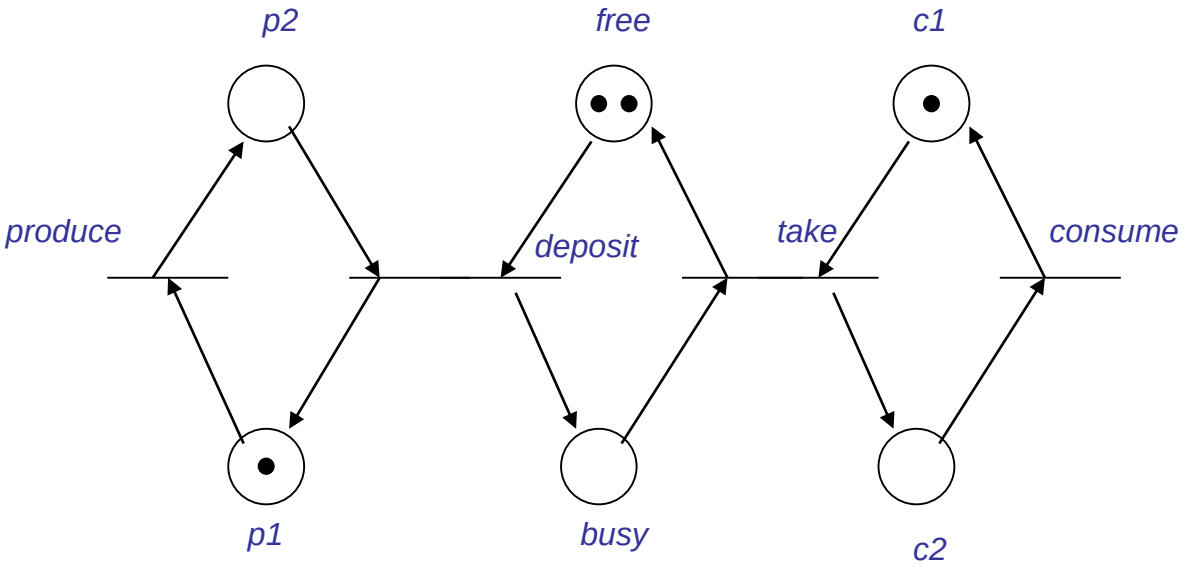
Reachable markings

Transitions never enabled

Conditions on reachable markings

……

Producer/Consumer example



Producer

Buffer
(2 slots)

Consumer

Petri nets

- 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

-assigning a delay (local time d) to each transition

-assigning a global time to the PN

When a transition is enabled a local timer is set to 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:

$$(\tau_{k1}, t_{k1}) \ldots (\tau_{kn}, t_{kn})$$

where

$\tau_{k1} <= \tau_{k2} <= \tau_{kn}$

$[\tau_{ki}, \tau_{ki+1})$ is the period of time between the firing of two transitions
in this period of time the net does not change the marking

STOCHASTIC PETRI NET:
when the delay d of a timed transition is a random variable

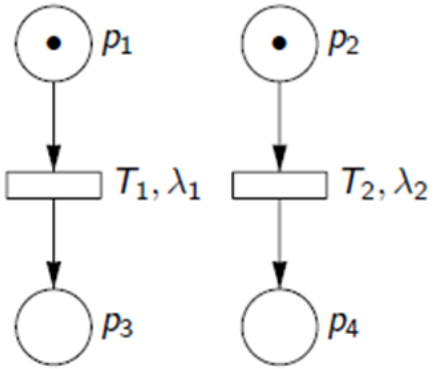
Stochastic Petri nets (SPN)

when the delay is a negative exponential distribution random variable
Reachability graph -> Markov chain

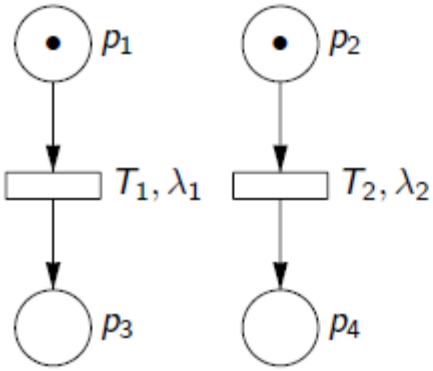
Transition rate: λk is the transition rate (positive real number) of transition Tk

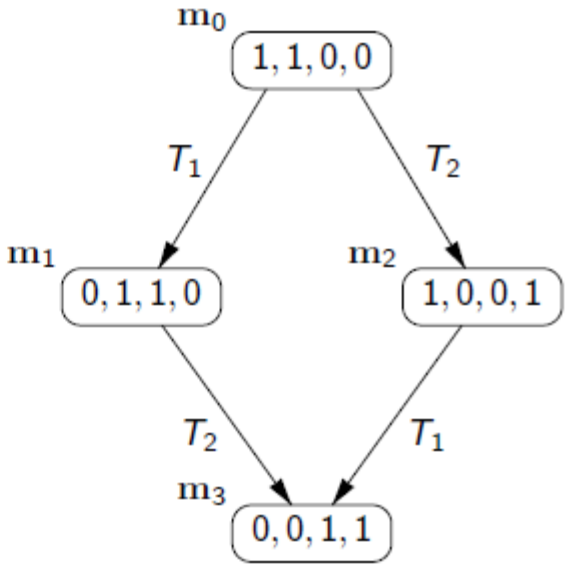
- delay of transition T1 random variable exponentially distributed with parameter $\lambda 1$
- delay of transition T2 random variable exponentially distributed with parameter $\lambda 2$

Assumption: delay of transitions independent



Stochastic Petri nets (SPN) – reachability graph

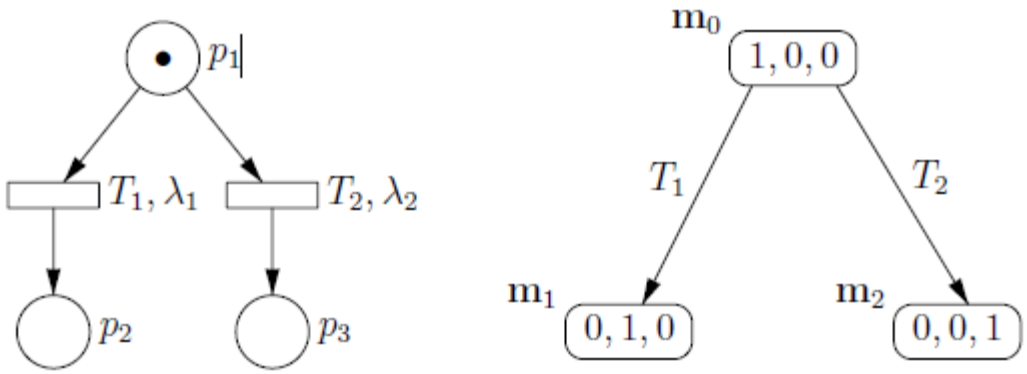




A timed transition T enabled at time t, with d the random value for the transition delay, fires at time t+d
if it remains enabled in the interval [t, t+d)

Stochastic Petri nets (SPN)

- Conflict resolution



The solution of the conflict depends on the delay of T1 and T2

Markov chain

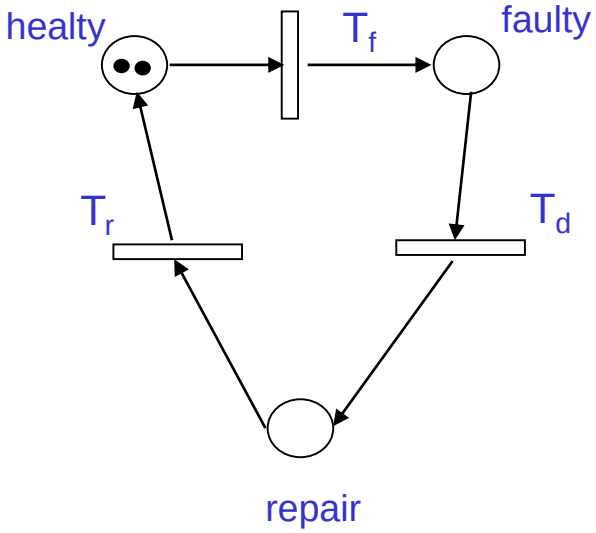
Random process {M(t), t>=0}

with M(0) = M_0 and M(t) the marking at time t

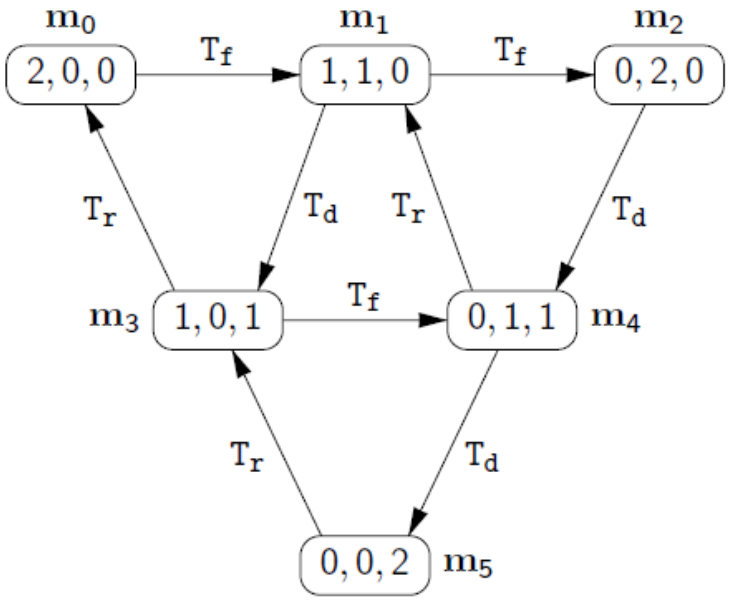
{M(t), t>=0} is a CTMC (memoriless property of exponential distributions)

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 μ



SPN



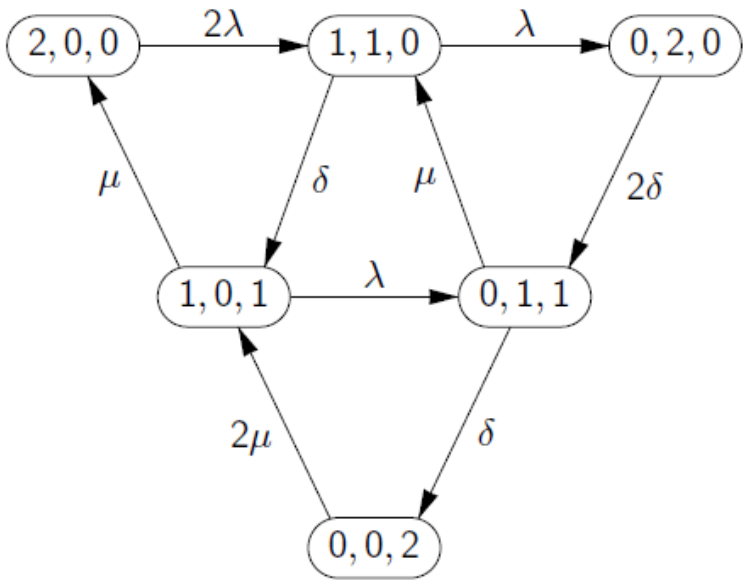
Reachability graph

Markov chain

failure rate = λ

detection rate = δ

repair rate = μ



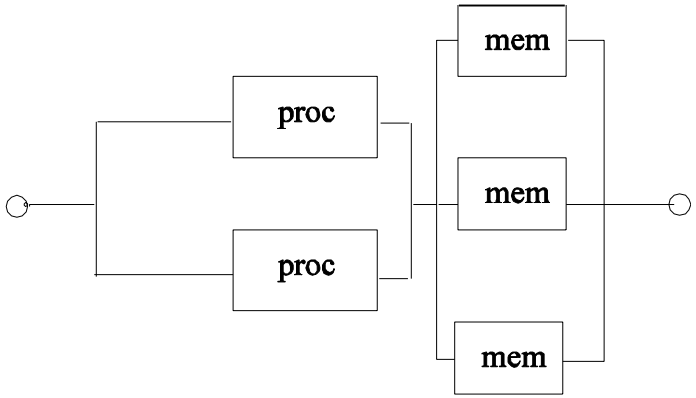
Properties

- 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.

Example

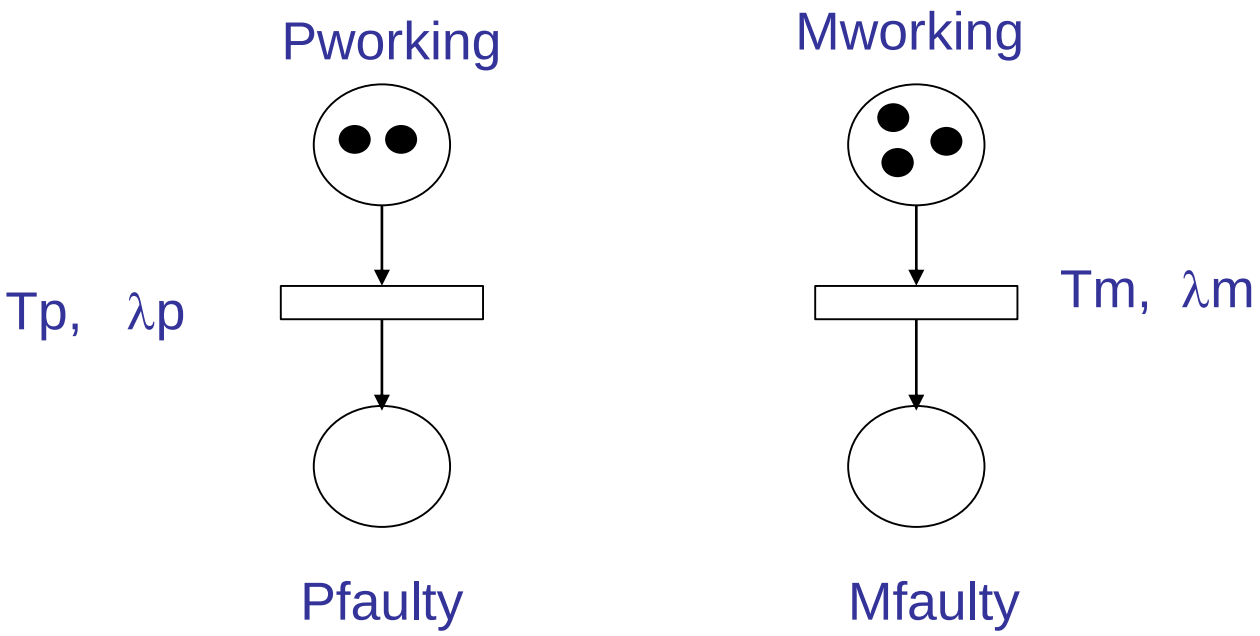
Multiprocessor system with 2 processors and 3 shared memories system. System is operational if at least one processor and one memory are operational.



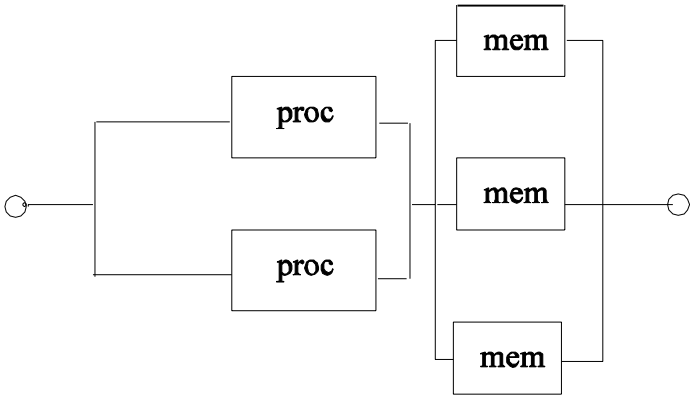
λ_m failure rate for memory

λ_p failure rate for processor

Stochastic Petri net (SPN)



Markov chain



Number of working processors
and number of working memories

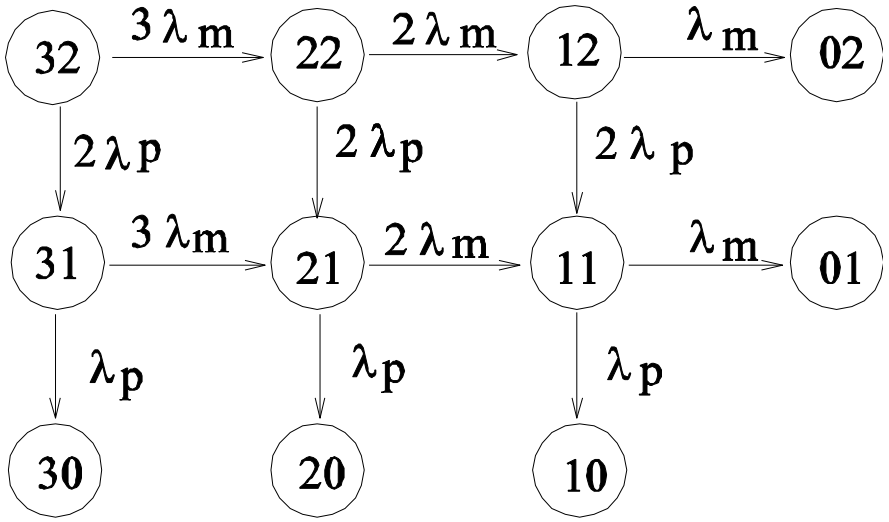
Given a state (i, j):

i is the number of operational memories;

j is the number of operational processors

S = {(3,2), (3,1), (3,0), (2,2), (2,1), (2,0), (1,2), (1,1), (1,0), (0,2), (0,1)}

Markov chain



λ_m failure rate for memory
λ_p failure rate for processor

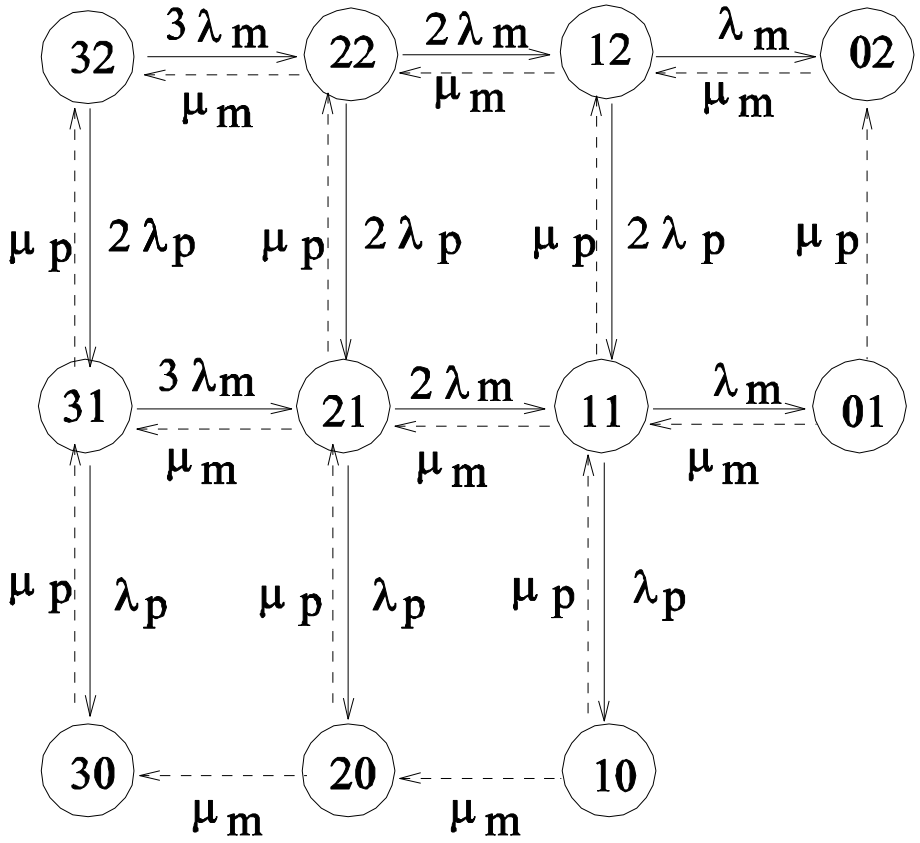
(3, 2) -> (2,2) after a memory fault

(3,0), (2,0), (1,0), (0,2), (0,1) are absorbent states

Availability modeling

- Assume that faulty components are replaced and we evaluate the probability that the system is operational at time t
- Constant repair rate μ (number of expected repairs in a unit of time)
- Strategy of repair:
  only one processor or one memory at a time can be substituted

Markov chain modelling the 2 processors and 3 shared memories system with repair.

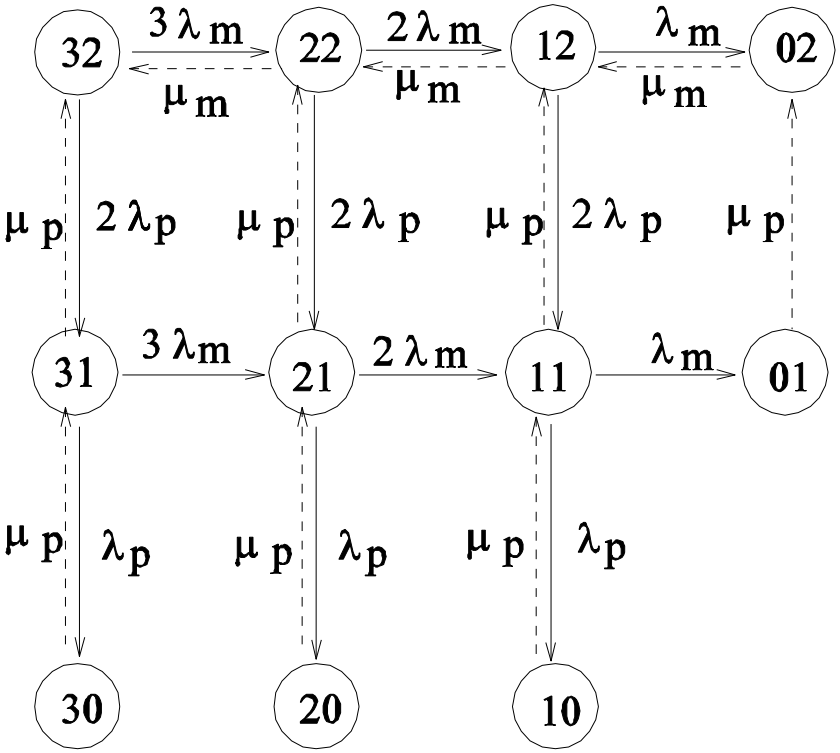


λm failure rate for memory
λp failure rate for processor
μm repair rate for memory
μp repair rate for processor

- Strategy of repair:

  only one component can be substituted at a time

  processor higher priority



exclude the lines μm representing memory repair in the case where there has been a process failure

Stochastic Activity Networks (SANs)

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.

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:

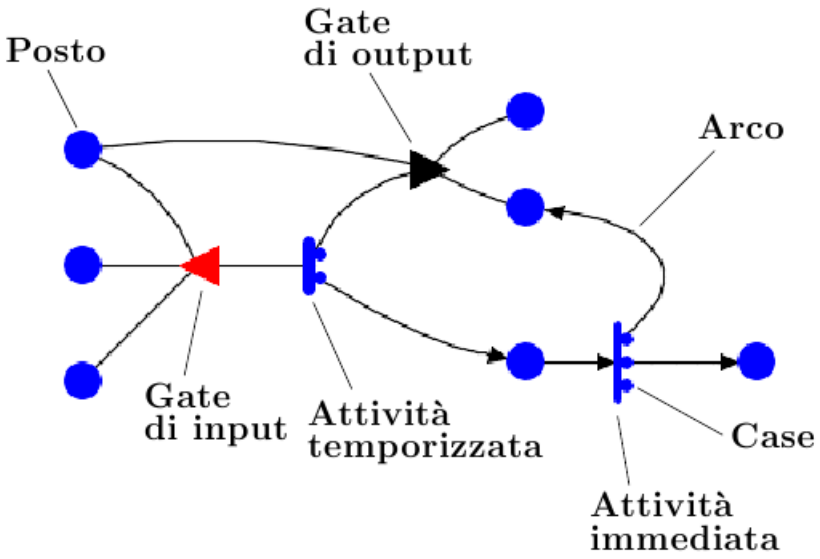
output function

Cases and case distribution:

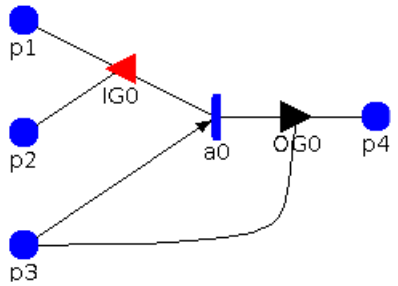
instantaneous or timed activity may have mutually exclusive outcomes, called cases, chosen probabilistically according to the case distribution of the activity. Cases can be used to model probabilistic behaviors.

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

Stochastic activity networks



Example

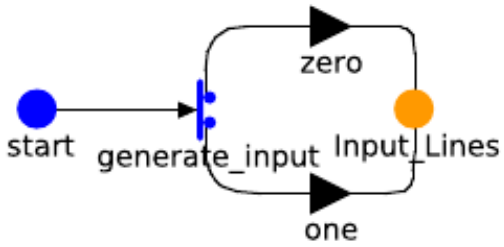


Gate	Definition
IG0	Enabling predicate if ((p1 → Mark() > 0 ∧ p2 → Mark() ==0) || (p1→ Mark()==0 ∧ p2 → Mark()> 0)) return 1; else return 0;
OG0	Output function p4 → Mark()++; if (p3 → Mark() == 0) p3 → Mark()=1;

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;

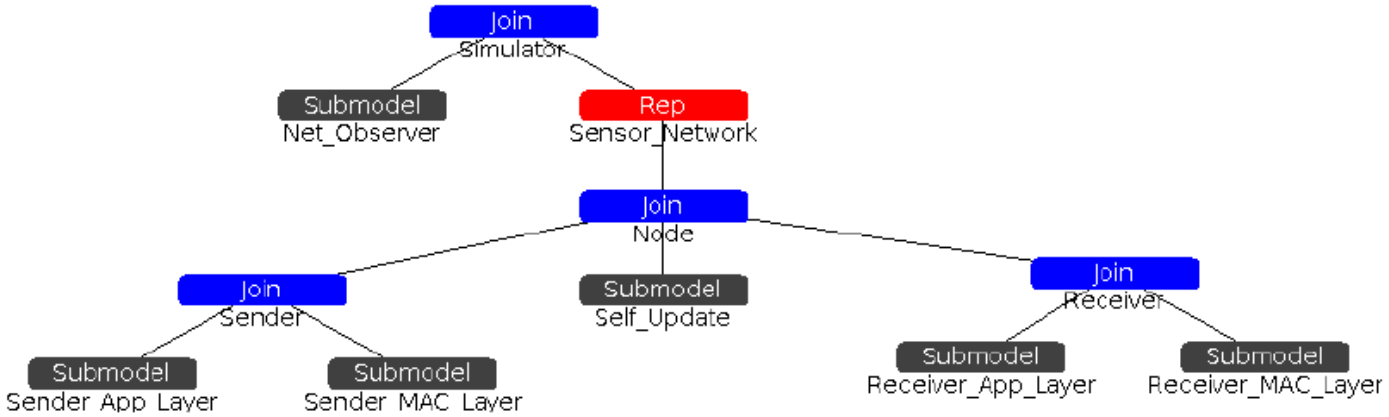
(ii) customised measures of system properties;

(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
sensor network protocol

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

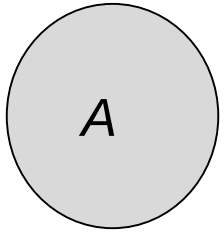
Simulation and Reward functions

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

ASSIGNING REWARDS TO STATES or TRANSITIONS defines a Markov reward model.

RATE rewards (in the case of assignment to states)
or
IMPLULSE rewards (in the case of assignment to transition)

Simulation and Reward functions



RATE rewards

Assume state A in the model is the only state that represents the failure of the system

Consider the performance variable Rel, which measures reliability

Generally the value of the performance variable is the **mean** of the values returned by the associated reward function

The reward funtion is:

```
if  (Model-name->A->Mark()==0) return 1;
else return 0;
```

Simulation and 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.

Confidence interval: accuracy of the measure of the property

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

Mobius tool: https://www.mobius.illinois.edu/