Petri nets

Petri nets

- Petri nets: High-level modelling formalism for concurrent and distributed systems (Petri, Carl A. (Ph. D. Thesis), University of Bonn, 1962)

- Stochastic Petri nets

  temporal and probabilistic information added to the model

  the approach aimed at equivalence between SPN and MC
  idea of associating an exponentially distributed random delay with the PN transitions (1990)

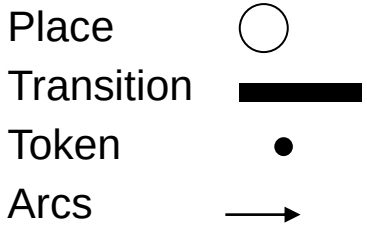
- SAN nets

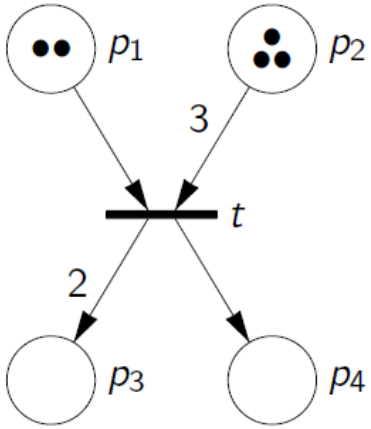
  Stochastic Activity networks
  extension to Stochastic Petri nets (data structure, complex firing rules, global variables, ….) (2001)

Petri nets

A Petri net consists of *places*, *transitions*, and *arcs*. Arcs are from a place to a transition or vice versa.
Places may contain a discrete number of marks called *tokens*.

Graphycally:

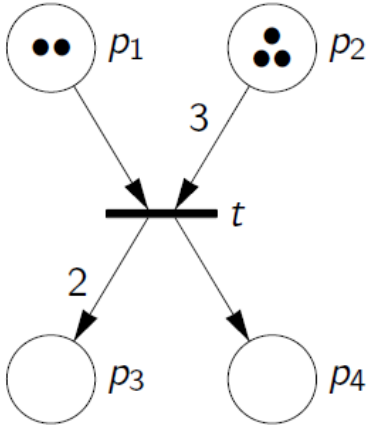




- the places with an arc that runs from the place to a transition are called the *input places* of the transition
- the places with an arc that runs from the transition to a place are called the *output places* of the transition

Petri nets

Marking of the network:
any distribution of tokens over the places. This represents a configuration of the network.



A transition of a Petri net may *fire* if it is *enabled*, i.e. there are sufficient tokens in all of its input places
This is named the *weight* of the arc (by default 1).

When the transition fires, it consumes the required input tokens, and creates tokens in its output places.

Petri nets are well suited for modeling the concurrent behavior of distributed systems.

Petri nets

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

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

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

P: places
T: transitions
F: arcs
W: weight
M_0 : initial marking

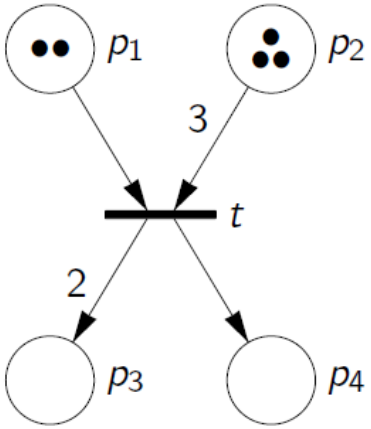
t, y transitions

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



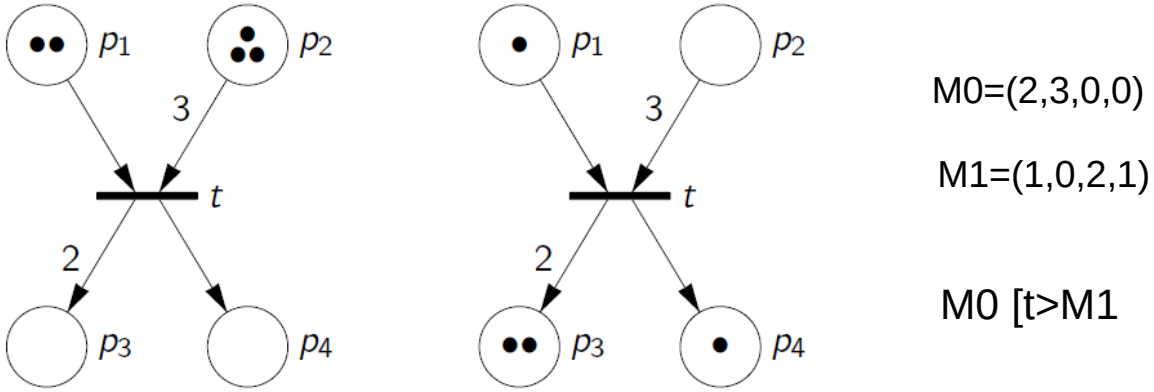
Transition firing

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=(1,0,2,1)

M0 [t>M1

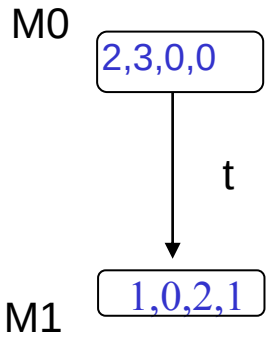
The firing of a transition is atomic (a single non-interruptible step)

Reachable marking $R_N(M)$:

$$M \in R_N(M)$$

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

Reachability graph



Transition sequence

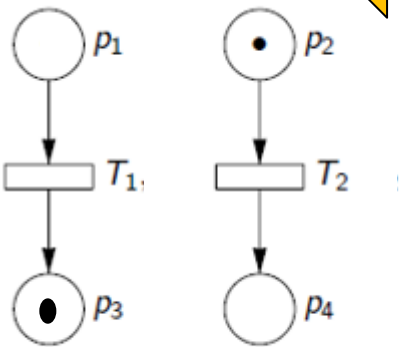
Let $M \,[t_1\rangle\, M' \wedge M' \,[t_2\rangle\, M''$ then $M \,[t_1 t_2\rangle\, M''$

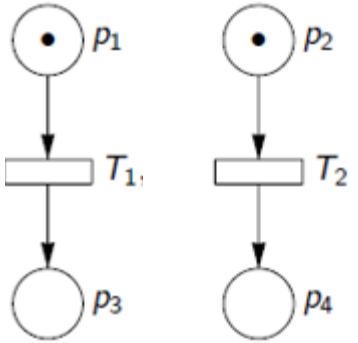
If $M \,[t_1 \ldots t_n\rangle\, M^{(n)}$ then $t_1 \ldots t_n$ is a transition sequence

Evolution of the net

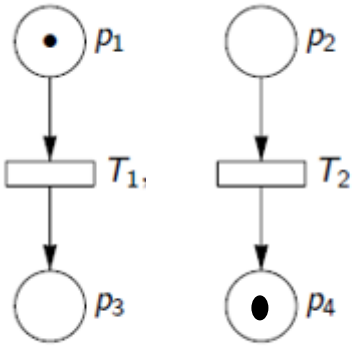
At any time, one of the *enabled* transitions is executed at a time
If multiple transitions are enabled, these transitions will fire in any order. The execution of Petri nets is *nondeterministic*

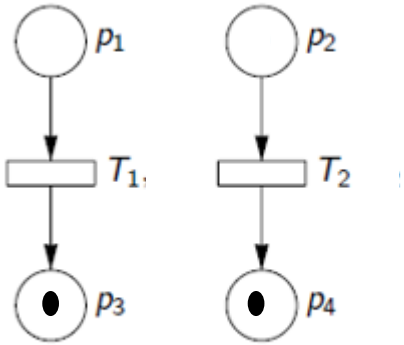
M0 [T1 T2> M**



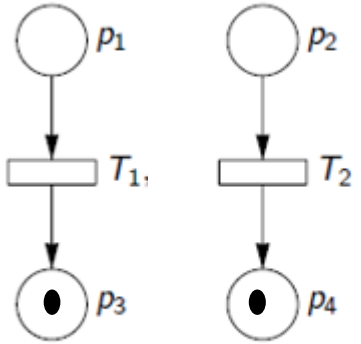


M0 [T2 T1> M''





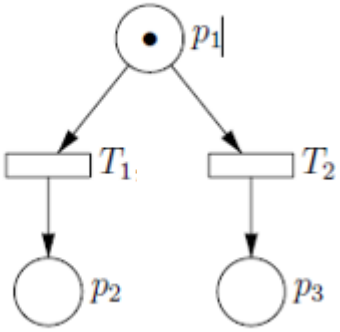
T1, T2 not in conflict
*M** = M''*

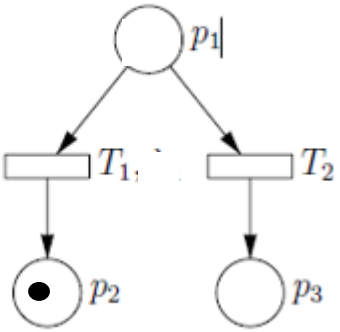


Evolution of the net

Conflict resolution

T1, T2 are in conflict
For example, when T1 fires,
T2 is no more enabled

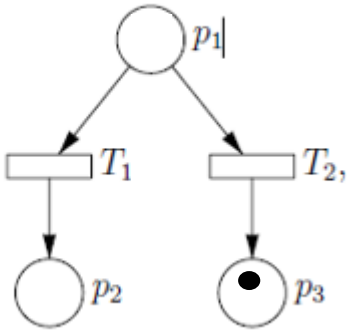




M0=(1,0,0,) M1=(0,1,0)

M0 [T1> M1

T2 not enabled in M1



M0=(1,0,0,) M2=(0,0,1)

M0 [T2> M2

T1 not enabled in M2

Analysis

Simulation

Reachable markings

Conditions on reachable markings

Transition invariants

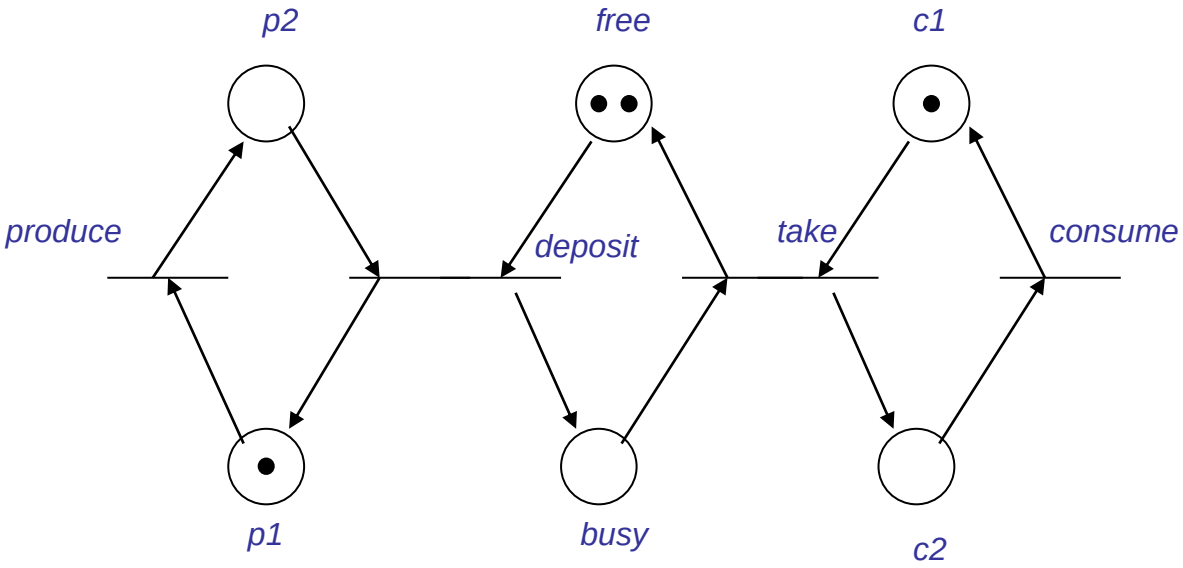
Place invariants

Transitions never enabled

Deadlock

……

Producer/Consumer example



Producer

Buffer
(2 slots)

Consumer

Dependabilty evaluation and 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

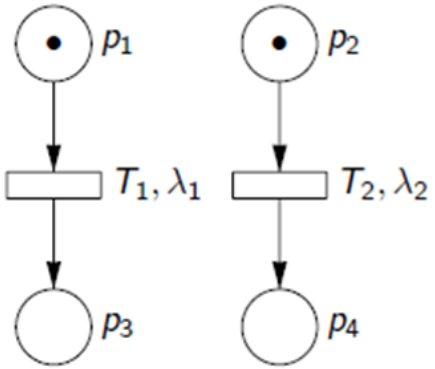
Stochastic Petri nets (SPN)

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

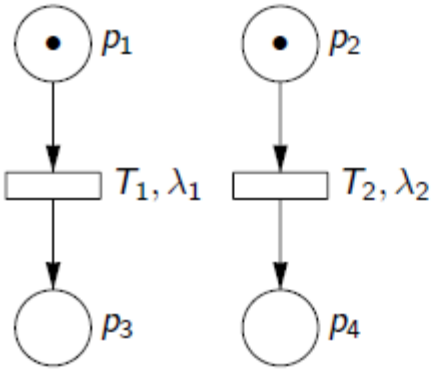
Transition rate: λ_k is the transition rate (positive real number) of transition T_k

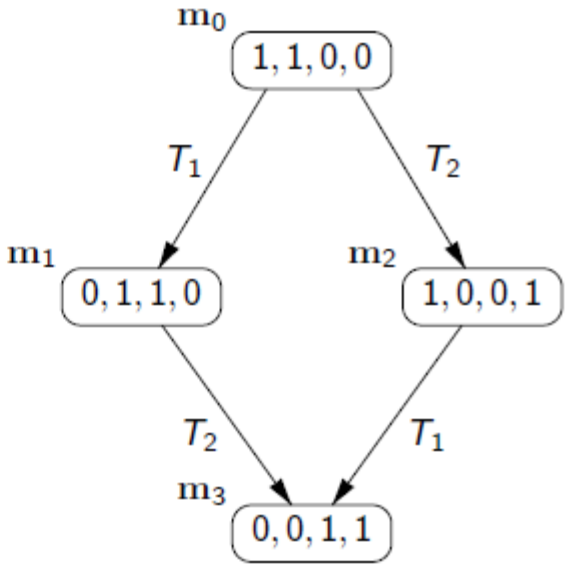
- delay of transition T_1 random variable exponentially distributed with parameter λ_1
- delay of transition T_2 random variable exponentially distributed with parameter λ_2

Assumption: delay of transitions independent



Stochastic Petri nets (SPN) – reachability graph

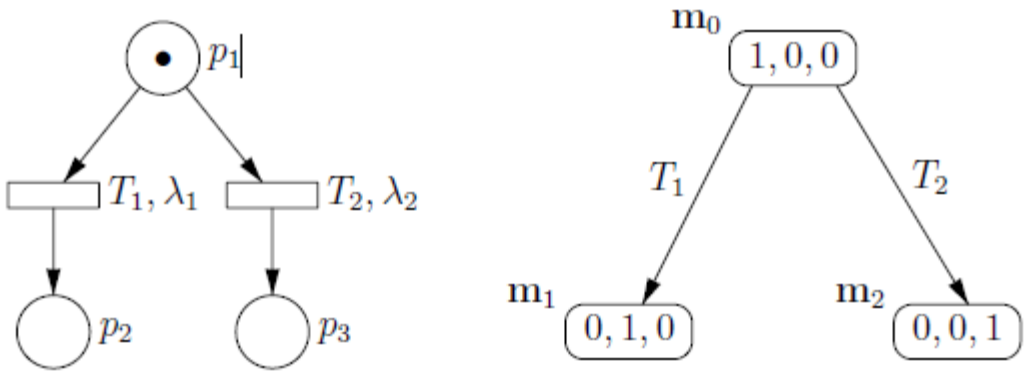




A timed transition T enabled at time t, with d the random value for the transition delay, fires at time t+d
if it remains enabled in the interval [t, t+d)

Stochastic Petri nets (SPN)

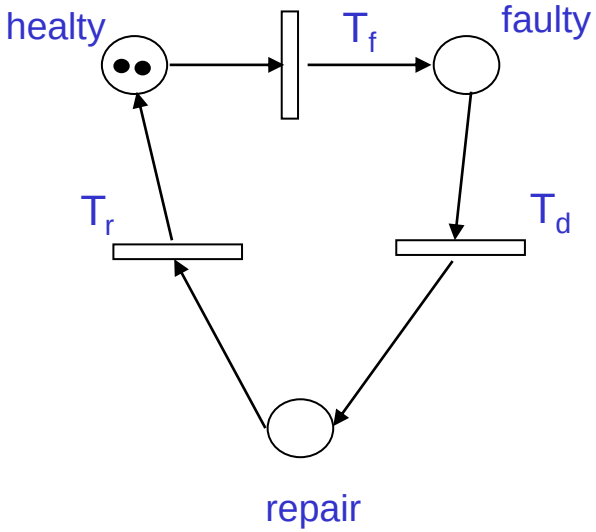
- Conflict resolution



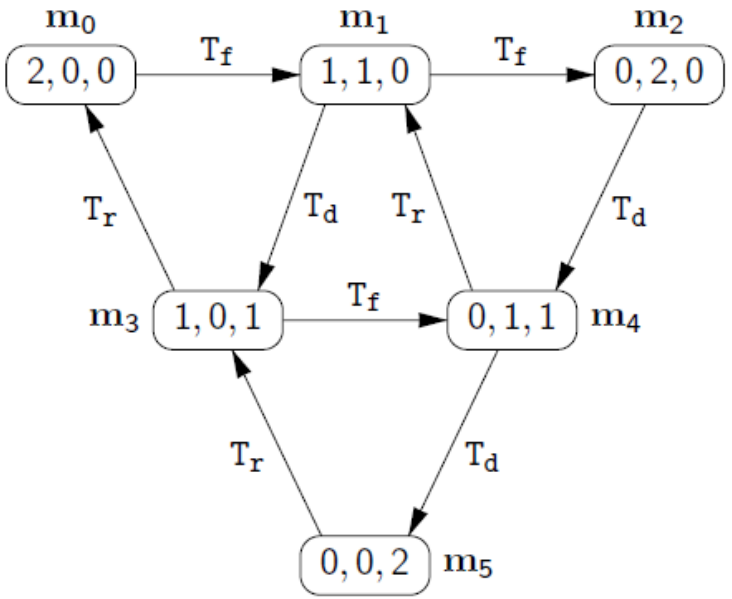
The solution of the conflict depends on the delay of T1 and T2

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



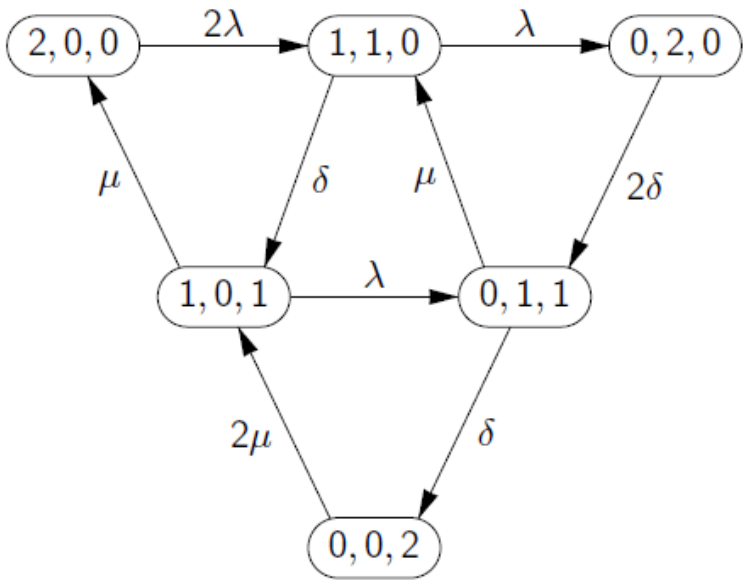
Reachability graph

Markov chain

failure rate = λ

detection rate = δ

repair rate = μ



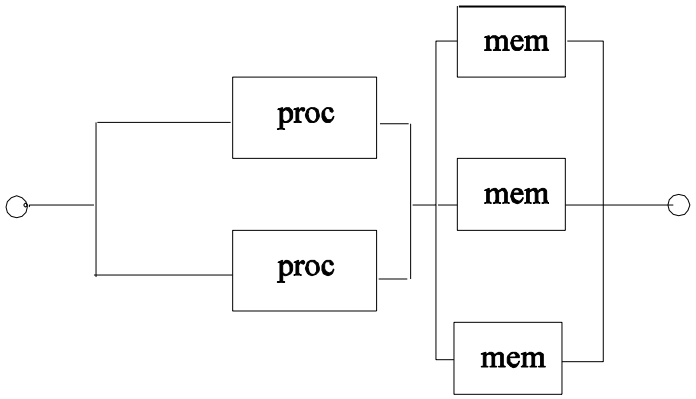
Properties

- 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

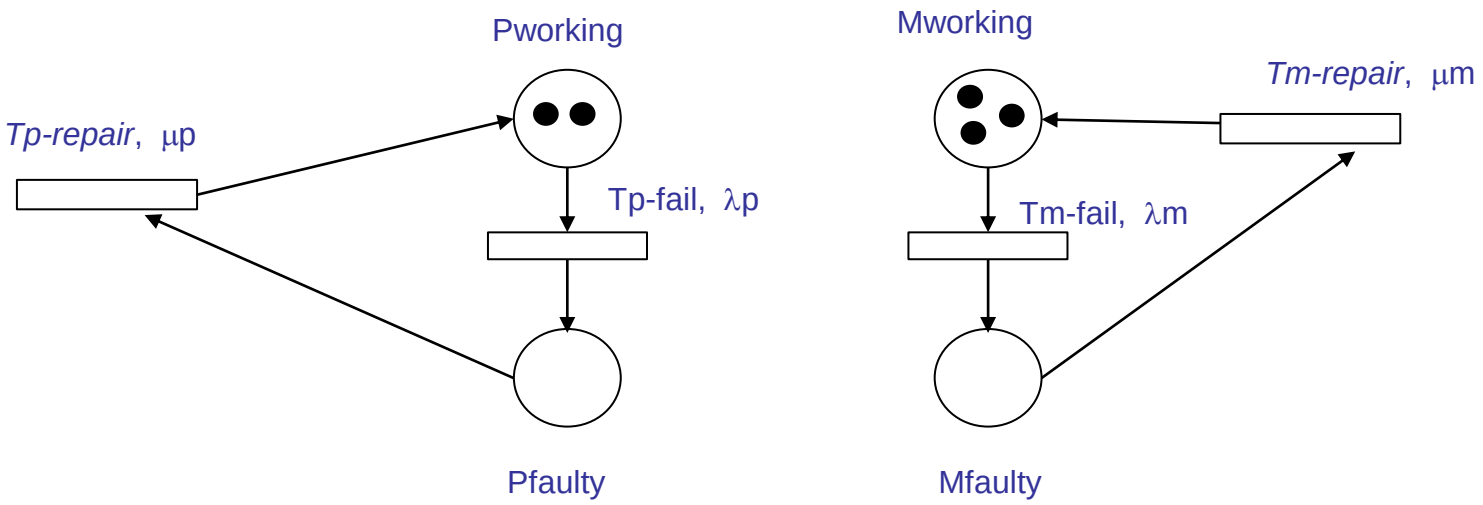
Multiprocessor system with 2 processors and 3 shared memories system. System is operational if at least one processor and one memory are operational.



λ_m failure rate for memory

λ_p failure rate for processor

Stochastic Petri net (SPN)



System state: (Pworking, Mworking, Pfaulty, Mfaulty)

Initial state (2,3,0,0)

(2,3,00) -> Tp-fail (1,3,1,0) -> Tm-fail (1,2,1,1) -> ….

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.

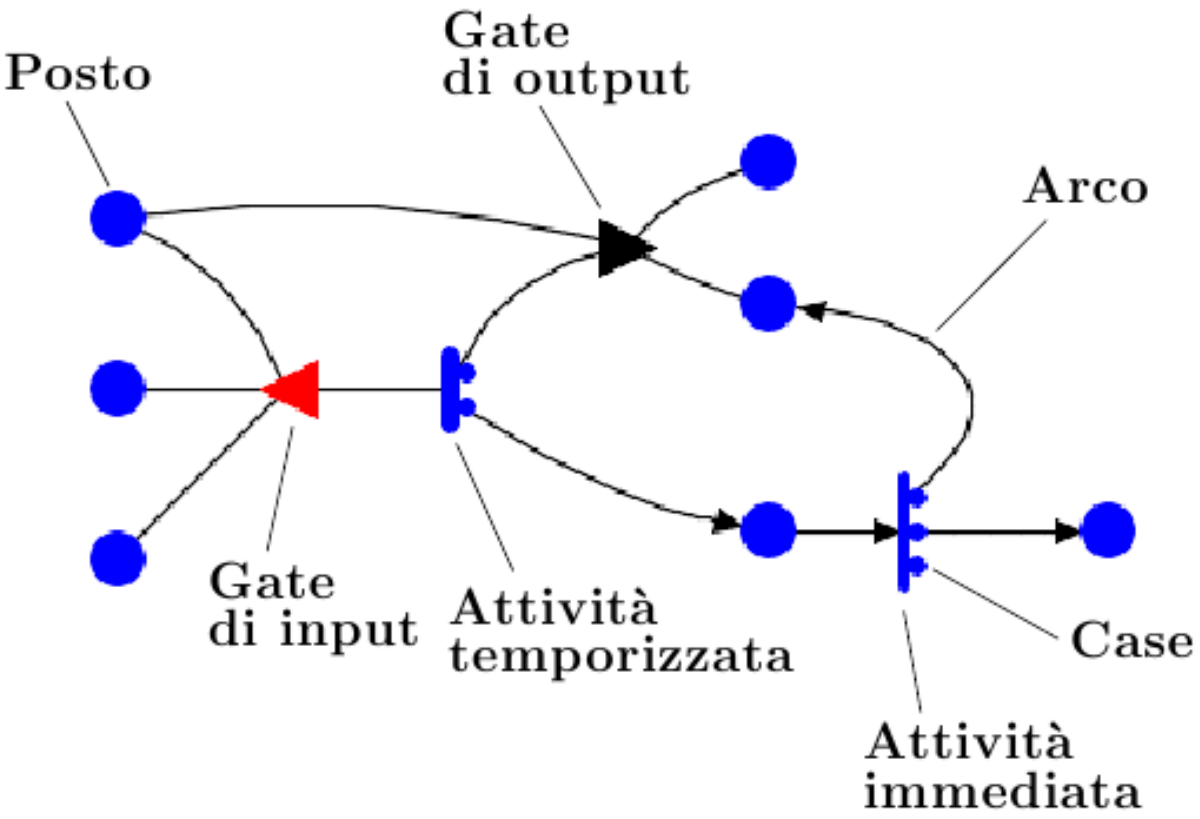
Output gates allow the definition of different next marking rules for the cases of the activity

Other extension to PN:

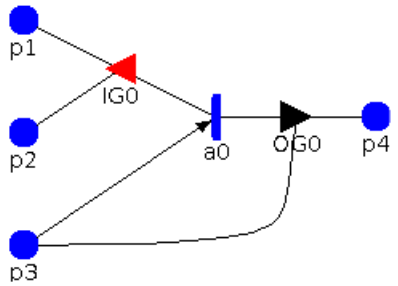
Shared variables
(global objects that can be used to exchange information among modules)

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

Stochastic activity networks



Example

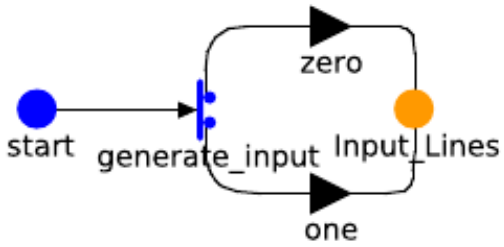


Gate	Definition
IG0	Enabling predicate if ((p1 → Mark() > 0 ∧ p2 → Mark() ==0) || (p1→ Mark()==0 ∧ p2 → Mark()> 0)) return 1; else return 0;
OG0	Output function p4 → Mark()++; if (p3 → Mark() == 0) p3 → Mark()=1;

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.