Performance Modelling — Lecture 8
More about GSPN Models

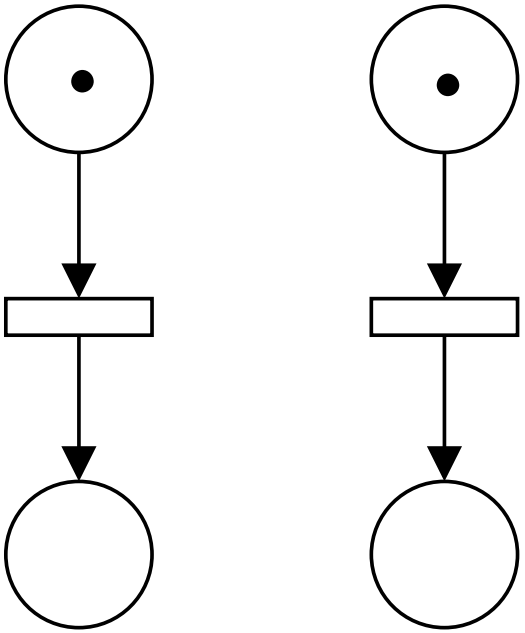
Jane Hillston
School of Informatics
The University of Edinburgh
Scotland

13th February 2017

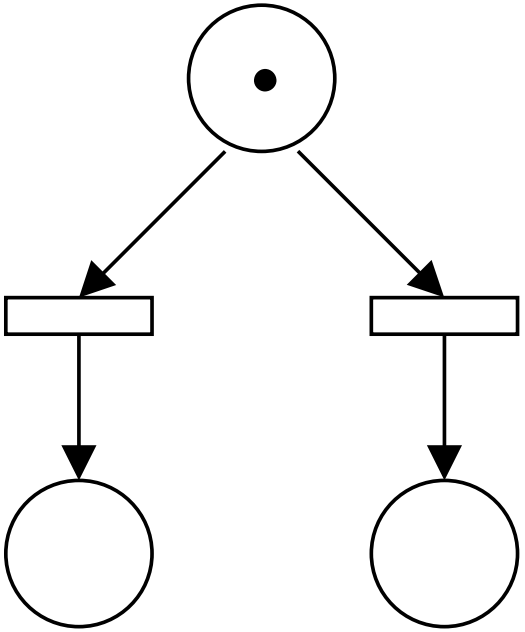
Petri Nets

Petri nets are a formal notation designed for modelling concurrency, causality and conflict.

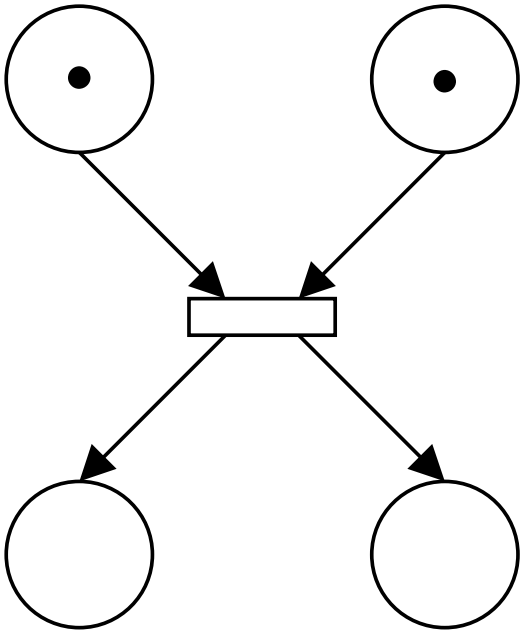
Concurrency



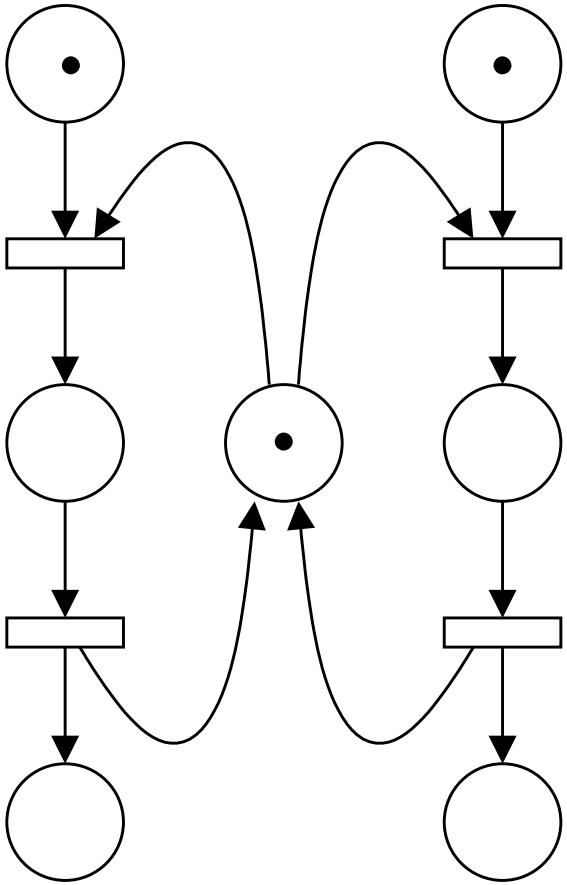
Conflict



Synchronisation



Mutual Exclusion



SPN and GSPN

In a stochastic Petri nets (SPN) we associate an exponentially distributed delay with the firing of each transition.

SPN and GSPN

In a stochastic Petri nets (SPN) we associate an exponentially distributed delay with the firing of each transition.

The delay occurs between when the transition becomes enabled and when it fires; indeed the instantaneous firing will only occur if the transition has remained enabled throughout the delay period.

SPN and GSPN

In a stochastic Petri nets (SPN) we associate an exponentially distributed delay with the firing of each transition.

The delay occurs between when the transition becomes enabled and when it fires; indeed the instantaneous firing will only occur if the transition has remained enabled throughout the delay period.

It is straightforward to show that the reachability graph of a SPN forms the state transition diagram of an underlying Markov process.

SPN and GSPN

In a stochastic Petri nets (SPN) we associate an exponentially distributed delay with the firing of each transition.

The delay occurs between when the transition becomes enabled and when it fires; indeed the instantaneous firing will only occur if the transition has remained enabled throughout the delay period.

It is straightforward to show that the reachability graph of a SPN forms the state transition diagram of an underlying Markov process.

Generalised stochastic Petri nets (GSPN) have additional modelling features which can make the representation of some systems less cumbersome.

SPN and GSPN

In a stochastic Petri nets (SPN) we associate an exponentially distributed delay with the firing of each transition.

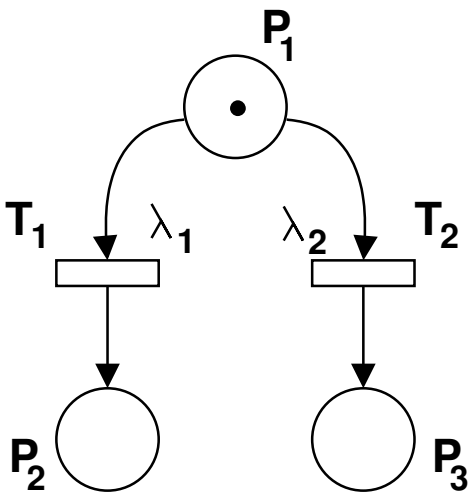
The delay occurs between when the transition becomes enabled and when it fires; indeed the instantaneous firing will only occur if the transition has remained enabled throughout the delay period.

It is straightforward to show that the reachability graph of a SPN forms the state transition diagram of an underlying Markov process.

Generalised stochastic Petri nets (GSPN) have additional modelling features which can make the representation of some systems less cumbersome.

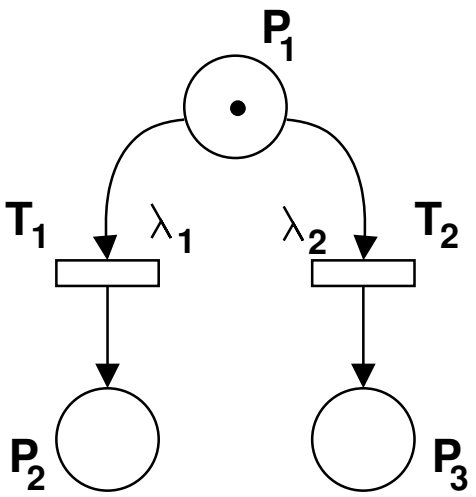
These are immediate transitions and inhibitor arcs.

Conflict Resolution: Timed



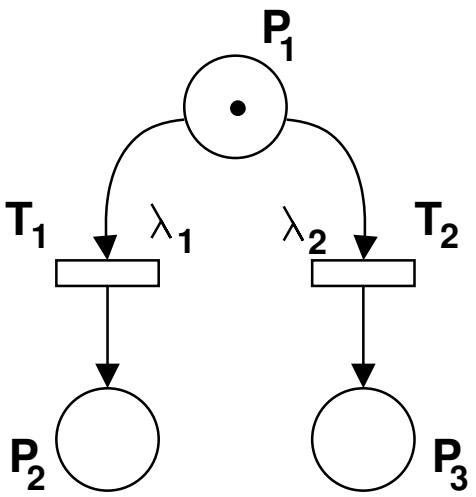
- When P_1 becomes marked both T_1 and T_2 become enabled.

Conflict Resolution: Timed



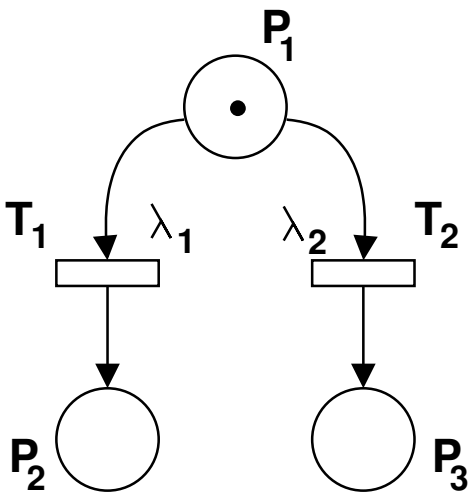
- When P_1 becomes marked both T_1 and T_2 become enabled.
- T_1 and T_2 will both sample their respective distributions to obtain t_1 and t_2.

Conflict Resolution: Timed



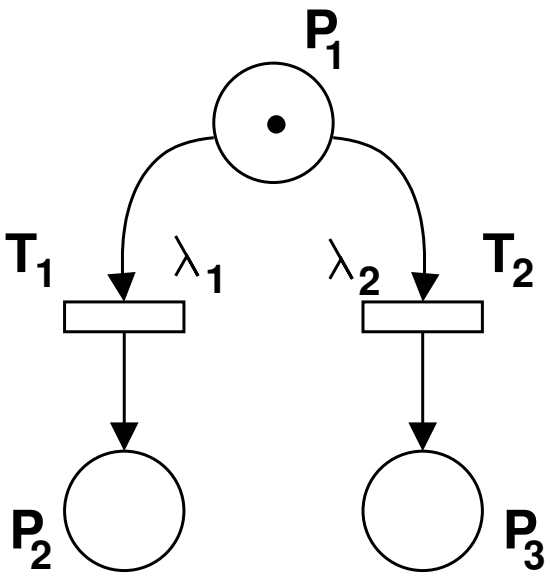
- When P_1 becomes marked both T_1 and T_2 become enabled.
- T_1 and T_2 will both sample their respective distributions to obtain t_1 and t_2.
- If $t_1 < t_2$ transition T_1 will fire after delay t_1

Conflict Resolution: Timed



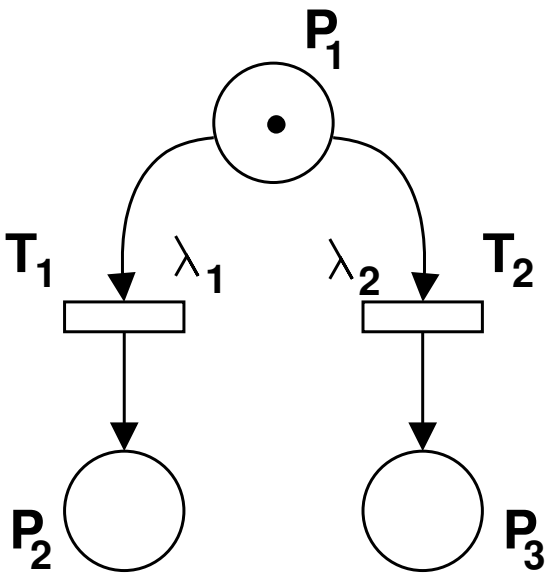
- When P_1 becomes marked both T_1 and T_2 become enabled.
- T_1 and T_2 will both sample their respective distributions to obtain t_1 and t_2.
- If $t_1 < t_2$ transition T_1 will fire after delay t_1
- If $t_2 < t_1$ transition T_2 will fire after delay t_2

Conflict Resolution: Timed



This is the RACE POLICY.

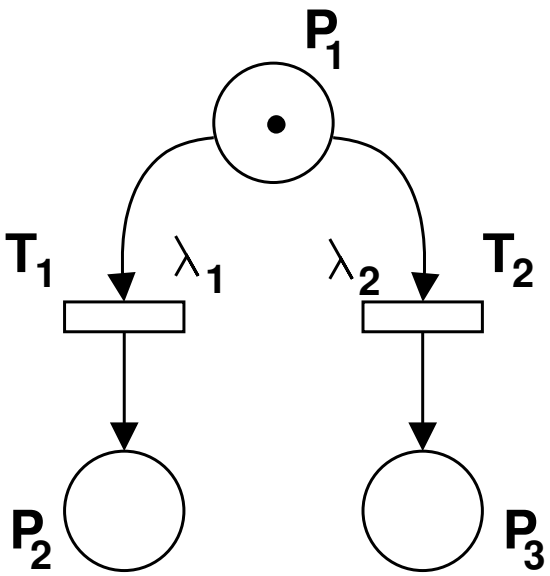
Conflict Resolution: Timed



This is the RACE POLICY.

T_1 will win the race with probability $\dfrac{\lambda_1}{\lambda_1 + \lambda_2}$.

Conflict Resolution: Timed

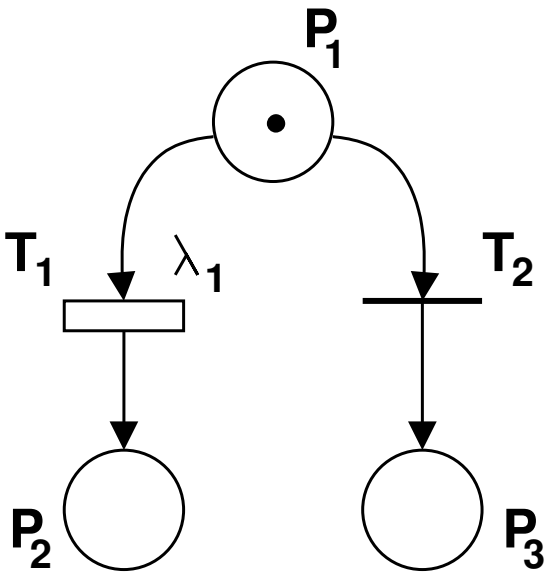


This is the RACE POLICY.

T_1 will win the race with probability $\dfrac{\lambda_1}{\lambda_1 + \lambda_2}$.

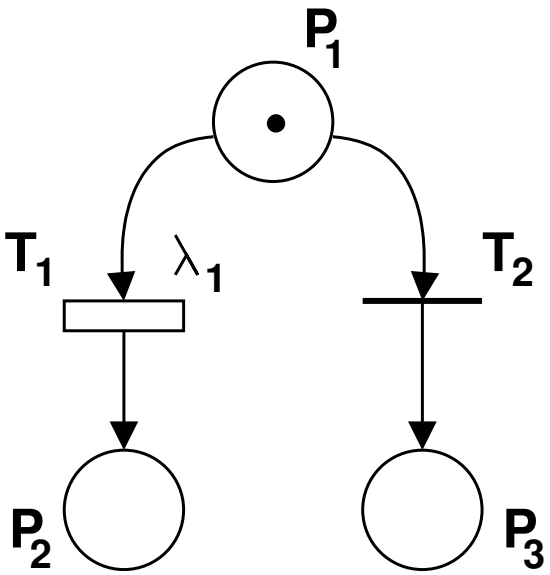
T_2 will win the race with probability $\dfrac{\lambda_2}{\lambda_1 + \lambda_2}$.

Conflict Resolution: Timed and Immediate



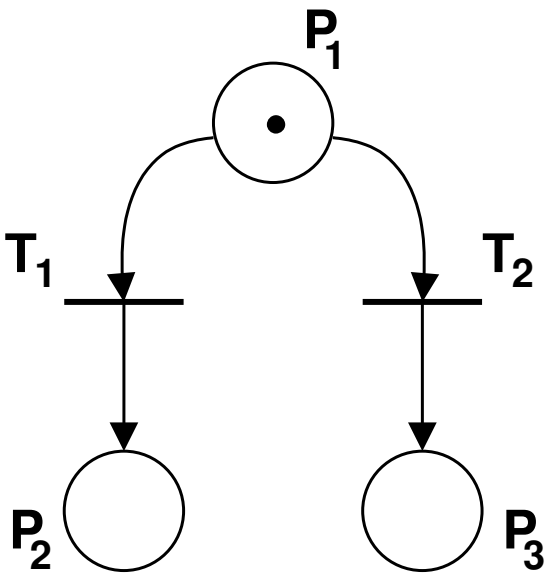
- For a conflict between a timed transition and an immediate transition the immediate one will always win.

Conflict Resolution: Timed and Immediate



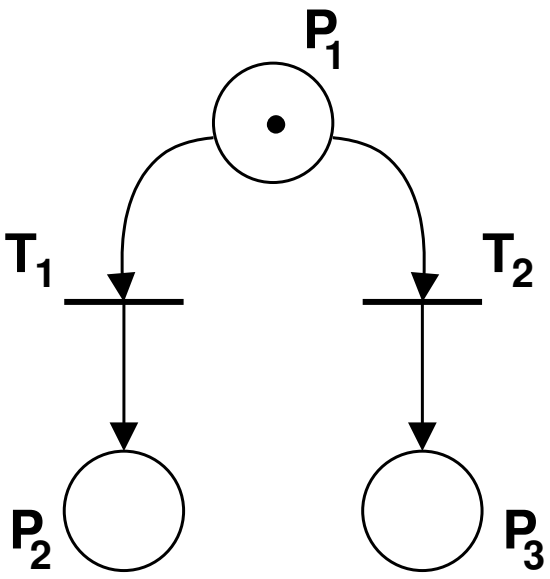
- For a conflict between a timed transition and an immediate transition the immediate one will always win.
- Immediate transitions have higher priority than timed ones and any transition is only enabled if no transitions of higher priority are enabled.

Conflict Resolution: Immediate



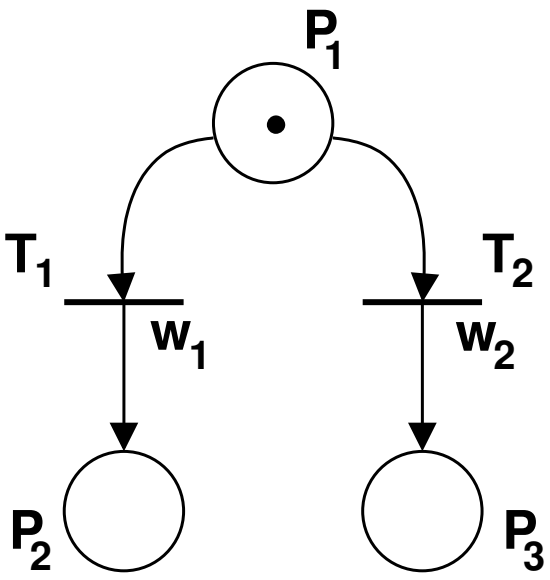
- Since both T_1 and T_2 are immediate transitions we cannot use a race policy.

Conflict Resolution: Immediate



- Since both T_1 and T_2 are immediate transitions we cannot use a race policy.
- How the conflict is to be resolved must be specified by attributing a weight to each transition.

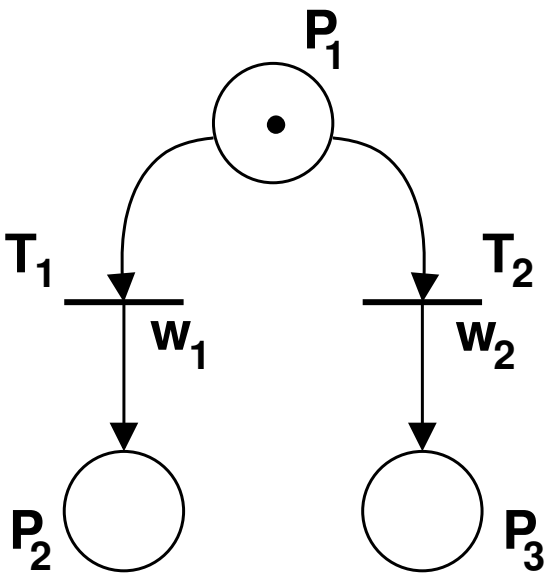
Conflict Resolution: Immediate



- The probability that T_1 fires is

$$\frac{w_1}{w_1 + w_2}.$$

Conflict Resolution: Immediate



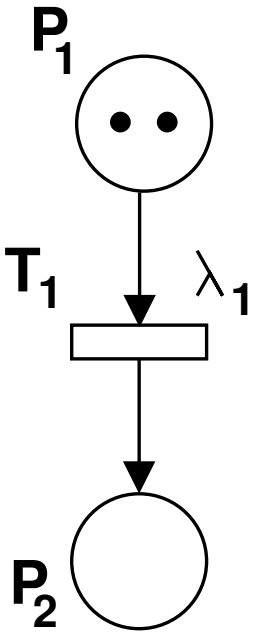
- The probability that T_1 fires is

$$\frac{w_1}{w_1 + w_2}.$$

- The probability that T_2 fires is

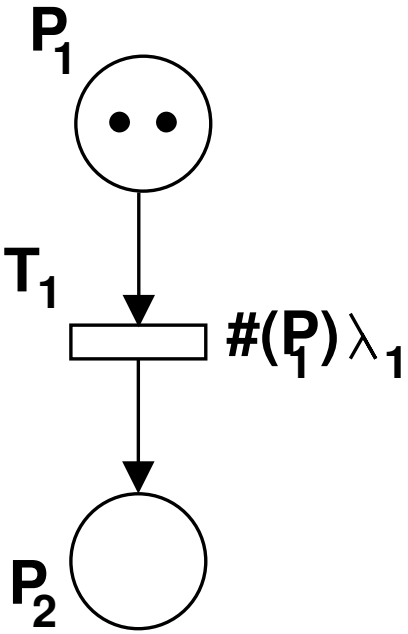
$$\frac{w_2}{w_1 + w_2}.$$

Server Semantics: Single Server



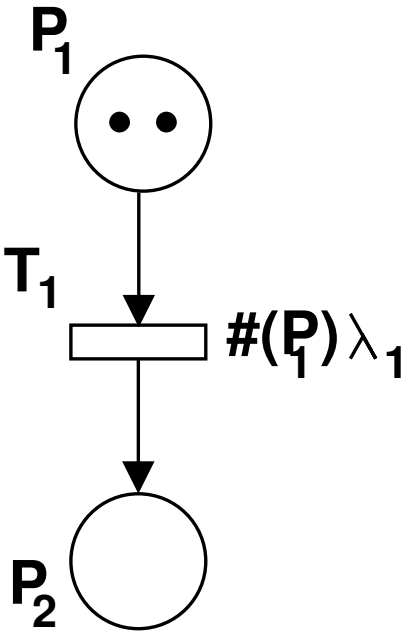
With single server semantics we assume T_1 can "serve" only one token at a time so the two tokens in P_1 are dealt with serially.

Server Semantics: Infinite Server



With infinite server semantics we assume T_1 can "serve" tokens concurrently — as if it provides a separate copy of itself for each token in P_1.

Server Semantics: Infinite Server



With infinite server semantics we assume T_1 can "serve" tokens concurrently — as if it provides a separate copy of itself for each token in P_1.

We then say that T_1 has a marking dependent firing rate.

Server Semantics

Single server semantics is the default.

Reader-Writer Example

- Consider a system in which there is a set of processes who share access to a common database.

Reader-Writer Example

- Consider a system in which there is a set of processes who share access to a common database.
- On any particular access a process may wish to perform either a read or a write.

Reader-Writer Example

- Consider a system in which there is a set of processes who share access to a common database.
- On any particular access a process may wish to perform either a read or a write.
- Any number of readers may access the database concurrently;

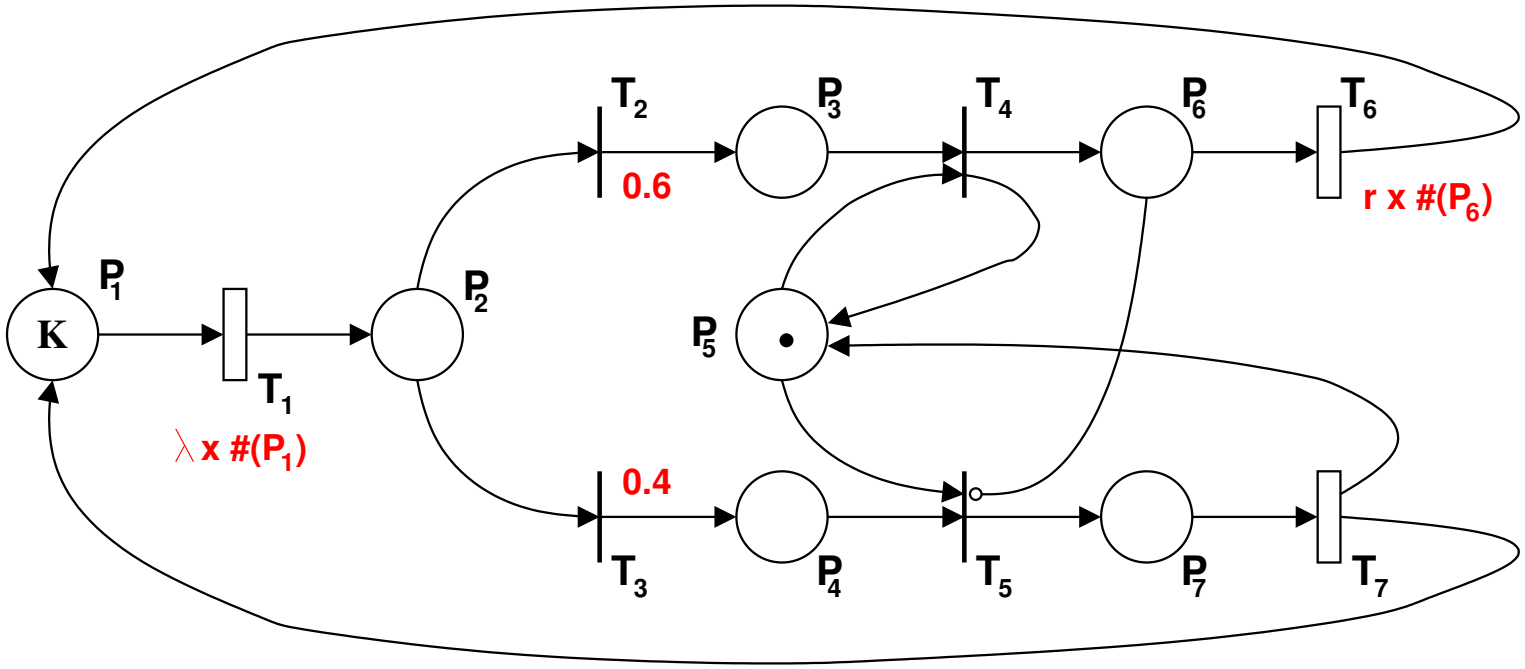
Reader-Writer Example

- Consider a system in which there is a set of processes who share access to a common database.
- On any particular access a process may wish to perform either a read or a write.
- Any number of readers may access the database concurrently;
- a writer requires exclusive access.

Reader-Writer Example

- Consider a system in which there is a set of processes who share access to a common database.
- On any particular access a process may wish to perform either a read or a write.
- Any number of readers may access the database concurrently;
- a writer requires exclusive access.
- Between accesses processes undertake independent work (concurrently).

GSPN model of the reader-writer system



The PIPE Tool

PIPE (Platform Independent Petri net Editor) is an open source, platform independent tool used of the creation and analysis of Petri Nets, and some of their extensions, developed at Imperial College.

The PIPE Tool

PIPE (Platform Independent Petri net Editor) is an open source, platform independent tool used of the creation and analysis of Petri Nets, and some of their extensions, developed at Imperial College.

It is implemented in Java and has a graphical user interface, which makes it very straightforward to use.

Installing PIPE

The most recent version is PIPEv4.3.0 and it is available for download from http://sourceforge.net/projects/pipe2/files/PIPEv4/PIPEv4.3.0/

Installing PIPE

The most recent version is PIPEv4.3.0 and it is available for download from
http://sourceforge.net/projects/pipe2/files/PIPEv4/PIPEv4.3.0/

Once you have unpacked the directory/folder `PIPEv4.3.0`, enter that directory and issue the command

./launch.sh or .\launch.bat

according to your operating system, to launch the PIPE tool.