

# Computing the throughput of probabilistic and replicated streaming applications

Anne Benoit, Fanny Dufossé, Matthieu Gallet, Bruno Gaujal  
and Yves Robert

Laboratoire de l'Informatique du Parallélisme  
Ecole Normale Supérieure de Lyon, France

Réunion Aleae, Bordeaux

# Outline

- 1 Introduction
- 2 Framework
- 3 Timed Petri Nets
- 4 Computing the throughput
- 5 Comparison results
- 6 Conclusion

# Outline

- 1 Introduction
- 2 Framework
- 3 Timed Petri Nets
- 4 Computing the throughput
- 5 Comparison results
- 6 Conclusion

# Problem description

- We are given
  - (i) a streaming application, dependence graph = linear chain;
  - (ii) a one-to-many mapping of application onto heterogeneous platform;
  - (iii) a set of I.I.D. (Independent and Identically-Distributed) variables to model computation/communication time in the mapping.
- How can we compute the throughput of the application, i.e., the rate at which data sets can be processed?
- Two execution models: **Strict** and **Overlap**

# Problem description

- We are given
  - (i) a streaming application, dependence graph = linear chain;
  - (ii) a one-to-many mapping of application onto heterogeneous platform;
  - (iii) a set of I.I.D. (Independent and Identically-Distributed) variables to model computation/communication time in the mapping.
- How can we compute the throughput of the application, i.e., the rate at which data sets can be processed?
- Two execution models: **Strict** and **Overlap**

# Problem description

- We are given
  - (i) a streaming application, dependence graph = linear chain;
  - (ii) a one-to-many mapping of application onto heterogeneous platform;
  - (iii) a set of I.I.D. (Independent and Identically-Distributed) variables to model computation/communication time in the mapping.
- How can we compute the throughput of the application, i.e., the rate at which data sets can be processed?
- Two execution models: **Strict** and **Overlap**

# Motivation

- No replication, i.e., one-to-one mapping: throughput dictated by critical hardware resource
- With replication, deterministic case: surprisingly difficult! (remember previous work, cases with no critical resources)
- Contributions:
  - (i) general method (exponential cost) to compute throughput with I.I.E. exponential laws;
  - (ii) bounds for arbitrary I.I.E. and N.B.U.E. (New Better than Used in Expectation) variables: between exponential and deterministic values;
  - (iii) the problem of finding the optimal mapping is NP-complete.

# Motivation

- No replication, i.e., one-to-one mapping: throughput dictated by critical hardware resource
- With replication, deterministic case: surprisingly difficult! (remember previous work, cases with no critical resources)
- Contributions:
  - (i) general method (exponential cost) to compute throughput with I.I.E. exponential laws;
  - (ii) bounds for arbitrary I.I.E. and N.B.U.E. (New Better than Used in Expectation) variables: between exponential and deterministic values;
  - (iii) the problem of finding the optimal mapping is NP-complete.

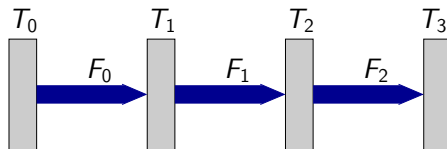


# Outline

- 1 Introduction
- 2 Framework**
- 3 Timed Petri Nets
- 4 Computing the throughput
- 5 Comparison results
- 6 Conclusion

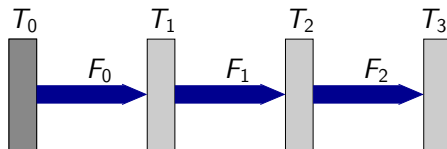
# Application

- A linear workflow with many instances



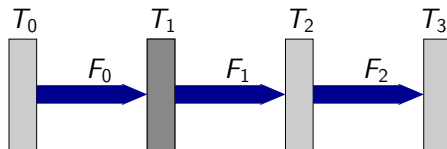
# Application

- A linear workflow with many instances



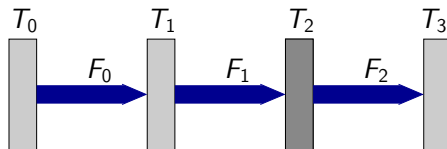
# Application

- A linear workflow with many instances



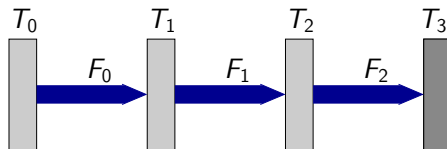
# Application

- A linear workflow with many instances



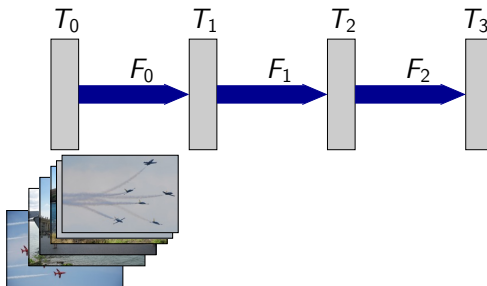
# Application

- A linear workflow with many instances



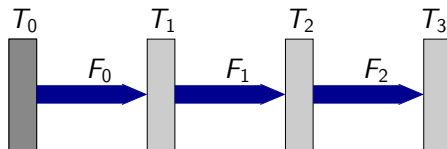
# Application

- A linear workflow with many instances



# Application

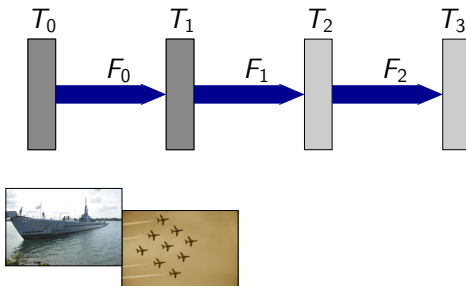
- A linear workflow with many instances





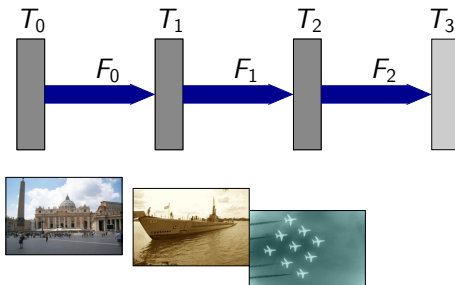
# Application

- A linear workflow with many instances



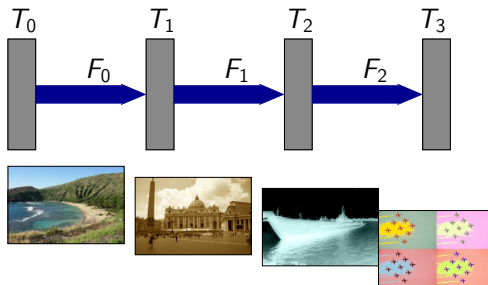
# Application

- A linear workflow with many instances



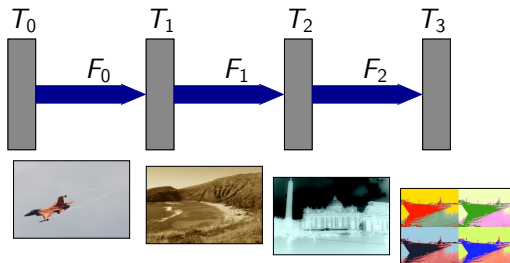
# Application

- A linear workflow with many instances



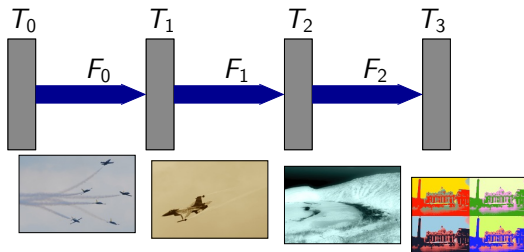
# Application

- A linear workflow with many instances



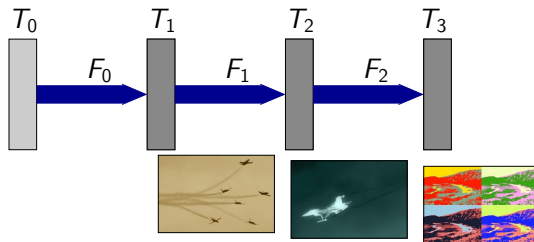
# Application

- A linear workflow with many instances



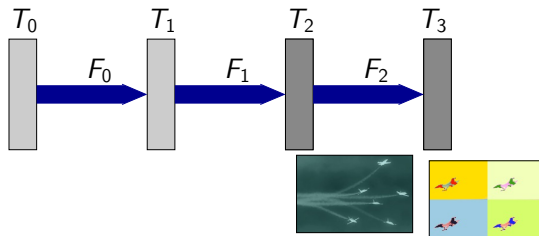
# Application

- A linear workflow with many instances



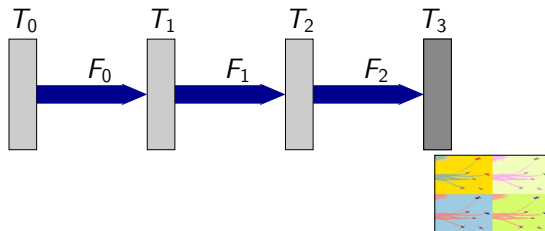
# Application

- A linear workflow with many instances



# Application

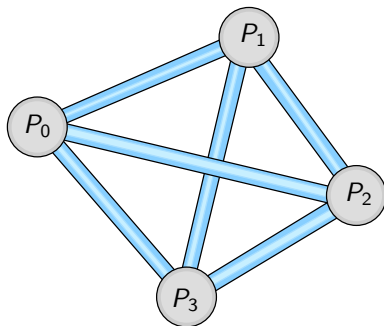
- A linear workflow with many instances





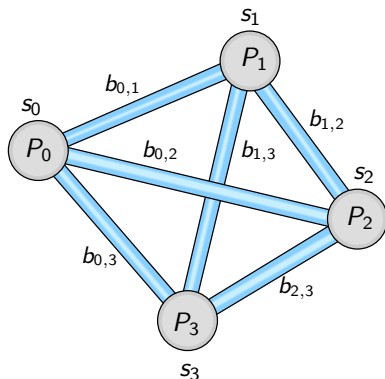
# Platform

- A fully connected platform
- Heterogeneous processors and communication links



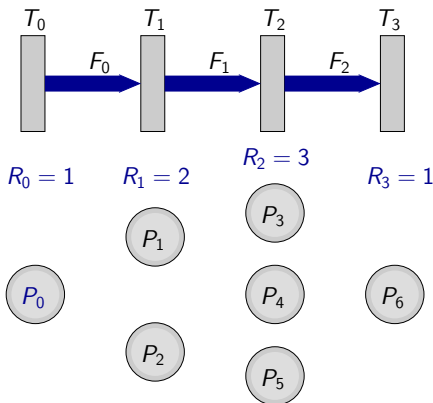
# Platform

- A fully connected platform
- Heterogeneous processors and communication links



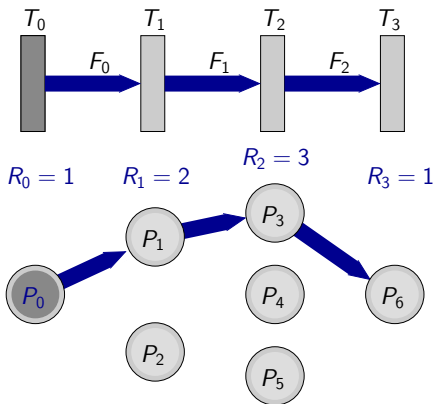
# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task



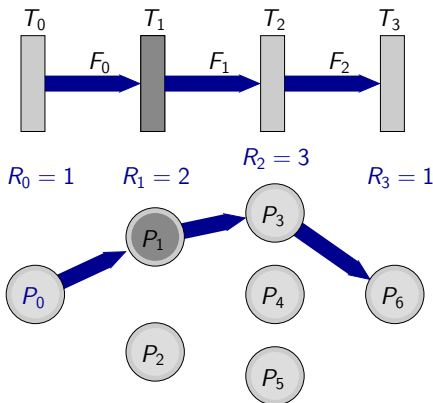
# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task



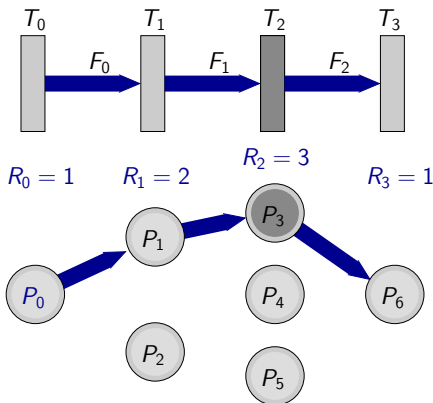
# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task



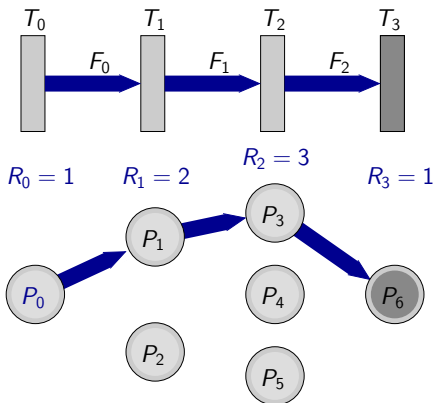
# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task



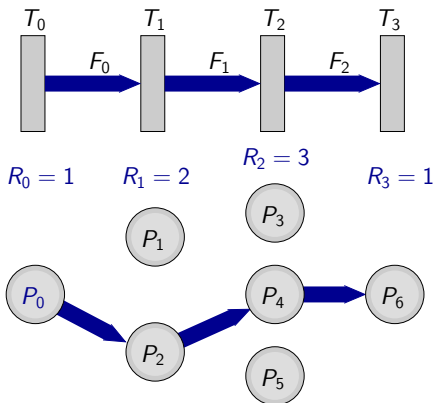
# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task



# Mapping

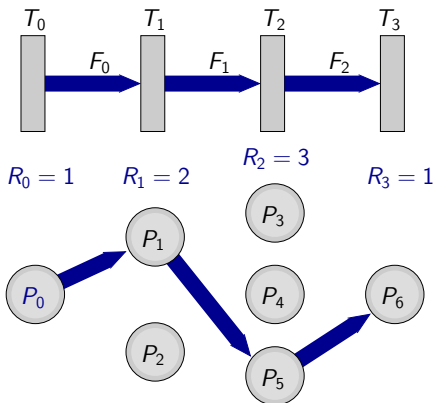
- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task





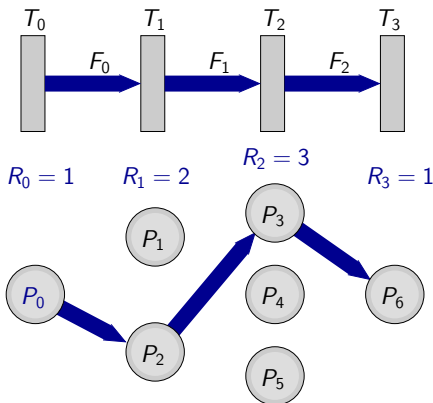
# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task



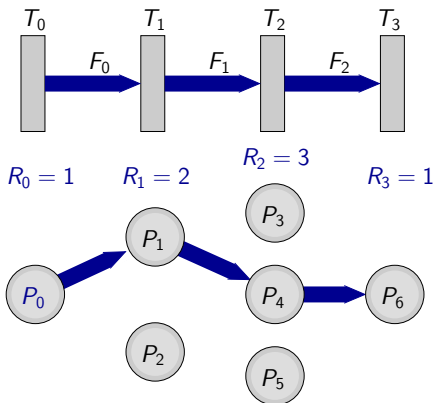
# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task



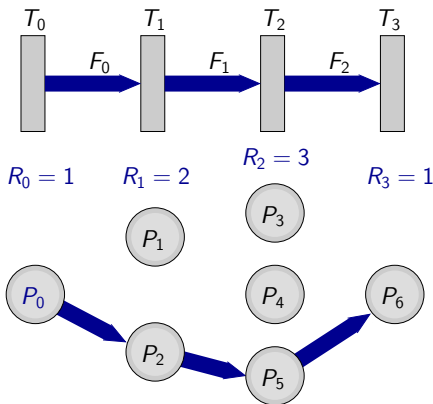
# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task



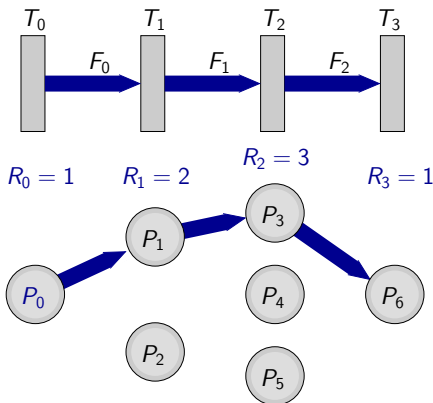
# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task



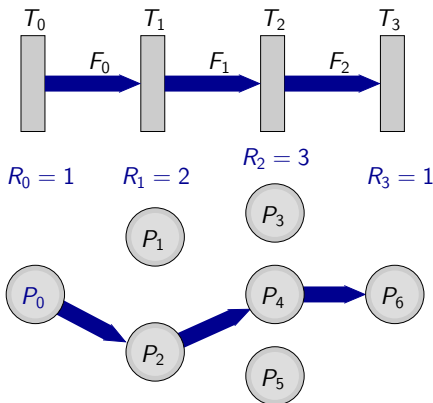
# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task



# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task



# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task

Input data	Path in the system
0	$P_0 \rightarrow P_1 \rightarrow P_3 \rightarrow P_6$
1	$P_0 \rightarrow P_2 \rightarrow P_4 \rightarrow P_6$
2	$P_0 \rightarrow P_1 \rightarrow P_5 \rightarrow P_6$
3	$P_0 \rightarrow P_2 \rightarrow P_3 \rightarrow P_6$
4	$P_0 \rightarrow P_1 \rightarrow P_4 \rightarrow P_6$
5	$P_0 \rightarrow P_2 \rightarrow P_5 \rightarrow P_6$
6	$P_0 \rightarrow P_1 \rightarrow P_3 \rightarrow P_6$
7	$P_0 \rightarrow P_2 \rightarrow P_4 \rightarrow P_6$

# Mapping

- A processor processes at most 1 task
- A task is mapped on possibly many processors
- Replication count of  $T_i$ :  $R_i$
- Round-Robin distribution of each task

## Theorem

*Assume that stage  $T_i$  is mapped onto  $R_i$  distinct processors. Then the number of paths is equal to  $R = \text{lcm}(R_0, \dots, R_{n-1})$ .*



# Communication models

- **Strict:**  
receptions, computations and transmissions are sequential
- **Overlap:**  
overlap of computations by communications

# Communication models

- **Strict:**  
receptions, computations and transmissions are sequential
- **Overlap:**  
overlap of computations by communications

# Random variables

- $X_i(n)$ : time required by  $P_i$  to process its  $n$ -th data set
- $Y_{i,j}(n)$ : time required by  $P_i$  to send its  $n$ -th file to  $P_j$
- Deterministic case
- Exponential variables
- I.I.D.: Independent and Identically-Distributed variables
- N.B.U.E.: New Better than Used in Expectation variables

# Random variables

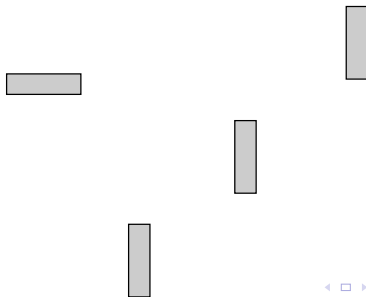
- $X_i(n)$ : time required by  $P_i$  to process its  $n$ -th data set
- $Y_{i,j}(n)$ : time required by  $P_i$  to send its  $n$ -th file to  $P_j$
- Deterministic case
- Exponential variables
- I.I.D.: Independent and Identically-Distributed variables
- N.B.U.E.: New Better than Used in Expectation variables

# Outline

- 1 Introduction
- 2 Framework
- 3 Timed Petri Nets**
- 4 Computing the throughput
- 5 Comparison results
- 6 Conclusion

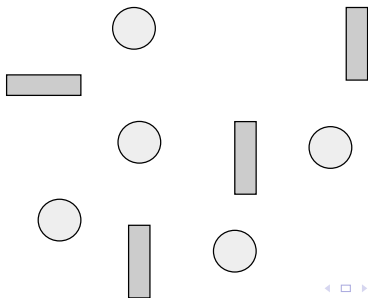
# Short presentation of Timed Petri Nets (TPN)

- Some transitions
- Some places
- Connections between transitions and places... and between places and transitions
- Some tokens allowing transitions to be fired
- Time between the consumption of the input tokens and the creation of the output tokens



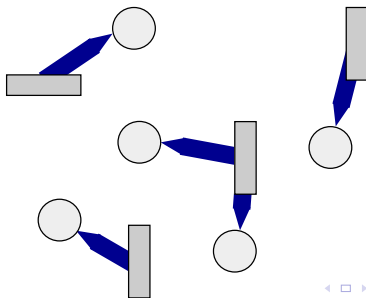
# Short presentation of Timed Petri Nets (TPN)

- Some transitions
- Some places
- Connections between transitions and places... and between places and transitions
- Some tokens allowing transitions to be fired
- Time between the consumption of the input tokens and the creation of the output tokens



# Short presentation of Timed Petri Nets (TPN)

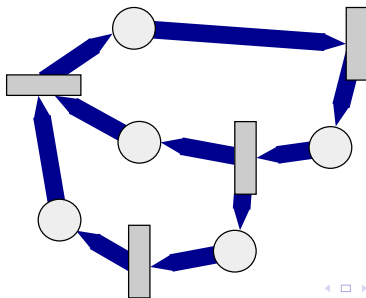
- Some transitions
- Some places
- Connections between transitions and places... and between places and transitions
- Some tokens allowing transitions to be fired
- Time between the consumption of the input tokens and the creation of the output tokens





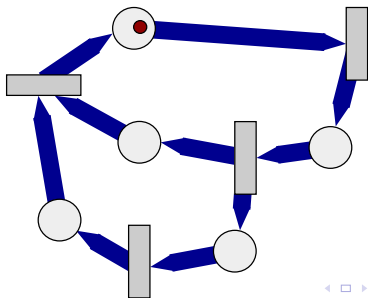
# Short presentation of Timed Petri Nets (TPN)

- Some transitions
- Some places
- Connections between transitions and places... and between places and transitions
- Some tokens allowing transitions to be fired
- Time between the consumption of the input tokens and the creation of the output tokens



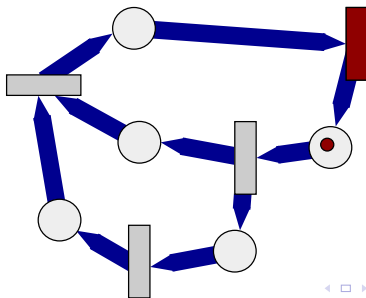
# Short presentation of Timed Petri Nets (TPN)

- Some transitions
- Some places
- Connections between transitions and places... and between places and transitions
- Some tokens allowing transitions to be fired
- Time between the consumption of the input tokens and the creation of the output tokens



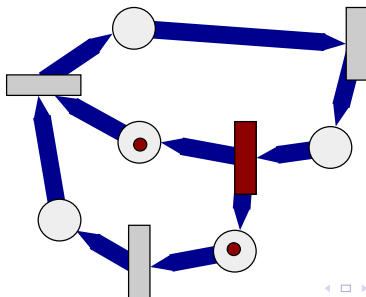
# Short presentation of Timed Petri Nets (TPN)

- Some transitions
- Some places
- Connections between transitions and places. . . and between places and transitions
- Some tokens allowing transitions to be fired
- Time between the consumption of the input tokens and the creation of the output tokens



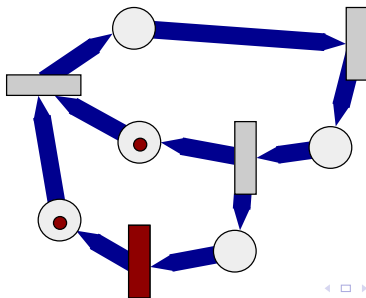
# Short presentation of Timed Petri Nets (TPN)

- Some transitions
- Some places
- Connections between transitions and places... and between places and transitions
- Some tokens allowing transitions to be fired
- Time between the consumption of the input tokens and the creation of the output tokens



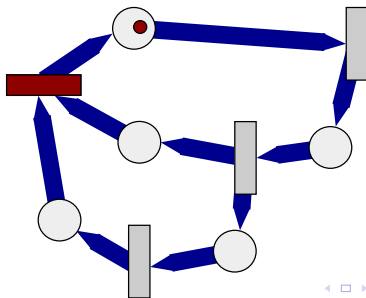
# Short presentation of Timed Petri Nets (TPN)

- Some transitions
- Some places
- Connections between transitions and places... and between places and transitions
- Some tokens allowing transitions to be fired
- Time between the consumption of the input tokens and the creation of the output tokens



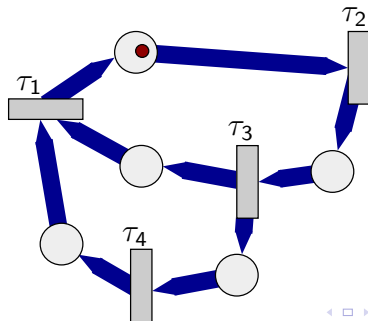
# Short presentation of Timed Petri Nets (TPN)

- Some transitions
- Some places
- Connections between transitions and places... and between places and transitions
- Some tokens allowing transitions to be fired
- Time between the consumption of the input tokens and the creation of the output tokens



# Short presentation of Timed Petri Nets (TPN)

- Some transitions
- Some places
- Connections between transitions and places. . . and between places and transitions
- Some tokens allowing transitions to be fired
- Time between the consumption of the input tokens and the creation of the output tokens



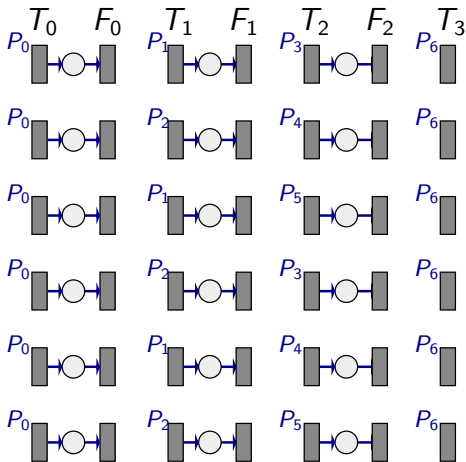
# Timed Event Graph model

- Transitions: communications and computations
- Places: dependences between two successive operations
- Each path followed by the input data must be fully developed in the TPN
- Exponential size of the TPN



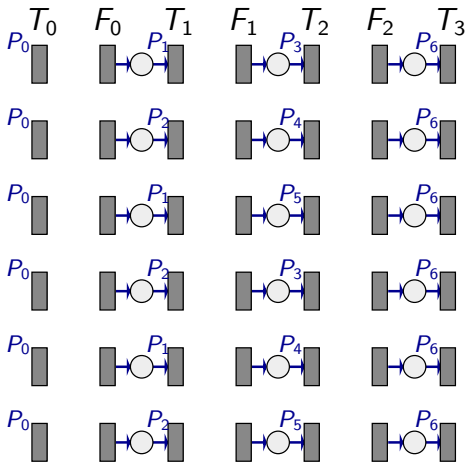
# Overlap model

A communication cannot begin before the end of the computation



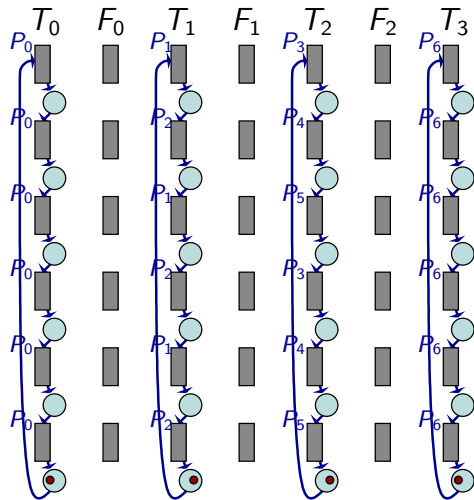
# Overlap model

A computation cannot begin before the end of the communication



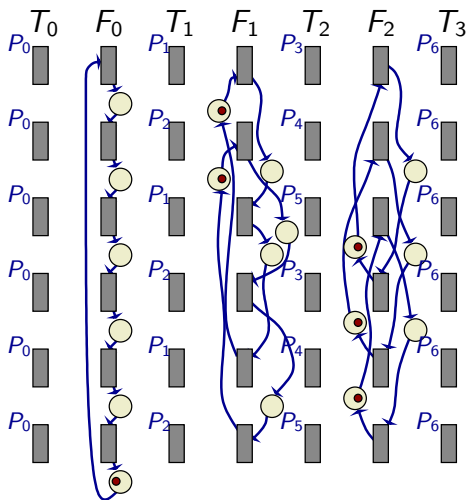
# Overlap model

Dependences due to the round-robin distribution of computations



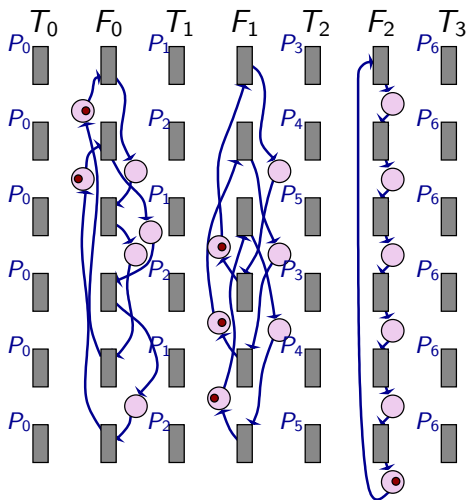
# Overlap model

Dependencies due to the round-robin distribution of outgoing communications



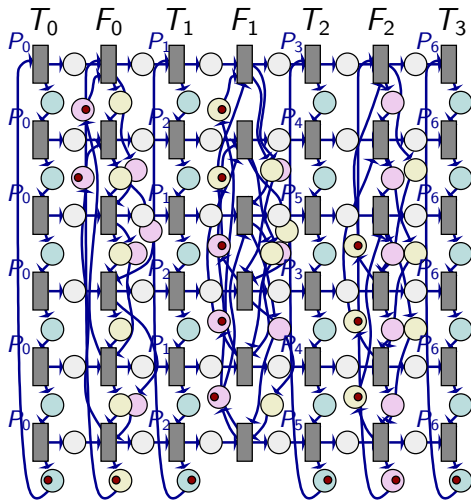
# Overlap model

Dependences due to the round-robin distribution of incoming communications



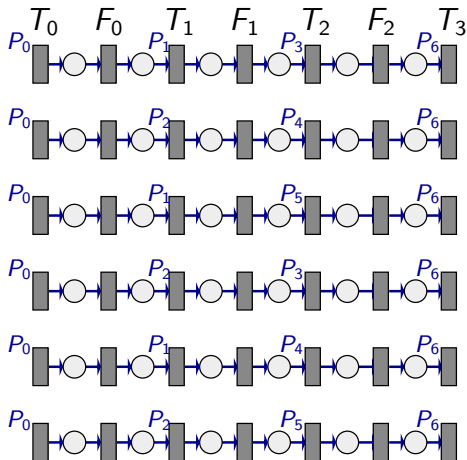
# Overlap model

All dependences!



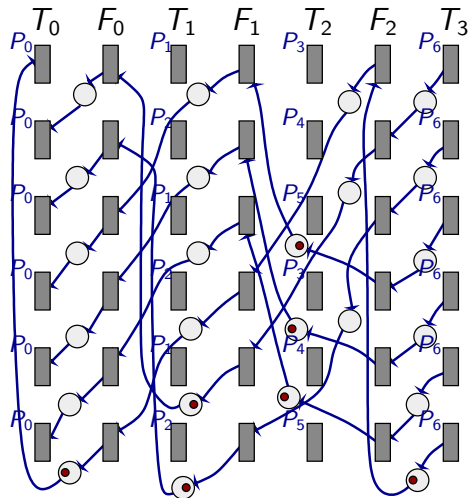
# Strict model

Dependences between communications and computations



# Strict model

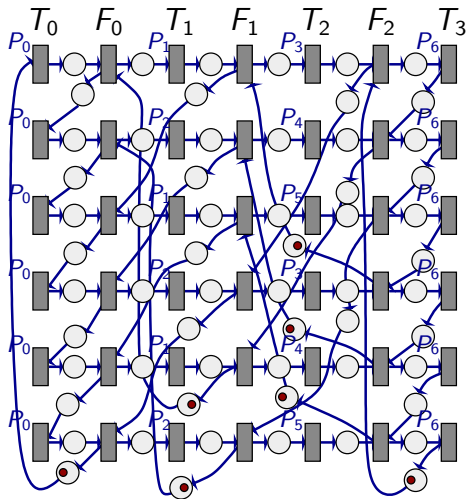
Dependencies due to the **Strict** model





# Strict model

All dependences!



# Outline

- 1 Introduction
- 2 Framework
- 3 Timed Petri Nets
- 4 Computing the throughput**
- 5 Comparison results
- 6 Conclusion

# Computing the throughput – deterministic case

- Equivalent to find critical cycles
- $\mathcal{C}$  is a cycle of the TPN
- $\mathcal{L}(\mathcal{C})$  is its length (total time of transitions)
- $t(\mathcal{C})$  is the total number of tokens in places traversed by  $\mathcal{C}$
- A critical cycle achieves the largest ratio  $\max_{\mathcal{C}_{\text{cycle}}} \frac{\mathcal{L}(\mathcal{C})}{t(\mathcal{C})}$
- This ratio gives the period  $\mathcal{P}$  of the system
- Can be computed in time  $\mathcal{O}(M^3 R^3)$   
( $R = \text{lcm}(R_0, \dots, R_{M-1})$ )

# Computing the throughput – deterministic case

(previous result)

- **Strict** model: the TPN has an exponential size!
- **Overlap** model:

## Theorem

*Consider a pipeline of  $M$  stages  $T_0, \dots, T_{M-1}$ , such that stage  $T_i$  is mapped onto  $R_i$  distinct processors. Then the average throughput of this system can be computed in time*

$$\mathcal{O}\left(\sum_{i=0}^{M-2} ((R_i R_{i+1})^3)\right).$$

# Computing the throughput – exponential laws

General case:

## Theorem

*Let us consider the system  $(X, Y)$  formed by the mapping of an application onto a platform. Then the throughput can be computed in time  $O(\exp(\text{lcm}_{1 \leq i \leq N}(R_i))^3)$ .*

# Computing the throughput – exponential laws

## General case:

- model the system by a timed Petri net  
Exponential in the size of the system
- transform this timed Petri net into a Markov chain  
Exponential in the size of the TPN
- compute the stationary measure of this Markov chain
- derive the throughput from the marginals of the stationary measure

# Computing the throughput – exponential laws

## General case:

- model the system by a timed Petri net  
Exponential in the size of the system
- transform this timed Petri net into a Markov chain  
Exponential in the size of the TPN
- compute the stationary measure of this Markov chain
- derive the throughput from the marginals of the stationary measure

# Computing the throughput – exponential laws

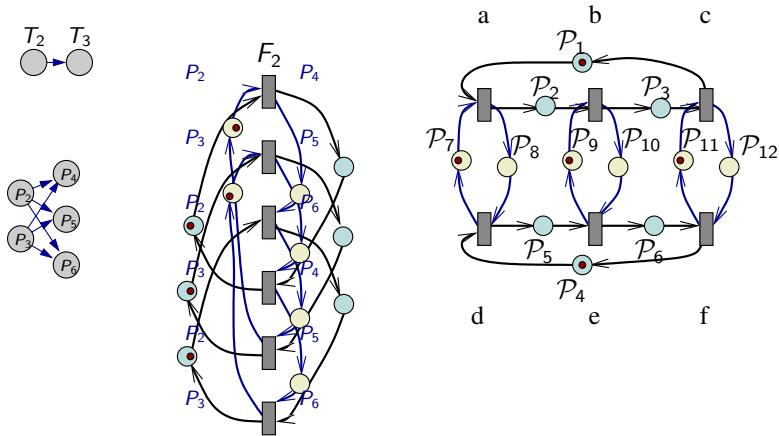
## General case:

- model the system by a timed Petri net  
Exponential in the size of the system
- **transform this timed Petri net into a Markov chain**  
Exponential in the size of the TPN
- compute the stationary measure of this Markov chain
- derive the throughput from the marginals of the stationary measure



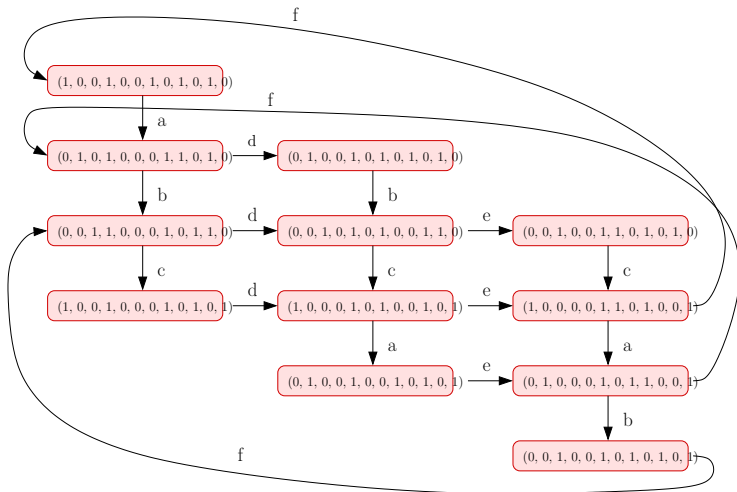
# Computing the throughput – exponential laws

Transformation into a Markov chain: each marking of the TPN becomes a state



# Computing the throughput – exponential laws

Transformation into a Markov chain: list of all possible states



# Computing the throughput – exponential laws

## Overlap model:

### Theorem

*Let us consider the system  $(X, Y)$  formed by the mapping of an application onto a platform. Then the throughput can be computed in time*

$$O\left(N \exp\left(\max_{1 \leq i \leq N} (R_i)\right)^3\right).$$

# Computing the throughput – exponential laws

## Overlap model:

- split the timed Petri net into columns  $C_i$ , with  $1 \leq i \leq 2N - 1$
- separately consider each column  $C_i$
- separately consider each connected component  $D_j$  of  $C_i$
- single component  $D_j$ : many copies of the same pattern  $\mathcal{P}_j$ , of size  $u_j \times v_j$
- transform  $\mathcal{P}_j$  into a Markov chain  $\mathcal{M}_j$
- determine a stationary measure of  $\mathcal{M}_j$
- compute the throughput of  $\mathcal{P}_j$  in isolation
- combine the inner throughputs of all components to get the global throughput of the system

# Computing the throughput – exponential laws

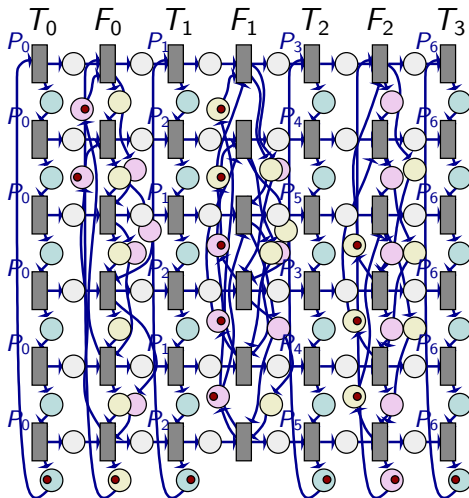
## Overlap model:

- split the timed Petri net into columns  $C_i$ , with  $1 \leq i \leq 2N - 1$
- separately consider each column  $C_i$
- separately consider each connected component  $D_j$  of  $C_i$
- single component  $D_j$ : many copies of the same pattern  $\mathcal{P}_j$ , of size  $u_j \times v_j$
- transform  $\mathcal{P}_j$  into a Markov chain  $\mathcal{M}_j$
- determine a stationary measure of  $\mathcal{M}_j$
- compute the throughput of  $\mathcal{P}_j$  in isolation
- combine the inner throughputs of all components to get the global throughput of the system

## Computing the throughput – exponential laws

## Overlap model:

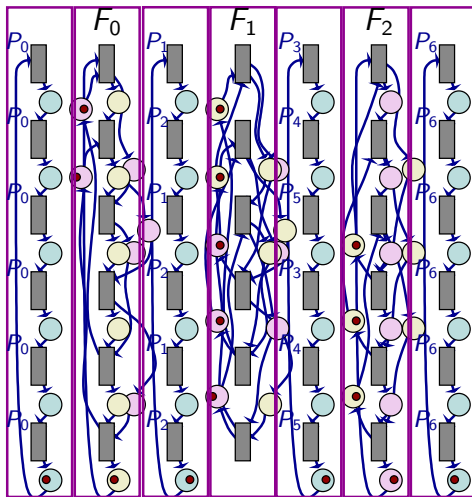
Communication column:



# Computing the throughput – exponential laws

**Overlap model:**

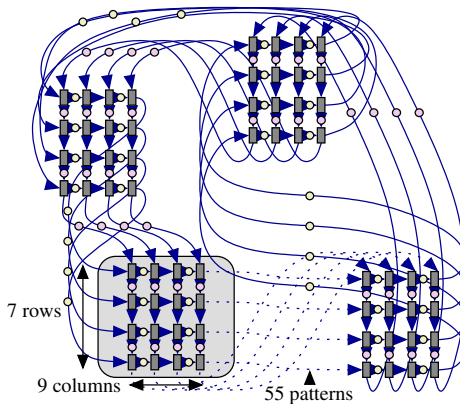
Communication column:



# Computing the throughput – exponential laws

## Overlap model:

Communication column:



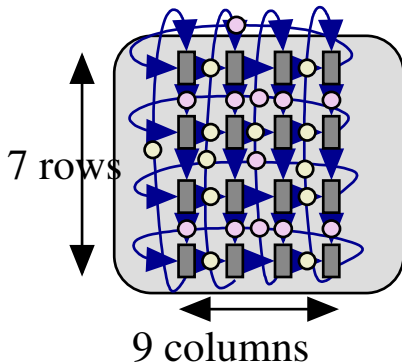
$$R_0 = 5, R_1 = 21, R_2 = 27, R_3 = 11$$



# Computing the throughput – exponential laws

**Overlap model:**

Communication column:



$$R_0 = 5, R_1 = 21, R_2 = 27, R_3 = 11$$

# Computing the throughput – exponential laws

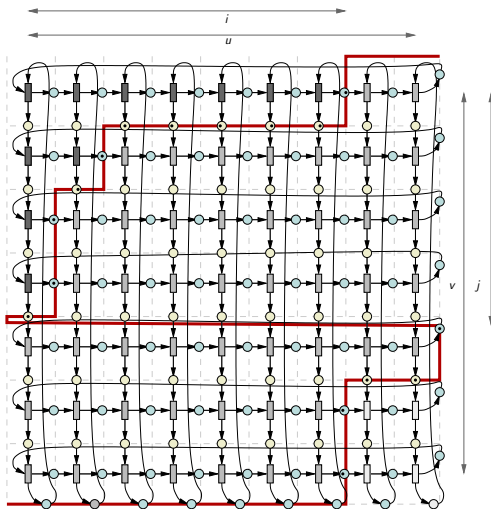
## Overlap model:

- split the timed Petri net into columns  $C_i$ , with  $1 \leq i \leq 2N - 1$
- separately consider each column  $C_i$
- separately consider each connected component  $D_j$  of  $C_i$
- single component  $D_j$ : many copies of the same pattern  $\mathcal{P}_j$ , of size  $u_j \times v_j$
- transform  $\mathcal{P}_j$  into a Markov chain  $\mathcal{M}_j$
- **determine a stationary measure of  $\mathcal{M}_j$**
- compute the throughput of  $\mathcal{P}_j$  in isolation
- combine the inner throughputs of all components to get the global throughput of the system

# Computing the throughput – exponential laws

## Overlap model:

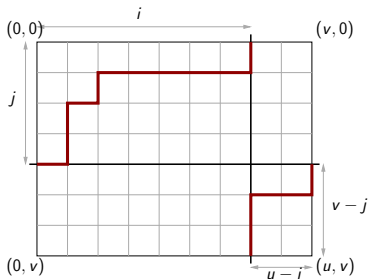
Representation of a valid marking on the TPN



# Computing the throughput – exponential laws

## Overlap model:

Representation of a valid marking with Young diagrams



⇒ Number of states easily determined

Polynomial number of states in each connected component

# Computing the throughput – exponential laws

**Overlap** model, homogeneous communication network:

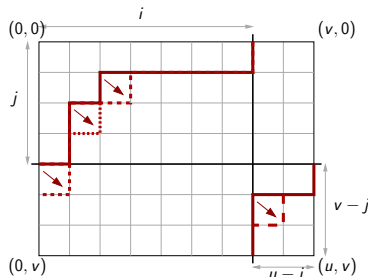
## Theorem

*Let us consider the system  $(X, Y)$  formed by the mapping of an application onto a platform, following the **Overlap** communication model with a homogeneous communication network. Then the throughput can be computed in polynomial time.*

## Computing the throughput – exponential laws

## Overlap model:

## Reachable states from a given position



- ⇒ Same number of incoming and outgoing states
- + Same firing rate (homogeneous communication network)
- = Invariant measure is uniform

# Outline

- 1 Introduction
- 2 Framework
- 3 Timed Petri Nets
- 4 Computing the throughput
- 5 Comparison results
- 6 Conclusion

# Comparison between two systems

## Theorem

Consider two systems  $(X^{(1)}, Y^{(1)})$  and  $(X^{(2)}, Y^{(2)})$ . If we have for all  $n$ ,

$$\forall 1 \leq p \leq M, X_p^{(1)}(n) \leq_{\text{st}} X_p^{(2)}(n) \text{ and}$$

$$\forall 1 \leq p, q \leq M, Y_{p,q}^{(1)}(n) \leq_{\text{st}} Y_{p,q}^{(2)}(n), \text{ then } \rho^{(1)} \geq \rho^{(2)}.$$



# Comparison between two systems with I.I.D. laws

## Theorem

*Let us consider two systems with I.I.D. communication and processing times  $(X^{(1)}, Y^{(1)})$  and  $(X^{(2)}, Y^{(2)})$ . If we have for all  $n$ ,  $\forall 1 \leq p \leq M, X_p^{(1)}(n) \leq_{\text{icx}} X_p^{(2)}(n)$  and  $\forall 1 \leq p, q \leq M, Y_{p,q}^{(1)}(n) \leq_{\text{icx}} Y_{p,q}^{(2)}(n)$ , then  $\rho^{(1)} \geq \rho^{(2)}$ .*

# Bounds on the expected throughput

## Theorem

Let us consider any system  $(X^{(1)}, Y^{(1)})$ , such that  $X_p^{(1)}(n)$  and  $Y_{p,q}^{(1)}(n)$  are N.B.U.E.. Let us also consider two new systems  $(X^{(2)}, Y^{(2)})$  and  $(X^{(3)}, Y^{(3)})$  such that:

$\forall 1 \leq p \leq M, X_p^{(2)}(n)$  has an exponential distribution, and  
 $\mathbf{E}[X_p^{(2)}(n)] = \mathbf{E}[X_p^{(1)}(n)],$

$\forall 1 \leq p, q \leq M, Y_{p,q}^{(2)}(n)$  has an exponential distribution, and  
 $\mathbf{E}[Y_{p,q}^{(2)}(n)] = \mathbf{E}[Y_{p,q}^{(1)}(n)],$

$\forall 1 \leq p \leq M, X_p^{(3)}(n)$  is deterministic and for all  $n$ ,  
 $X_p^{(3)}(n) = \mathbf{E}[X_p^{(1)}(n)],$

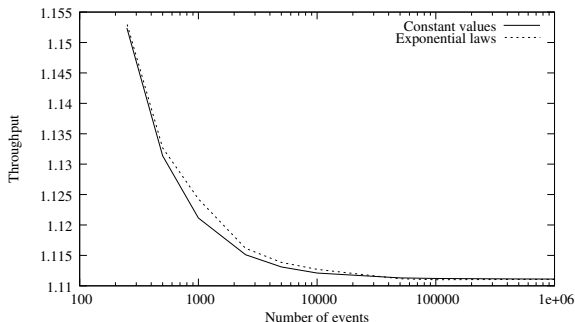
$\forall 1 \leq p, q \leq M, Y_{p,q}^{(3)}(n)$  is deterministic and for all  $n$ ,  
 $Y_{p,q}^{(3)}(n) = \mathbf{E}[Y_{p,q}^{(1)}(n)].$

Then we have:

$$\rho^{(3)} \geq \rho^{(1)} \geq \rho^{(2)}.$$

# Numerical experiments

Evolution of the measured throughput with the number of samples



Distribution	Constant value $c$	Exponential mean $c$	Uniform $c/2 - 3c/2$	Uniform $c/10 - 19c/10$	Pareto mean $c$
Throughput	2.0299	2.0314	2.0304	2.0305	2.0300

**Table:** Throughput obtained with several distributions of same mean.

# Outline

- 1 Introduction
- 2 Framework
- 3 Timed Petri Nets
- 4 Computing the throughput
- 5 Comparison results
- 6 Conclusion**

# Conclusion and future work

- Even if the mapping is given, the throughput is hard to determine
- Expectation of the throughput can be computed in many cases:
  - General case with exponential laws: exponential time
  - **Overlap** model with exponential laws: smaller exponential time
  - **Overlap** model, homogeneous communications: polynomial time
  - General case, N.B.U.E. laws: bounds can be established
- Determining the mapping that maximizes the throughput is an NP-complete problem, even in the simpler deterministic case with no communication costs

Future work:

- Design efficient mapping heuristics

# Conclusion and future work

- Even if the mapping is given, the throughput is hard to determine
- Expectation of the throughput can be computed in many cases:
  - General case with exponential laws: exponential time
  - **Overlap** model with exponential laws: smaller exponential time
  - **Overlap** model, homogeneous communications: polynomial time
  - General case, N.B.U.E. laws: bounds can be established
- Determining the mapping that maximizes the throughput is an NP-complete problem, even in the simpler deterministic case with no communication costs

Future work:

- Design efficient mapping heuristics

# Conclusion and future work

- Even if the mapping is given, the throughput is hard to determine
- Expectation of the throughput can be computed in many cases:
  - General case with exponential laws: exponential time
  - **Overlap** model with exponential laws: smaller exponential time
  - **Overlap** model, homogeneous communications: polynomial time
  - General case, N.B.U.E. laws: bounds can be established
- Determining the mapping that maximizes the throughput is an NP-complete problem, even in the simpler deterministic case with no communication costs

Future work:

- Design efficient mapping heuristics