Performance modelling and evaluation

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Mendoza, Argentina
Noviembre 2006

Course details

- 20 lectures of 50 minutes
- Topic:
  - performance evaluation based on formal models
  - "performance" in Spanish: *prestaciones, rendimiento* o *desempeño*
- Slides available at:
  - http://webdiis.unizar.es/~jcampos/mendoza06.pdf
- Books:
  - Jain, R.: *The Art of Computer Systems Performance Analysis*. Wiley, 1991
  - Kant, K.: *Introduction to Computer Systems Performance Evaluation*. McGraw-Hill, 1992.
  - Balbo, G.; Conte, G.; Ajmone Marsan, M.; Donatelli, S.; Franceschinis, G.: *Modelling with Generalized Stochastic Petri Nets*. John Wiley, 1995.
- Orientation:
  - Both graduate and doctoral students (master/PhD)

Contents

Performance modelling and evaluation

1. Introduction

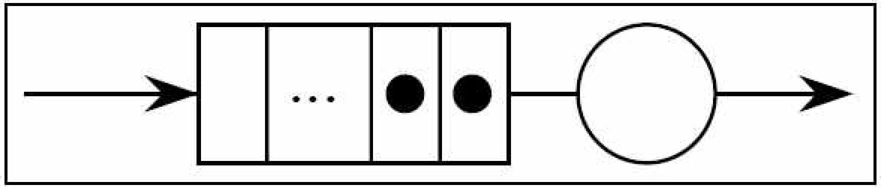
Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Motivating examples
- Why performance?
- When, where and how performance?
- Performance metrics
- Evaluation techniques
- Analysis techniques of models
- Modelling formalisms

Motivating examples

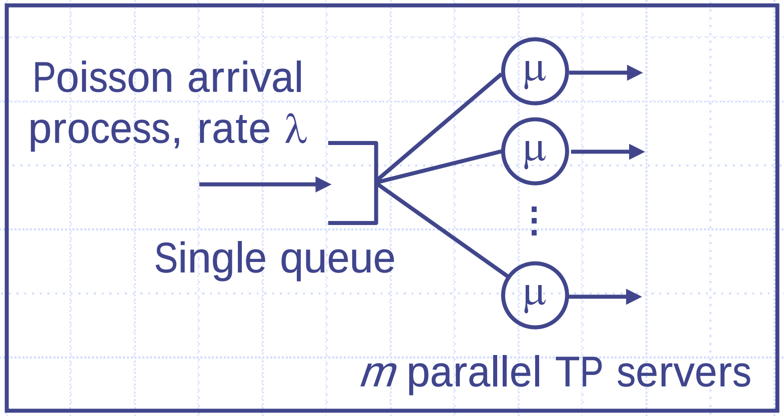
- A simple telecommunication protocol (TP) example
  - A TP system accepts and processes a stream of transactions, mediated through a (large) buffer



  - Transactions arrive "randomly" at some specified rate (e.g., 15 tps)
  - The TP server is capable of servicing transactions at a given service rate (e.g., 58.37 ms)
  - Q1: If both the arrival rate and service rate are doubled, what happens to the mean response time?
  - Q2: What happens to the mean response time if the arrival rate increases by 10%?

Motivating examples

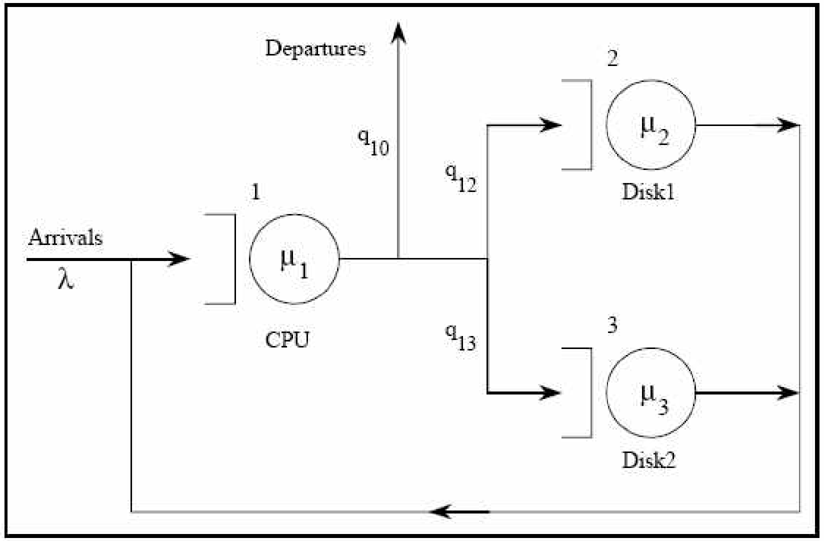
- A simple multiprocessor TP system
  - Consider our TP system but this time with multiple transaction processors



  - The arrival rate is 15 tps
  - The mean service time per transaction is 58.37 ms
  - Q: By how much is the system response time reduced by adding one processor?

Motivating examples

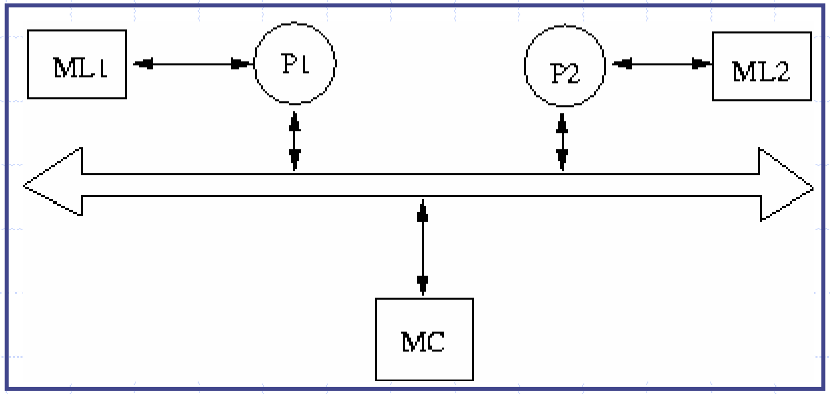
- A Simple Computer Model
  - Consider an open uniprocessor CPU system with just disks
  - Each submitted job makes 121 visits to the CPU, 70 to disk 1 and 50 to disk 2 on average

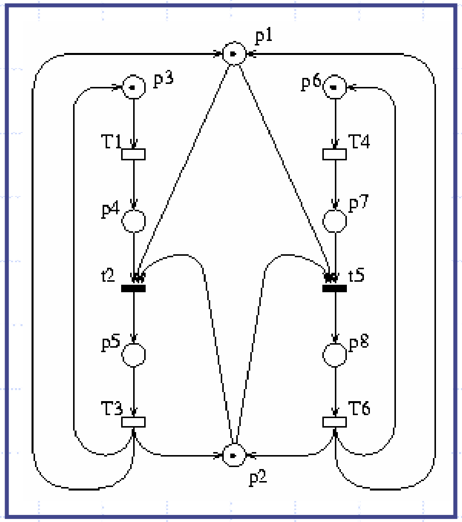


  - The mean service times are 5 ms for the CPU, 30 ms for disk 1 and 37 ms for disk 2
  - Q: What is the effect of replacing the CPU with one twice the speed?

Motivating examples

- A very simple shared memory multiprocessor





- Both processors behave in a similar way:
  - A cyclic sequence of: local activity, then
  - an access request to the shared memory, and then
  - accessing the shared memory (in mutual exclusion)

- Q: What is the “processing power”? (average number of processors effectively -locally- working)

Outline

- Motivating examples
- Why performance?
- When, where and how performance?
- Performance metrics
- Evaluation techniques
- Analysis techniques of models
- Modelling formalisms

Why performance?

- Functional requirements of a system
  - "Does a system work?"
- Qualitative analysis
  - Correctness ("it works")
  - Verification of logical properties:
    - deadlock-freeness, liveness, boundedness, home state existence, synchronic lead, mutual exclusions
- But correctness is not a sufficient condition to make a system acceptable...

Why performance?

- Non-functional requirements:
  - "How well does a system work?"
  - Quality requirements like accuracy, performance, security, modifiability, easiness of use...
- Quantitative analysis:
  - Performance evaluation
    - "How quickly can the system accomplish a given task?"
    - "How much is the system being used?"
    - *Responsiveness*: ability to meet its objectives for response time or throughput
    - *Scalability*: ability to continue to meet responsiveness as the demand for the software functions increases
  - Reliability evaluation
    - "Would the system remain continuously operational for the duration of a mission?"
    - "How dependable is the system over the long run?"

Why performance?

- **System users, designers and administrators aim to obtain or provide the highest performance at the lowest cost**

- Typical problems faced by system designers and administrators that can be addressed through performance evaluation include:
  - Specifying performance requirements
  - Evaluating design alternatives
  - Comparing two or more systems
  - Determining the optimal value of a parameter (system tuning)
  - Finding the performance bottleneck (bottleneck identification)
  - Characterizing the load on the system (workload characterization)
  - Determining the number and size of the components (capacity planning)
  - Predicting the performance at future loads (forecasting)

Outline

- Motivating examples
- Why performance?
- When, where and how performance?
- Performance metrics
- Evaluation techniques
- Analysis techniques of models
- Modelling formalisms

When, where and how performance?

- When, where?

System Lifecycle	Performance Objectives
Design	- compare design alternatives - sensitivity analysis - design optimization
↓ Manufacturing	- Use measurement/models to verify if system meets performance requirements
↓ Configuration Selection (sales/purchase)	- verify vendor's claims - compare different systems
↓ System Operational (Usage)	- system tuning - bottleneck identification - load balancing
↓ Upgrade	- evaluate upgrade alternatives - forecasting

When, where and how performance?

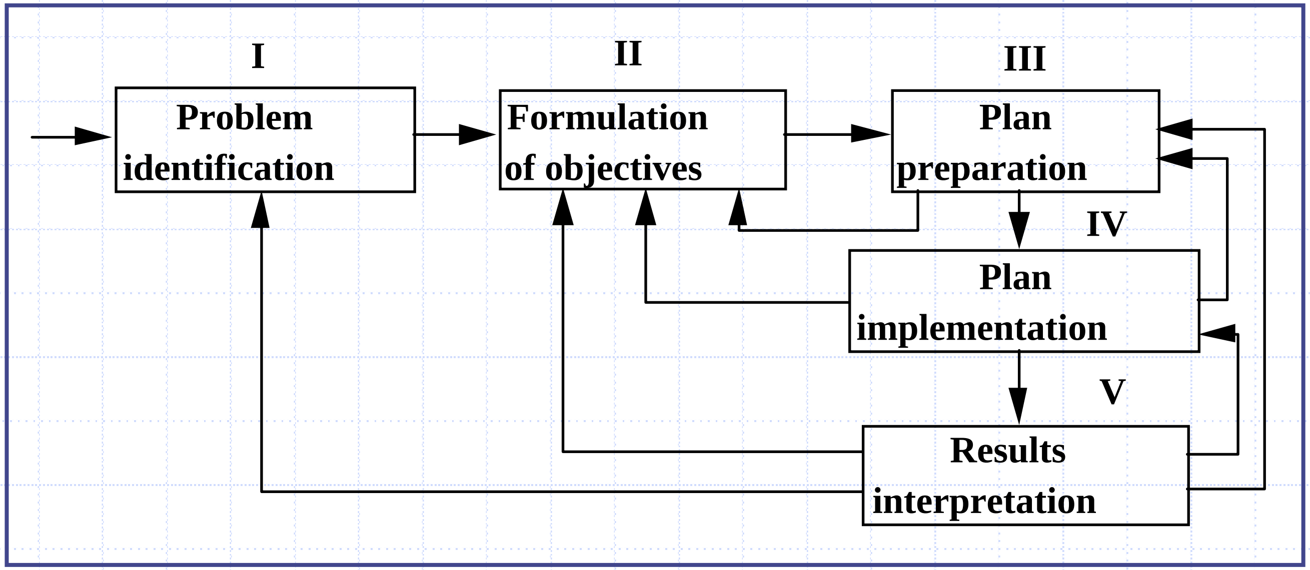
- How?
  - Select appropriate evaluation techniques, performance metrics and workloads for a system
  - Conduct performance measurements correctly
  - Use proper statistical techniques to compare several alternatives
  - Design measurement and simulation experiments to provide the most information with the least effort
  - Perform simulations correctly
  - Use simple queueing models to analyze the performance of systems

When, where and how performance?

- Steps in performance evaluation study
  1. State the goals of the study and define the system boundaries
  2. List system services and possible outcomes
  3. Select performance metrics
  4. List system and workload parameters
  5. Select factors and their values
  6. Select evaluation techniques
  7. Select the workload
  8. Design the experiments
  9. Analyze and interpret the data
  10. Present the results. Start over, if necessary.

When, where and how performance?

- Performance as an Engineering Activity



I. The need for the study arises.
II. Careful estimation of the costs and
III. possible benefits of the study
IV.
V.

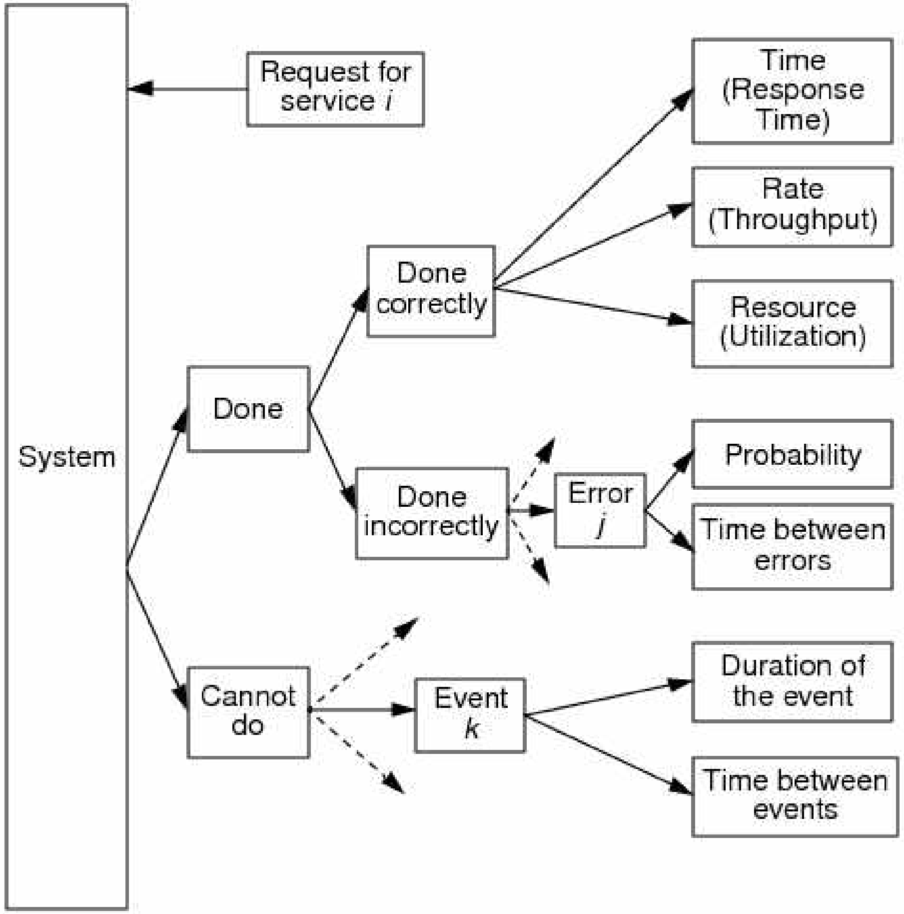
iterative procedure

Outline

- Motivating examples
- Why performance?
- When, where and how performance?
- Performance metrics
- Evaluation techniques
- Analysis techniques of models
- Modelling formalisms

Performance metrics

- Quantifiable descriptor used to represent the performance of a system or some of its aspects

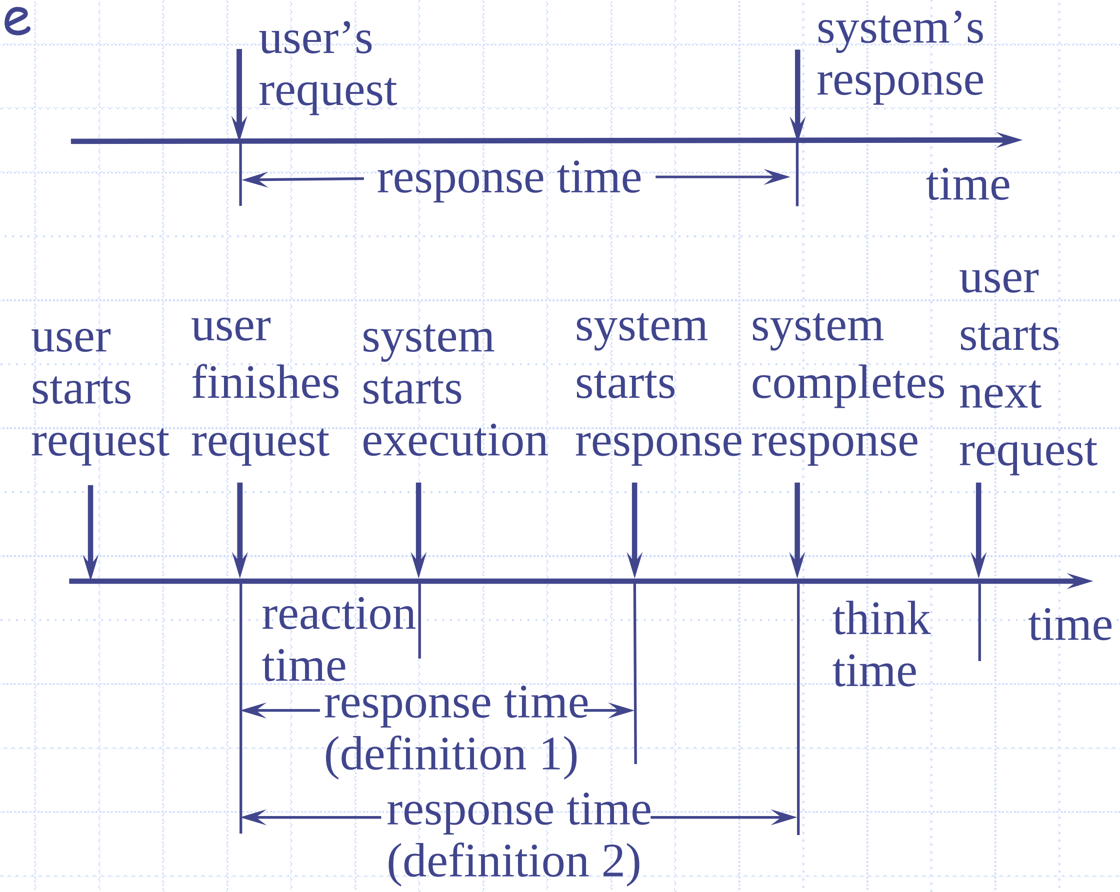


Performance metrics

Index class	Examples of indices	General definition
Productivity	Throughput rate Production rate Capacity (maximum throughput rate) Instruction execution rate Data-processing rate	The volume of information processed by the system in the unit time
Responsiveness	Response time Turnaround time Reaction time	The time between the presentation of an input to the system and the appearance of a corresponding output
Utilization	Hardware module (CPU, memory, I/O channel, I/O device) utilization Operating system module utilization Public software module (e.g., compiler) utilization Data base utilization	The ratio between the time a specified part of the system is used during a given interval of time and the duration of that interval

Performance metrics

- **Response time:**
  - the interval between a user's request and the system response



Performance metrics

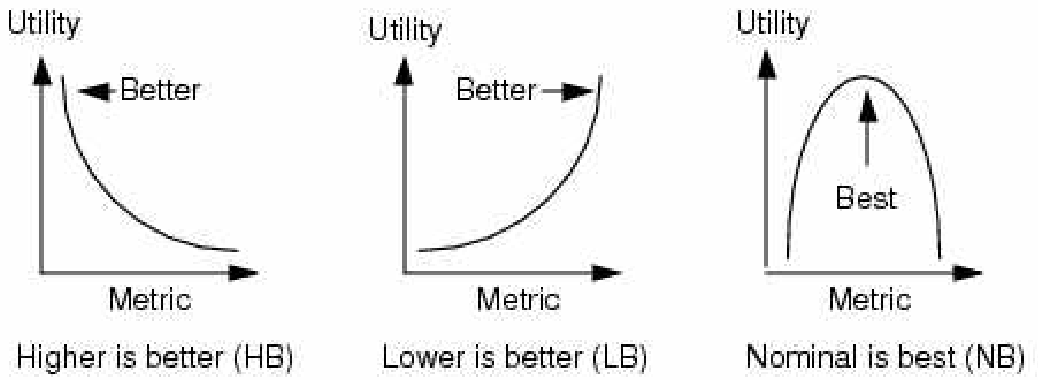
- **Throughput**:
  - productivity measure
  - rate at which requests can be seviced by the system
  - amount of work performed per unit of time
- **Efficiency**:
  - ratio of the maximum achievable throughput to nominal capacity
    - nominal capacity (or bandwidth in the case of computer networks): max achievable throughput under ideal workload conditions

Performance metrics

- **Utilization** of a resource:
  - is measured as the fraction of time the resource is busy servicing requests
  - **bottleneck**: the resource with a maximum utilization in a system; it is the resource slowing down the system
- **Reliability** metrics:
  - measure the period of operation without a single error
  - for example, the probability of an error not occuring by time t, or the mean time between errors
- **Availability** measures:
  - are interested in computing the fraction of the time the system is available to service users' requests
  - these include the system uptime, downtime, and mean time between failures

Performance metrics

- Utility classification of performance metrics
  - Higher is better (HB) metrics: higher values of such metrics preferred, e.g., throughput.
  - Lower is better (LB) metrics: lower values of such metrics preferred, e.g., response time.
  - Nominal is best (NB) metrics: both high and low values are undesirable, e.g., utilization



Performance metrics

- Specification problems (of performance metrics)
  - Nonspecific: no clear numbers are specified
  - Nonmeasurable: no way to verify if the system meets the requirements
  - Nonactionable: metrics are not easy-to-understand, it is not clear which direction is "good" and which is "bad", so you don't know when to take action
  - Nonrelevant: metrics measure things that are not important
  - Nontimely: metrics for which you cannot get the data when you need it
  - Metrics should be
    - **S**pecific
    - **M**easurable
    - **A**ctionable
    - **R**elevant
    - **T**imely

Performance metrics

- An example of specification of performance metrics: performance requirements of a high speed local area network (LAN)
  - A LAN basically transport packets to a specific destination station
  - Three possible outcomes
    - The data arrive correctly to the destination station
    - The data does not arrive correctly
    - The data does not arrive

Performance metrics

- Performance requirements:
  - Speed: if the data arrive correctly to the destination station,
    - Arrival time to any destination < 1 s
    - Throughput > 80 Mbps
  - Reliability:
    - Probability of error of a bit < 10^{-7}
    - Probability bad packet detected < 1%
    - Probability bad packet not detected <10^{-15}
    - Probability packet directed to a bad destination <10^{-18}
    - Probability packet duplicated <10^{-5}
    - Probability lost packet < 1%
  - Availability:
    - Mean time for reinitialization < 15 ms
    - Mean time between consecutive reinitializations > 1 min
    - Mean time for network repair < 1 h

Outline

- Motivating examples
- Why performance?
- When, where and how performance?
- Performance metrics
- Evaluation techniques
- Analysis techniques of models
- Modelling formalisms

Evaluation techniques

- Measurement techniques
  - Tracing a real system
  - Measuring a prototype
- System simulation
  - Trace-driven
  - Discrete-event
- Analytical modelling
  - Markov chains
  - Queueing networks
  - Petri nets
  - Process algebras

Evaluation techniques

- Measurement techniques: Tracing a real system
  - Observation of system operation during a period of time and registering values of relevant variables for the evaluation
  - Need:
    - To get approval of operators
    - To instrument the system for measurement
      - End-to-end metrics and component-wise metrics
      - Non-interference with servicing of user requests
    - Instrumentation should be able to keep up with system load
  - ☺:
    - Most accurate estimation of metrics
  - ☹:
    - Lack of control over parameters/workloads
    - Non-repetitive measurements
    - No insights into “future” operation/design

Evaluation techniques

- Measurement techniques: Measuring a prototype
  - Observation of a prototype of the system and registering values of relevant variables for the evaluation
  - Need:
    - A prototype
    - To instrument the prototype for measurement
      - End-to-end metrics and component-wise metrics
      - Non-interference with servicing of user requests
    - To design and generate workload
    - To select/tune system parameters
  - ☺ :
    - Accurate estimation of metrics
    - More control over parameters and workload
  - ☹ :
    - No insights into "future" system designs

Evaluation techniques

- System simulation
  - Developing a computer program to simulate the system behaviour and measurement of the program execution
  - Need:
    - A simulator
      - Programming skills, right level of detail
    - To design and generate workload
    - To select/tune system parameters
  - ☺ :
    - High control over parameters and workload
    - Possible to incorporate future system designs as well
      - Takes effort though
  - ☹ :
    - Less accuracy
    - Large effort

Evaluation techniques

- Analytical modelling
  - Building a mathematical model of the system and analysing the model
  - Need:
    - A model
      - Probabilistic and statistical modeling skills
    - To design and generate workload
    - To select/tune system parameters
  - ☺ :
    - Least effort
    - High control over parameters and workload
    - Relatively easy to incorporate future system designs as well
  - ☹ :
    - Least accurate
      - Unrealistic assumptions

Evaluation techniques

❑ Comparison of techniques / selection

Technique Criterion	Modeling	Simulation	Measure prototype	Tracing real syst.
When? (stage)	Anytime	Anytime	Post-prototype	Post-deployment
Time required	Small	Medium	Usually less than simulation	Less than measurement
Tools	Analysts	Programming languages	Instrumentation	Instrumentation
Accuracy	Low	Moderate	fn(env. params)	High
Tradeoff-evaluation/ Forecasting Ability	Easy	Moderate	Difficult	Very difficult
Environment Control	High control	High control	Low control over inputs to sub-components	Little control
Cost	Small	Medium	High	High
Saleability	Low	Medium	High	Highest

Evaluation techniques

- Common mistakes in selecting techniques:
  - Use of wrong technique
    - Analysts prefer technique they are comfortable with
    - Use it to solve every performance evaluation problem
      - A model that they can best solve, not one that can best solve the problem
    - Should have basic knowledge of all four techniques
  - Always use two or more techniques
    - Do not believe models till validated by simulation
    - Do not believe simulation till validated by measurement
    - Do not believe measurement till predicted by model or simulation

Be aware of limitations of each technique!

Outline

- Motivating examples
- Why performance?
- When, where and how performance?
- Performance metrics
- Evaluation techniques
- Analysis techniques of models
- Modelling formalisms

Analysis techniques of models

- Classification according to formalism:
  - Markov chains
  - Queueing systems and queueing networks
  - Stochastic Petri nets
  - Stochastic process algebras
- Classification according to the object of study:
  - Transient state analysis
  - Steady-state analysis

Analysis techniques of models

- Classification according to solution technique:
  - Enumerative (state-space based)
  - Reduction/transformation-based
  - Structurally based (high level model-based)

- Classification according to quality of results:
  - Exact values
  - Approximations
  - Bounding techniques

Outline

- Motivating examples
- Why performance?
- When, where and how performance?
- Performance metrics
- Evaluation techniques
- Analysis techniques of models
- Modelling formalisms

Modelling formalisms

- Markov chains
  - Based on the concept of state of the system
  - Solution techniques:
    - Enumerative
    - Transient and steady-state analysis
    - Exact and approximate analysis
  - ☹ :
    - Low abstraction lever
    - Model size equals number of states of the system
    - Only in very particular cases aggregation techniques exist

Modelling formalisms

- Queueing networks
  - High abstraction level
    - The number of states characterizing the system grows exponentially on the model size
  - Solution techniques:
    - Enumerative (based on Markov chains)
    - Reduction/transformation-based
    - Structurally based ("product-form solution", exact)
    - Transient and steady-state analysis
    - Exact, approximate and bounds
  - ☹ :
    - Lack of synchronization primitive
    - Extensions exist but destroying analysis possibilities

Modelling formalisms

- Stochastic Petri nets
  - Abstraction level similar to queueing networks
  - With synchronization primitive

    "SPN = Petri nets + timing interpretation =

    = queueing networks + synchronizations"
  - Wide range of qualitative (logical properties) analysis techniques
    - Enumerative
    - Reduction/transformation-based
    - Structurally based
  - Petri nets as a formal modelling paradigm
    - a conceptual framework to obtain specific formalisms based on common concepts and principles at different life-cycle phases

  ...

Modelling formalisms

- Stochastic Petri nets (cont.)
  - Analysis techniques:
    - Exact: mainly enumerative (based on Markov chains)
    - Bounding techniques (structurally based)
    - Approximation techniques (reduction/transformation)
  - ☹ :
    - Lack of a "product-form" solution for efficient exact analysis in most cases

Performance modelling and evaluation

2. Stochastic processes, the Poisson process

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Why stochastic processes?
- The Poisson process
  - Exponential distribution
  - Properties of exponential r.v. and Poisson process

Why stochastic processes?

- Computer systems are
  - Dynamic: they can pass through a **succession** of states as time progresses.
  - Influenced by events which we consider here as **random** phenomena.

- *Definition*: A **stochastic process** is a family of random variables

$$\{X(t) \in \Omega \mid t \in T\}$$

each defined on some (the same for each) sample space Ω for a parameter space T.

Why stochastic processes?

- T, Ω may be either discrete or continuous.
  - Discrete state and continuous state processes:
    - A process is called discrete or continuous state depending upon the values its states can take, i.e., whether the values (Ω) are finite and countable, or any value on the real line.
  - Discrete and continuous (time) parameter processes:
    - A process is called discrete or continuous (time) parameter process depending on whether the index set T is discrete or continuous.

Why stochastic processes?

- T is normally regarded as time
  - real time: continuous
  - every month or after job completion: discrete

- Ω is the set of values each $X(t)$ may take
  - bank balance: discrete
  - number of active tasks: discrete
  - time delay in communication network: continuous

Why stochastic processes?

- Example:
  - Suppose we observe $n(t)$, the number of jobs at the CPU as a function of time, then the process
    $$\{n(t),\ t \in [0,\infty)\}$$
    is a stochastic process, where $n(t)$ is a random variable, and $n(t) \in \{0,1,2,...\}$
  - The values assumed by the random variable are called **states**, and the set of all possible values forms the state space of the process.
  - In this example time is continuous and state space is discrete.

Why stochastic processes?

- Description of a stochastic process:
  - Probabilistic description of a random variable X is given by its probability density function (pdf)

$$f_X(x) = \frac{d}{dx} \underbrace{P\{X \leq x\}}, \quad -\infty < x < \infty$$

Probability Distribution Function (PDF)
also called cumulative distribution function (cdf)

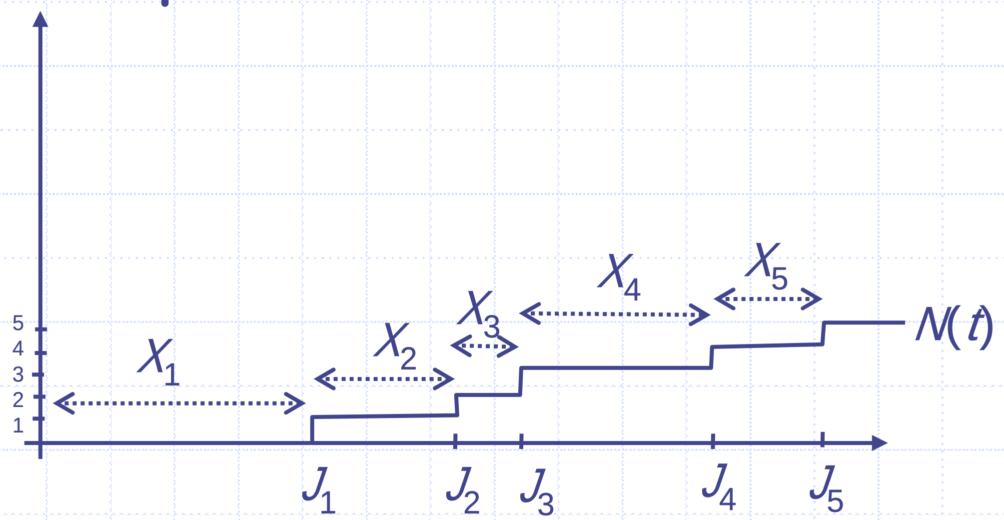
  - Probabilistic description of a stochastic process is given by the **joint pdf** of any set of random variables selected from the process.
    - Thus, in the general case, the detailed description of a stochastic process is unfeasable.

Outline

- Why stochastic processes?
- The Poisson process
  - Exponential distribution
  - Properties of exponential r.v. and Poisson process

The Poisson process

- A large class of stochastic processes are *renewal processes*. This class of processes are used to model independent identically distributed occurrences.
- *Definition*: Let $X_1, X_2, X_3, \ldots$ be independent identically distributed and positive random variables, and set $J_n = X_1 + \ldots + X_n$.
  Then process $N(t)$, $t \geq 0$, where $N(t) = \max\{ n \mid J_n \leq t \}$ is called a **renewal process**.

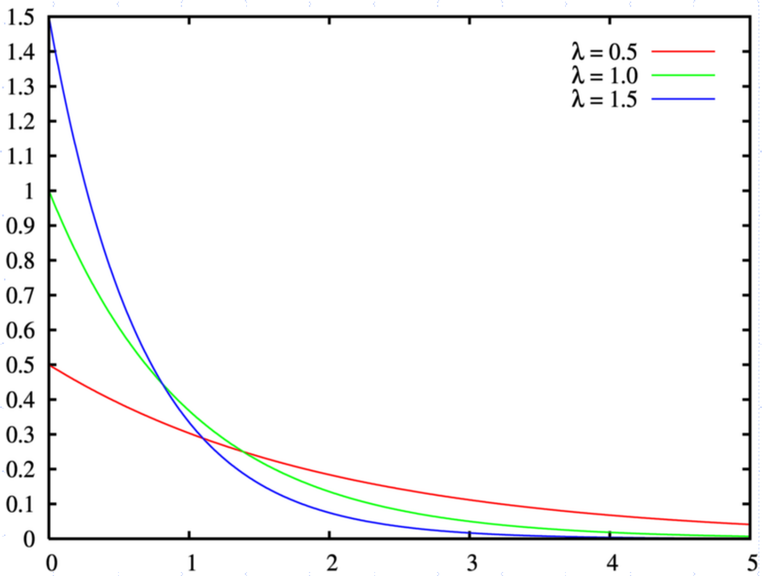


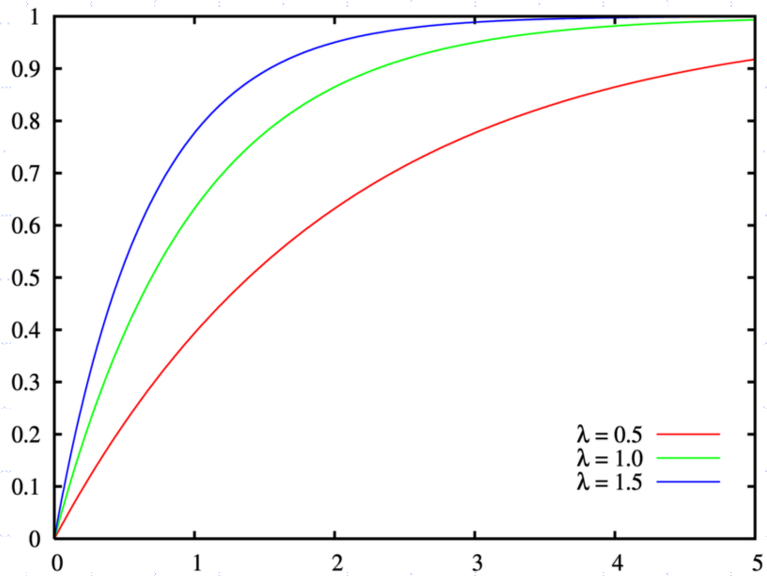
The Poisson process

- *Definition*: The (time-homogeneous, one-dimensional) **Poisson process** is a special case of a renewal process where the time between occurrences is exponentially distributed.

- The pdf and PDF of an **exponentially distributed random variable** *X* are:

$$f_X(x) = \lambda e^{-\lambda x} \quad (x \geq 0) \qquad F_X(x) = P(X \leq x) = 1 - e^{-\lambda x} \quad (x \geq 0)$$





The Poisson process

- Properties of exponential distribution
  - The mean value and variance

$$E[X] = \frac{1}{\lambda} \qquad V[X] = \frac{1}{\lambda^2}$$

  - The minimum of exponentials is exponential (sum of rates)

$$f_X(x) = \lambda e^{-\lambda x} \quad (x \geq 0)$$

$$f_Y(y) = \mu e^{-\mu y} \quad (y \geq 0)$$

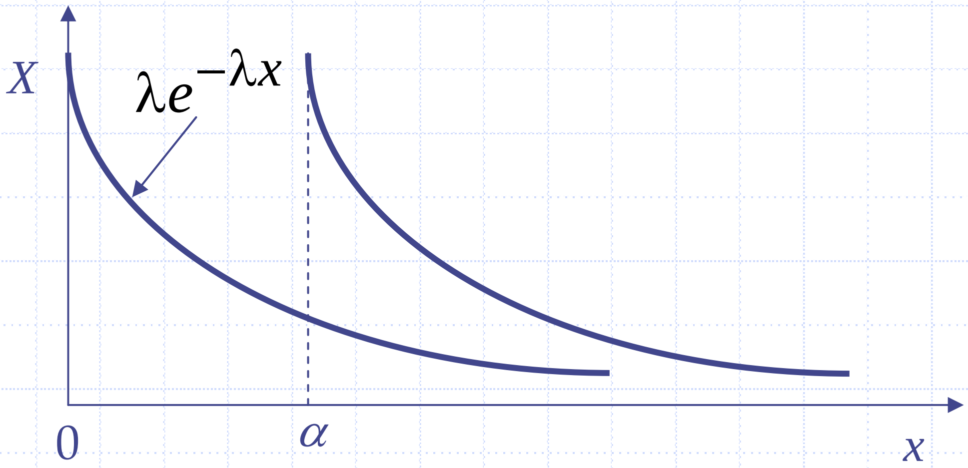
$$Z = \text{mín}\{X, Y\}$$

$$f_Z(z) = (\lambda + \mu) e^{-(\lambda+\mu) z} \quad (z \geq 0)$$

The Poisson process

- Memoryless property

$$P\{X \geq x+\alpha \mid X \geq \alpha\} = P\{X \geq x\}$$



$$\text{(Right) } P(T > 40 \mid T > 30) = P(T > 10).$$

It does *not* mean

$$\text{(Wrong) } P(T > 40 \mid T > 30) = P(T > 40).$$

The Poisson process

- Properties of Poisson process
  - Residual life
    - If you pick a random time point during a Poisson process, what is the time remaining *R* to the next instant (arrival)?
    - E.g. when you get to a bus stop, how long will you have to wait for the next bus?
    - If process is Poisson, *R* has the same distribution as *X* (the time between ocurrences) by the memoryless property of exponential
    - it doesn't matter when the last bus went!

      contrast constant interarrival times in a perfectly regular bus service

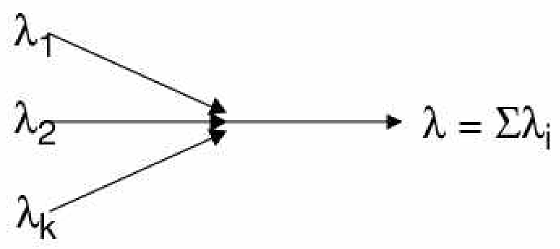
The Poisson process

- Infinitesimal definition of Poisson process
  - P(arrival in $(t, t+\Delta t)$) = $P(R \leq \Delta t)$ = $P(X \leq \Delta t)$ for all t
    $= 1 - e^{-\lambda \Delta t}$
    $= \lambda \Delta t + o(\Delta t)$
  - Therefore
    - Probability of an arrival in $(t, t+\Delta t)$ is $\lambda \Delta t + o(\Delta t)$ regardless of process history before t
    - Probability of more than one arrival in $(t, t+\Delta t)$ is $o(\Delta t)$ regardless of process history before t
- The Poisson distribution
  - Distribution of number of arrivals in time t

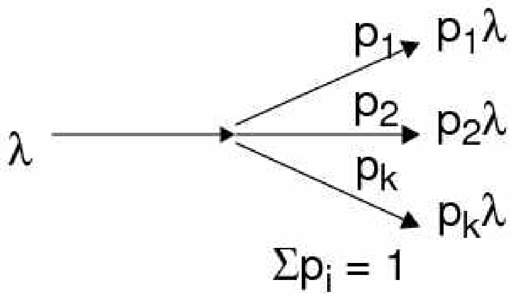
$$P\big(N(t)=n\big)=\frac{(\lambda t)^n}{n!}e^{-\lambda t}$$

The Poisson process

- Superposition property (merging)



- Decomposition property (splitting)



The Poisson process

- Central limit theorem for counting processes:
  - Let $A_1(t)$, …, $A_k(t)$ be independent counting processes (with arbitrary distributions), then

$$X(t) = \frac{\sum_{i=1}^{k} A_i(t)}{k}$$

    is a Poisson process when k → ∞ (under certain "technical conditions")
  - Interpretation: independently of the behaviour of individual countings, the average counting behaviour is Poisson if population is big

Performance modelling and evaluation

3. Discrete time Markov chains

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Basic definitions
- Representations
- Multi-step transition probabilities
- Classification of states
- Steady-state behaviour
- Example

Basic definitions

- Markov processes: special class of stochastic processes that satisfy the **Markov Property** (MP):
  - Given the state of the process at time t, its state at time $t + s$ has probability distribution which is a function of s only.
  - i.e. the future behaviour after t is independent of the behaviour before t.
  - Often intuitively reasonable, yet sufficiently "special" to facilitate effective mathematical analysis.

Basic definitions

- We consider Markov processes with discrete state (sample) space.

  They are called **Markov chains**.

  - If time parameter is discrete $\{t_0, t_1, t_2 \ldots\}$ they are called **Discrete Time Markov Chains** (DTMC).

  - If time is continuous ($t \geq 0$, $t \in \mathbb{R}$), they are called **Continuous Time Markov Chains** (CTMC).

Basic definitions

- Let $X = \{X_n \mid n = 0,1,\ldots;\ X_i \in I\!N,\ i \geq 0\}$ be a non-negative integer valued Markov chain with discrete time parameter n.

  Markov Property states that:

  $P(X_{n+1} = j \mid X_0 = x_0, \ldots, X_n = x_n) =$

  $= P(X_{n+1} = j \mid X_n = x_n)$, for $j,n = 0,1\ldots$

Basic definitions

- Evolution of a DTMC is completely described by its 1-step transition probabilities

$$p_{ij}(n) = P(X_{n+1} = j \mid X_n = i) \text{ for } i,j,n \geq 0$$

- If the conditional probability is invariant with respect to the time origin, the DTMC is said to be time-homogeneous

$$p_{ij}(n) = p_{ij}$$

$$\sum_{j \in \Omega} p_{ij} = 1, \ \forall i \in \Omega$$

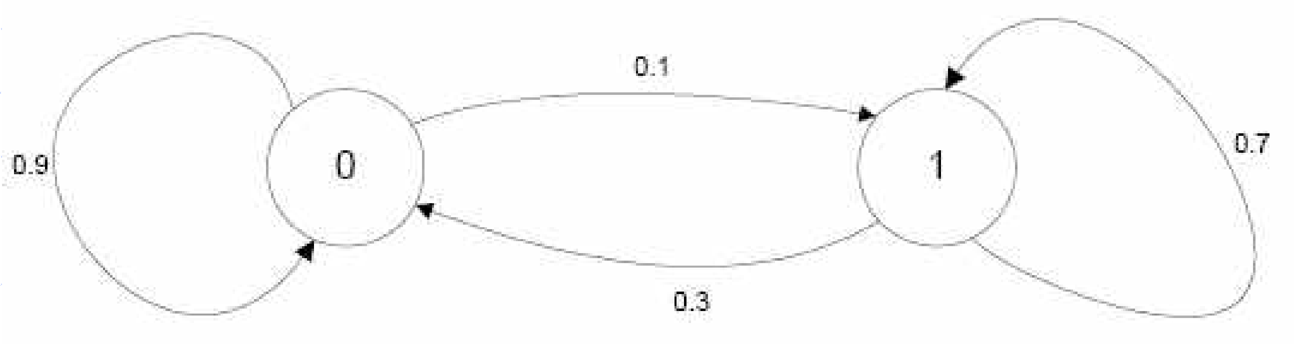
Outline

- Basic definitions
- Representations
- Multi-step transition probabilities
- Classification of states
- Steady-state behaviour
- Example

Representations

- State transition diagram
  - Directed graph
    - number of nodes = number of states (if Ω finite)
    - An arc from i to j if and only if $p_{ij} > 0$

Telephone line example:
line is either idle (state 0) or busy (state 1)



Representations

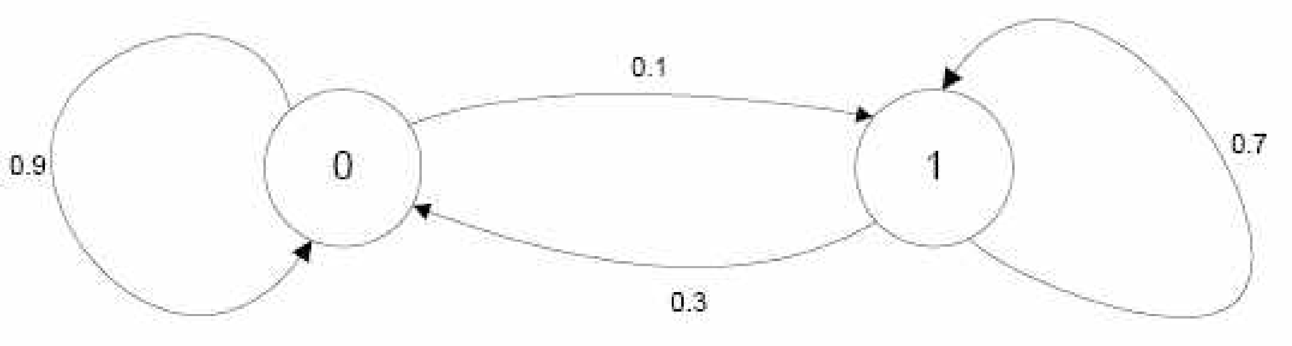
- Transition probability matrix

$$P = \begin{bmatrix} p_{00} & p_{01} & & \cdots \\ p_{10} & p_{11} & & \cdots \\ \vdots & & & \\ p_{i0} & & p_{ii} & \cdots \\ \vdots & & \vdots & \end{bmatrix} \quad \text{in which all rows sum to 1}$$

  - dimension = number of states in Ω if finite, otherwise countably infinite
  - conversely, any real matrix P s.t. $p_{ij} \geq 0$, $\Sigma_j\, p_{ij} = 1$ (called a stochastic matrix) defines a MC

Representations

Telephone line example

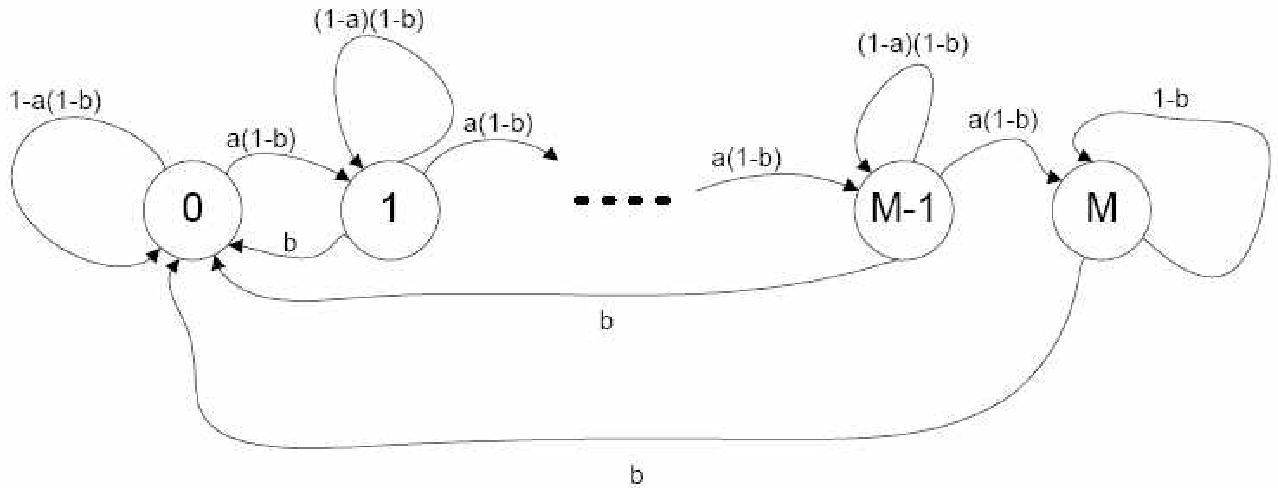


$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{bmatrix}$$

Representations

- Example: I/O buffer, capacity M records

  New record added in any unit of time with prob. a (if not full).
  Buffer emptied in any unit of time with prob. b.
  If both occur in same interval, insertion done first.
  Let X_n be the number of records in buffer at (discrete) time n. Then, assuming that insertions and emptying are independent of each other and of their own past histories, $\{X_n \mid n=0,1,...\}$ is a MC with state space $\{0,1,...,M\}$ and state diagram:



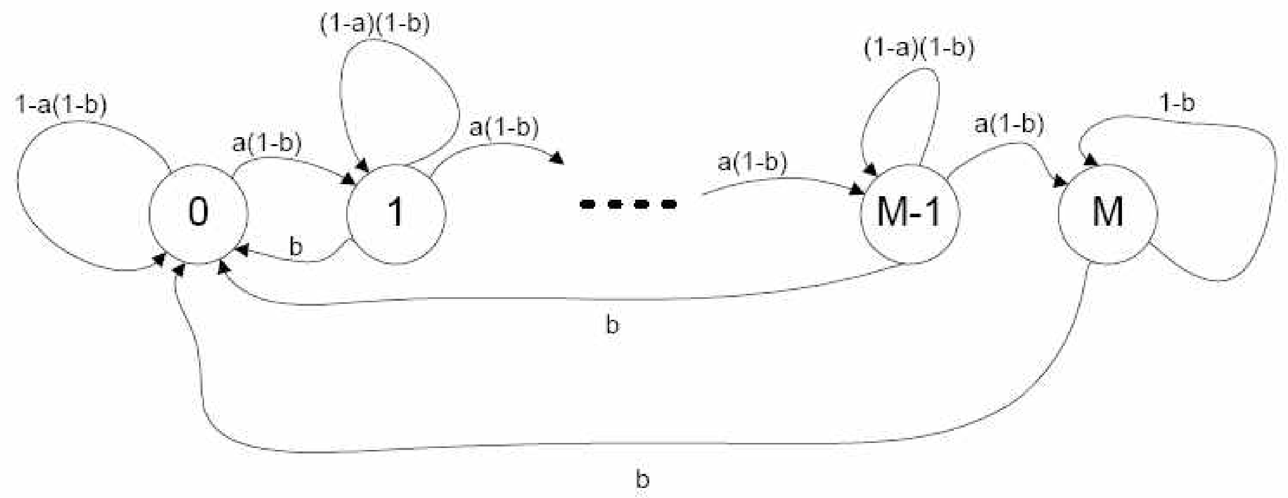
Representations

- The transition rate matrix follows immediately, e.g.:

$$p_{12} = a(1 - b) = p_{n,n+1}\ ,\ 0 \leq n \leq M - 1$$

$$p_{MM} = 1 - b$$

etc.

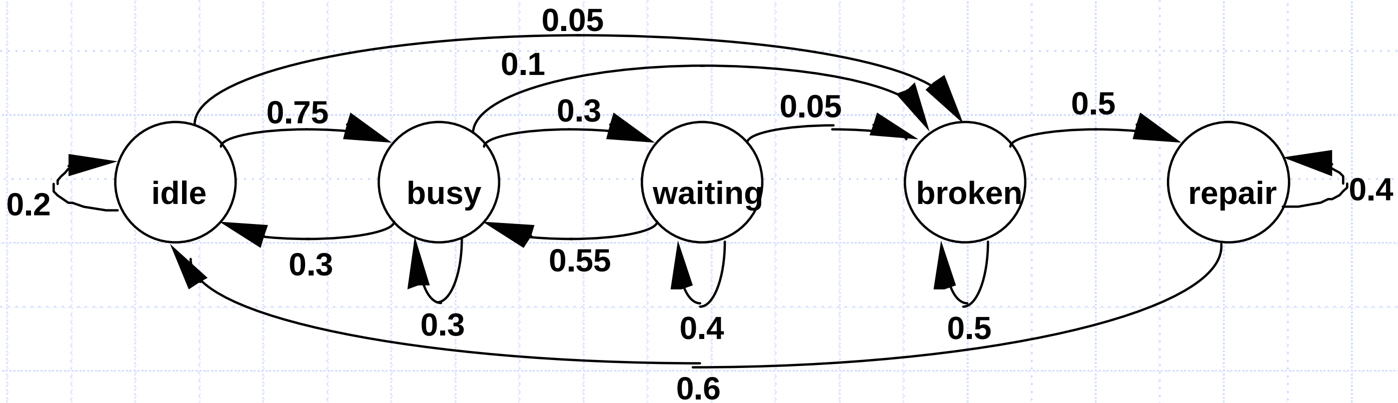


Representations

- Example:

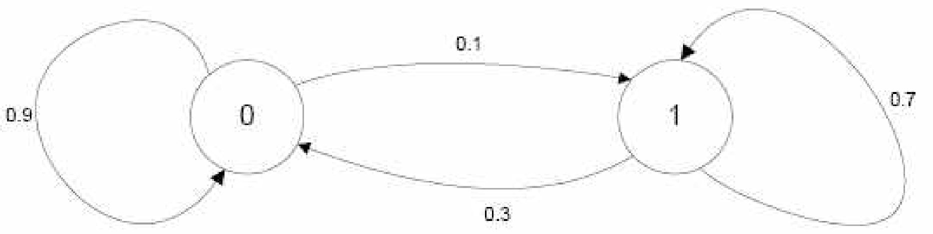
  A system that can be
  - Idle
  - Busy
  - Waiting for a resource
  - Broken
  - Repairing

$$P = \begin{array}{c} \begin{array}{ccccc} \text{idle} & \text{busy} & \text{wait} & \text{broken} & \text{repair} \end{array} \\ \begin{bmatrix} 0.2 & 0.75 & 0.0 & 0.05 & 0.0 \\ 0.3 & 0.3 & 0.3 & 0.1 & 0.0 \\ 0.0 & 0.55 & 0.4 & 0.05 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.5 & 0.5 \\ 0.6 & 0.0 & 0.0 & 0.0 & 0.4 \end{bmatrix} \end{array}$$



Representations

- Time spent in a state:



$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{bmatrix}$$

- T_0 = random variable "*time spent in state* 0"

$P(T_0=0) = (1-p_{00})$
$P(T_0=1) = p_{00}\,(1-p_{00})$
$P(T_0=2) = p_{00}^2\,(1-p_{00})$
…
$P(T_0=n) = p_{00}^n\,(1-p_{00})$

→ Geometrically distributed random variable
Is the discrete analogue of exponential distribution → memoryless

Outline

- Basic definitions
- Representations
- Multi-step transition probabilities
- Classification of states
- Steady-state behaviour
- Example

Multi-step transition probabilities

- Let the 2-step transition probability be

$$
\begin{aligned}
p_{ij}^{(2)} &= P(X_{n+2} = j \mid X_n = i) \\
&= \sum_{k\in\Omega} P(X_{n+1} = k, X_{n+2} = j \mid X_n = i) \ \text{ by law of tot. prob.} \\
&= \sum_{k\in\Omega} P(X_{n+2} = j \mid X_n = i, X_{n+1} = k) P(X_{n+1} = k \mid X_n = i) \\
&= \sum_{k\in\Omega} P(X_{n+2} = j \mid X_{n+1} = k) P(X_{n+1} = k \mid X_n = i) \ \text{ by MP} \\
&= \sum_{k\in\Omega} p_{ik} p_{kj} \\
&= \left(P^2\right)_{ij}
\end{aligned}
$$

Multi-step transition probabilities

- Similarly, the *n*-step transition probability

$$
\begin{aligned}
p_{ij}^{(n)} &= P(X_n = j \mid X_0 = i) \\
&= \sum_{k \in \Omega} P(X_n = j \mid X_{n-1} = k, X_0 = i) P(X_{n-1} = k \mid X_0 = i) \\
&= \sum_{k \in \Omega} p_{ik}^{(n-1)} p_{kj}
\end{aligned}
$$

In matrix form:

$$P^{(n)} = P^{(n-1)} \times P$$

If n=2:

$$P^{(2)} = P^{(1)} \times P = P^2$$

And in general: $P^{(n)} = P^n$ i.e. $p_{ij}^{(n)} = \left(P^n\right)_{ij}$

Multi-step transition probabilities

- A more general version of previous equations
  - Chapman-Kolmogorov equations

$$p_{ij}^{(n+m)} = \sum_{k\in\Omega} p_{ik}^{(n)} p_{kj}^{(m)}$$

Because

$$p_{ij}^{(n+m)} = \sum_{k\in\Omega} P(X_{n+m} = j \mid X_n = k, X_0 = i) P(X_n = k \mid X_0 = i)$$

Thus

$$P^{(n+m)} = P^{(n)} \times P^{(m)} = P^n \times P^m$$

Multi-step transition probabilities

- Computation of transient distribution
  - Probabilistic behaviour of the Markov chain over any finite period time, given the initial state

$$P(X_n = j \mid X_0 = i) = p_{ij}^{(n)} = \left(P^n\right)_{ij}$$

  - E.g., in the example of the I/O buffer with capacity of *M* records, the average number of records in the buffer at time 50 is

$$E(X_{50} \mid X_0 = 0) = \sum_{j=1}^{\min(M,50)} j\, q_{0j}^{(50)}$$

Multi-step transition probabilities

- Computation of transient distribution
  - n*th*-step distribution:

$$\pi_j(n) = P(X_n = j) = \sum_{i\in\Omega} P(X_0 = i) P(X_n = j \mid X_0 = i)$$
$$= \sum_{i\in\Omega} \pi_i(0)\, p_{ij}^{(n)} = \sum_{i\in\Omega} \pi_i(0) (P^n)_{ij}$$

  - in matrix form:

$$\pi(n)^{\mathsf{T}} = \pi(0)^{\mathsf{T}} P^n$$

  - Problem: computationally expensive!

Outline

- Basic definitions
- Representations
- Multi-step transition probabilities
- Classification of states
- Steady-state behaviour
- Example

Classification of states

- State j is **accessible** from state i (writen written $i \rightarrow j$) if

$$p_{ij}^{(n)} > 0, \text{ for some } n$$

- A state i is said to **communicate** with state j (writen written $i \leftrightarrow j$) if i is accessible from j and j is accesible from i
- A set of states C such that each pair of states in C communicates is a **communicating class**
- A communicating class is **closed** if the probability of leaving the class is zero (no state out of C is accesible from states in C)
- A Markov chain is **irreducible** if the state space is a communicating class
- State i is an **absorbing** state if there is no state reachable from i

Classification of states

- Periodicity:
  - A state *i* has **period** *k* if any return to state *i* must occur in some multiple of *k* time steps.

$$k = \gcd\{n : P(X_n = i \mid X_0 = i) > 0\}$$

  - If $k = 1$, then the state is **aperiodic**; otherwise ($k>1$), the state is **periodic with period *k***.
  - It can be shown that every state in a communicating class must have the same period.
  - An irreducible Markov chain is **aperiodic** if its states are aperiodic.

Classification of states

- Recurrence
  - A state i is **transient** if, given that we start in state i, there is a non-zero probability that we will never return back to i.
    - Formally, next return time to state i ("hitting time"):

$$T_i = \min\{n : X_n = i \mid X_0 = i\}$$

    - State i is transient if $P(T_i < \infty) < 1$
  - If a state i is not transient (it has finite hitting time with probability 1), then it is said to be **recurrent**.
  - Let M_i be the expected (average) return time, $M_i = E[T_i]$
    - Then, state i is **positive recurrent** if M_i is finite; otherwise, state i is **null recurrent**.
  - It can be shown that a state is recurrent iff $\sum_{n=0}^{\infty} p_{ii}^{(n)} = \infty$

Classification of states

- In a finite DTMC:
  - All states belonging to a closed class are positive recurrent.
  - All states not belonging to a closed class are transient.
  - There are not null recurrent states.
- In an irreducible DTMC:
  - Either all states are transient or recurrent
- Ergodicity:
  - A state *i* is said to be **ergodic** if it is aperiodic and positive recurrent.
  - If all states in a DTMC are ergodic, the chain is said to be ergodic.

Outline

- Basic definitions
- Representations
- Multi-step transition probabilities
- Classification of states
- Steady-state behaviour
- Example

Steady-state behaviour

- Transient behaviour: computationally expensive
- Easier and maybe more interesting to determine the **limit** or **steady-state** distribution

$$\pi_j = \lim_{n\to\infty} \pi_j(n)$$

In vector form

$$\pi = \lim_{n\to\infty} \pi(n)$$

- Does it exist?
- Is it unique?
- Is it independent of the initial state?

Steady-state behaviour

- If limit distribution exists… we know how to compute it!

$$\pi(n+1) = \pi(0)P^{n+1} = \pi(n)P$$

$$\lim_{n\to\infty} \pi(n+1) = \lim_{n\to\infty} \pi(n)P$$

i.e., it must be equal to the **stationary distribution**, the solution of:

$$\boxed{\begin{array}{l} \pi^T P = \pi^T \\ \pi^T e = 1 \end{array}}$$

→ *balance* equations

→ *normalizing* equat.

where $e = (1,1,\ldots,1)^T$, and the initial distribution does not affect the limit distribution

Steady-state behaviour

- Other interpretation:
  - The solution of balance equations can be seen as the proportion of time that the process enters in each state in the long run
    - Let $N_j(n)$ be the number of visits of the process to the state j until instant n
    - The **occupation distribution** can be defined as

$$\pi_j = \lim_{n\to\infty} \frac{E[N_j(n)]}{n+1}$$

    - Of course, its inverse is the mean interval between visits, or **mean return time** ($1/\pi_j$)
  - If the occupation distribution exists, it verifies

$$\pi^T P = \pi^T \,; \quad \pi^T e = 1$$

Steady-state behaviour

- But,
  - Does limit distribution exist?
  - Is it unique?
  - Is it independent of the initial state?

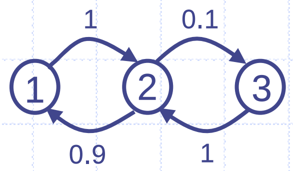
We know some cases where the answer is no

We know some cases where the answer is yes

Steady-state behaviour

- If a unique limit distribution exists, all rows of P^n must be equal in the limit, in this way the distribution of X_n does not depend on the initial distribution
- Example

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0.1 & 0 & 0.9 \\ 0 & 1 & 0 \end{bmatrix}$$



$$P^{2n} = \begin{bmatrix} 0.1 & 0 & 0.9 \\ 0 & 1 & 0 \\ 0.1 & 0 & 0.9 \end{bmatrix} \qquad P^{2n-1} = \begin{bmatrix} 0 & 1 & 0 \\ 0.1 & 0 & 0.9 \\ 0 & 1 & 0 \end{bmatrix}$$

If a is the initial distribution, then the distribution of X_n, $n \geq 1$ is:

$(0.1(a_1+a_3),\quad a_2,\quad 0.9(a_1+a_3))$, if n is odd
$(\quad 0.1a_2,\quad a_1+a_3,\quad 0.9a_2\quad)$, if n is even

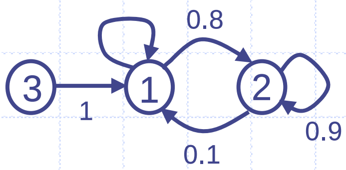
Thus, the DTMC has not limit distribution.
If balance and normalization equations are solved, we get a unique solution $\pi = (0.05, 0.5, 0.45)$.

This means: if π is assumed as initial distribution, then π is also the distribution for X_n, for all n.

Steady-state behaviour

- Example: limit and stationary distributions may be non unique

$$P = \begin{bmatrix} 0.2 & 0.8 & 0 \\ 0.1 & 0.9 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Then,

$$\lim_{n\to\infty} P^n = \begin{bmatrix} 0.1111 & 0.8889 & 0 \\ 0.1111 & 0.8889 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Limit distribution exists, but it is not unique since it depends on the initial distribution: if a is the initial distribution

$$\pi = (0.111(a_1+a_2), 0.8889(a_1+a_2), a_3)$$

is a limit distribution for X_n, and it is also a stationary distribution.

Steady-state behaviour

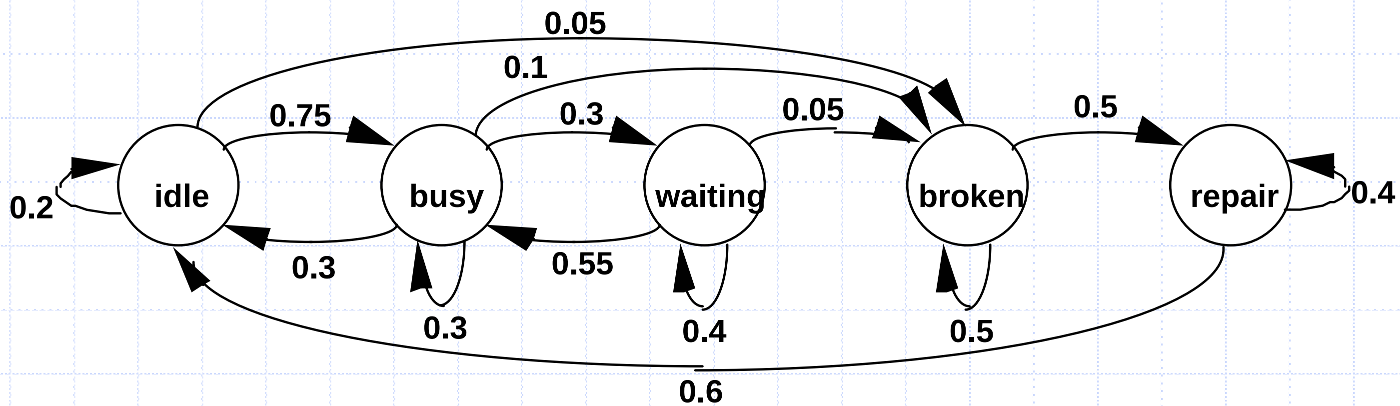
- Finite & irreducible DTMC $\Rightarrow$ there exists a unique stationary distribution
- Finite & irreducible DTMC $\Rightarrow$ there exists a unique occupation distribution, and it is equal to the stationary distribution
- Finite, irreducible & aperiodic DTMC $\Rightarrow$ it has a unique limit distribution, and it is equal to the stationary distribution
- Positive recurrent & aperiodic DTMC $\Rightarrow$ there exists limit distribution
  - If in addition DTMC is irreducible, the limit distribution is independent of the initial probability
- Irreducible, positive recurrent & periodic DTMC with period $d \Rightarrow \lim_{n\to\infty} p_{ij}^{(nd)} = d\pi_j$
- An irreducible & aperiodic DTMC is positive recurrent $\Leftrightarrow$ there exists a unique solution of balance equation
- Irreducible, aperiodic & null recurrent DTMC $\Rightarrow \lim_{n\to\infty} p_{ij}^{(n)} = 0$

Outline

- Basic definitions
- Representations
- Multi-step transition probabilities
- Classification of states
- Steady-state behaviour
- Example

Example

- A processor has certain tasks to perform
  - State transition diagram. Possible states:
    - idle (no task to do)
    - busy (working on a task)
    - waiting (stopped for some resource)
    - broken (no longer operational)
    - repair (fixing the failure)

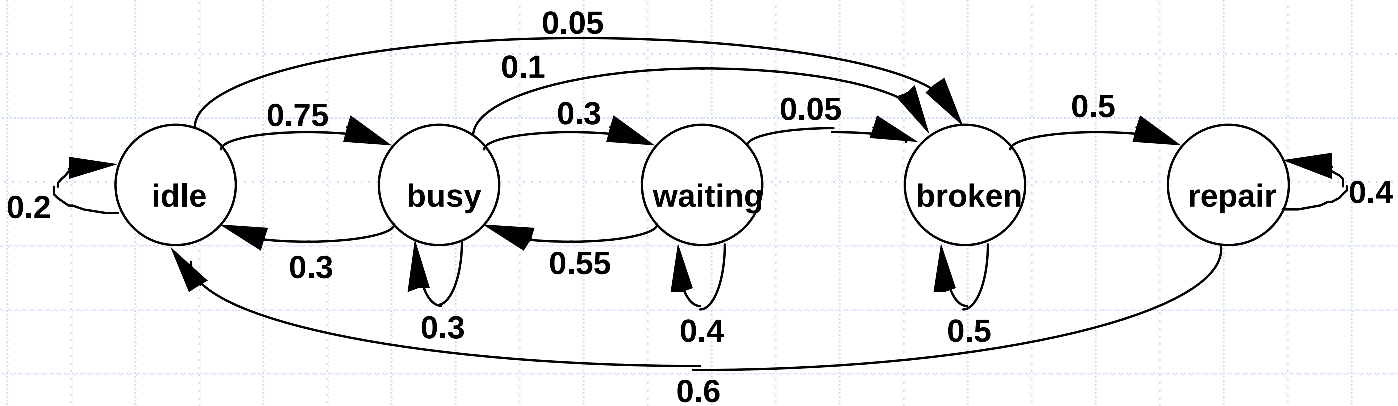


Example

- Transition probability matrix representation

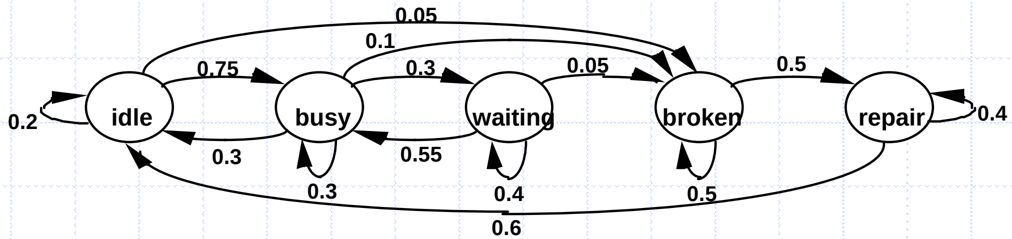
$$P = \begin{array}{c} \\ \\ \end{array}\begin{array}{ccccc} \text{idle} & \text{busy} & \text{wait} & \text{broken} & \text{repair} \end{array}$$

$$P = \begin{bmatrix} 0.2 & 0.75 & 0.0 & 0.05 & 0.0 \\ 0.3 & 0.3 & 0.3 & 0.1 & 0.0 \\ 0.0 & 0.55 & 0.4 & 0.05 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.5 & 0.5 \\ 0.6 & 0.0 & 0.0 & 0.0 & 0.4 \end{bmatrix}$$



Example

- Properties: finite state space, irreducible, aperiodic



$\Rightarrow$ it has a unique limit distribution,
and it is equal to the stationary distribution

- Solution:

$$\pi = \lim_{n \to \infty} \pi(n)$$

$$\pi^T P = \pi^T$$
$$\pi^T e = 1$$

$$\pi = (0.2155, 0.3804, 0.1902, 0.1167, 0.0972)^T$$

Example

$$\pi = (0.2155,\ 0.3804,\ 0.1902,\ 0.1167,\ 0.0972)^T$$

- Other performance indices:
  - *Availability*: P(idle + busy + wait) = 0.7861
    (in other words, 78.61% of the time)
  - So, not available: 21.39% of the time
  - *Working time*: P(busy + wait) = 0.57

Performance modelling and evaluation

4. Continuous time Markov chains

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Definitions
- Steady-state distribution
- Examples

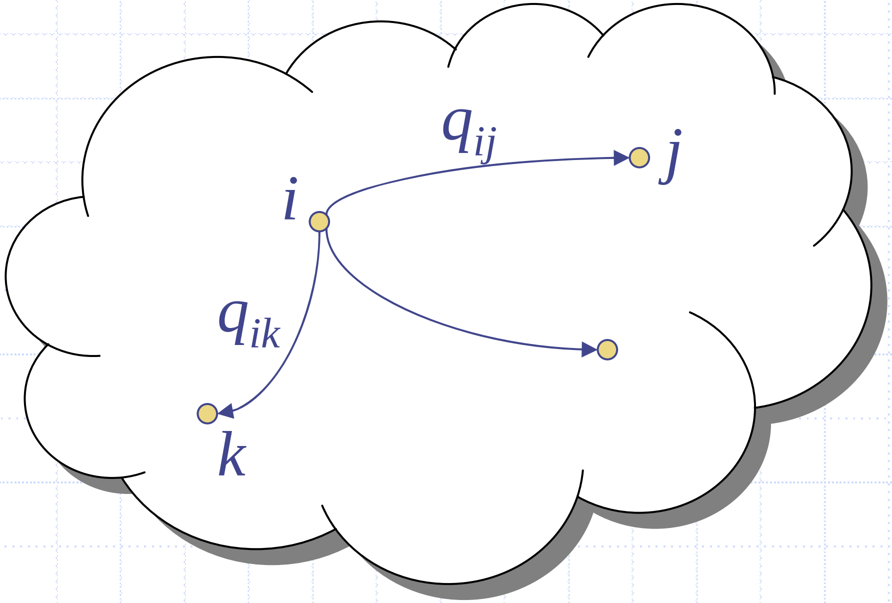
Definitions

- Remember DTMC
  - p_{ij} is the transition probability from *i* to *j* over one time slot
  - The time spent in a state is geometrically distributed
    - Result of the Markov (memoryless) property
  - When there is a jump from state *i*, it goes to state *j* with probability

$$\frac{p_{ij}}{\sum_{k \neq i} p_{ik}}$$

Definitions

- Continuous time version



- q_{ij} is the **transition rate** from state i to state j

Definitions

- Formally:
  - A CTMC is a stochastic process $\{X(t) \mid t \geq 0, t \in IR\}$ s.t. for all $t_0,\ldots,t_{n-1},t_n,t \in IR$, $0 \leq t_0 < \ldots < t_{n-1} < t_n < t$, for all $n \in IN$

$$P(X(t)=x \mid X(t_n)=x_n, X(t_{n-1})=x_{n-1},\ldots, X(t_0)=x_0)=$$
$$=P(X(t)=x \mid X(t_n)=x_n)$$

  - Alternative (equivalent) definition: $\{X(t) \mid t \geq 0, t \in IR\}$ s.t. for all $t,s \geq 0$

$$P(X(t+s)=x \mid X(t)=x_t, X(u), 0 \leq u \leq t)=$$
$$=P(X(t+s)=x \mid X(t)=x_t)$$

Definitions

- Homogeneity
  - We are considering discrete state (sample) space, then we denote

$$p_{ij}(t,s) = P(X(t+s)=j \mid X(t)=i), \text{ for } s > 0.$$

  - A CTMC is called (time-)homogeneous if

$$p_{ij}(t,s) = p_{ij}(s) \quad \text{for all } t \geq 0$$

Definitions

- Time spent in a state:
  - Markov property and time homogeneity imply that if at time *t* the process is in state *j*, the time remaining in state *j* is independent of the time already spent in state *j* : memoryless property.

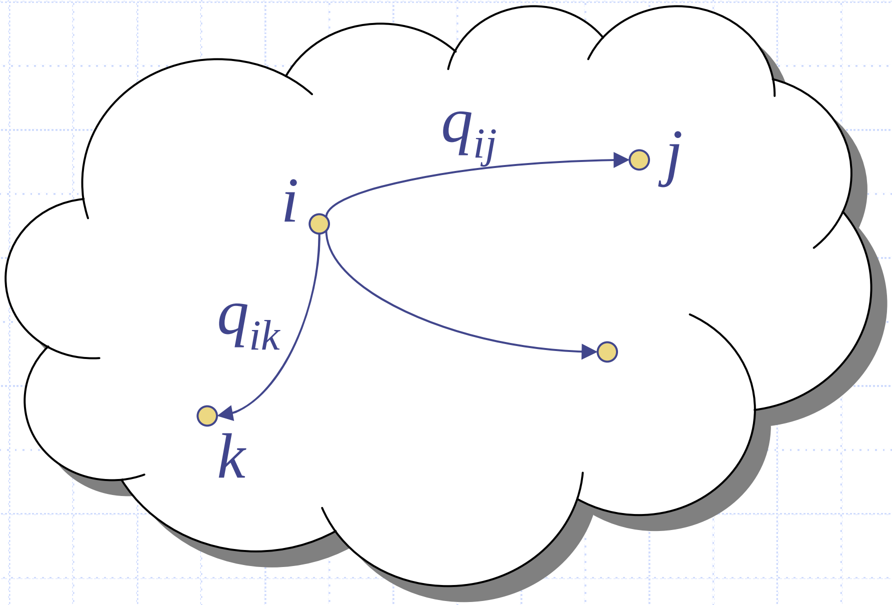
$$
\begin{aligned}
P(S > t+s | S > t) &= P(X_{t+u} = j, 0 \le u \le s | X_u = j, 0 \le u \le t) \\
& \quad \text{where S} = \text{time spent in state } j \\
& \quad \text{state } j \text{ entered at time } 0 \\
&= P(X_{t+u} = j, 0 \le u \le s | X_t = j) \text{ by MP} \\
&= P(X_u = j, 0 \le u \le s | X_0 = j) \text{ by T.H.} \\
&= P(S > s)
\end{aligned}
$$

$\Rightarrow$ time spent in state *j* is exponentially distributed.

Definitions

- Transition rates:
  - In time-homogeneous CTMC, $p_{ij}(s)$ is the probability of jumping from i to j during an interval time of duration s.
  - Therefore, we define the instantaneous transition rate from state i to state j as:

$$q_{ij} = \lim_{\Delta t \to 0} \frac{p_{ij}(\Delta t)}{\Delta t}$$



  - And the exit rate from state i as $-q_{ii}$

$$q_{ii} = -\sum_{j \neq i} q_{ij} = \lim_{\Delta t \to 0} \frac{p_{ii}(\Delta t) - 1}{\Delta t}$$

  - $Q = [q_{ij}]$ is called **infinitesimal generator matrix** (*Q matrix*)

Outline

- Definitions
- Steady-state distribution
- Examples

Steady-state distribution

- Kolmogorov differential equation:

  Denote the distribution at instant t: $\pi_i(t) = P(X(t)=i)$
  And denote in matrix form: $P(t) = [p_{ij}(t)]$

  Then $\pi(t) = \pi(u)P(t-u)$, for $u < t$
  (we omit vector transposition to simplify notation)

  Substituting $u = t-\Delta t$ and substracting $\pi(t-\Delta t)$:

  $\pi(t) - \pi(t-\Delta t) = \pi(t-\Delta t)\,[P(\Delta t) - I]$, with I the identity matrix

  Dividing by Δt and taking the limit
  $$\frac{d}{dt}\pi(t) = \pi(t)\lim_{\Delta t \to 0}\frac{P(\Delta t) - I}{\Delta t}$$

  Then, by definition of $Q = [q_{ij}]$, we obtain the
  Kolmogorov differential equation
  $$\frac{d}{dt}\pi(t) = \pi(t)\,Q$$

Steady-state distribution

- Since also $\pi(t)e = 1$, with $e = (1,1,\ldots,1)$

  If the following limit exists

  $$\lim_{t\to\infty} \pi(t)$$

  then taking the limit of Kolmogorov differential equation we get the equations for the steady-state probabilities:

$$\boxed{\begin{aligned} \pi Q &= 0 \\ \pi e &= 1 \end{aligned}}$$

(*balance* equations)

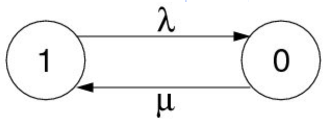
(*normalizing* equation)

Outline

- Definitions
- Steady-state distribution
- Examples

Examples

- Example 1: A 2-state CTMC
  - Consider a simple two-state CTMC



  - The corresponding Q matrix is given by

$$Q = \begin{bmatrix} -\mu & \mu \\ \lambda & -\lambda \end{bmatrix}$$

  - The Kolmogorov differential equation yields:

$$\frac{d}{dt}\pi_0(t) = -\mu\pi_0(t) + \lambda\pi_1(t)$$

$$\frac{d}{dt}\pi_1(t) = \mu\pi_0(t) - \lambda\pi_1(t)$$

$$\pi_0(t) + \pi_1(t) = 1$$

  - Given that $\pi_1(0) = 1$, we get the transient solution:

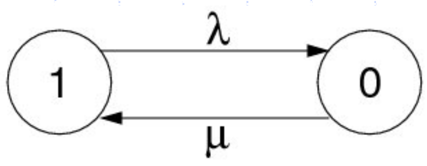
$$\pi_1(t) = \frac{\mu}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)t}$$

$$\pi_0(t) = \frac{\lambda}{\lambda+\mu} - \frac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)t}$$

Examples

- And the steady-state solution comes from $\pi Q = 0$; $\pi e = 1$:

$$\lambda\pi_1 - \mu\pi_0 = 0$$
$$\mu\pi_0 - \lambda\pi_1 = 0$$
$$\pi_0 + \pi_1 = 1$$



balance eq.

- We get:

$$\pi_0 = \frac{\lambda}{\lambda+\mu}; \qquad \pi_1 = \frac{\mu}{\lambda+\mu}$$

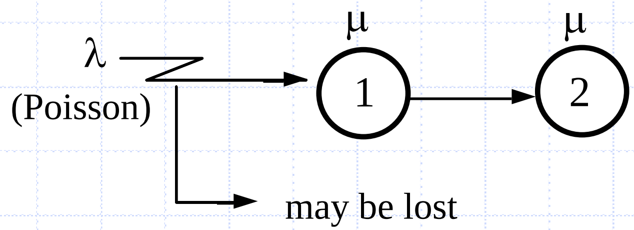
- Which can also be obtained by taking the limits as $t \to \infty$ of the equations for $\pi_1(t)$ and $\pi_0(t)$.

$$\pi_1 = \lim_{t\to\infty} \pi_1(t) = \lim_{t\to\infty} \frac{\mu}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)t} = \frac{\mu}{\lambda+\mu}$$

$$\pi_0 = \lim_{t\to\infty} \pi_0(t) = \lim_{t\to\infty} \frac{\lambda}{\lambda+\mu} - \frac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)t} = \frac{\lambda}{\lambda+\mu}$$

Examples

- Example 2:
  A simple open system with loss



- Works enter to the system with exponentially distributed (parameter λ) interarrival time (Poisson process)
- The service time in both processing stations is exponentially distributed with rate μ
- If a work ends in station 1 when station 2 is busy, station 1 is blocked
- If station 1 is busy or blocked when a work arrives, arriving work is lost
- Questions:
  - proportion of lost works?
  - mean number of working stations?
  - mean number of works in the system?

Examples

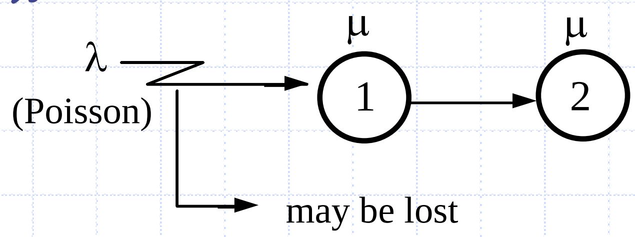
- The set of states of the system:

  S={(0,0), (1,0), (0,1), (1,1), (b,1)}

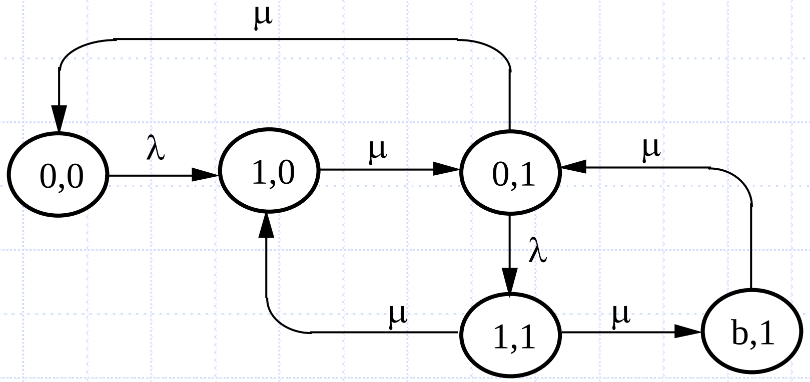
  0 → empty station

  1 → working station

  b → blocked station



State transition diagram:



Infinitesimal generator matrix:

$$Q = \begin{array}{c} 00 \\ 10 \\ 01 \\ 11 \\ b1 \end{array} \begin{bmatrix} -\lambda & \lambda & 0 & 0 & 0 \\ 0 & -\mu & \mu & 0 & 0 \\ \mu & 0 & -(\lambda+\mu) & \lambda & 0 \\ 0 & \mu & 0 & -2\mu & \mu \\ 0 & 0 & \mu & 0 & -\mu \end{bmatrix}$$

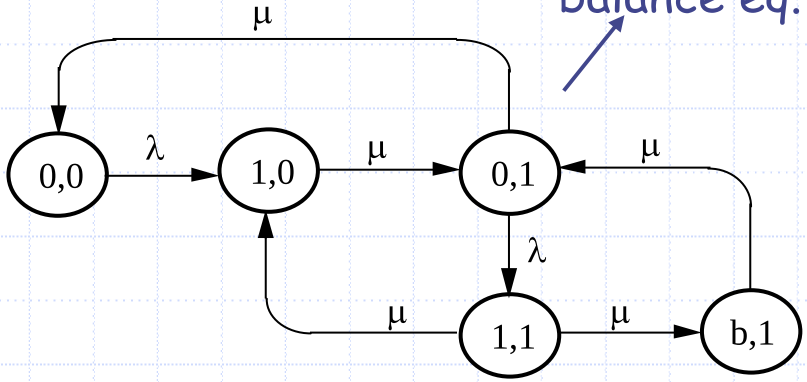
Examples

- Steady-state solution:

$$\left.\begin{aligned}\pi Q &= 0\\ \pi e &= 1\end{aligned}\right\} \longrightarrow \left|\begin{aligned}&\lambda\pi_{00} = \mu\pi_{01}\\ &\lambda\pi_{00} + \mu\pi_{11} = \mu\pi_{10}\\ &\mu\pi_{10} + \mu\pi_{b1} = (\lambda+\mu)\pi_{01}\\ &\lambda\pi_{01} = 2\mu\pi_{11}\\ &\mu\pi_{11} = \mu\pi_{b1}\end{aligned}\right.$$

$$\pi_{00} + \pi_{10} + \pi_{01} + \pi_{11} + \pi_{b1} = 1$$

balance eq.



$$\Rightarrow \pi = \left(\frac{2}{\Delta}, \frac{2\rho+\rho^2}{\Delta}, \frac{2\rho}{\Delta}, \frac{\rho^2}{\Delta}, \frac{\rho^2}{\Delta}\right) \quad \text{where } \rho = \frac{\lambda}{\mu} \quad \text{and} \quad \Delta = 3\rho^2 + 4\rho + 2$$

Examples

- Proportion of lost works?
  - Is the probability of the event “when a new work arrives, the first station is non-empty”, i.e.:

$$\pi_{10} + \pi_{11} + \pi_{b1} = \frac{3\rho^2 + 2\rho}{3\rho^2 + 4\rho + 2}$$

- Mean number of working stations?
  - In state (0,0) there is no working station and in state (1,1) there are two; in the rest of states there is only one, thus

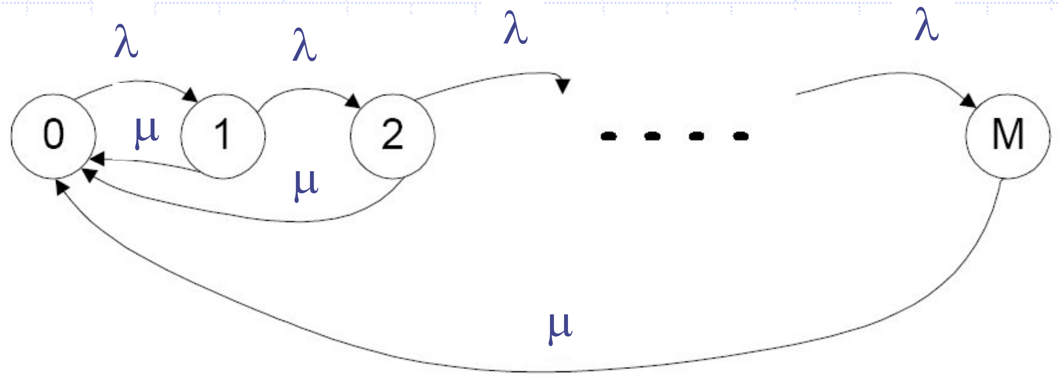
$$B = \pi_{01} + \pi_{10} + \pi_{b1} + 2\pi_{11} = \frac{4\rho^2 + 4\rho}{3\rho^2 + 4\rho + 2}$$

- Mean number of works in the system?
  - In state (0,0) there is no one; in states (1,1) and (b,1) there are two and in the rest there is only one, thus

$$L = \pi_{01} + \pi_{10} + 2\pi_{b1} + 2\pi_{11} = \frac{5\rho^2 + 4\rho}{3\rho^2 + 4\rho + 2}$$

Examples

- Example 3: I/O buffer with limited capacity
  - Records arrive according to a Poisson process (rate λ)
  - Buffer capacity: M records
  - Buffer cleared at times spaced by intervals which are exponentially distributed (parameter μ) and independent of arrivals



Examples

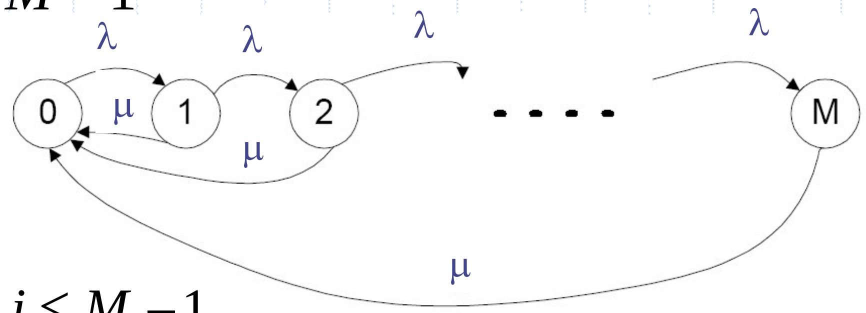
- Steady-state solution:

$$\lambda\pi_0 = \mu(\pi_1 + \cdots + \pi_M)$$

$$(\lambda+\mu)\pi_i = \lambda\pi_{i-1},\ 1 \le i \le M-1$$

$$\mu\pi_M = \lambda\pi_{M-1}$$

$$\pi_0 + \cdots + \pi_M = 1$$



$$\Rightarrow \pi_i = \left(\frac{\lambda}{\lambda+\mu}\right)^i \frac{\mu}{\lambda+\mu},\ 0 \le i \le M-1$$

$$\pi_M = \left(\frac{\lambda}{\lambda+\mu}\right)^M$$

Thus, for example, the mean number of records in the buffer in steady-state:

$$B = M\alpha^M + \sum_{i=0}^{M-1} i\alpha^i \frac{\mu}{\lambda+\mu}, \quad \text{where } \alpha = \frac{\lambda}{\lambda+\mu}$$

Performance modelling and evaluation

5. Birth-death processes

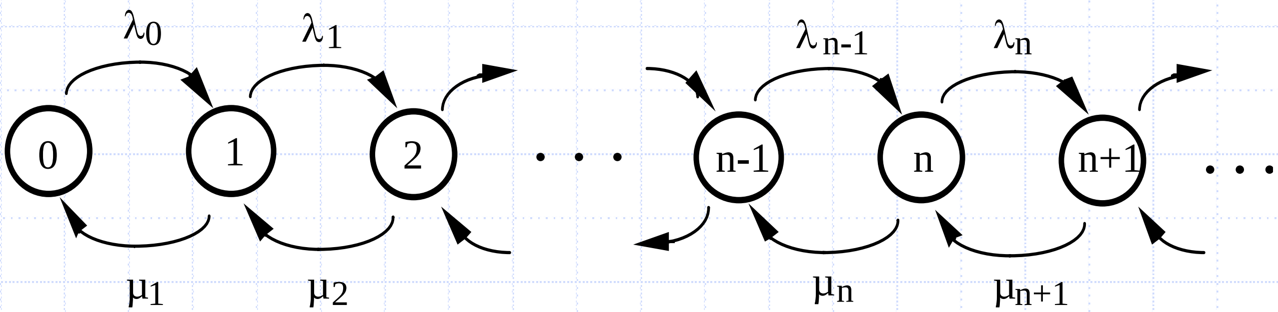
Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Definition
- Steady-state distribution
- Example

Definition

- A CTMC with state space $S = \{0,1,\ldots\}$ is called a ***birth-death process*** if the only non-zero transition rates are $q_{i,i+1}$ and $q_{i+1,i}$, $i \geq 0$, representing births and deaths, respectively.



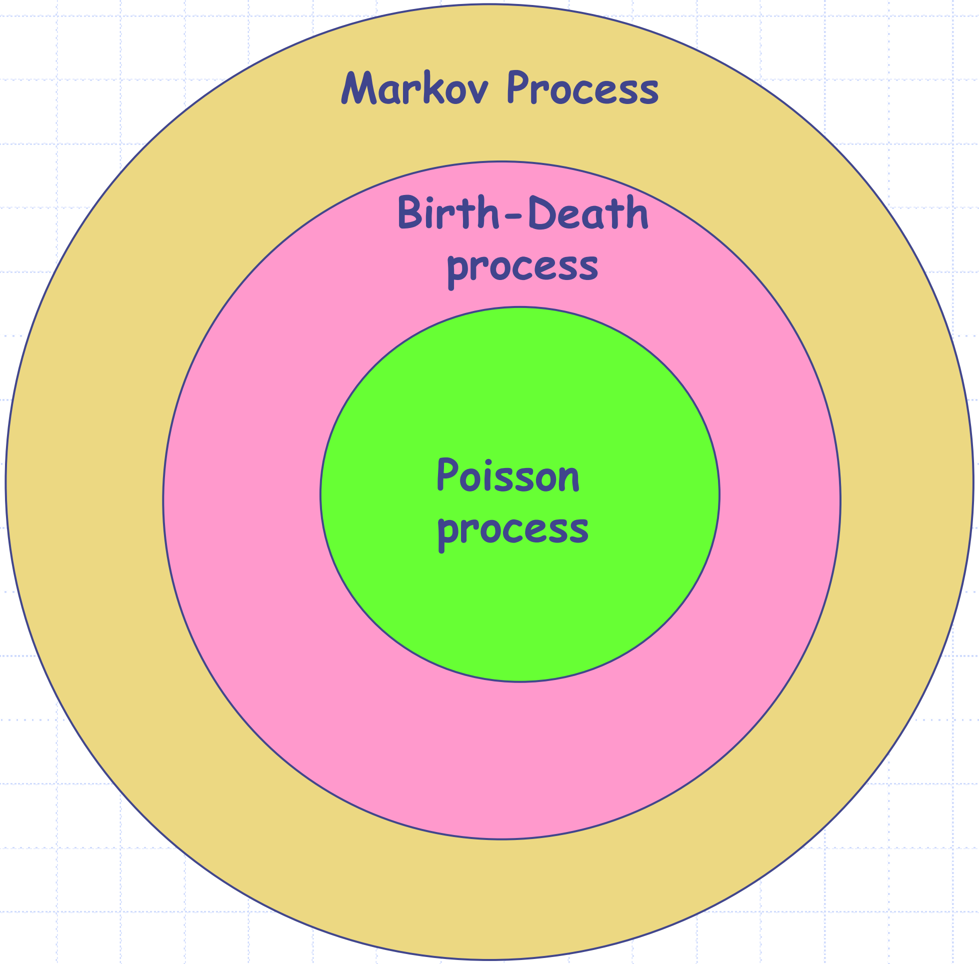
λ_n = birth rate in state n (i.e., $\lambda_n = q_{n,n+1}$)

μ_n = death rate in state n (i.e., $\mu_n = q_{n,n-1}$)

Population model

(Poisson process is a particular case of pure birth process)

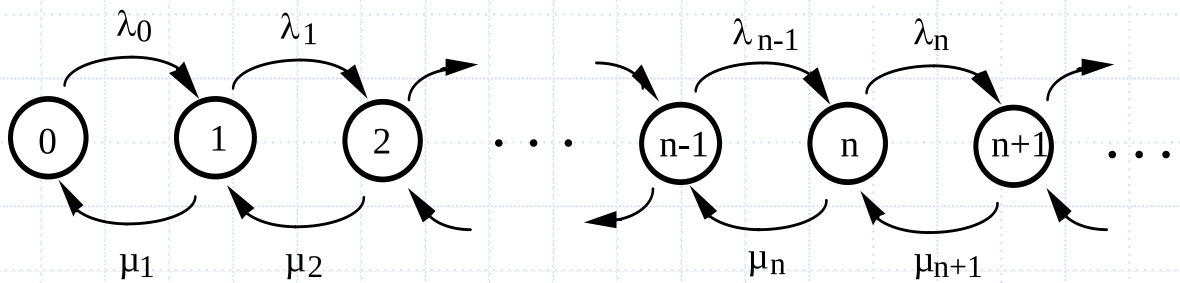
Definition



Definition

- Infinitesimal generator matrix:

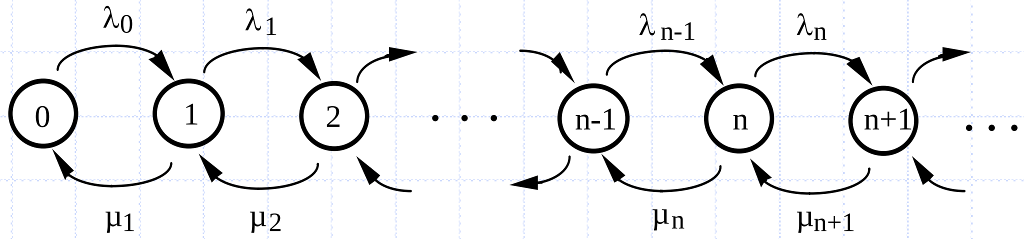
$$Q = \begin{bmatrix} -\lambda_0 & \lambda_0 & 0 & 0 & 0 & \cdots \\ \mu_1 & -(\lambda_1 + \mu_1) & \lambda_1 & 0 & 0 & \cdots \\ 0 & \mu_2 & -(\lambda_2 + \mu_2) & \lambda_2 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$



Outline

- Definition
- Steady-state distribution
- Example

Steady-state distribution



- Balance equations: $\pi Q = 0$

$$\lambda_0 \pi_0 = \mu_1 \pi_1$$

$$(\lambda_1 + \mu_1)\pi_1 = \lambda_0 \pi_0 + \mu_2 \pi_2$$

$$(\lambda_2 + \mu_2)\pi_2 = \lambda_1 \pi_1 + \mu_3 \pi_3$$

$$\cdots$$

$$(\lambda_{n-1} + \mu_{n-1})\pi_{n-1} = \lambda_{n-2} \pi_{n-2} + \mu_n \pi_n$$

$$(\lambda_n + \mu_n)\pi_n = \lambda_{n-1} \pi_{n-1} + \mu_{n+1} \pi_{n+1}, \quad n \geq 1$$

Steady-state distribution

- Then,

$$\Rightarrow \pi_{n+1} = \frac{\lambda_n}{\mu_{n+1}} \pi_n, \quad \mathrm{n} \geq 0 \quad \Rightarrow \quad \pi_n = \frac{\lambda_0 \ldots \lambda_{n-1}}{\mu_1 \ldots \mu_n} \pi_0, \quad \mathrm{n} \geq 1$$

- And normalizing equation, $\pi e = 1$, then,

$$\pi_0 = \frac{1}{1 + \sum_{n=1}^{\infty} \prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}}}$$

$$\pi_n = \frac{\prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}}}{1 + \sum_{n=1}^{\infty} \prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}}}, \quad n \geq 1$$

ergodicity condition:
$\pi_n > 0$, for all $n \geq 0$, i.e.

$$\sum_{n=1}^{\infty} \prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}} < \infty$$

Outline

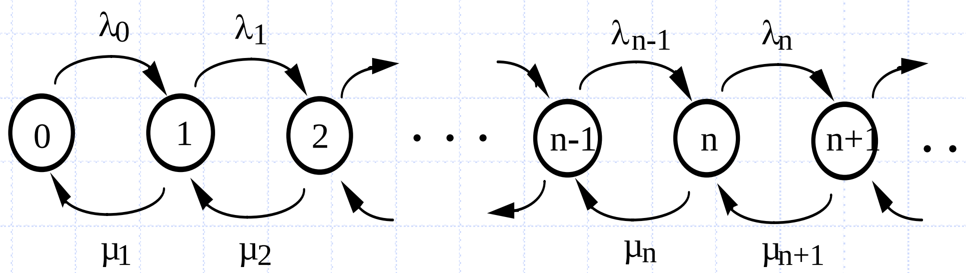
- Definition
- Steady-state distribution
- Example

Example

- Let $X(t)$ be the number of bacteria in a colony at instant t.
- Evolution of the population is described by:
  - the time that each of the individuals takes for division in two (binary fission), independently of the other bacteria, and
  - the life time of each bacterium (also independent)
- Assume:
  - Time for division is exponentially dist. (rate λ)
  - Life time is also exponentially dist. (rate μ)

Example

- Then, $X(t)$ is a birth-death process with



$\lambda_n = n\lambda$, n=0,1,2… and $\mu_n = n\mu$, n=1,2…

$\lambda_0 = 0 \Rightarrow$ state 0 is an absorbent state!

$$\lambda_0\pi_0 = \mu_1\pi_1$$
$$(\lambda_1 + \mu_1)\pi_1 = \lambda_0\pi_0 + \mu_2\pi_2$$
$$(\lambda_2 + \mu_2)\pi_2 = \lambda_1\pi_1 + \mu_3\pi_3$$
$$\ldots$$
$$(\lambda_{n-1} + \mu_{n-1})\pi_{n-1} = \lambda_{n-2}\pi_{n-2} + \mu_n\pi_n$$
$$(\lambda_n + \mu_n)\pi_n = \lambda_{n-1}\pi_{n-1} + \mu_{n+1}\pi_{n+1}, \quad n \geq 1$$

$\Rightarrow \pi_n = 0$, n>0

¿?

Example

- Define $P(t)$ as the average population at time t, then

$$P'(t) = (\lambda - \mu)P(t)$$

(the derivative is defined as the *instantaneous rate of change* of a function)

$$\Rightarrow P(t) = P(0)e^{(\lambda-\mu)t}$$

Then,

- if $\lambda < \mu$: the population tends to 0
- if $\lambda > \mu$: the population tends to infinity

Performance modelling and evaluation

6. Queueing models

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Basic definitions
- Notation
- General results
- M/M/1 queue
- M/M/c queue
- M/M/∞ queue
- M/M/c/B queue
- M/M/1/B/P queue
- An Erlangian model: M/E_k/1
- Models with general distributions

Basic definitions

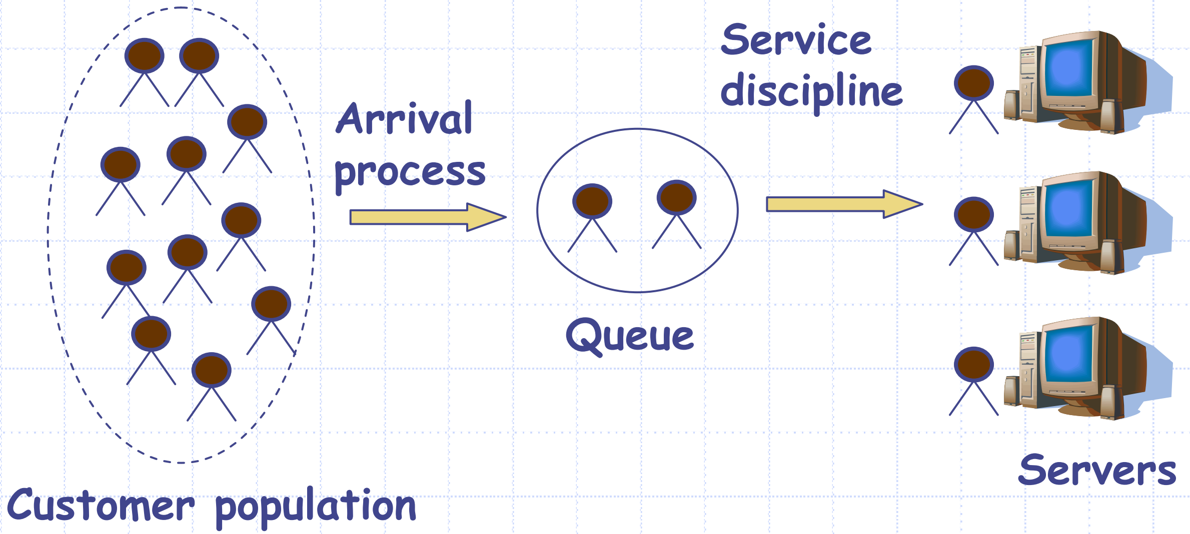
- Lipsky: "... a queue is a line of customers waiting to be served"
- Gross and Harris: "A Queueing System can be described as customers arriving for service, waiting for service if it is not immediate, and if having waited for service, leaving the system after being served"
- Cooper: "The term Queueing Theory is often used to describe the more specialized mathematical theory of waiting lines"

Basic definitions

- In computer systems, many jobs share the system resources such as CPU, disks, and other devices.
- When the resource is in use by one job, all other jobs wanting to use the resource have to wait in queues
- Queueing theory helps determine the amount of time spent by jobs in various queues, and in turn helps predict the response time, device utilizations, and throughput

Basic definitions

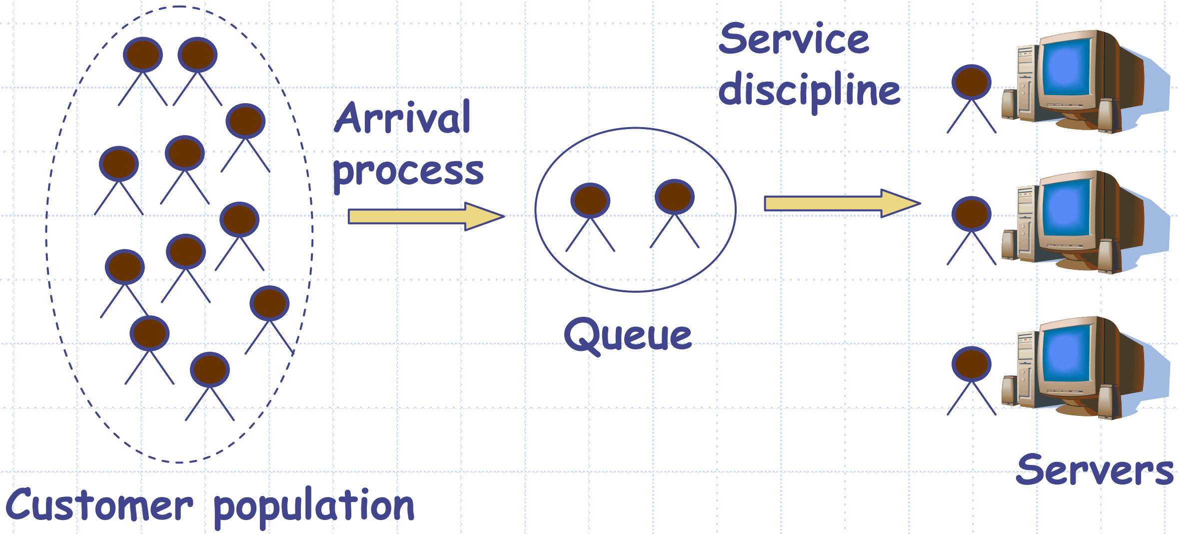
- Basic queueing model



- **State** of the model: the number of customers in the queue (including those being served)
- The state space may be unbounded, i.e. infinite (if an infinite population is assumed)

Basic definitions

- The following should be specified (to enable analysis):
  - Customer population
  - Arrival process
  - Number of servers
  - Service discipline
  - Service time distribution
  - System capacity



Basic definitions

- Population size
  - Potential customers who can enter the queue
  - Real systems have finite population
  - However, if population is large, assume infinite for ease of analysis
- Arrival process
  - Customers arrive at $t_1, t_2, \dots, t_j$
  - Interarrival time $\tau_j := t_j - t_{j-1}$
  - Assume interarrival times τ_j are IID random variables
  - E.g., Poisson process, Erlang, hyperexponential, general
- Service time distribution
  - Assume IID random variables
  - E.g., exponential, Erlang, hyperexponential, and general

Basic definitions

- Service disciplines
  - FIFO (or FCFS), first in first out most common
  - LIFO (or LCFS), last in first out
  - RR, Round Robin, a small fraction of time corresponds to each customer, cyclically
  - PS, Processor Sharing, limit situation of RR when the fraction of time tends to 0
  - Random
  - Priority disciplines…
    - Non preemptive: an ongoing service is not interrupted
    - Preemptive-resume: it is interrupted and resumes later on
    - Preemptive-restart: it is interrupted and restarts later on
  - Number of servers
    - One/many (identical) servers
- System capacity
  - Waiting space + number in service
  - Infinite assume if capacity is large

Outline

- Basic definitions
- Notation
- General results
- M/M/1 queue
- M/M/c queue
- M/M/∞ queue
- M/M/c/B queue
- M/M/1/B/P queue
- An Erlangian model: M/Ek/1
- Models with general distributions

Notation

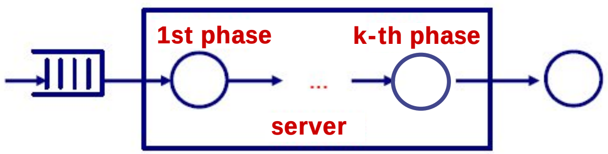
- Kendall notation
  - A/S/m/B/K/SD
  - A is interarrival time distribution
  - S is service time distribution
  - m is number of servers
  - B is number of buffers (system capacity)
  - K is population size
  - SD is service discipline

Notation

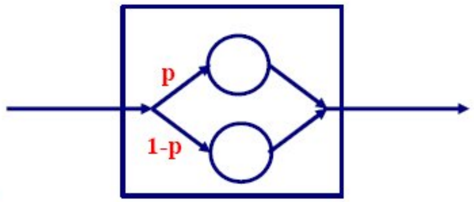
- Some common abbreviations
  - M (Markov): denotes exponential (and thus "memoryless" distribution)
  - D (Deterministic): values are constant
  - E_k (Erlang): Erlang distribution with k phases
  - H_k (Hyperexponential): Hyperexponential distribution with k branches
  - PH, phase type distribution
  - G (General): denotes distribution not specified; results are valid for all distributions
  - Bulk arrivals denoted using superscripts. E.g. $M^{[x]}$

Notation

- Coefficient of variation, $CV^2=\sigma^2/\mu^2$, gives a measure of the degree of irregularity of a positive random variable compared with an expon. dist. r.v.
  - $CV^2=1$: exponential model; the most frequently used pattern; good mathematical properties
  - $CV^2=0$: deterministic model
  - $CV^2=1/k$: Erlang-k model; intermediate between exponential and deterministic



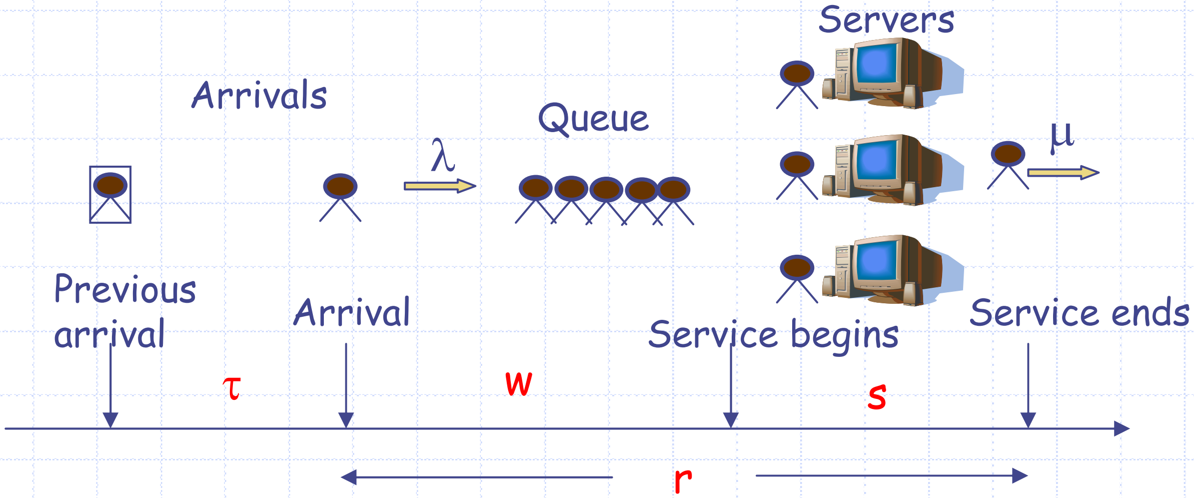
  - $CV^2>1$: Hyperexponential model; associated with parallel servers



Notation

- Examples
  - M/M/3/20/1500/FCFS is a single queue system such that:
    - Interarrivals are exponentially distributed
    - Service times are exponentially distributed
    - There are three servers
    - System capacity is 20; max. queue size is 20 – 3 = 17
    - Population size is 1500
    - Service discipline is FCFS
  - M/M/3: Typically, assume infinite system capacity, infinite population, and FIFO service. In such cases, last three parameters dropped.

Notation



- τ = interarrival time
- λ = mean arrival rate = $1/E[\tau]$
  - May depend upon # of jobs in system
- s = service time per job
- S = E[s] = mean service time
- μ = mean service rate per server = 1/E[s]; total service rate = $m\mu$
- X = throughput (mean number of completions per time unit)

They are r.v.'s (except mean values)

- n_q = # of jobs waiting in queue
- L= $E[n_q]$
- n_s = # of jobs recv. service
- n = # of jobs in system
  - $n = n_q + n_s$ is the queue length
- N = E[n]
- r = response time
  - Waiting for service plus time receiving service
- R = E[r]
- w = waiting time in queue
- W = E[w]

Outline

- Basic definitions
- Notation
- General results
- M/M/1 queue
- M/M/c queue
- M/M/∞ queue
- M/M/c/B queue
- M/M/1/B/P queue
- An Erlangian model: M/Ek/1
- Models with general distributions

General results

- Stability Condition
  - For stability, mean arrival rate should be less than mean service rate:

$$\lambda < m\mu$$

  - Does not apply to finite population systems and finite capacity systems
    - Queues for finite population systems cannot grow indefinitely
    - By definition, queues for finite buffer systems cannot grow indefinitely

General results

- Number in System versus Number in Queue
  - Number of jobs in system equals those waiting plus those being served
    - $n = n_q + n_s$
    - $E[n] = E[n_q]+E[n_s]$
    - That is, mean number of jobs in system equals mean number in queue plus mean number being served
- If service rate of servers independent of number of jobs in the queue, then
  - $Var[n] = Var[n_q]+Var[n_s]$
  - $Cov(n_q,n_s) = 0$

General results

- Time in System versus Time in Queue
  - Time spent by a job in system is sum of waiting time and service time. That is, $r = w + s$
    - Mean response time equals sum of the mean waiting time and the mean service time. $R = W + S$
  - If service rate independent of # of jobs in queue
    - $\mathrm{Cov}(w,s) = 0$
    - $\mathrm{Var}[r] = \mathrm{Var}[w] + \mathrm{Var}[s]$

General results

- Utilization Law
  - Utilization of a system (U) = fraction of time the system is busy

$$U = \frac{\text{busy time during period } T}{T}$$
$$= \frac{(\text{number of completions}) \cdot (\text{busy time during period } T)}{T \cdot (\text{number of completions})}$$
$$= X \cdot S$$

Assuming number of arrivals equal number of completions ($\lambda = X$): $U = \lambda \cdot S$

In case of queues with m servers: $U = \lambda / (m\mu)$

General results

- A disk is serving 50 requests/sec; each request requires 0.005 seconds of service.
  1) What is the Utilization?

     $U = 50 \times 0.005 = 0.25 \quad (25\%)$

  2) Maximum possible service rate?

     $U = 1 = (0.005)X$

     $X = 200$ requests/sec

General results

- A router forwards 100 packets/second onto a link. The transmission time (i.e., time to put packets onto the link), on average, is 1 ms.

  1) What is the link utilization?

     Link throughput: $X = 100$ packets/sec

     Service time: $S = 0.001$ sec

     $U = XS = 0.1 \quad (10\%)$

  2) Link capacity?

     $U = 1 = 0.0001X \Rightarrow X = 1000$ packets/sec

General results

- Number versus Time: Little´s Law
  - Assume **Job Flow Balance**: number of arrivals equal number of completions ($\lambda = X$)
    - New jobs not generated in the system
    - Jobs not lost (forever) in the system
    - If jobs lost in the system (e.g., due to finite capacity), law applies with adjusted arrival rate
  - Mean # in system = arrival rate x mean response time

$$N = \lambda R$$

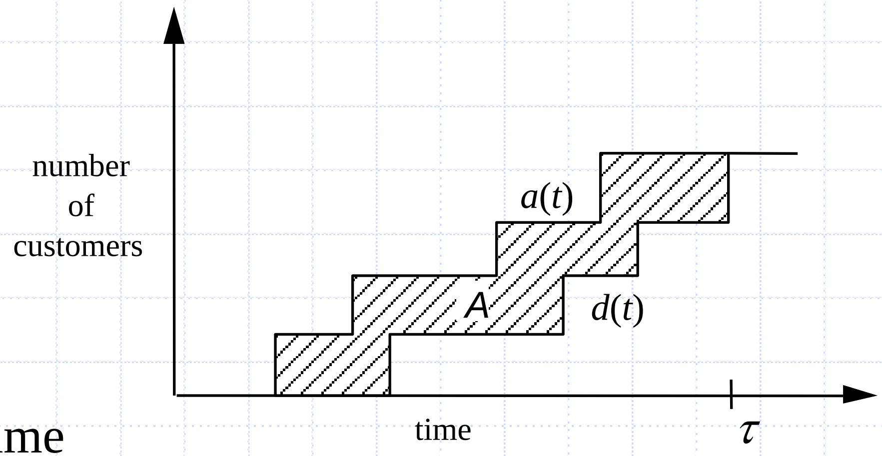
  - Mean # in queue = arrival rate x mean waiting time

$$L = \lambda W$$

  - We will intuitively derive this law …

General results

Intuitively...



$$\lambda = \frac{a(\tau)}{\tau} = \text{ mean arrival rate}$$

$$R = \frac{1}{a(\tau)} \sum_{k=1}^{a(\tau)} t_k = \text{ mean response time}$$

$$N = \frac{1}{\tau} \int_0^\tau n(t)dt = \text{ mean number of customers in system}$$

$$A = \int_0^\tau n(t)dt = \sum_{k=1}^{a(\tau)} t_k$$

$$\Rightarrow \tau\, N = a(\tau)\, R = \tau\, \lambda\, R$$

$$\Rightarrow \boxed{N = \lambda\, R}$$

General results

- A spy working for *Burguer King* tries to know how many clients are inside of a *McDonald´s*. He cannot enter but he observes:
  - Each hour, 32 clients arrive in average
  - Each one stays inside 12 minutes in average

By Little's Law, the average number of customers inside *McDonald´s* is

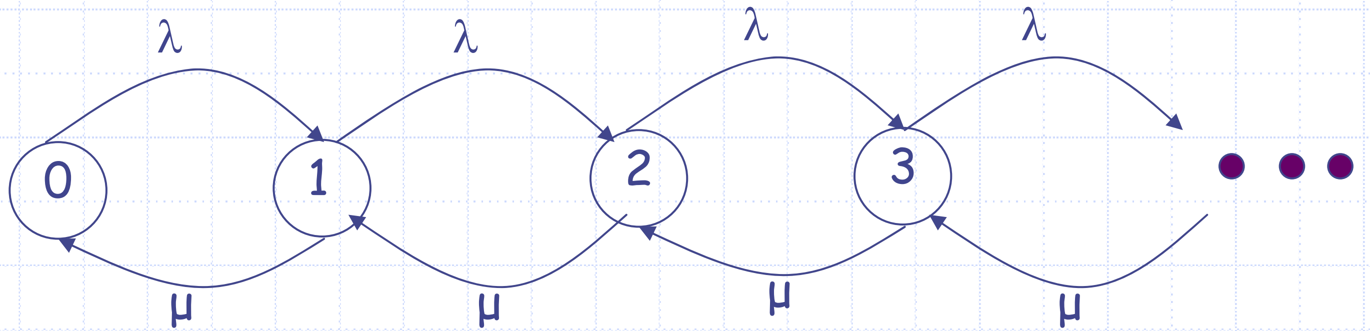
$$N = \lambda R = 0.53 \text{ customers/min} * 12 \text{ min} = 6.4 \text{ customers}$$

Outline

- Basic definitions
- Notation
- General results
- M/M/1 queue
- M/M/c queue
- M/M/∞ queue
- M/M/c/B queue
- M/M/1/B/P queue
- An Erlangian model: M/Ek/1
- Models with general distributions

M/M/1 queue

- One server, one queue, FIFO service
- exponentially distributed interarrival and service times
- infinite population, infinite capacity
- Can be model as a birth-death process
  - Constant birth and death rates



State n represents n customers in the system.
Remember notation: π_n = steady-state probability of being in state n.

M/M/1 queue

- Steady-state solution:
  - From balance equations

$$\pi_0 = \frac{1}{1 + \sum_{n=1}^{\infty} \left(\frac{\lambda}{\mu}\right)^n}$$

$$\pi_n = \left(\frac{\lambda}{\mu}\right)^n \pi_0, \quad n \geq 1$$

  - Traffic intensity

$$\rho = \frac{\lambda}{\mu}$$

  - Ergodicity condition

$$\sum_{n=1}^{\infty} \left(\frac{\lambda}{\mu}\right)^n < \infty \Leftrightarrow \lambda < \mu \Leftrightarrow \rho < 1$$

  - Bad cases
    - $\lambda > \mu$: transient MC
    - $\lambda = \mu$: null recurrent MC
  - Then, if $\rho < 1$

$$\pi_0 = \frac{1}{1 + \frac{\rho}{1-\rho}} = 1 - \rho$$

Thus

$$\boxed{\begin{aligned} \pi_0 &= 1 - \rho \\ \pi_n &= \rho^n (1 - \rho) \end{aligned}}$$

(geometric distribution)

M/M/1 queue

- Performance indices:
  - Utilization: prob. of one or more jobs in system (= mean # jobs in service)

$$U = 1 - \pi_0 = \rho$$

  - Mean # jobs in system

$$N = \sum_{n=1}^{\infty} n\pi_n = \sum_{n=1}^{\infty} n(1-\rho)\rho^n = \frac{\rho}{1-\rho}$$

  - Mean # jobs in queue

$$L = N - U = \frac{\rho}{1-\rho} - \rho = \frac{\rho^2}{1-\rho}$$

M/M/1 queue

- Mean response time (by Little's Law)

$$N = \lambda R \Rightarrow R = \frac{N}{\lambda} = \frac{\rho}{1-\rho} \cdot \frac{1}{\lambda} = \frac{1/\mu}{1-\rho}$$

- Mean waiting time in queue (Little's Law)

$$W = R - \frac{1}{\mu} = \frac{1/\mu}{1-\rho} - \frac{1}{\mu} = \frac{\lambda/\mu^2}{1-\rho}$$

- Prob. of finding *n* or more jobs in system

$$P(\#\text{ in system } \geq n) = \sum_{j=n}^{\infty} \pi_j = \sum_{j=n}^{\infty} (1-\rho)\rho^j = \ldots = \rho^n$$

M/M/1 queue

- Waiting time and response time distributions
  - Waiting times in queue exponentially distributed

$$P(w \leq t) = 1 - \rho\, e^{-\mu t(1-\rho)}$$

  - Response times exponentially distributed

$$P(r \leq t) = 1 - e^{-\mu t(1-\rho)}$$

M/M/1 queue

- Burke theorem:
  - If $\lambda < \mu$ then the departure process of a M/M/1 queue is a Poisson process with parameter λ (like the arrival process).
  - Proof:
    - The **reversed process** of a stochastic process is a dual process
      - with the same state space
      - in which the direction of time is reversed

      (like vieweing a video film backwards)
    - If the reversed process is stochastically identical to the original process, that process is called **reversible**.

M/M/1 queue

- A necessary and sufficient condition for reversibility are the **detailed balance equations**:

$$\pi_i q_{ij} = \pi_j q_{ji} \quad \text{for all states} \quad i \neq j$$

- An ergodic ($\lambda < \mu$) M/M/1 queue satisfies the detailed balance equations:
  - If i,j are not adjacent then q_{ij} and q_{ji} are null
  - If i,j are adjacent then, $i = n$, $j = n+1$, $q_{ij} = \lambda$, $q_{ji} = \mu$ and

$$\pi_n = \rho^n (1-\rho) \text{ and } \pi_{n+1} = \rho^{n+1}(1-\rho)$$

$$\text{then } \pi_n \lambda = \pi_{n+1}\mu$$

- Thus, an ergodic M/M/1 queue is reversible.
- The departure process of an M/M/1 queue is equal to the arrival process of the reversed queue.
- Since the reversed queue is again an M/M/1 queue then its arrival process is Poisson.

M/M/1 queue

- Example
  - Packets arrive at 100 packets/second at a router. The router takes 1 ms to transmit the incoming packets to an outgoing link. Using an M/M/1 model, answer the following:
    - What is utilization?
    - Probability of *n* packets in router?
    - Mean time spent in the router?
    - Probability of buffer overflow if router could buffer only 5 packets?
    - Buffer requirement to limit packet loss to 10^{-6}?

M/M/1 queue

- Arrival rate

  $\lambda = 100$ pps
- Service rate

  $\mu = 1/0.001 = 1000$pps
- Traffic intensity

  $\rho = 0.1$
- Mean packet residence time at router

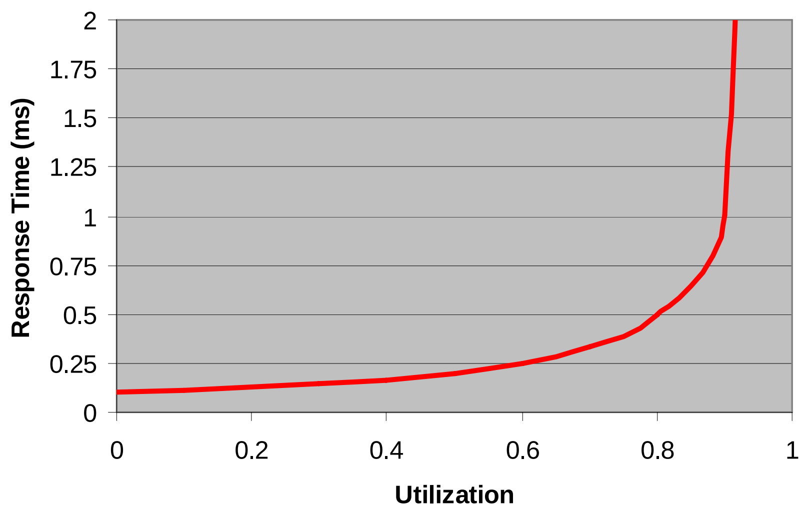
  $r = (1/\mu)(1/(1-\rho))$

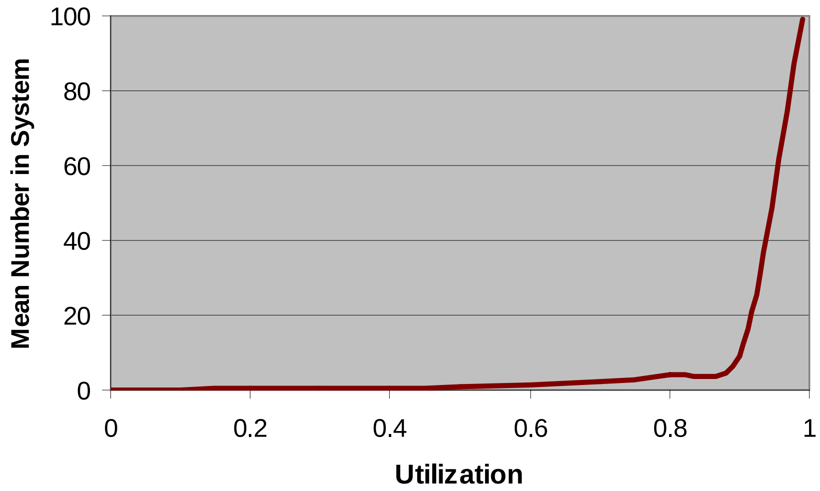
  $= 1.01$ ms
- Prob. of buffer overflow

  $P(\# \geq 6) = \rho^6 = 10^{-12}$
- To limit loss to less than 10^{-6}

  $\rho^n \leq 10^{-6}$

  $n > \log(10^{-6})/\log(0.1) > 3$



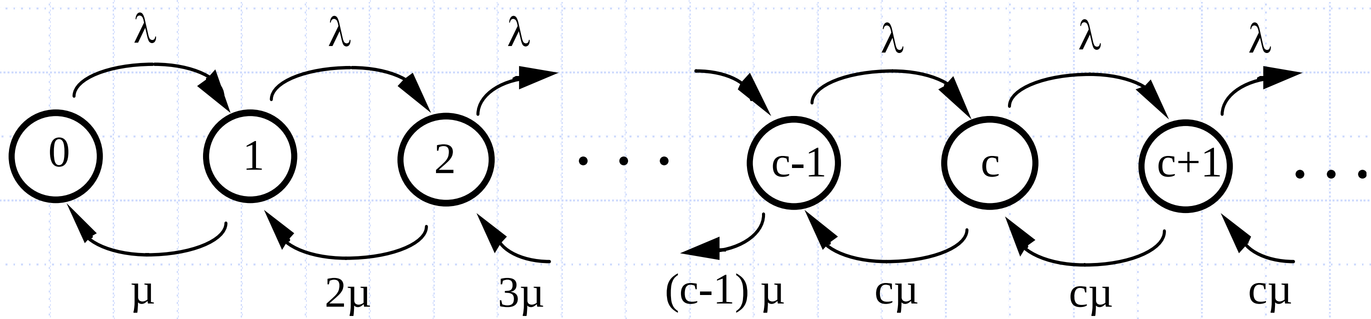


Outline

- Basic definitions
- Notation
- General results
- M/M/1 queue
- M/M/c queue
- M/M/∞ queue
- M/M/c/B queue
- M/M/1/B/P queue
- An Erlangian model: M/Ek/1
- Models with general distributions

M/M/c queue

- c servers, one queue, FIFO service, exponentially distributed interarrival and service times
- infinite population, infinite capacity
- Can be model as a birth-death process



State transition diagram for M/M/c queue

M/M/c queue

- Mean arrival rate: λ
- Mean service rate: $c\mu$
- Traffic Intensity (avg. utilization):

  $\rho = \lambda/(c\mu)$

  $\rho < 1$ for stability
- Flow balance equations yield:

  $\pi_n = ((c\rho)^n/n!)\pi_0,\ n = 1,\dots,c\text{-}1$

  $\pi_n = ((c\rho)^n/(c!c^{n-c}))\pi_0,\ n \geq c$
- And the probability of zero jobs in system

$$\pi_0 = \left[1 + \frac{(c\rho)^c}{c!(1-\rho)} + \sum_{n=1}^{c-1} \frac{(c\rho)^n}{n!}\right]^{-1}$$

M/M/c queue

- Other performance measures
  - Newly arrivals wait if all servers are busy, i.e., if c or more jobs are in the system

    $$P(\# \geq c \text{ jobs}) = \pi_c + \pi_{c+1} + \pi_{c+2} + \ldots$$

    $$C(\rho,c) = [(c\rho)^c]/[c!(1-\rho)]\ \pi_0$$

    $C(\rho,c)$ is known as **Erlang's C formula**

  - Mean # of jobs in system

    $$N = \mathrm{E}[n] = \Sigma n\pi_n,\ n = 0, 1, \ldots, \infty$$

    $$= [\pi_0 (c\rho)^c]/[c!(1-\rho)^2] + c\rho$$

    $$= c\rho + \rho\, C(\rho,c)/(1-\rho)$$

M/M/c queue

- Mean # of jobs in queue

  $L = E[n_q] = \Sigma(n-c)\pi_n,\ n = c, \ldots, \infty$

  $= [\pi_0\rho(c\rho)^c]/[c!(1-\rho)^2]$

  $= \rho\, C(\rho,c)/(1-\rho)$

- Mean response time (Little's law)

  $N = \lambda R$

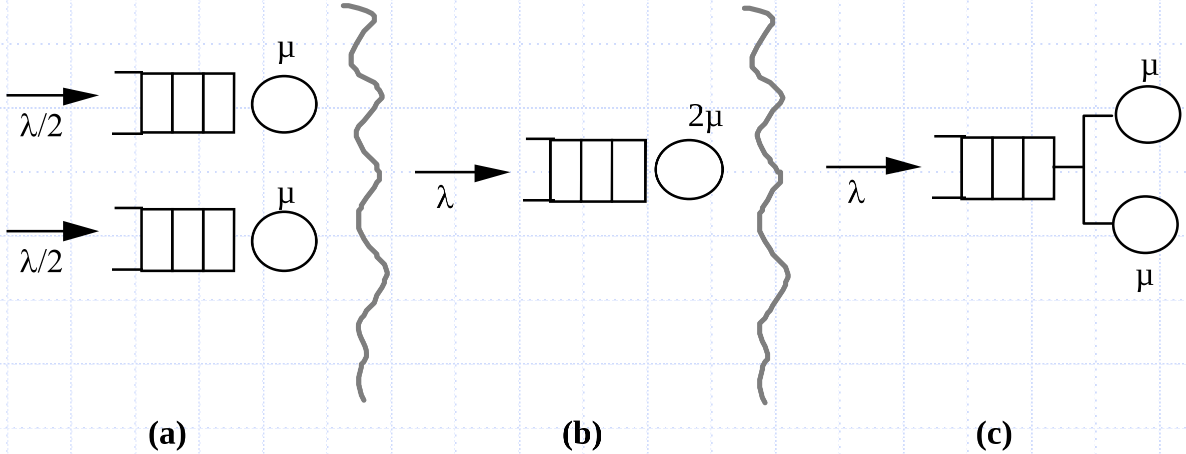
  $R = 1/\mu + C(\rho,c)/[c\mu(1-\rho)]$

- Mean waiting time in queue (Little's law)

  $W = L/\lambda = [\rho\, C(\rho,c)/(1-\rho)]/\lambda = C(\rho,c)/[c\mu(1-\rho)]$

M/M/c queue

- Exercise
  - Compare the following three systems
    a) Two independent M/M/1 queues with arrival rate $\lambda/2$ and service rate μ.
    b) One M/M/1 queue with arrival rate λ and service rate 2μ.
    c) One M/M/2 queue with arrival rate λ and service rate μ each server.

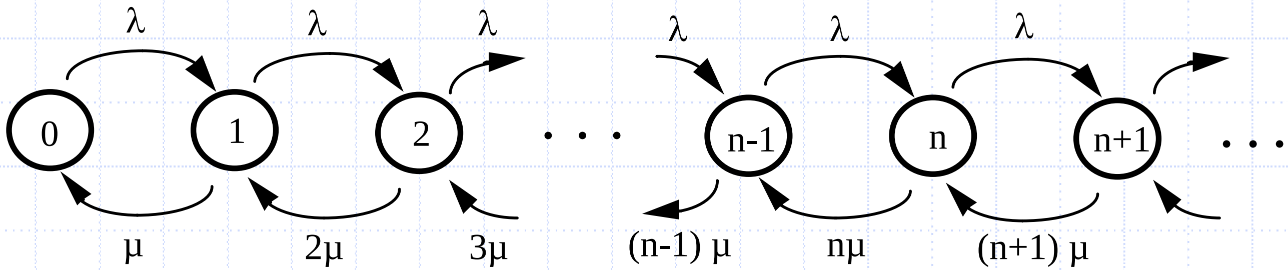


Outline

- Basic definitions
- Notation
- General results
- M/M/1 queue
- M/M/c queue
- M/M/∞ queue
- M/M/c/B queue
- M/M/1/B/P queue
- An Erlangian model: M/Ek/1
- Models with general distributions

M/M/∞ queue

- As many servers as customers arrive to the system
- one queue, FIFO service, exponentially distributed interarrival and service times
- infinite population, infinite capacity
- can be model as a birth-death process



$$\lambda_n = \lambda \qquad \forall n \in \mathbb{N}$$

$$\mu_n = n\mu \qquad \forall n \in \mathbb{N}$$

M/M/∞ queue

- Steady-state solution:

$$\pi_n = \frac{(\lambda/\mu)^n}{n!} e^{-\lambda/\mu} \qquad \text{(Poisson distribution)}$$

- Performance indices:
  - Mean # of jobs in system

$$N = \sum_{n=0}^{\infty} n\pi_n = \frac{\lambda}{\mu}$$

  - Mean response time (Little's law)

$$R = \frac{N}{\lambda} = \frac{1}{\mu}$$

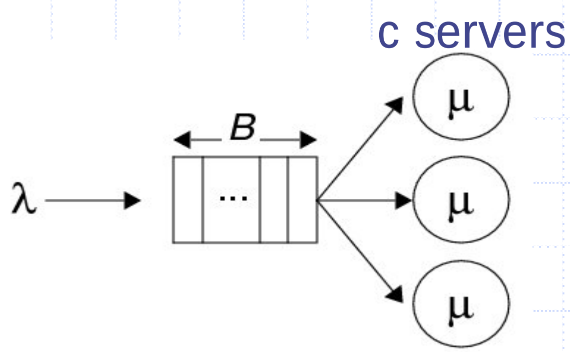
Obviously equal to mean service time

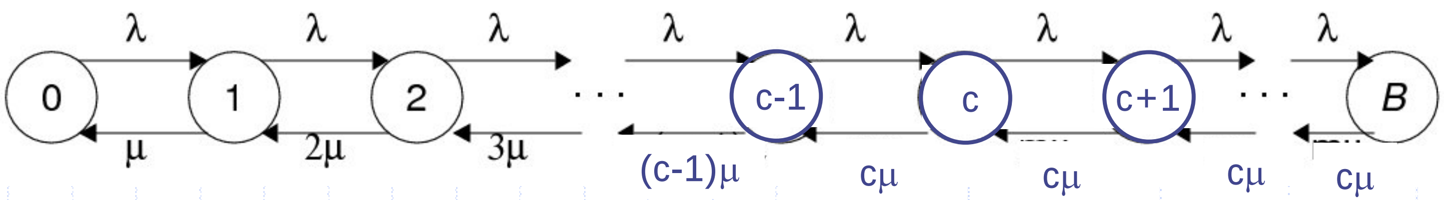
Outline

- Basic definitions
- Notation
- General results
- M/M/1 queue
- M/M/c queue
- M/M/∞ queue
- M/M/c/B queue
- M/M/1/B/P queue
- An Erlangian model: M/Ek/1
- Models with general distributions

M/M/c/B queue

- *c* servers, one queue, FIFO service, exponentially distributed interarrival and service times
- infinite population
- capacity of system equal to B
- can be model as a finite birth-death process





M/M/c/B queue

- Steady-state distribution:

$$\pi_n = \begin{cases} \dfrac{1}{n!}\left(\dfrac{\lambda}{\mu}\right)^n \pi_0, & n = 1,2,\ldots,c-1 \\ \dfrac{1}{c!\,c^{n-c}}\left(\dfrac{\lambda}{\mu}\right)^n \pi_0, & n = c,c+1,\ldots,B \end{cases}$$

$$\pi_0 = \begin{cases} \left(\displaystyle\sum_{n=0}^{c-1} \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n + \frac{(\lambda/\mu)^c}{c!}\cdot\frac{1-(\lambda/c\mu)^{B-c+1}}{1-\lambda/c\mu}\right)^{-1}, & \text{if } \lambda/c\mu \neq 1 \\ \left(\displaystyle\sum_{n=0}^{c-1} \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n + \frac{(\lambda/\mu)^c}{c!}\cdot(B-c+1)\right)^{-1}, & \text{if } \lambda/c\mu = 1 \end{cases}$$

Exercise: taking the limit as $B \to \infty$ and restricting $(\lambda/(c\mu)<1$ we get the result for M/M/c/∞.

M/M/c/B queue

- Performance measures:
  - Effective arrival rate (rate of jobs actually entering the system)
  - Expected # of jobs in system
  - Mean # jobs waiting in queue
  - Mean response time and mean witing time

$$\lambda' = \sum_{n=0}^{B-1} \lambda \pi_n = \lambda(1 - \pi_B)$$

$$N = \sum_{n=0}^{B} n \pi_n$$

$$L = \sum_{n=c+1}^{B} (n - c)\pi_n$$

$$R = N / \lambda'$$

$$W = L / \lambda'$$

M/M/c/B queue

- The particular case of M/M/1/B

$$\pi_n = \begin{cases} \dfrac{(1-\lambda/\mu)(\lambda/\mu)^n}{1-(\lambda/\mu)^{B+1}}, & \text{if } \lambda/\mu \neq 1 \\ \dfrac{1}{k+1}(\lambda/\mu)^n, & \text{if } \lambda/\mu = 1 \end{cases}$$

- Expected # of jobs in system

$$N = \frac{\lambda/\mu}{1-\lambda/\mu} - \frac{(B+1)(\lambda/\mu)^{B+1}}{1-(\lambda/\mu)^{B+1}}$$

- Mean # jobs waiting in queue

$$L = N - \frac{(\lambda/\mu)\left(1-(\lambda/\mu)^B\right)}{1-(\lambda/\mu)^{B+1}}$$

Outline

- Basic definitions
- Notation
- General results
- M/M/1 queue
- M/M/c queue
- M/M/∞ queue
- M/M/c/B queue
- M/M/1/B/P queue
- An Erlangian model: M/Ek/1
- Models with general distributions

M/M/1/B/P queue

- Single server model with system capacity B and population size P (potential customers)
- If there are $P-n$ individuals available:

arrival rate is $(P-n)\lambda$

Thus, it can be modeled as a finite birth-death process with birth/deadth rates:

$$\lambda_n = \begin{cases} (P-n)\lambda, & \text{if } n < B \\ 0, & \text{if } n \geq B \end{cases}$$

$$\mu_n = \mu$$

M/M/1/B/P queue

- Steady state:

$$\pi_n = \left(\prod_{j=0}^{n-1} \frac{(P-j)\lambda}{\mu} \right) \pi_0 = \frac{P!}{(P-n)!} \left(\frac{\lambda}{\mu} \right)^n \pi_0, \quad \text{if} \;\; n \leq B$$

$$\pi_0 = \frac{1}{1 + \sum_{j=1}^{B} \frac{P!}{(P-j)!} \left(\frac{\lambda}{\mu} \right)^j}$$

- Mean # customers in system

$$N = \frac{\sum_{n=1}^{B} n \frac{(\lambda/\mu)^n}{(P-n)!}}{\sum_{j=0}^{B} \frac{(\lambda/\mu)^j}{(P-j)!}}$$

M/M/1/B/P queue

- Special case:
  - $B = P$: "machine-repairman" model

    P independent machines each of which fails as Poisson process.

    Then they can be viewed as queueing up for a single repairman who repairs machines in an exponentially distributed time.

    Availability of the system is defined as the probability of finding the system capable of doing some work (at least one machine running)

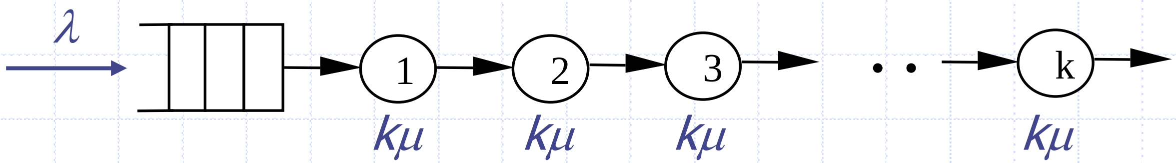
$$\text{availability} = 1 - \pi_P$$

Outline

- Basic definitions
- Notation
- General results
- M/M/1 queue
- M/M/c queue
- M/M/∞ queue
- M/M/c/B queue
- M/M/1/B/P queue
- An Erlangian model: M/Ek/1
- Models with general distributions

An Erlangian model: $M/E_k/1$

- ❑ Exponentially distributed interarrivals
- ❑ Erlang phase *k* service time

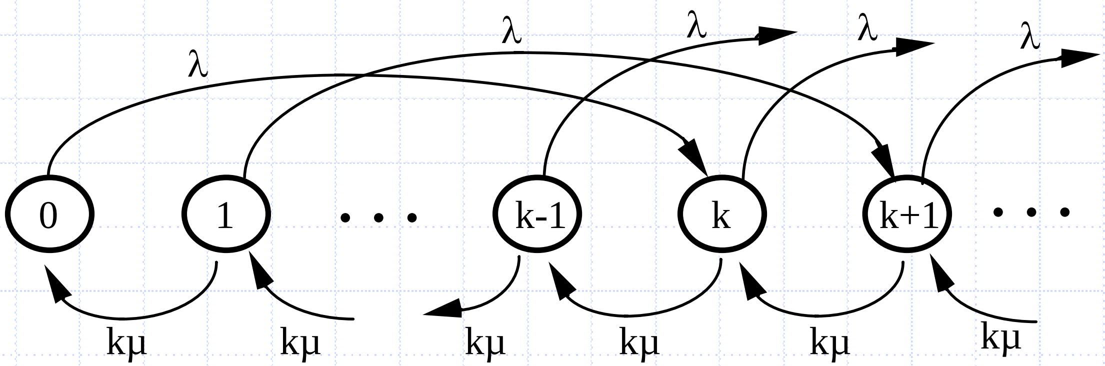


$E[X] = 1/\mu; \quad CV^2 = 1/k$

- ❑ Erlang is nor memoryless (for $k > 1$)
  $\Rightarrow$ the normal description of the state of the queue results in a non-Markovian process
- ❑ We change definition of state
  - ❑ State = number of stages of service to be completed which are currently in the system

An Erlangian model: $M/E_k/1$

- We get a CTMC that is not a birth-death process



An Erlangian model: $M/E_k/1$

- Steady-state solution: Return to balance equations… $\pi Q = 0$

  Solving equations and defining $\rho=\lambda/\mu$

$$E[N_{\text{stages}}] = \frac{(k+1)\rho}{2(1-\rho)}$$

$$L = \frac{(k+1)\rho^2}{2k(1-\rho)}$$

etc.

Outline

- Basic definitions
- Notation
- General results
- M/M/1 queue
- M/M/c queue
- M/M/∞ queue
- M/M/c/B queue
- M/M/1/B/P queue
- An Erlangian model: M/Ek/1
- Models with general distributions

Models with general distributions

- M/G/1 queue

$$W = \frac{\lambda \overline{x^2}/2}{1-\rho}$$

where $\overline{x^2}$ is the second order moment of service time

and $\rho = \lambda/\mu$ (with μ the inverse of the mean)

- G/M/1 queue

$$W = \frac{\sigma}{\mu(1-\sigma)}$$

where σ is the unique root in the range $0 \le \sigma < 1$

of the functional equation $\sigma = A^*(\mu - \mu\sigma)$

where A^* is the Laplace transform of the pdf of the interarrival time

Performance modelling and evaluation

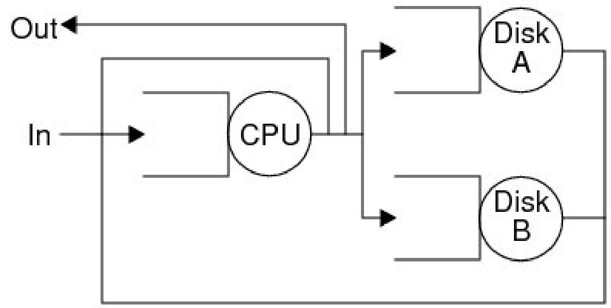
7. Queueing networks

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Basic concepts
- Open queueing networks
- Closed queueing networks
- BCMP networks

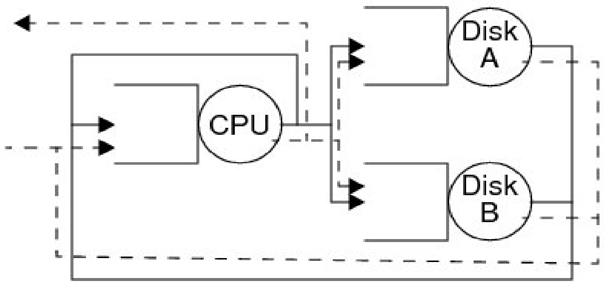
Basic concepts



- A QN is a collection of servers/queues interconnected according to some topology where jobs departing from one server arrive at another queue for service.
- Servers may be
  - processing elements in a computer, e.g. CPU or I/O devices,
  - stations/nodes in a communication network (may be widely separated geographically),
  - machines in a flexible manufacturing system,
  - semaphores in a traffic map of a city, etc., etc.
- Topology represents the possible routes taken by tasks through the system.

Basic concepts

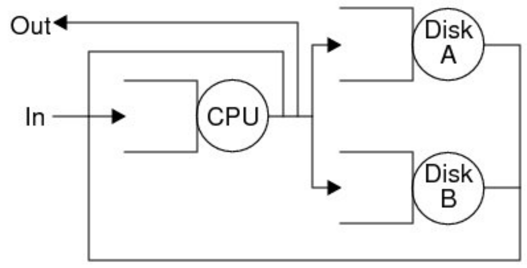
- May be several different classes of tasks (**multiclass** network):
  - different service requirements at each node,
  - different routing behaviours,
  - more complex notation, but straightforward generalisation of the single-class network in principle,
  - we will consider only the single class case.



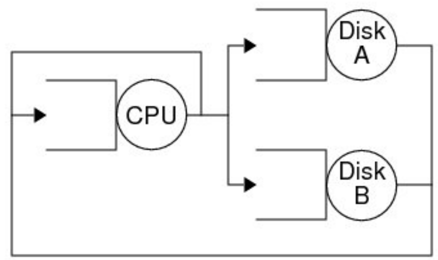
multiclass network

Basic concepts

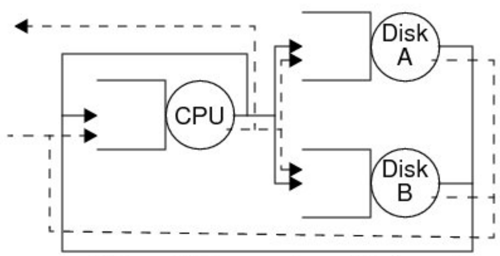
- Types of networks:
  - An **open** QN has external arrivals and departures.
  - A **closed** QN has no external arrivals or departures. The number of customers circulating remains constant (population) .
  - A **mixed** QN is open for some workloads (classes) and closed for others (multiclass QN case).



Open Queueing Network



Closed Queueing Network



Mixed Queueing Network

Basic concepts

- QN input parameters:
  - The number of stations within the network.
  - The service-time distribution (we assume exponential but could be different), may be specified as the average service time s_i, or the service rate $\mu_i = 1/s_i$.
  - The scheduling or queueing discipline at each station: FCFS, LCFS-PR, PS (processor sharing), IS (infinite number of servers, i.e. a delay node: no queueing)
  - The routing probabilities of the customers among all the stations, specified as q_{ij}, which gives the fraction of the customers completing service at station i that join queue j.
  - The population size N for closed queueing networks, or the interarrival-time distribution for the open networks (given as the external arrival rate λ_i).

Basic concepts

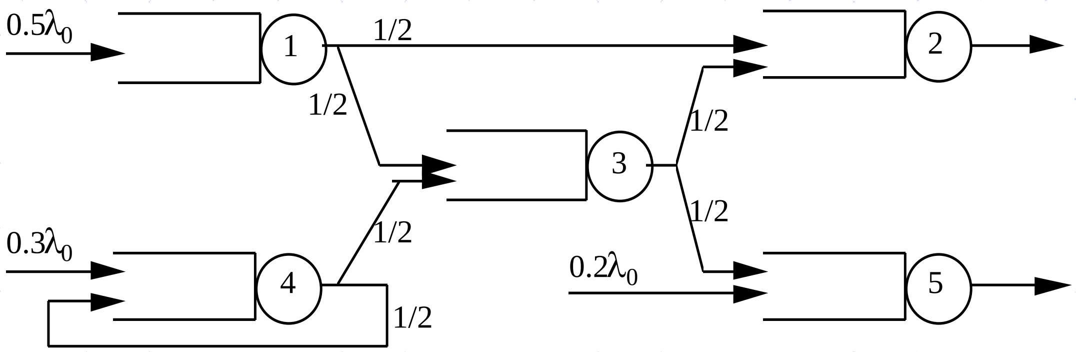
- QN output parameters (performance measures) include:
  - Resource utilization U_i representing the fraction of the time that the resource at station i is busy.
  - Throughput λ_i denoting the average number of job completions per unit time at station i.
  - Average queue length Q_i denoting the number of customers in the queue i including the customer in service.
  - Average response time R_i denoting the amount of time that a customer spends at a station i.
  - Average waiting line length L_i denoting the number of customers waiting in line i excluding the customer in service.
  - Average waiting time W_i denoting the amount of time a customer spends waiting for service at station i.

Outline

- Basic concepts
- Open queueing networks
- Closed queueing networks
- BCMP networks

Open queueing networks

- Assume single class of customers.
- M servers, 1,2…,M, with FCFS discipline and exponential service times, mean μ_i.
- External Poisson arrivals into node j, rate γ_j ($1 \le j \le M$) (= 0 if no arrivals). Total external arrivals: Poisson with rate $\lambda_0 = \gamma_1 + \ldots + \gamma_M$. For instance $\lambda_0 = 10$.
- State space of network $S = \{(n_1, \ldots, n_M) \mid n_i \ge 0\}$
  - queue length vector random variable is $(N_1, \ldots, N_M)$
  - $p(n) = p(n_1, \ldots, n_M) = P(N_1 = n_1, \ldots, N_M = n_M)$



Open queueing networks

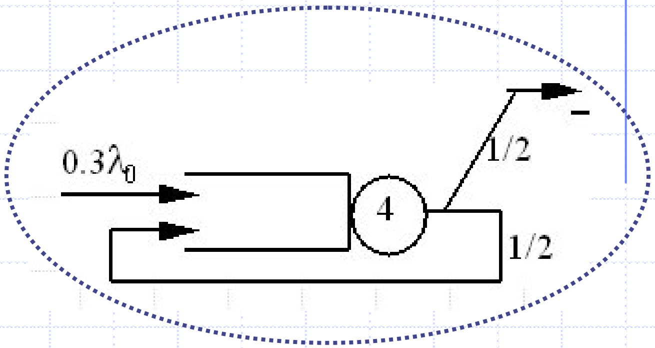
- The outside of the network can be represented as a new queue labelled 0.
- Routing probability matrix $Q = \{q_{ij} \mid 0 \le i,j \le M\}$
  - $q_{0j} = \gamma_j$, $j = 1 \ldots M$; $q_{00} = 0$.
  - q_{ij} = probability that on leaving *node i* a task goes to node j independently of past history ($1 \le i,j \le M$).
  - $q_{i0} = 1 - (q_{i1} + \ldots + q_{iM})$, $i = 1 \ldots M$.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1/2 & 0 & 0 & 3/10 & 1/5 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Open queueing networks

- Traffic equations:
  - For each queue *i*:

$$\lambda_i = \gamma_i + \sum_{j=1}^{M} \lambda_j q_{ji}, \text{ for } 1 \le i \le M$$



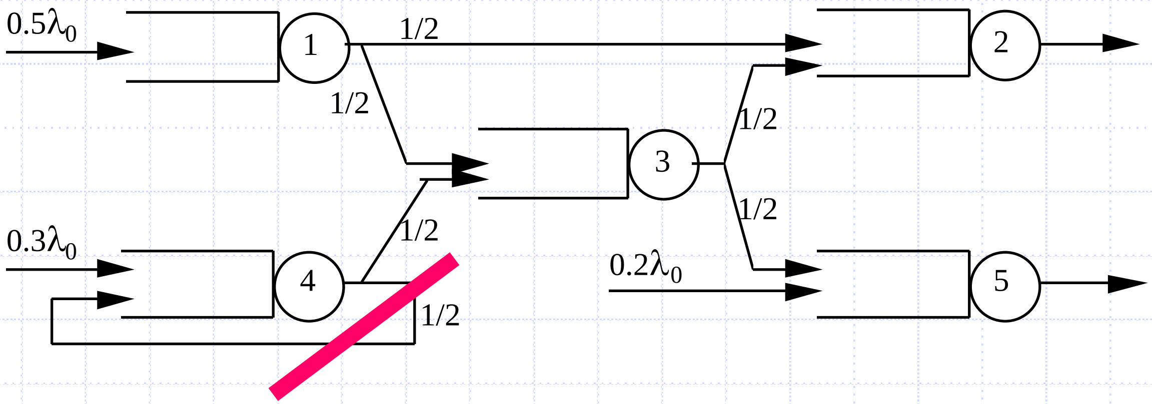
  - Example: $\lambda_4 = 0.3\lambda_0 + \frac{1}{2}\lambda_4 \Rightarrow \lambda_4 = 0.6\lambda_0$
  - In matrix form and adding virtual "queue" 0:

$$\Lambda = \Lambda Q$$

    - independent of Poisson assumption since we are only considering mean numbers of arrivals in unit time
    - assumes only the existence of a steady state

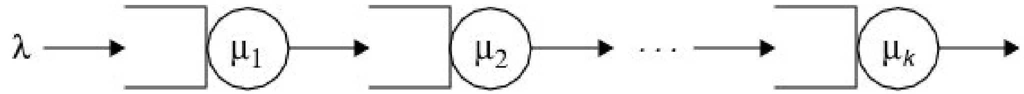
Open queueing networks

- Arrivals to a node are not in general Poisson.
- If there is no feedback then all arrival processes are Poisson because
  1. departure process of M/M/1 queue is Poisson
  2. superposition of independent Poisson processes is Poisson



Open queueing networks

- Example: consider the simple open QN



  - Each queue behaves as an **independent** M/M/1 with arrival rate λ and service rate μ_i.
    - i.e., the probability of n_i jobs in the queue i in steady state is $(1-\rho_i)\rho_i^{n_i}$ with $\rho_i=\lambda/\mu_i$ (utilization of queue i).
  - Then, the joint probability is just the product:

$$\pi_{n_1 n_2 \ldots n_k} = (1-\rho_1)\rho_1^{n_1}(1-\rho_2)\rho_2^{n_2}\ldots(1-\rho_k)\rho_k^{n_k}$$

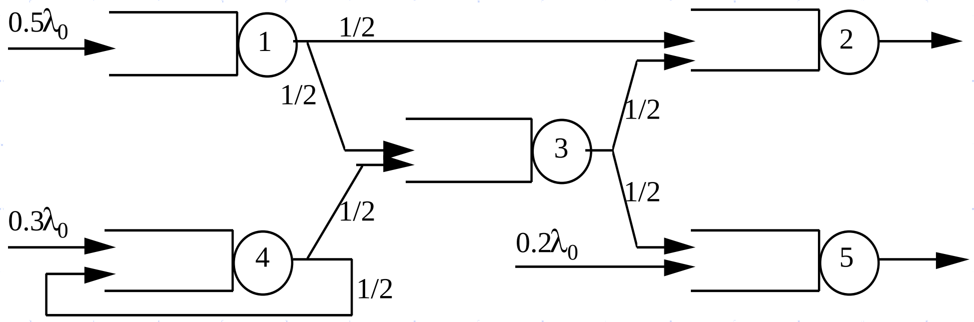
Any QN exhibiting such a property is called a **product form QN** (PF-QN or QN with a product form solution).

Open queueing networks

- Visit ratios: they are the relative throughputs, normalized for a given queue (for instance queue 1).
- In the example

$$v = vQ$$

$$v_1 = 1$$



$$\left.\begin{aligned} \tfrac{1}{3}v_2 &= v_1 \\ \tfrac{1}{2}v_1 + \tfrac{1}{2}v_3 &= v_2 \\ \tfrac{1}{2}v_1 + \tfrac{1}{2}v_3 + \tfrac{1}{2}v_4 &= v_3 \\ \tfrac{1}{3}v_2 + v_5 &= v_4 \\ \tfrac{1}{3}v_2 + \tfrac{1}{2}v_4 &= v_5 \\ v_1 &= 1 \end{aligned}\right\} \Rightarrow v = (1, 3, 5, 4, 3)$$

Open queueing networks

- Steady-state queue length probabilities
  - Jackson's theorem (1963)
    - The number of tasks at any server is **independent** of the number of tasks at every other server in the steady state.
    - Node i behaves ***as if*** it were subjected to Poisson arrivals, rate i ($1 \le i \le M$).

      Thus, even though arrivals at each node are not, in general, Poisson, we can treat the system as if it were a collection of M **independent** M/M/1 queues (PF-QN).

remember: M/M/1 queue

$$\pi_n = \rho^n (1-\rho)$$

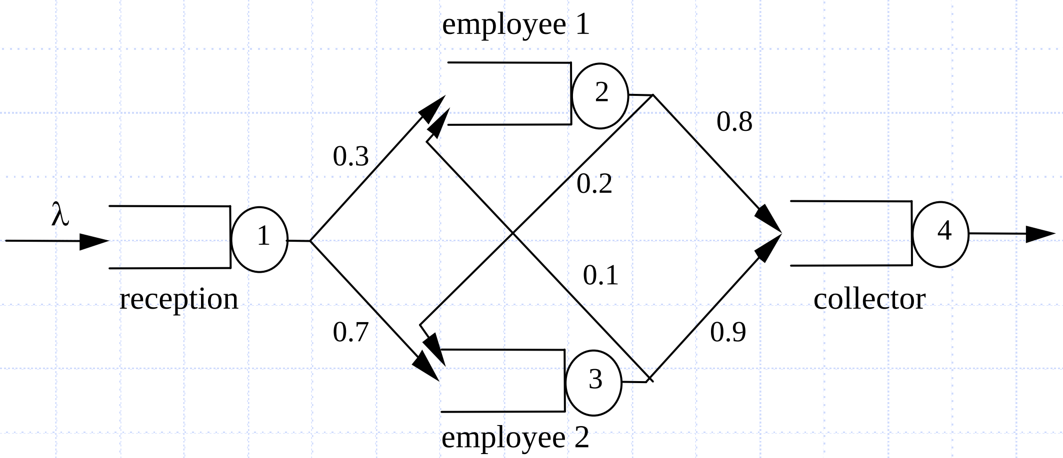
$$\pi_{n_1 n_2 \dots n_M} = \frac{\rho_1^{n_1} \rho_2^{n_2} \dots \rho_M^{n_M}}{C}$$

$$C = 1/\pi_{0,0 \dots 0} = \frac{1}{(1-\rho_1)(1-\rho_2)\dots(1-\rho_M)}$$

C is a normalization constant

Open queueing networks

- Example
  - Registry of motor vehicles in a city
    - Reception: a person directs the customer to the right employee – 20 seconds in average
    - Employees: execute the tasks – two different employees depending of the type of vehicle – 10 minutes / 5 minutes
    - Collector: collects the money – 1 minute



Open queueing networks

- Throughput (traffic equations):

$$\left.\begin{aligned} \lambda_1 &= \lambda \\ \lambda_2 &= 0.3\lambda + 0.1\lambda_3 \\ \lambda_3 &= 0.7\lambda + 0.2\lambda_2 \\ \lambda_4 &= 0.8\lambda_2 + 0.9\lambda_3 \end{aligned}\right\} \Rightarrow \begin{aligned} \lambda_1 &= \lambda \\ \lambda_2 &= 0.38\lambda \\ \lambda_3 &= 0.78\lambda \\ \lambda_4 &= \lambda \end{aligned}$$

- Utilizations:

$$\begin{aligned} \rho_1 &= s_1\lambda_1 = & 20\lambda \\ \rho_2 &= s_2\lambda_2 = 600 * 0.38\lambda = & 228\lambda \\ \rho_3 &= s_3\lambda_3 = 300 * 0.78\lambda = & 234\lambda \\ \rho_4 &= s_4\lambda_4 = & 60\lambda \end{aligned}$$

- Steady-state condition: all utilizations < 1

  $\lambda < 0.00427$ customers/sec. (15.38 cust/hour)

  bottleneck station: employee 2

Open queueing networks

- Steady-state queue length probabilities :

$$\pi_{n_1,n_2,n_3,n_4} = (1-20\lambda)(20\lambda)^{n_1} \cdot (1-228\lambda)(228\lambda)^{n_2} \cdot$$

$$\cdot(1-234\lambda)(234\lambda)^{n_3} \cdot (1-60\lambda)(60\lambda)^{n_4}$$

- Bounds for the mean queue lengths (using λ < 0.00427)

$$E[n_1] = \frac{\rho_1}{1-\rho_1} = \frac{20\lambda}{1-20\lambda} \leq 0.093$$

reception is busy at most 8,5% of the time

$$E[n_2] = \frac{\rho_2}{1-\rho_2} = \frac{228\lambda}{1-228\lambda} \leq 36.82$$

$$E[n_3] = \frac{\rho_3}{1-\rho_3} = \frac{234\lambda}{1-234\lambda} \leq \infty$$

bottleneck

$$E[n_4] = \frac{\rho_4}{1-\rho_4} = \frac{60\lambda}{1-60\lambda} \leq 0.34$$

collector is busy at most 25,6% of the time

Open queueing networks

- Response time (waiting time of a customer in the system), by Little's law:

$$T = \frac{N}{\lambda} = \frac{E[n_1] + E[n_2] + E[n_3] + E[n_4]}{\lambda} =$$

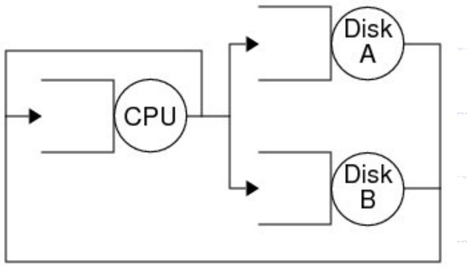
$$= \frac{20}{1-20\lambda} + \frac{228}{1-228\lambda} + \frac{234}{1-234\lambda} + \frac{60}{1-60\lambda}$$

Outline

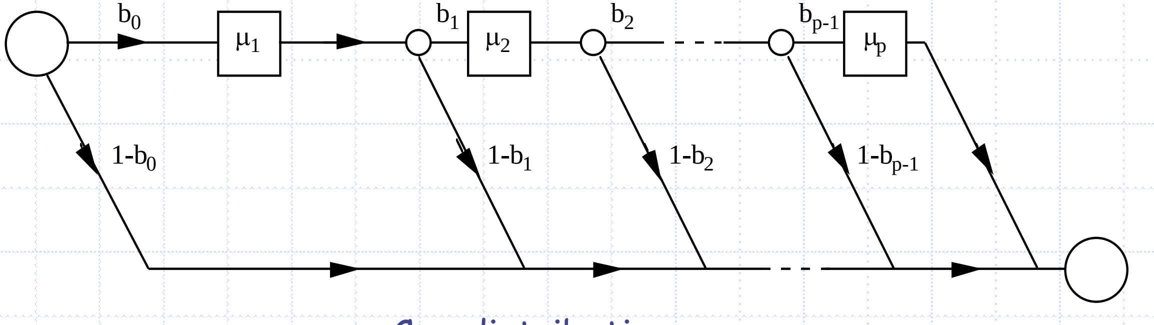
- Basic concepts
- Open queueing networks
- Closed queueing networks
- BCMP networks

Closed queueing networks

- Closed network:
  - Assume single class of customers.
  - Closed: no external arrivals or departures (no γ_i terms). N customers.
  - The M queues/servers must be of one of these kinds:
    - FCFS with exponentially distributed service times
    - LCFS preemptive-resume with Cox service times
    - PS with Cox service times
    - IS with Cox service times



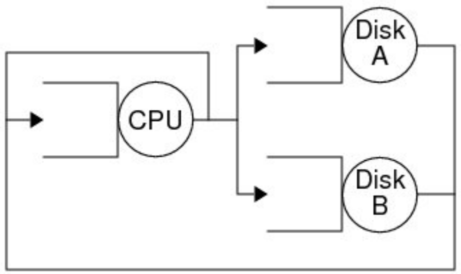
Closed Queueing Network



Cox distribution

Closed queueing networks

- Routing probabilities satisfy $\sum_{j=1}^{M} q_{ij} = 1, \text{ for } 1 \le i \le M$
- State space $S = \{(n_1, \ldots, n_M) \mid n_i \ge 0; \sum_{j=1}^{M} n_j = N\}$ for population N.
  - $|S|$ = number of ways of putting N balls into M bags
    $$\binom{N+M-1}{M-1}$$
  - finiteness of $S \Rightarrow$ steady state always exists



Closed Queueing Network

Closed queueing networks

- Traffic equations:
  - For each queue *i*:
    $$\lambda_i = \sum_{j=1}^{M} \lambda_j q_{ji}, \text{ for } 1 \le i \le M$$
    - homogeneous linear equations with an infinity of solutions which differ by a multiplicative factor (because $|I - Q| = 0$ since rows all sum to zero)
  - Visit ratios: they are the relative throughputs, normalized for a given queue (for instance queue 1).
    $$v = vQ$$
    $$v_1 = 1$$
  - Relative utilization of a server or service demand:
    - Is the visit ratio (or relative throughput) weighted by the mean service time

$$u_i = s_i v_i = v_i / \mu_i$$

Closed queueing networks

- Steady-state queue length probabilities:
  - Gordon-Newell's theorem (1967)

$$\pi_{n_1,n_2,\ldots,n_M} = \frac{1}{G(N)} \prod_{i=1}^{M} \frac{u_i^{n_i}}{\beta_i(n_i)}$$

$$\beta_i(n_i) = \begin{cases} n_i! & si \quad n_i \le c_i \\ c_i!\, c_i^{n_i - c_i} & si \quad n_i \ge c_i \end{cases}$$

c_i = number of servers in station i

- $G(N)$ is a normalization constant

$$G(N) = \sum_{\forall n, \sum_{i=1}^{M} n_i = N} \prod_{i=1}^{M} \frac{u_i^{n_i}}{\beta_i(n_i)}$$

Closed queueing networks

- Other performance indices:

$$P\{n_i \geq n\} = u_i^n \frac{G(N-n)}{G(N)}$$

- In particular, the actual **utilization**:

$$\rho_i = P\{n_i \geq 1\} = u_i \frac{G(N-1)}{G(N)}$$

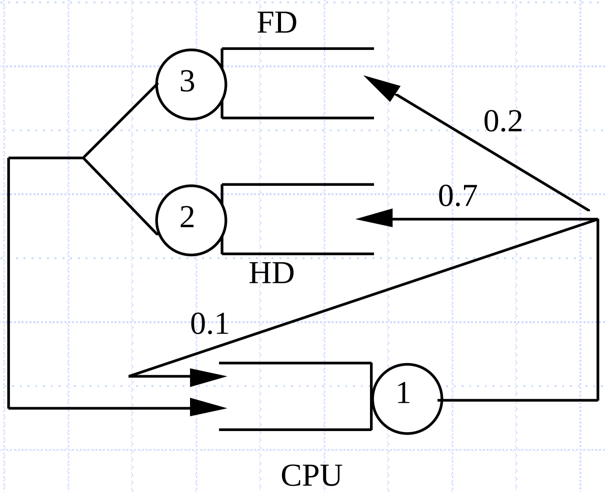
- **Mean queue length**

$$E[n] = \sum_{n=1}^{\infty} nP\{n\} = \sum_{n=1}^{\infty}\sum_{k=n}^{\infty} P\{k\} = \sum_{n=1}^{\infty} P\{k \geq n\}$$

then
$$E[n_i] = \sum_{n=1}^{N} u_i^n \frac{G(N-n)}{G(N)}$$

Closed queueing networks

- Example: A small computer system with
  - Floppy disk: mean access time 280 msec FCFS
  - Hard disk: mean access time 40 msec FCFS
  - CPU: mean computing time 28 msec RR (1msec)

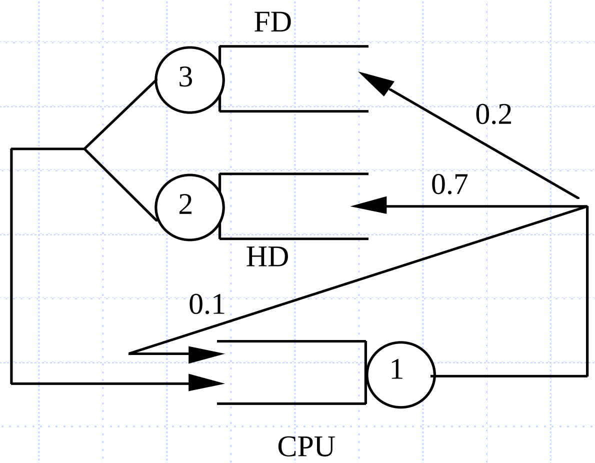


Closed queueing networks

- Relative utilizations:

$$v_2 = 0.7 v_1 \Rightarrow \frac{1}{40} u_2 = 0.7 \frac{1}{28} u_1 \Rightarrow u_2 = u_1$$

$$v_3 = 0.2 v_1 \Rightarrow \frac{1}{280} u_3 = 0.2 \frac{1}{28} u_1 \Rightarrow u_3 = 2 u_1$$



- Normalization constant for different populations (computed using the formula ➔ computat. expensive)

G(0)	G(1)	G(2)	G(3)	G(4)	G(5)	G(6)
1	4	11	26	57	120	247

- Steady-state solution for 6 works

$$\pi_{n_1, n_2, 6-n_1-n_2} = \frac{1}{247} \cdot 1^{n_1} \cdot 1^{n_2} \cdot 2^{6-n_1-n_2} = \frac{2^{6-n_1-n_2}}{247}$$

Closed queueing networks

- Actual utilizations:

$$\rho_i = u_i \frac{G(N-1)}{G(N)} \Rightarrow \rho_1 = \rho_2 = 1 \cdot \frac{120}{247} = 0.4858$$

$$\rho_3 = 2 \cdot \frac{120}{247} = 0.9717$$

- Throughput:

$$\lambda_1 = \rho_1 \cdot \mu_1 = \frac{0.4858}{0.028} = 17.35$$

$$\lambda_2 = \rho_2 \cdot \mu_2 = \frac{0.4858}{0.040} = 12.145$$

$$\lambda_3 = \rho_3 \cdot \mu_3 = \frac{0.9717}{0.280} = 3.47$$

Closed queueing networks

- Average queue lengths:

$$E[n_1] = E[n_2] = \frac{1+4+11+26+57+120}{247} = 0.8866$$

$$E[n_3] = \frac{2^6 \cdot 1 + 2^5 \cdot 4 + 2^4 \cdot 11 + 2^3 \cdot 26 + 2^2 \cdot 57 + 2 \cdot 120}{247} = 4.2267$$

- Average response time of each station (Little) :

$$R_1 = \frac{E[n_1]}{\lambda_1} = 0.0511 \quad \text{seconds}$$

$$R_2 = \frac{E[n_2]}{\lambda_2} = 0.073 \quad \text{seconds}$$

$$R_3 = \frac{E[n_3]}{\lambda_3} = 1.218 \quad \text{seconds}$$

Closed queueing networks

- Main problem:

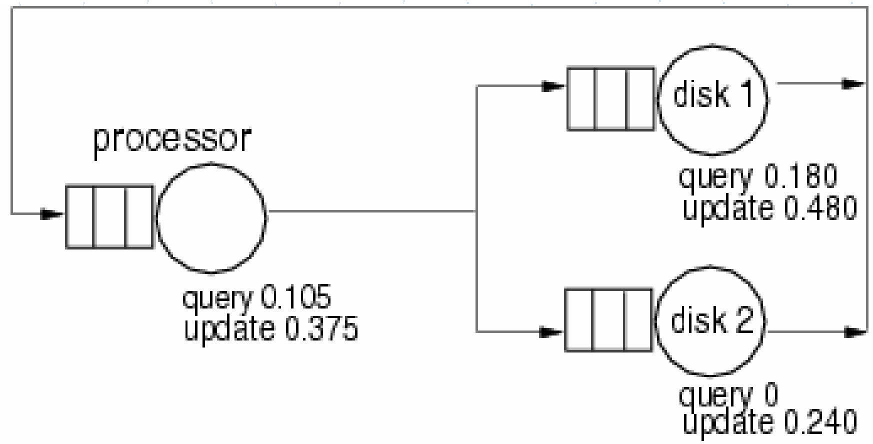
  ➔ computation of the normalization constant $G(N)$

Outline

- Basic concepts
- Open queueing networks
- Closed queueing networks
- BCMP networks

BCMP networks

- Multiclass QN



Server with N transactions in execution

- Questions:
  - What is the predicted increase in the throughput of query transactions if the load of update transactions is moved to off-peak hours?
  - How will the response time change if the total I/O load of query transactions is moved to disk 2?

BCMP networks

- Generalization of product form solution for networks with **different classes of customer** and extension to several service disciplines
- Customers are allowed to change class membership ➔ different chains of classes
  - **Chain**: subset of classes in which a customer can change (i.e., changes from one chain to another are not allowed)
- Classes can be open or closed (mixed QN).
  - In open networks, the time between successive arrivals of a class is exponentially distributed.

BCMP networks

- Service stations can obey any of the four following possibilities:
  - Type 1: FCFS, single server, exponentially distributed with service rate dependent on the total number of customers at the station but the same mean for all classes of customer
  - Type 2: Processor sharing, Cox distribution (may be distinct for each class of customer)
  - Type 3: Infinite-server, Cox distribution (may be distinct for each class of customer)
  - Type 4: LCFS, preemptive-resume, Cox distribution (may be distinct for each class of customer)

BCMP networks

- Baskett-Chandy-Muntz-Palacios Gomez's theorem (1975)
  - Under the previous conditions, the steady-state probability distribution has a product form…

$$P(\bar{N} = \bar{n}) = \frac{1}{G} A(\bar{n}) \prod_{i=1}^{M} p_i(\bar{n}_i),$$

where G is a normalizing constant (it assures that the probabilities sum to one), $A(\bar{n})$ is a function of the external arrival processes only, and the functions, $p_i(\bar{n}_i)$ are the "per-node" steady-state distributions.

BCMP networks

- The important point of this result is that there are explicit expressions for the p_i functions. They are as follows:
  - When node i is of type FCFS, we have in the load-independent case

$$p_i(\bar{n}_i) = n_i! \left(\prod_{r=1}^{R} \frac{1}{n_{i,r}!} V_{i,r}^{n_{i,r}} \right) \left(\frac{1}{\mu_i} \right)^{n_i},$$

  - and in the load-dependent case

$$p_i(\bar{n}_i) = n_i! \left(\prod_{r=1}^{R} \frac{1}{n_{i,r}!} V_{i,r}^{n_{i,r}} \right) \prod_{j=1}^{n_i} \frac{1}{\mu_i(j)}$$

BCMP networks

- When node *i* is of type PS or LCFS-PR, we have in the load-independent case

$$p_i(\bar{n}_i) = n_i! \prod_{r=1}^{R} \frac{1}{n_{i,r}!} \left(\frac{V_{i,r}}{\mu_{i,r}}\right)^{n_{i,r}},$$

- and in the load-dependent case

$$p_i(\bar{n}_i) = n_i! \prod_{r=1}^{R} \frac{1}{n_{i,r}!} V_{i,r}^{n_{i,r}} \prod_{j=1}^{n_i} \frac{1}{\mu_{i,r}(j)}.$$

BCMP networks

- When node *i* is of type IS, we have in the load-independent case

$$p_i(\bar{n}_i) = \prod_{r=1}^{R} \frac{1}{n_{i,r}!} \left(\frac{V_{i,r}}{\mu_{i,r}} \right)^{n_{i,r}}$$

- and in the load-dependent case

$$p_i(\bar{n}_i) = \prod_{r=1}^{R} \frac{1}{n_{i,r}!} V_{i,r}^{n_{i,r}} \prod_{j=1}^{n_i} \frac{1}{\mu_{i,r}(j)}.$$

BCMP networks

- Finally, the term $A(\overline{n})$ is determined by the arrival processes in the following manner.
  - If all chains are closed, then $A(\overline{n}) = 1$.
  - If the arrivals depend on the total system population, then it is equal to
    $$A(\overline{n}) = \prod_{j=0}^{k-1} \lambda(j)$$
    where k is the network population.
  - If the arrivals are per chain, then
    $$A(\overline{n}) = \prod_{c=1}^{N_C} \prod_{j=0}^{k_c-1} \lambda_c(j)$$
    where N_C is the number of routing chains and k_c is the population in routing chain c.
- Notation is hard, but it can be programed…

Performance modelling and evaluation

8. Computational algorithms for closed QN

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Computation of normalization constant: Convolution algorithm
- Mean value analysis algorithm

Reminder

- Gordon-Newell's theorem for closed QN with a single server in each station

$$\pi_{n_1,n_2,\ldots n_M} = \frac{1}{G(N)} \prod_{i=1}^{M} u_i^{n_i}$$

where u_i are relative utilizations, $u_i = s_i v_i = v_i / \mu_i$
with v_i the relative throughput, i.e. solution of

$$v = vQ\,; \quad v_1 = 1$$

thus, u_i are real positive solutions of

$$\mu_j u_j = \sum_{i=1}^{M} \mu_i u_i q_{ij}, \quad 1 \le j \le M$$

$G(N)$ is a normalization constant: $G(N) = \sum_{\substack{n_i \ge 0 \\ \sum_{i=1}^{M} n_i = N}} \prod_{i=1}^{M} u_i^{n_i}$

Convolution algorithm

- Buzen, 1973: single class of customers
- Bruell and Balbo, 1980: multiclass QN
- Define $g(n, M) = G(n), n = 0,1,\ldots, N$

  Then (a) $$g(n,m) = \sum_{\substack{n_i \geq 0 \\ \sum_{i=1}^{m} n_i = n \\ n_m = 0}} \prod_{i=1}^{m} u_i^{n_i} + \sum_{\substack{n_i \geq 0 \\ \sum_{i=1}^{m} n_i = n \\ n_m > 0}} \prod_{i=1}^{m} u_i^{n_i} =$$

  $$= g(n, m-1) + u_m g(n-1, m), \text{ if } m > 1, n > 0$$

  (b) $$g(n,1) = u_1^n, \ n = 0,1,\ldots, N$$

  (c) $$g(0,m) = 1, \ m = 1,2,\ldots, M$$

Convolution algorithm

- Algorithm for $g(n,m)$ = iterative relationship (a) together with initial conditions (b) and (c)

$n \backslash m$	u_1	u_2	$\cdots$	u_m	$\cdots$	u_M
0	1	1	$\cdots$	1	$\cdots$	1
1	u_1					
2	u_1^2					
3	u_1^3					
$\vdots$	$\vdots$			$g(n-1,m)$		
$\vdots$	$\vdots$			$\downarrow \cdot u_m$		
n	u_1^n		$g(n,m-1) \rightarrow$	$g(n,m)$		
$\vdots$	$\vdots$					
$\vdots$	$\vdots$					
N	u_1^N					$g(N,M)=G(N)$

The entire rightmost column is of interest since $g(n,M)=G(n)$, $n=0,1\ldots,N$, thus the values $G(n)$ are by-products from the computation of $G(N)$.

Observe: the leftmost column can be computed in the same way if it is assumed that there is a column of zeros immediately to the left.

Convolution algorithm

- Space complexity: $O(N)$
  - Only one column at a time!
- Algorithm:

$\{$**Assumed that** $u_m, m=1,\ldots,M$ **are known**$\}$
$c_0 := 1;$
for $n:=1$ **to** N **do** $c_n:=0;$
for $m:=1$ **to** M **do**
  for $n:=1$ **to** N **do**
    $c_n := c_n + u_m * c_{n-1};$

- Time complexity: $O(N \cdot M)$
  - $N \cdot M$ additions and multiplications for the computation of $G(1), \ldots, G(N)$

Convolution algorithm

- Numerical considerations
  - $G(N)$ depends on the relative utilizations u_i, $i=1,\dots,M \Rightarrow$ choosing u_i's much bigger or smaller than 1 will surely lead to problems
  - Table for the example presented in previous lecture

$u_1=u_2=1;\ u_3=2$

	u_1	u_2	u_3
0	1	1	1
1	1	2	4
2	1	3	11
3	1	4	26
4	1	5	57
5	1	6	120
6	1	7	247

$\Rightarrow G(n),\ n=0,\dots,6$

Outline

- Computation of normalization constant: Convolution algorithm
- Mean value analysis algorithm

Mean value analysis algorithm

- Objective: to avoid the computation of G(N)
  - We saw that the mean queue length in station *i* is

$$E[n_i \mid N] = \sum_{n=1}^{N} u_i^n \frac{G(N-n)}{G(N)}$$

  - Then:

$$E[n_i \mid N] = \sum_{n=1}^{N} u_i^n \frac{G(N-n)}{G(N)} = u_i \frac{G(N-1)}{G(N)} + \sum_{n=2}^{N} u_i^n \frac{G(N-n)}{G(N)} =$$

$$= \rho_i(N) + \sum_{n=1}^{N-1} u_i^{n+1} \frac{G(N-n-1)}{G(N)} = \rho_i(N) + u_i \frac{G(N-1)}{G(N)} \sum_{n=1}^{N-1} u_i^n \frac{G(N-n-1)}{G(N-1)} =$$

$$= \rho_i(N) + \rho_i(N) \sum_{n=1}^{N-1} u_i^n \frac{G(N-n-1)}{G(N-1)}$$

$$\Rightarrow E[n_i \mid N] = \rho_i(N)\left(1 + E[n_i \mid N-1]\right)$$

  - And the average response time at station (Little's law):

$$R_i(N) = \frac{E[n_i \mid N]}{\lambda_i(N)} \quad \Rightarrow \quad \boxed{R_i(N) = \frac{1}{\mu_i}\left(1 + E[n_i \mid N-1]\right)}$$

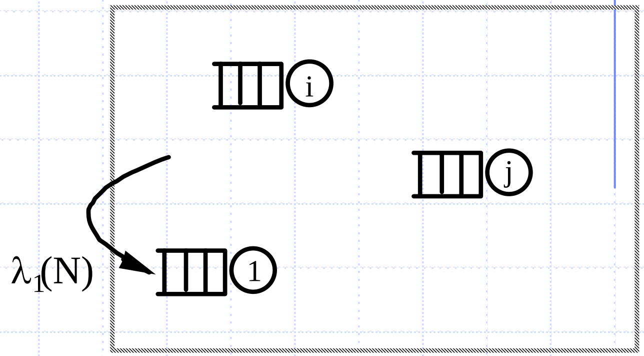
"Mean Value Theorem"

Mean value analysis algorithm

- Mean value theorem $$R_i(N) = \frac{1}{\mu_i}\left(1 + E[n_i \mid N-1]\right)$$
  - Interpretation: The queue length distribution seen by an arriving customer is the steady-state distribution with himself removed from the network.
    Thus the average number of customers found by an arriving customer is simply $E[n_i \mid N\text{-}1]$ and the average delay is $(1/\mu_i)(1+ E[n_i \mid N\text{-}1])$

Mean value analysis algorithm

- Computation of the throughput
  - Apply Little's law to all the net:
    - Average number of customers = N
    - Average delay = $\sum_{i=1}^{M} v_i R_i(N)$



normalized for station 1

$$\Rightarrow \quad \lambda_1(N) = \frac{N}{\sum_{i=1}^{M} v_i R_i(N)}$$

$$\lambda_i(N) = v_i \lambda_1(N) = v_i \frac{N}{\sum_{i=1}^{M} v_i R_i(N)}, \quad i = 2,\ldots, M$$

- Computation of actual utilizations:

$$\rho_i(N) = \frac{\lambda_i(N)}{\mu_i}, \quad i = 1,\ldots, M$$

Mean value analysis algorithm

- And remember that the mean queue length (Little's law):

$$E[n_i \mid N] = \lambda_i(N)R_i(N), \; i = 1,\ldots,M$$

$$\text{and} \quad E[n_i \mid 0] = 0, \; i = 1,\ldots,M$$

Mean value analysis algorithm

- Putting all together: MVA algorithm
  1. Compute visit ratios v: $v = vQ$; $v_1 = 1$
  2. $E[n_i \mid 0]:=0$, $i=,\dots,M$
  3. For n:=1 to N do

$$R_i(n) = \frac{1}{\mu_i}\left(1 + E[n_i \mid n-1]\right),\ i = 1,\dots,\ M$$

$$\lambda_1(n) = \frac{n}{\sum_{i=1}^{M} v_i R_i(n)}$$

$$\lambda_i(n) = v_i \lambda_1(n),\ i = 2,\dots,\ M$$

$$\rho_i(n) = \frac{\lambda_i(n)}{\mu_i},\quad i = 1,\dots,\ M$$

$$E[n_i \mid n] = \lambda_i(n) R_i(n),\ i = 1,\dots,\ M$$

Mean value analysis algorithm

- Complexity of MVA algorithm
  - Equal to the Convolution algorithm for the computation of $G(N)$.
  - Requires less storage than Convolution algorithm since no memory is allocated for $G(n)$, $n=1,\dots,N$, constants.

- Advantages of MVA algorithm:
  - It is more robust as compared to Convolution algorithm since it never computes G (avoids overflow/underflow problem).
  - It computes directly all the interesting average performance measures for each value of the population in the network.

Mean value analysis algorithm

- Disadvantage of MVA algorithm:
  - It is a method for computing the average values, so it is impossible to construct the complete description of the steady-state probability distribution function

    (therefore it is impossible to get other measures on the system such as "what is the probability of queue 3 having two or more customers?")

Performance modelling and evaluation

9. Performance bounds for QN

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Introduction to the use of bounds
- Asymptotic bound analysis
- Balanced job bounds

Introduction to the use of bounds

- Preliminary design phases of a system:
  - many of the system parameters are not known accurately
  - the number of alternative designs that need to be considered may be very large

$\Rightarrow$ exact solution can be very expensive and not justifiable

Introduction to the use of bounds

- Performance bounds:
  - require much less computation as compared to the exact
  - allow to quickly evaluate several alternatives and reject those that are clearly bad
  - there exist techniques that can provide increasingly tighter bounds at the expense of increasing computation
  - most of the techniques are valid only for product form queueing networks

Introduction to the use of bounds

- Here we see...
  - the less accurate and quickest technique, valid for any network: asymptotic bound analysis, and
  - the first known technique for product form networks: balanced job bound analysis

  ... for the case of networks with:
    - single class of customers, and
    - single-server (fixed rate) nodes and possibly delay nodes (infinite-server)
- They allow to obtain bounds for the throughput of the network.

Introduction to the use of bounds

- Terminology and notation:
  - If the network contains several delay nodes, we can merge all of them into one delay node

    → it suffices to consider only one delay node
  - We index the delay node as 0 and denote the relative utilization of this node by Z.
  - The single-server stations will be indexed as $1,\ldots,M$, with u_i being the relative utilization of node i.

Introduction to the use of bounds

- Without loss of generality:

$u_1 \leq \ldots \leq u_M$ (node M is the bottleneck)

Hence $Q_1(N) \leq \ldots \leq Q_M(N)$ for any population N

because $Q_i(N) = E[n_i \mid N] = \sum_{k=1}^{N} u_i^k \dfrac{G(N-k)}{G(N)}$

Introduction to the use of bounds

- Visit ratio at station 1 is 1, then the throughput of the network will be computed with respect to this station.
- Additinal notation:
  - Total utilization (only for single server stations): $$L = \sum_{i=1}^{M} u_i$$
  - Average relative utilization (only for single server stations): $$u_a = \frac{L}{M}$$
  - (Relative) residence time at node *i* (visit ratio x average delay): $$T_i(N) = v_i R_i(N)$$

Outline

- Introduction to the use of bounds
- Asymptotic bound analysis
- Balanced job bounds

Asymptotic bound analysis

- Kleinrock, 1976
- It does not require product form property to hold
- The bounds are obtained by considering two extreme situations:
  - no queueing takes place at any node, and
  - all nodes are loaded as heavilly as the bottleneck node

Asymptotic bound analysis

- Upper bound on throughput (of station 1):
  - Without any queueing, the average delay time is $L + Z$ therefore, by Little's formula,

$$\lambda(N) \leq \frac{N}{L+Z}$$

  - However, this bound may not satisfy the restriction that the utilization of any node cannot exceed 1
    - → Additional constraint (M is the bottleneck): $\lambda(N) \leq \frac{1}{u_M}$
  - Then the upper bound on throughput is:

$$\boxed{\lambda(N) \leq \min\left\{\frac{N}{L+Z}, \frac{1}{u_M}\right\}}$$

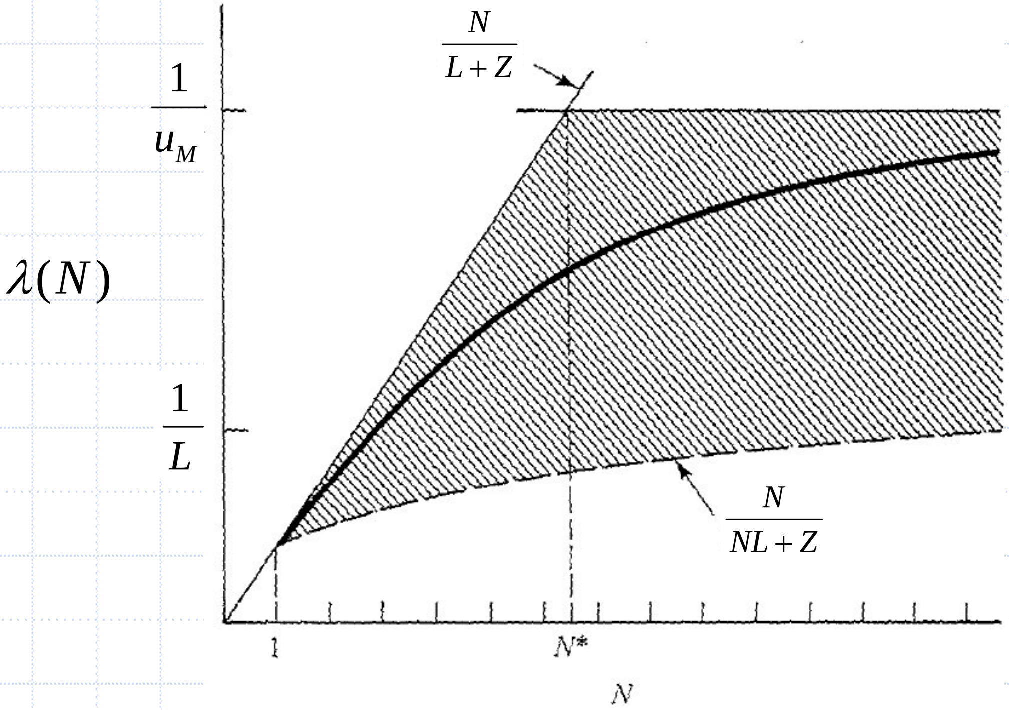
Asymptotic bound analysis

- Lower bound on throughput:
  - the throughput will be minimum if every customer had to wait behind the remaining $N-1$ other customers at every single-server station in the network…

$$\boxed{\lambda(N) \geq \frac{N}{NL+Z}}$$

- Notice that:
  - we did not use the product form assumption in deriving these (upper and lower) bounds,
  - the bounds could however be rather loose,
  - subsequent bounds are tighter but do require the product form assumption.

Asymptotic bound analysis



$$\frac{N}{NL + Z} \leq \lambda(N) \leq \min\left\{\frac{N}{L + Z}, \frac{1}{u_M}\right\}$$

Outline

- Introduction to the use of bounds
- Asymptotic bound analysis
- Balanced job bounds

Balanced job bounds

- Zahorjan et al., 1982
- Mean Value Theorem is used, thus the analysis is valid only for product form queueing networks.
- Remember: $$R_i(N) = \frac{1}{\mu_i}\left(1 + E[n_i \mid N-1]\right)$$
- Then, the relative residence time at node *i* is:

$$T_i(N) = v_i R_i(N) = u_i\left(1 + E[n_i \mid N-1]\right) = u_i\left(1 + Q_i(N-1)\right)$$

Balanced job bounds

- Summing over all nodes we get the "average cycle time" of the network, i.e., the average delay in all stations (assuming that there are no delay nodes):

$$CT(N) = L + \sum_{i=1}^{M} u_i \left(1 + Q_i(N-1)\right)$$

- Since $u_M \geq u_i$ for all i, we have

$$CT(N) \leq L + u_M (N-1)$$

Balanced job bounds

- Lower bound for the throughput: Applying Little's law to the whole net ($L=\lambda W$).

  $N = \lambda(N)\, CT(N) \Rightarrow \lambda(N) = N / CT(N) \Rightarrow$

  $$\boxed{\lambda(N) \geq \frac{N}{L + (N-1)u_M}}$$

  Notice that this bound essentially assumes that all nodes are loaded as heavily as the bottleneck node.

Balanced job bounds

- Upper bound for the throughput.
  - Lemma: if x_i, y_i, $i = 1,\dots,n$, are such that $x_1 \le \dots \le x_n$ and $y_1 \le \dots \le y_n$, then

$$\sum_{i=1}^{n} x_i y_i \ge \frac{1}{n} \sum_{i=1}^{n} x_i \sum_{j=1}^{n} y_j$$

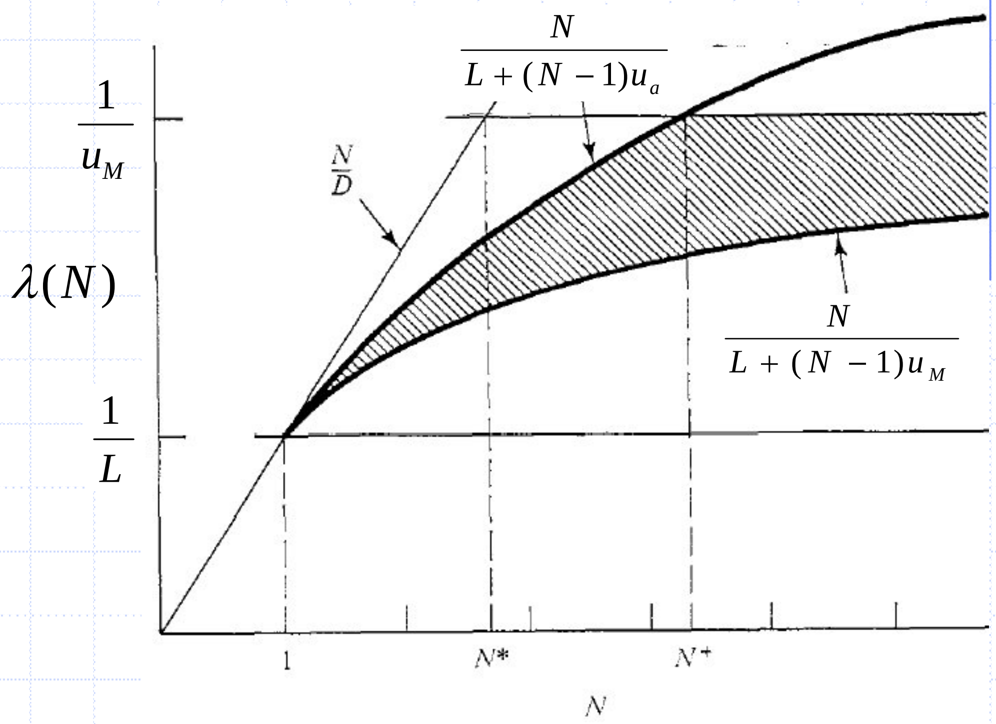
  - Setting $x_i = u_i$ and $y_i = Q_i(N)$: $$\sum_{i=1}^{M} u_i Q_i(N) \ge u_a \sum_{i=1}^{M} Q_i(N)$$
  - Hence, since $u_a = L/M$ and from $CT(N) = L + \sum_{i=1}^{M} u_i \left(1 + Q_i(N-1)\right)$ we get $$CT(N) \ge L + (N-1)u_a$$
  - Then, the throughput upper bound is (Little's law):

$$\boxed{\lambda(N) \le \frac{N}{L + (N-1)u_a}}$$

Balanced job bounds

$$\frac{N}{\left(\frac{L}{u_M}+N-1\right)u_M} \le \lambda(N) \le$$

$$\le \frac{N}{(M+N-1)u_a}$$



Since $R_i(N) = u_i\left(1 + Q_i(N-1)\right)$, the following interpretation can be made:

- The upper bound on throughput is equal to the throughput of a system composed of M centers each with relative utilization u_a.
- The lower bound on throughput is equal to the throughput of a system composed of "L/u_M centers" (which is not integer, in general) each with relative utilization u_M.

Performance modelling and evaluation

10. Petri nets

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Justification
- Definitions
- Functional properties and analysis

Justification

- In real computer systems **cooperation** and **competence** relationships are usual.
- **Synchronization primitives** are necessary for expressing these relationships.
- Product form queueing networks **do not** allow the explicit representation of a general synchronization primitive.

Justification

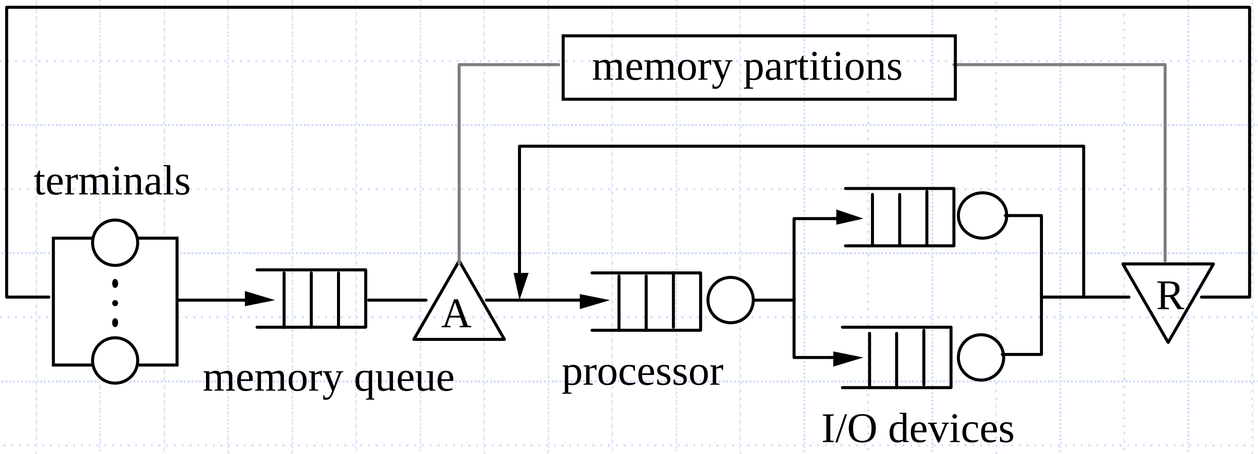
- Example: **blocking phenomena** in computer systems, arise because a job requires more than one resource before it can be processed
  1. Holding a channel and a disk drive before data transfer can occur.
  2. Obtaining a memory partition before job processing can occur.
  3. Obtaining a database lock before the data item can be read from the disk.

Justification

- Possible solutions to the problem of lack of expressivity of QN:
  - Ad hoc extensions of QN
  - Define a new formalism with synchronization primitive (like Petri nets or process algebra)

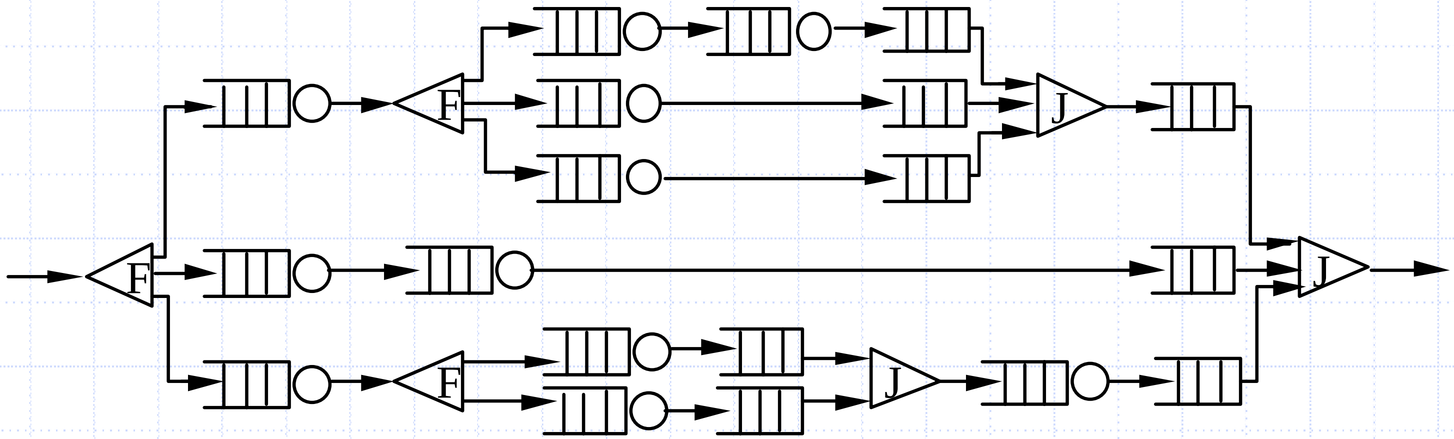
Justification

- Ad hoc extensions of QN
  - Passive resources
    - Example: a memory limited system



Justification

- Ad hoc extensions of QN (cont.)
  - Forks and joins
    - Example: multitasking feature of operating systems



Justification

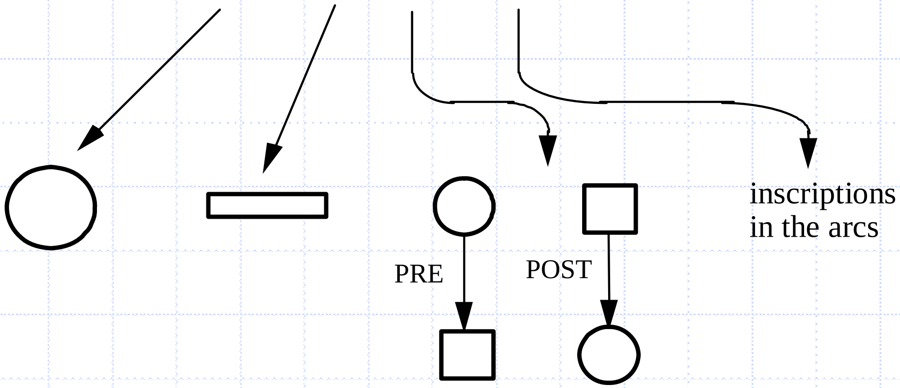
- In general, product form solution does not hold for the previous ad hoc extensions.
- A formal and unified definition is needed as well as computation techniques.

$\Rightarrow$Define a new formalism with synchronization primitive
(like Petri nets or process algebra)

Definitions

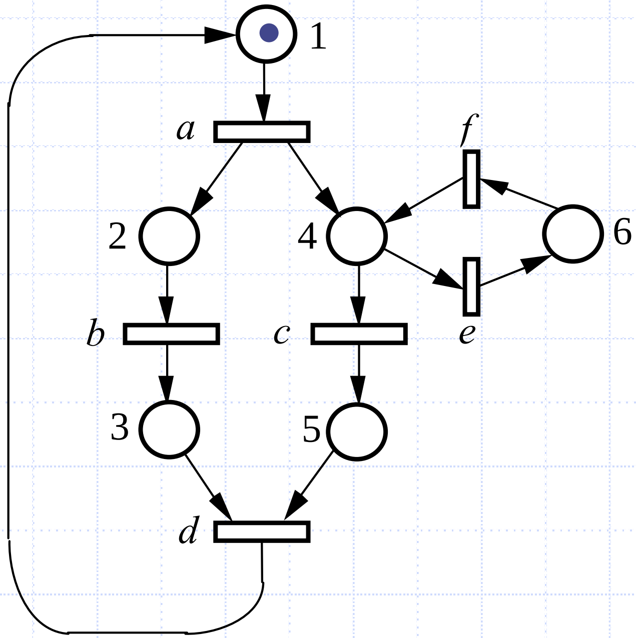
- Autonomous Petri nets (place/transition nets or P/T nets)
  - Petri Nets is a bipartite valued graph
    - Places: states/data (*P*)
    - Transitions: actions/algorithms (*T*)
    - Arcs: connecting places and transitions (*F*)
    - Weights: labeling the arcs (*W*)

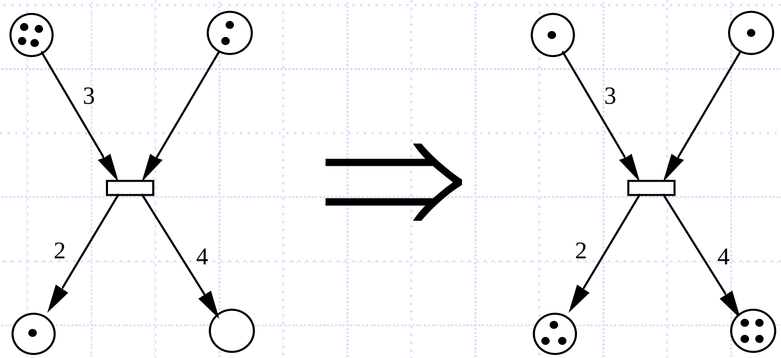
$$N = \langle P, T, F, W \rangle$$



Definitions

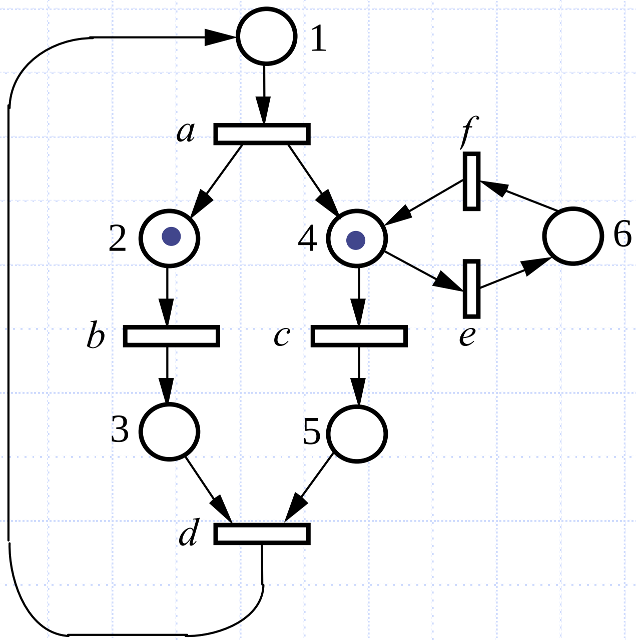
- Net ➔ Static part
  - Places : State variables (names)
  - Transitions: Changes in the state (conditions)
- Marking ➔ Dynamic part
  - Marking : State variables (values)
- Event/Firing
  - **Enabling**: the pre-condition is verified
  - **Firing**: change in the marking
  - the pre-condition "consumes" tokens
  - the post-condition "produces" tokens





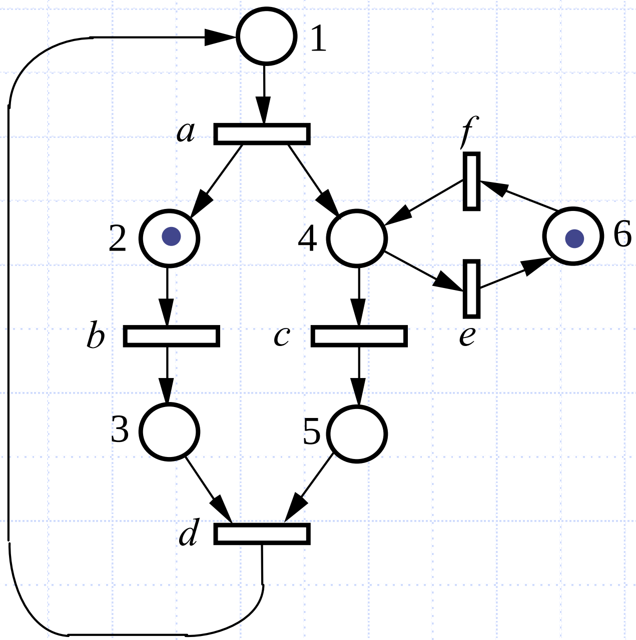
Definitions

- Net ➔ Static part
  - Places : State variables (names)
  - Transitions: Changes in the state (conditions)
- Marking ➔ Dynamic part
  - Marking : State variables (values)
- Event/Firing
  - **Enabling**: the pre-condition is verified
  - **Firing**: change in the marking
  - the pre-condition "consumes" tokens
  - the post-condition "produces" tokens



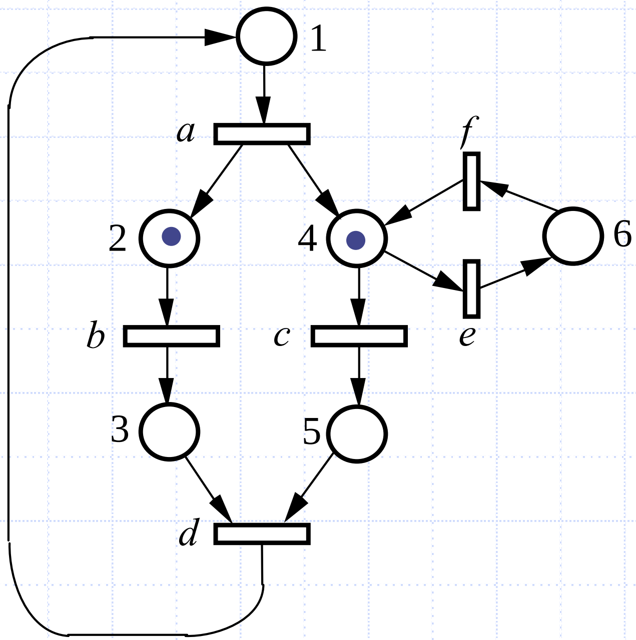
Definitions

- Net ➔ Static part
  - Places : State variables (names)
  - Transitions: Changes in the state (conditions)
- Marking ➔ Dynamic part
  - Marking : State variables (values)
- Event/Firing
  - **Enabling**: the pre-condition is verified
  - **Firing**: change in the marking
  - the pre-condition "consumes" tokens
  - the post-condition "produces" tokens



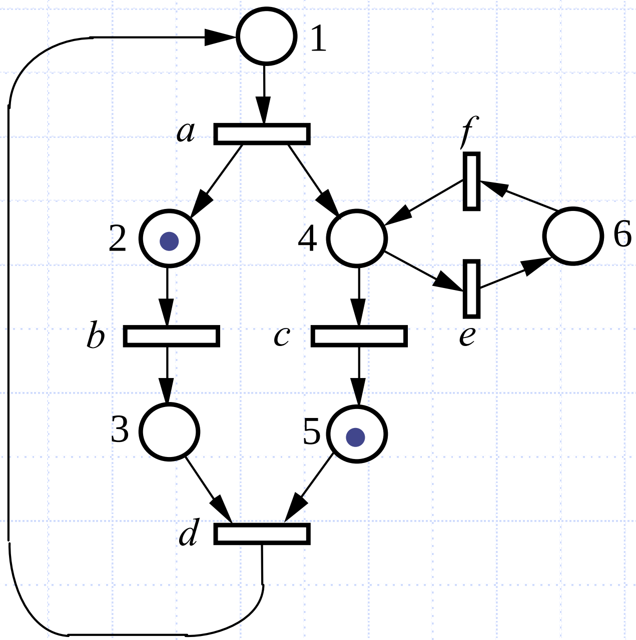
Definitions

- Net ➔ Static part
  - Places : State variables (names)
  - Transitions: Changes in the state (conditions)
- Marking ➔ Dynamic part
  - Marking : State variables (values)
- Event/Firing
  - **Enabling**: the pre-condition is verified
  - **Firing**: change in the marking
  - the pre-condition "consumes" tokens
  - the post-condition "produces" tokens



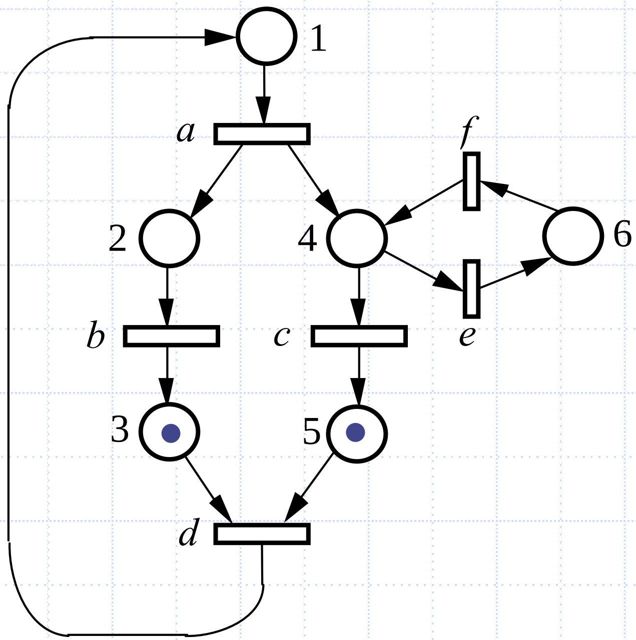
Definitions

- Net ➔ Static part
  - Places : State variables (names)
  - Transitions: Changes in the state (conditions)
- Marking ➔ Dynamic part
  - Marking : State variables (values)
- Event/Firing
  - **Enabling**: the pre-condition is verified
  - **Firing**: change in the marking
  - the pre-condition "consumes" tokens
  - the post-condition "produces" tokens



Definitions

- Net ➔ Static part
  - Places : State variables (names)
  - Transitions: Changes in the state (conditions)
- Marking ➔ Dynamic part
  - Marking : State variables (values)
- Event/Firing
  - **Enabling**: the pre-condition is verified
  - **Firing**: change in the marking
  - the pre-condition "consumes" tokens
  - the post-condition "produces" tokens



Definitions

- PN and its algebraic representation based on state equation
  - Linear representation of PNs, the structure:

$$\mathcal{N} = < P, T, \mathbf{Pre}, \mathbf{Post} >$$

  - Pre-incidence matrix

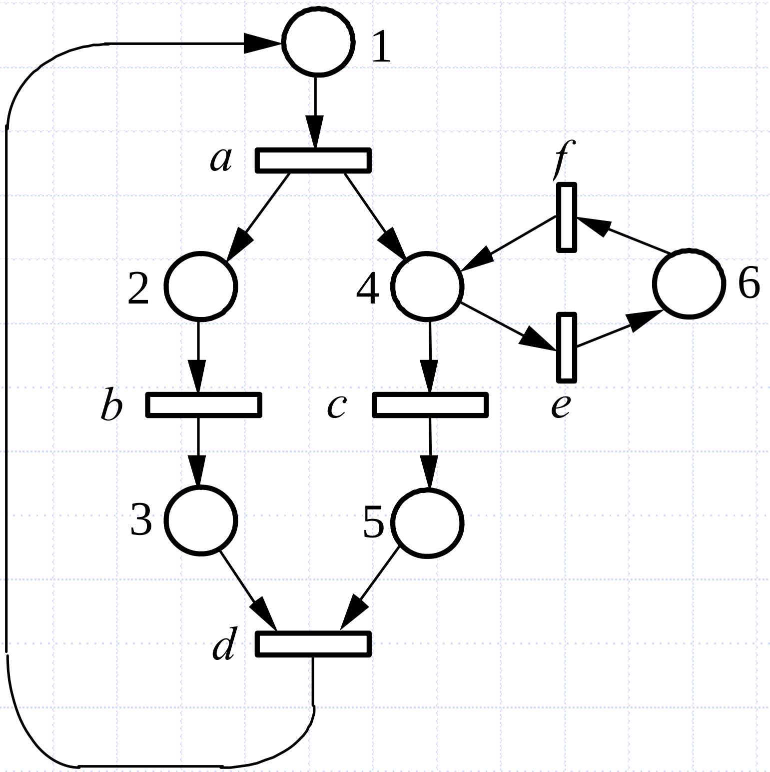
$$\mathbf{Pre}(p,t): P x T \rightarrow \mathbb{N}^{+}$$

  - Post-incidence matrix

$$\mathbf{Post}(p,t): P x T \rightarrow \mathbb{N}^{+}$$

  - Incidence matrix, *C* = Post - Pre
    (marked) Petri Net is finally defined by: $\Sigma = \langle \mathcal{N}, m_0 \rangle$

Definitions



$$
\textbf{Pre} = \begin{array}{c} \\ p1 \\ p2 \\ p3 \\ p4 \\ p5 \\ p6 \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}
$$

$$
\textbf{Post} = \begin{array}{c} \\ p1 \\ p2 \\ p3 \\ p4 \\ p5 \\ p6 \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}
$$

Incidence matrix C (= Post – Pre) cannot "see" self loops

Definitions

- State equation definition

$$m(k)\,[t > m(k+1) \Leftrightarrow \begin{array}{l} m(k+1) = m(k) + \mathbf{C}(t) = \\ = m(k) + \mathbf{Post}(t) + \mathbf{Pre}(t) \geq 0 \end{array}$$

Integrating in one execution (sequence of firing)

$$m_0\,[\sigma > m(k) \Rightarrow \boxed{m(k) = m_0 + \mathbf{C} \cdot \boldsymbol{\sigma}}$$

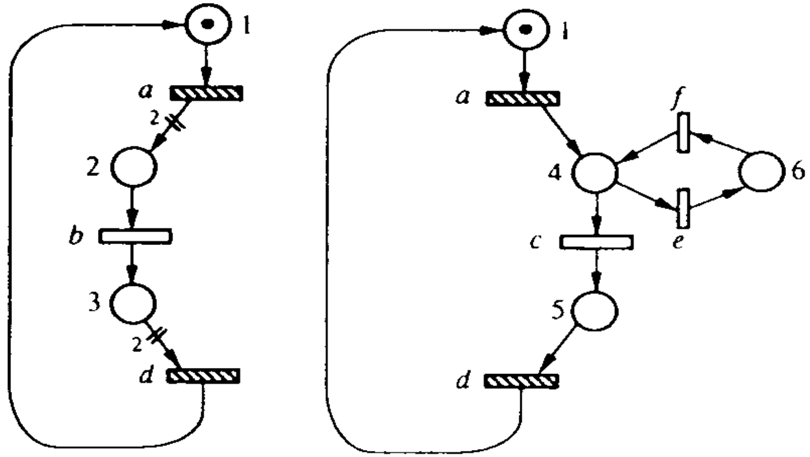
where σ (bold) is the firing counting vector of σ

Very important: unfortunately...

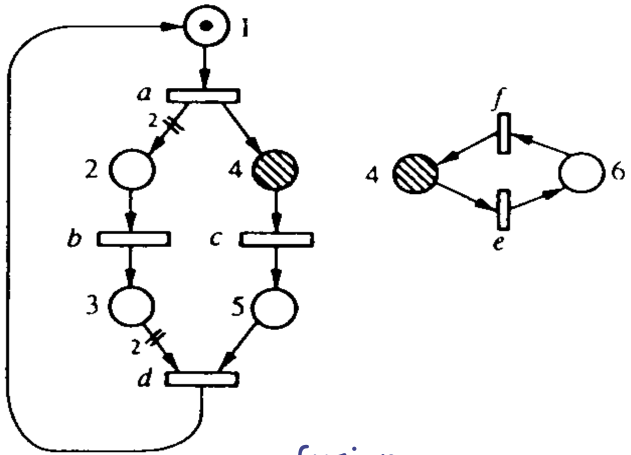
$$m(k) = m_0 + \mathbf{C} \cdot \boldsymbol{\sigma} \geq 0,\ \boldsymbol{\sigma} \geq 0 \not\Rightarrow m_0\,[\sigma > m(k)$$

Definitions

- Design methodologies:
  1. Parallel composition by…



synchronization

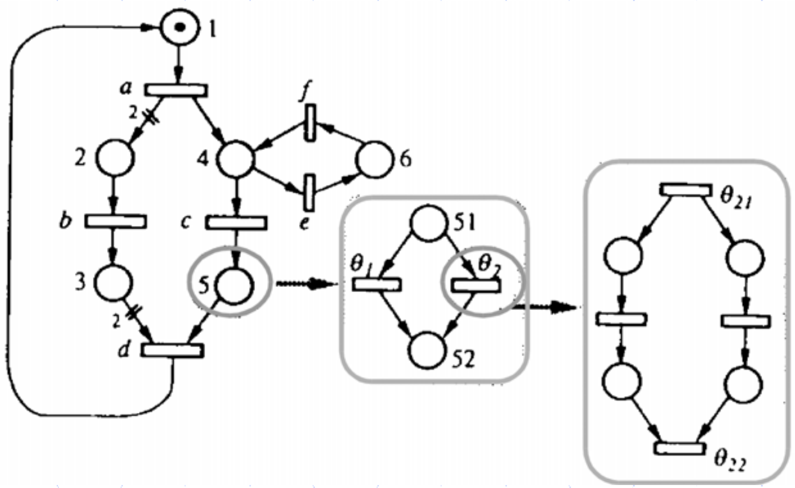


fusion

+ bottom-up methodology

Definitions

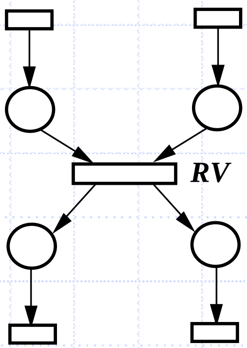
- Design methodologies (cont):
  2. Sequential composition by refinement



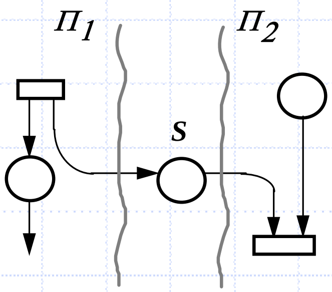
+ top-down methodology

Definitions

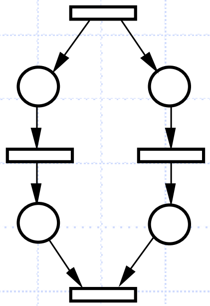
- Design methodologies (cont): typical synchronization schemes



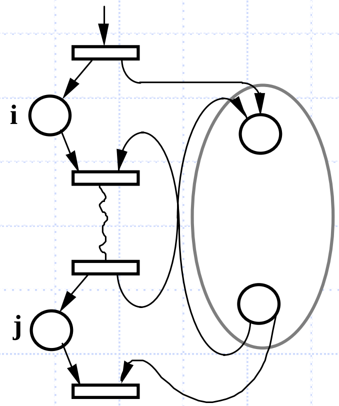
1. Rendezvous, RV



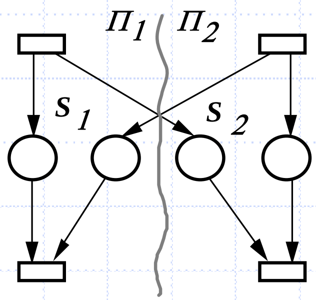
2. Semáforo, S



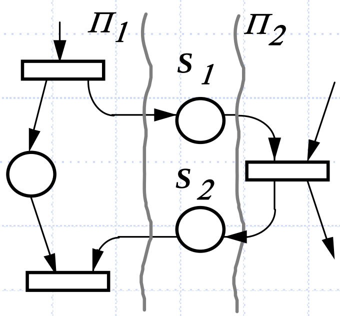
5. Fork-Joint



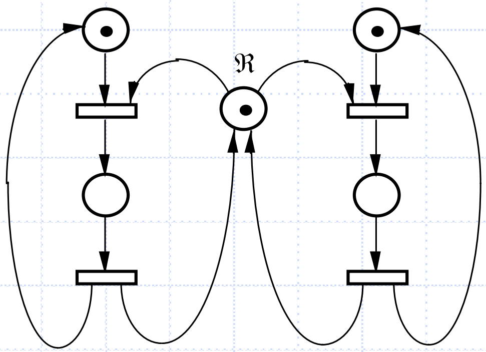
6. Sub programa (p_i ,p_j están en mutex)



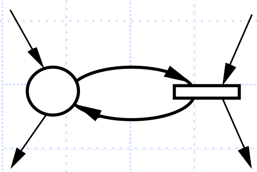
3. RV/Semáforo simétrico



4. RV/Semáforo asimétrico (master/slave)



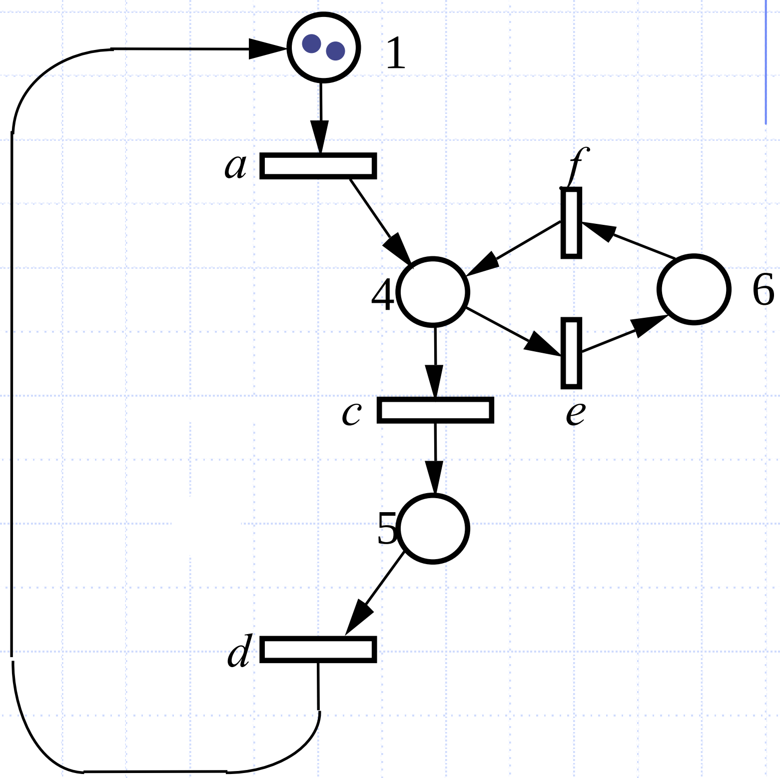
7. Recurso compartido (ℜ)



8. Guarda (condición de lectura)

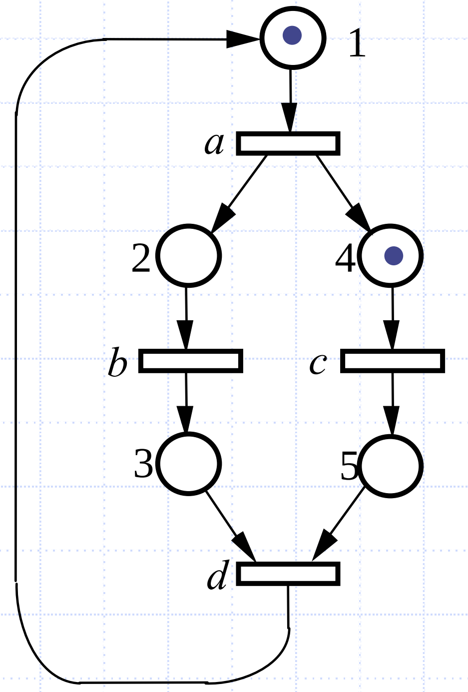
Definitions

- PN syntactic subclasses
  - **State machines**
    - Subclass of ordinary PN (arc weights = 1)
    - Neither synchronizations nor structural parallelism allowed
    - Model systems with a finite number of states
    - Their analysis and synthesis theory is well-known



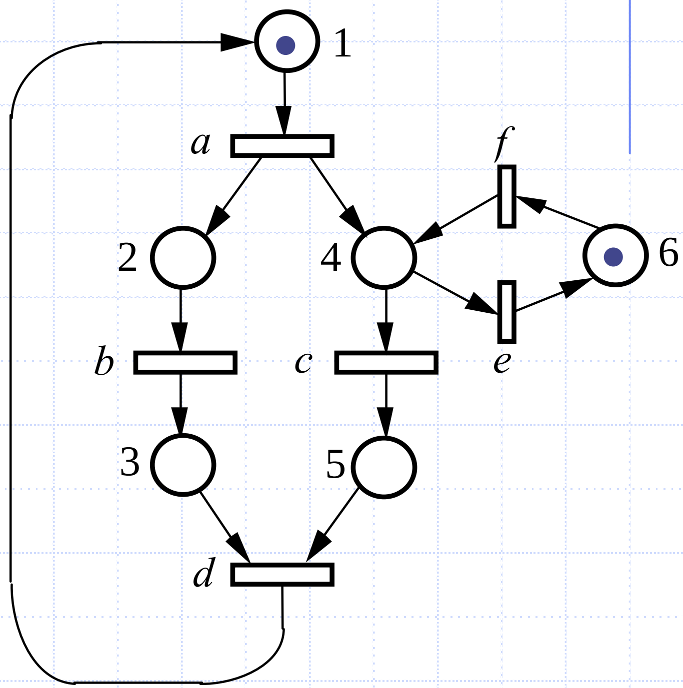
Definitions

- PN syntactic subclasses (cont.)
  - **Marked Graphs**
    - Subclass of ordinary PN (arc weights = 1)
    - Allow synchronizations and parallelism but not allow decisions
    - No conflicts present
    - Allow the modeling of infinite number of states
    - Their analysis and synthesis theory is well-known



Definitions

- PN syntactic subclasses (cont.)
  - **Free-Choice nets**
    - Subclass of ordinary PN (arc weights = 1)
    - Allow synchronizations, parallelism and choices
    - Choices and synchronizations cannot be present in the same transition
    - Their analysis and synthesis theory is well-known



  - There are other syntactic subclasses…

Functional properties and analysis

- Functional basic properties
  - **Boundedness**: finiteness of the state space, i.e. the marking of all places is bounded

$$\forall p \in P \quad \exists k \in N \ \text{ such that } \ \mathbf{m}(p) \le k$$

  - **Safeness** = 1-boundedness (binary marking)
  - **Mutual Exclusion**: two or more places cannot be marked simultaneously (problem of shared resources)
  - **Deadlock**: situation where there is no transition enabled
  - **Liveness**: infinite potential activity of all transitions

$$\forall t \in T, \forall \mathbf{m} \text{ reachable}, \ \exists \mathbf{m}', \mathbf{m}\,[\sigma > \mathbf{m}' \ \text{ such that } \ \mathbf{m}'\,[t >$$

  - **Home state**: a marking that can be recovered from every reachable marking
  - **Reversibility**: recovering of the initial marking

$$\forall \mathbf{m} \text{ reachable}, \ \exists \sigma \ \text{ such that } \mathbf{m}\,[\sigma > \mathbf{m}_0$$

Functional properties and analysis

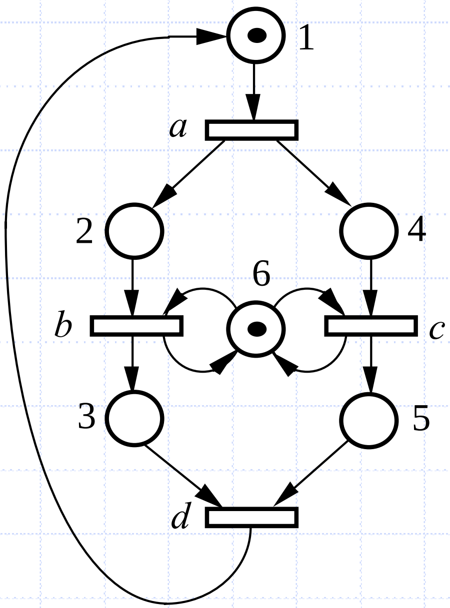
- Structural basic properties:
  - N is **structurally bounded** if for all m_0, $\langle N, m_0 \rangle$ is bounded
  - N is **structurally live** if there exists a m_0 for which $\langle N, m_0 \rangle$ is live

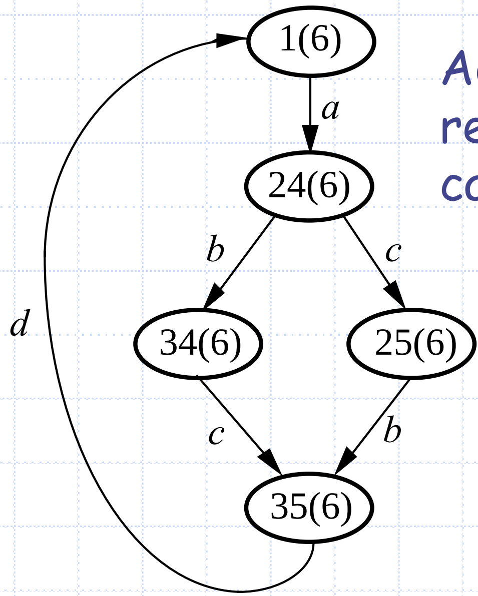
Functional properties and analysis

- Analysis techniques (for the computation of functional properties)
  - Enumerative: based on reachability graph
  - Structural: based on the structure of the model, considering m_0 as a parameter
  - Reduction/transformation: rules that preserve a given property and simplify the model

Functional properties and analysis

- Enumerative analysis: exhaustive sequential enumeration of reachable states
  - Problem 1: state explosion problem
  - Problem 2: lost of information about concurrent behaviour



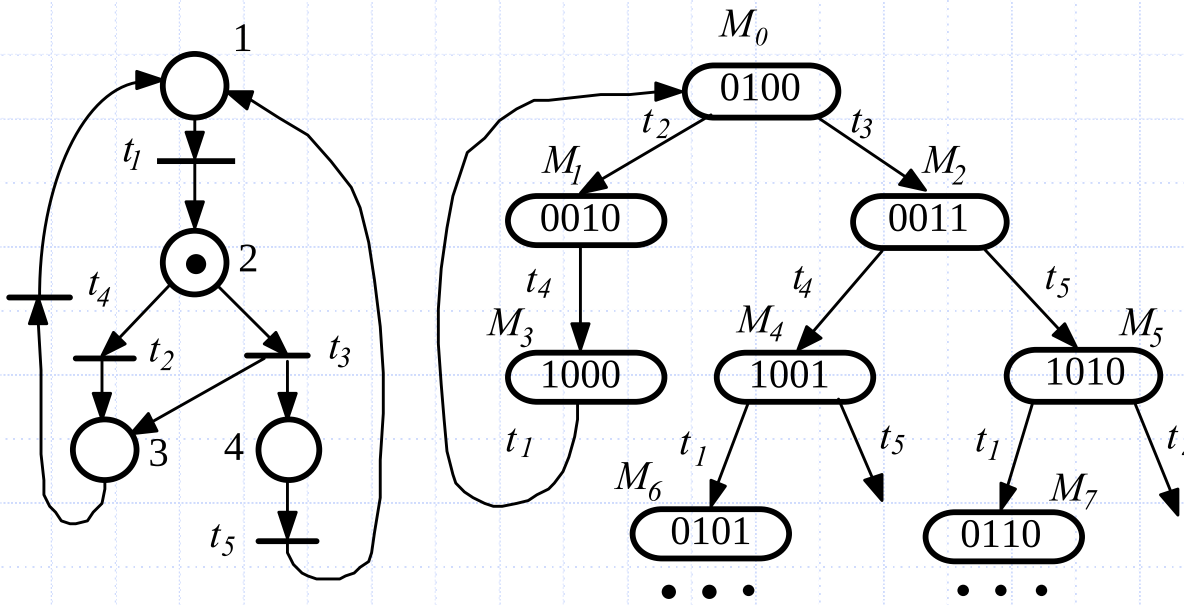


Adding place 6 does not modify reachability graph but b and c cannot fire simultaneously.

reachability graph

Functional properties and analysis

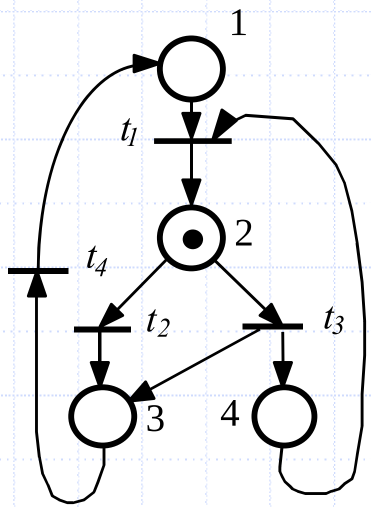
- Enumerative analysis (cont.):
  - Bounded system ⇔ finite reachability graph

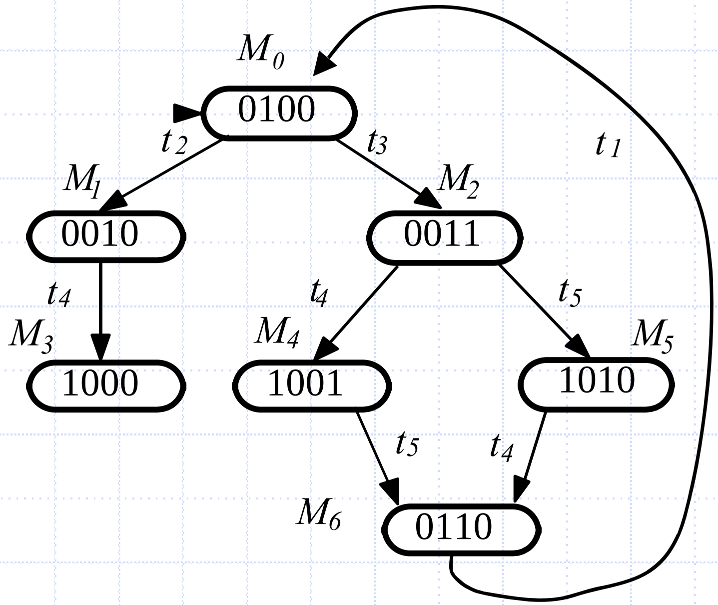


unbounded system

Functional properties and analysis

- Enumerative analysis (cont.):
  - Deadlock exists ⇔ There exists a terminal node in the RG

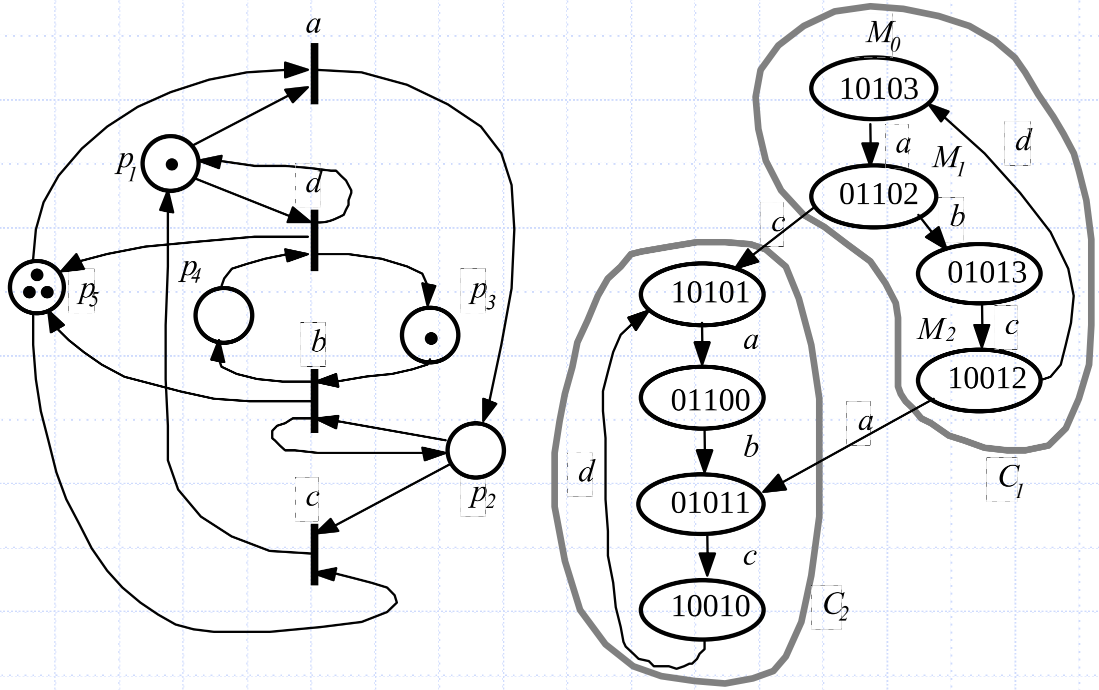




M_3 is a deadlock

Functional properties and analysis

- Enumerative analysis (cont.):
  - Live net ⇔ in all the strongly connected components of the RG all transitions can be fired
  - Reversible net ⇔ there is only one strongly connected component in the RG



live and non-reversible system

Functional properties and analysis

- Structural analysis:
  - Based either on convex geometry (linear algebra and linear programming), or
  - Based on graph theory

  → We concentrate on first approach.

- Definitions:

  P-semiflow: $y \geq 0, \; y^T.C = 0$

  T-semiflow: $x \geq 0, \; C.x = 0$

Functional properties and analysis

- Properties:
  1. If y is a P-semiflow, then the next token conservation law holds (or P-invariant):

$$\text{for all } m \in RS(N, m_0) \text{ and for all } m_0 \Rightarrow$$
$$\Rightarrow y^T . \, m = y^T . \, m_0 .$$

Proof: if $m \in RS(N, m_0)$ then $m = m_0 + C.\sigma$, and pre-multiplying by y^T:

$$y^T . \, m = y^T . \, m_0 + y^T . C . \sigma = y^T . m_0$$

P-semiflows ➔ Conservation of tokens

Functional properties and analysis

- Properties (cont.):
  2. If m is a reachable marking in N, σ a fireable sequence with $\sigma = x$, *and* x a T-semiflow, the next property follows (or T-invariant):

     $$m\,[\sigma > m$$

     Proof: if is a T-semiflow, $m = m_0 + C.x = m_0$

  T-semiflows ➔ Repetitivity of the marking

- P and T-semiflows can be computed using algorithms based in Convex Geometry (linear algebra and linear programming)

Functional properties and analysis

- Definitions:
  - N is conservative $\Leftrightarrow \exists y > 0, \ y^T.C = 0$
  - N is structurally bounded $\Leftrightarrow \exists \ y \geq 1, y^T.C \leq 0$
    (computable in polynomial time)

- Properties: pre-multiplying by y the state equation
  - N conservative $\Rightarrow y^T.\ m = y^T.\ m_0$
    (token conservation)
  - N structurally bounded $\Rightarrow y^T.\ m \leq y^T.\ m_0$
    (tokens limitation)

Functional properties and analysis

- Definitions:
  - N is consistent $\Leftrightarrow \exists\ x > 0,\ C.x = 0$
  - N is structurally repetitive $\Leftrightarrow \exists\ x \geq 1,\ C.x \geq 0$
- Properties:
  - $\langle N, m_0 \rangle$ repetitive $\Rightarrow$ N structurally repetitive
  - N structurally live $\Rightarrow$ N structurally repetitive
  - N structurally live and structurally bounded $\Rightarrow$ structurally repetitive and structurally bounded $\Leftrightarrow$ consistent and conservative

Performance modelling and evaluation

11. Stochastic Petri nets: exact analysis

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Petri nets with temporal interpretation
- Basic exact analysis of SPN

Petri nets with temporal interpretation

- Addition of temporal interpretation to autonomous Petri nets:
  - Timed Petri nets (TPN):
    - Ramchandani, 1974
  - (Interval) Time Petri nets (ITPN):
    - Merlin and Faber, 1976
  - Stochastic Petri nets (SPN):
    - Symons, 1978; Natkin, 1980; Molloy, 1981
  - Generalized stochastic Petri nets (GSPN)
    - Ajmone Marsan, Balbo, Conte, 1984

Petri nets with temporal interpretation

- Basic idea:
  - Queueing network (QN) =

    = structure

    queue rooms, stations,
    and deterministic routing

    \+ distribution of customers (or population)

    (distributed) state of the model

    \+ stochastic interpretation

    random routing, service times,
    number of servers

    $\Rightarrow$ dynamic behaviour: movement of customers

Petri nets with temporal interpretation

- Stochastic Petri net (SPN) =

  = structure

  places, transitions,
  and arcs

  + distribution of tokens (or marking)

  (distributed) state of the model

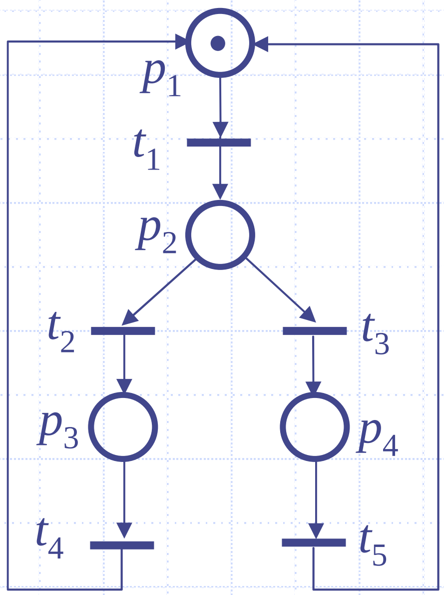
  + stochastic interpretation

  random routing, service times,
  number of servers

  $\Rightarrow$ dynamic behaviour: movement of tokens (firing rule)

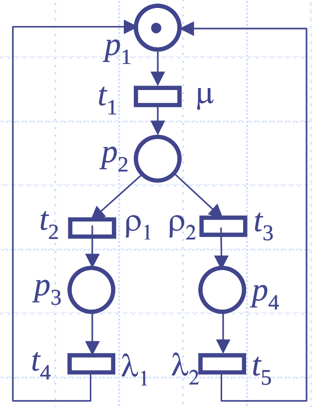
Petri nets with temporal interpretation

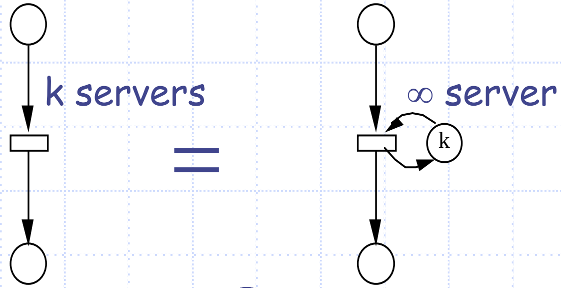
- Autonomous Petri nets
  - Non-determinism with respect to
    - duration of activities and
    - routing
  - Not valid for performance evaluation (quantitative analysis: throughput, response time, average marking)

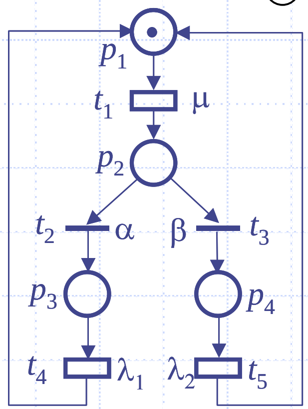


Petri nets with temporal interpretation

- Reduction of the non-determinism
  - Define duration of activities (elapsed time from enabling to firing of a transitions)
    - Constant → *Timed* Petri nets
    - Interval → *Time* Petri nets
    - Random (exponentially distrib.) → → *Stochastic* Petri nets
    - Random or immediate → → *Generalized Stochastic* PNs
  - Define server semantics (single/multiple/infinite)
  - Define routing at conflicts
    - Race between timed transitions
    - Random choice







Outline

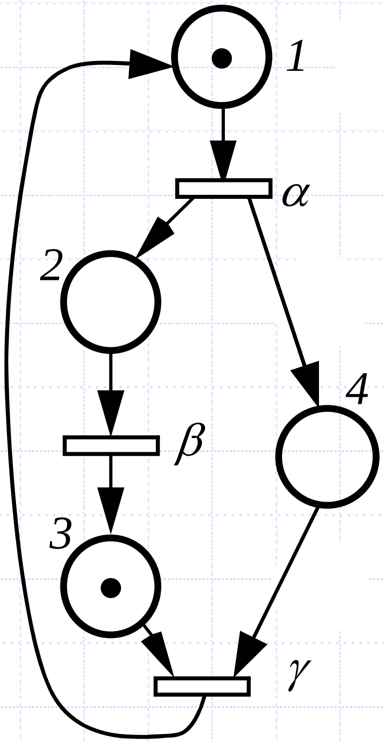
- Petri nets with temporal interpretation
- Basic exact analysis of SPN

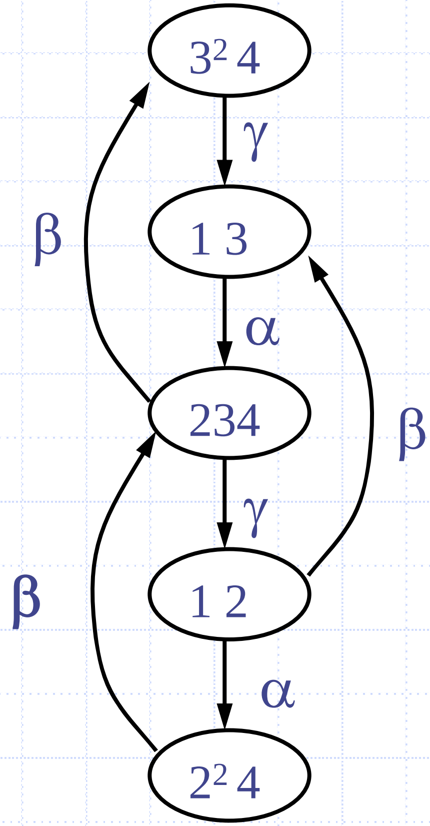
Basic exact analysis of SPN

- Stochastic Petri nets
  - Duration of activities: exponentially distributed random variables
  - Single server semantics at each transition
  - Conflicts resolution: race policy

The reachability graph of the SPN is isomorphic to a Continuous Time Markov Chain

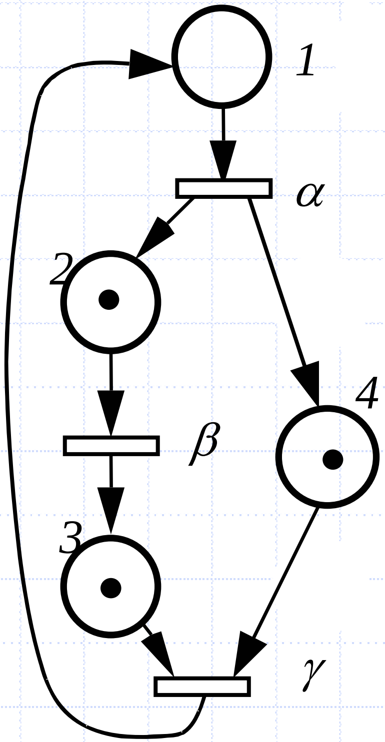
Basic exact analysis of SPN

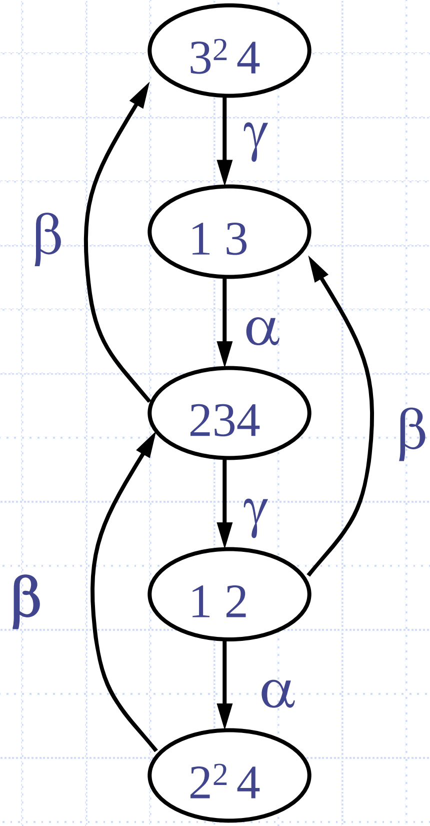




The reachability graph of the SPN is isomorphic to a Continuous Time Markov Chain

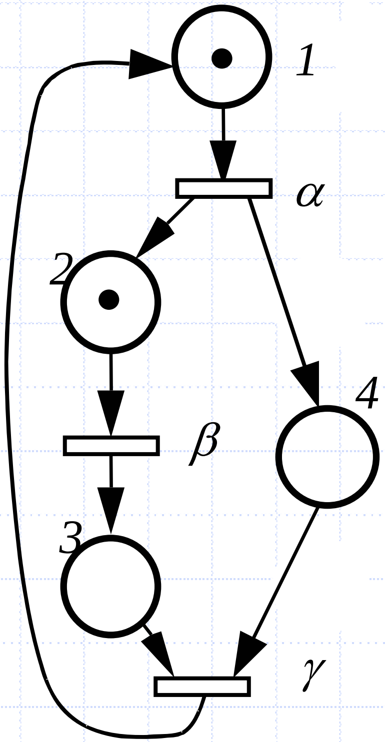
Basic exact analysis of SPN

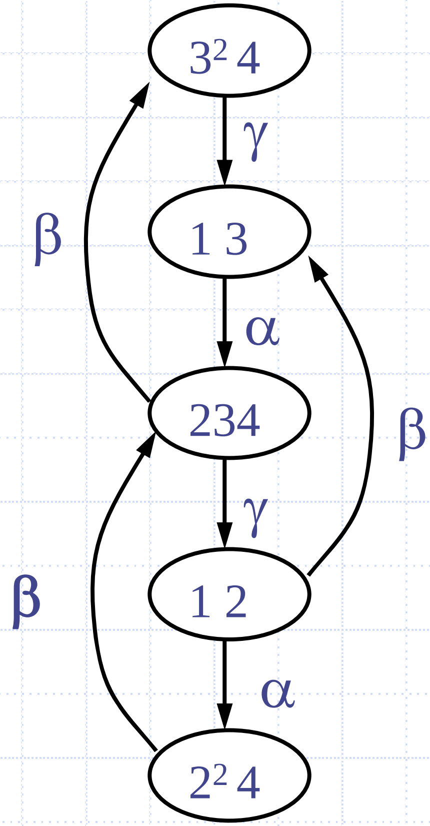




The reachability graph of the SPN is isomorphic to a Continuous Time Markov Chain

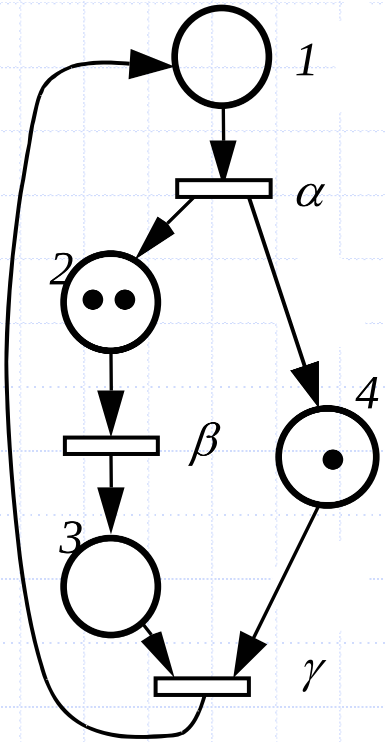
Basic exact analysis of SPN

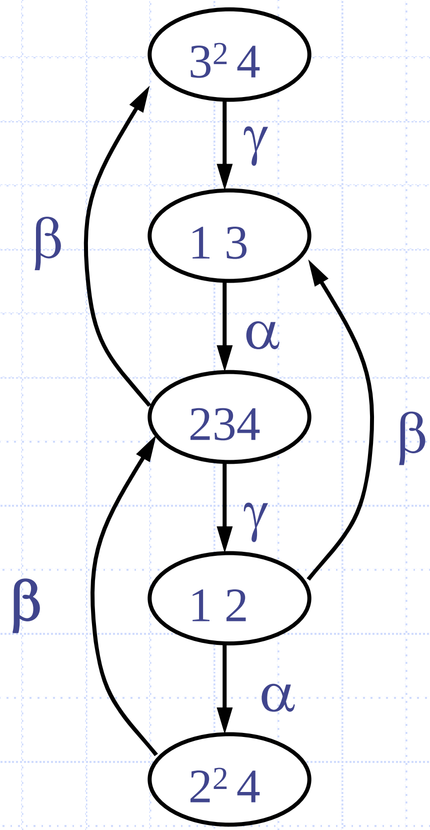




The reachability graph of the SPN is isomorphic to a Continuous Time Markov Chain

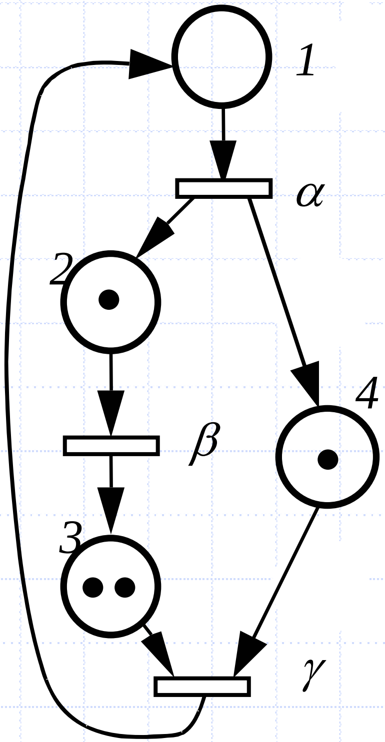
Basic exact analysis of SPN

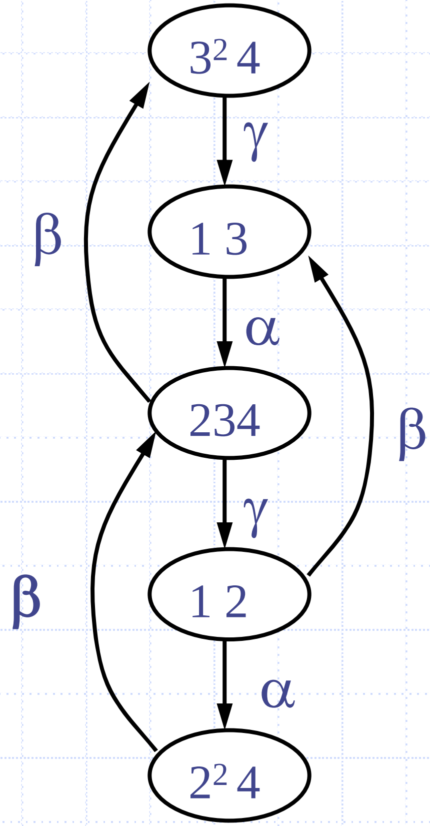




The reachability graph of the SPN is isomorphic to a Continuous Time Markov Chain

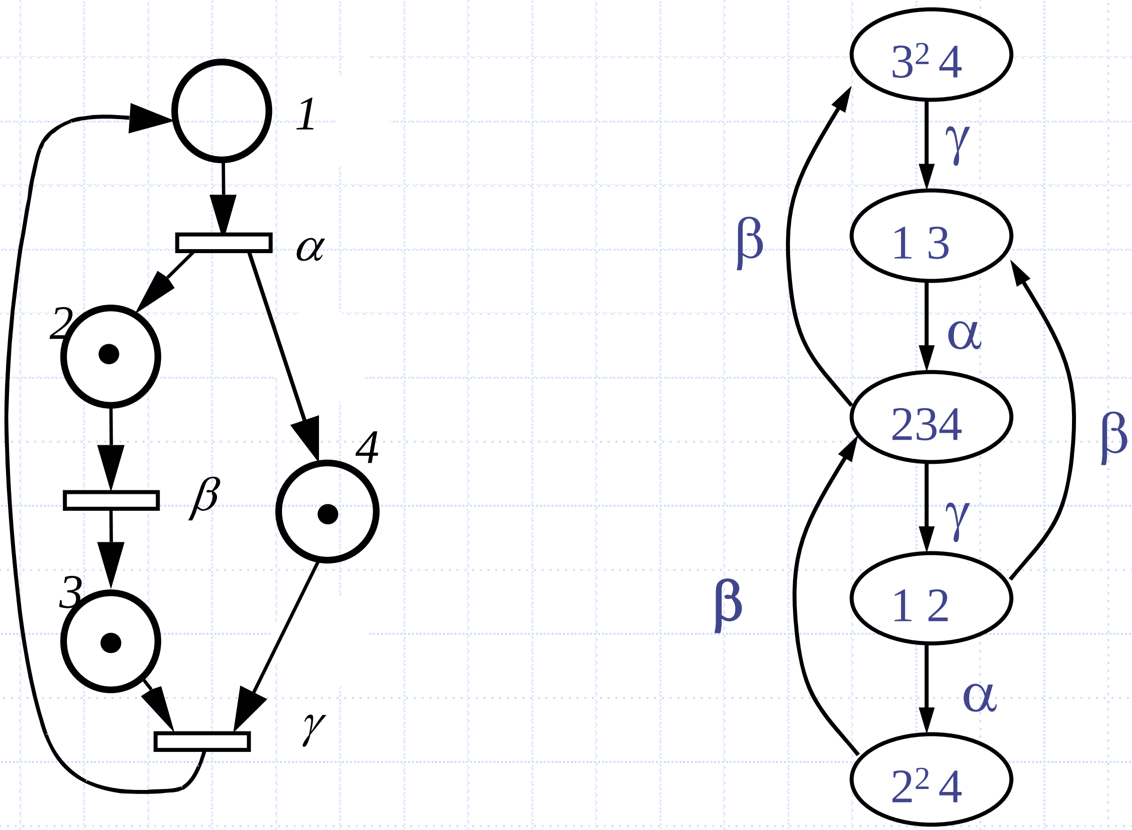
Basic exact analysis of SPN





The reachability graph of the SPN is isomorphic to a Continuous Time Markov Chain

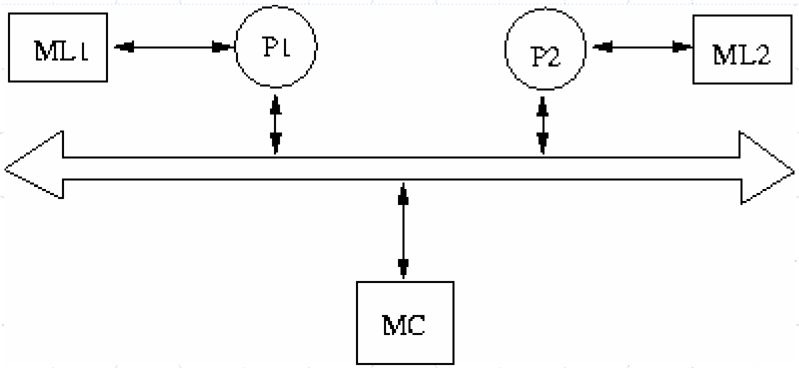
Basic exact analysis of SPN



The reachability graph of the SPN is isomorphic to a Continuous Time Markov Chain

Basic exact analysis of SPN

- Shared memory multiprocessor



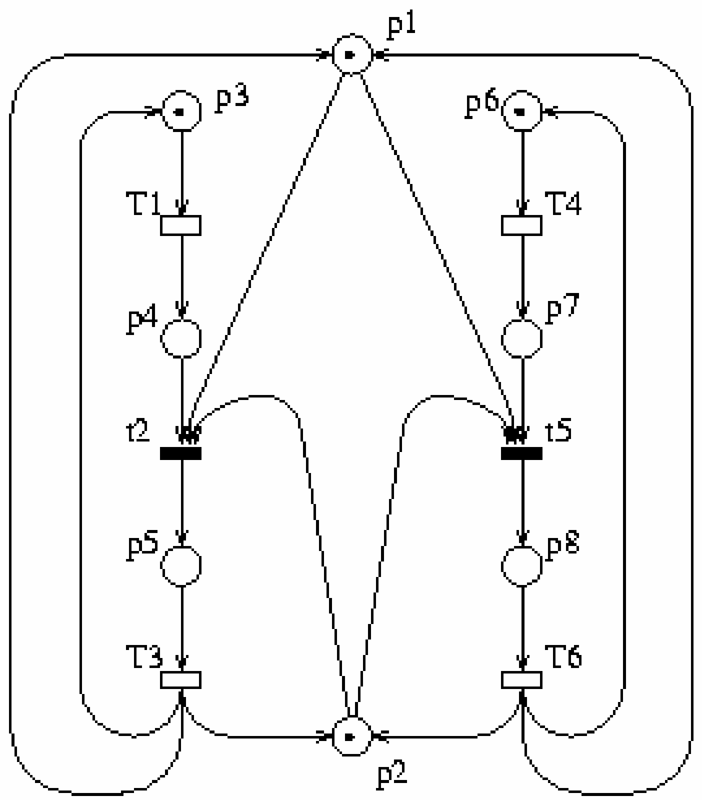
Both processors behave in a similar way:

- A cyclic sequence of: local activity, then
- an access request to the shared memory, and then
- accessing the shared memory

Basic exact analysis of SPN

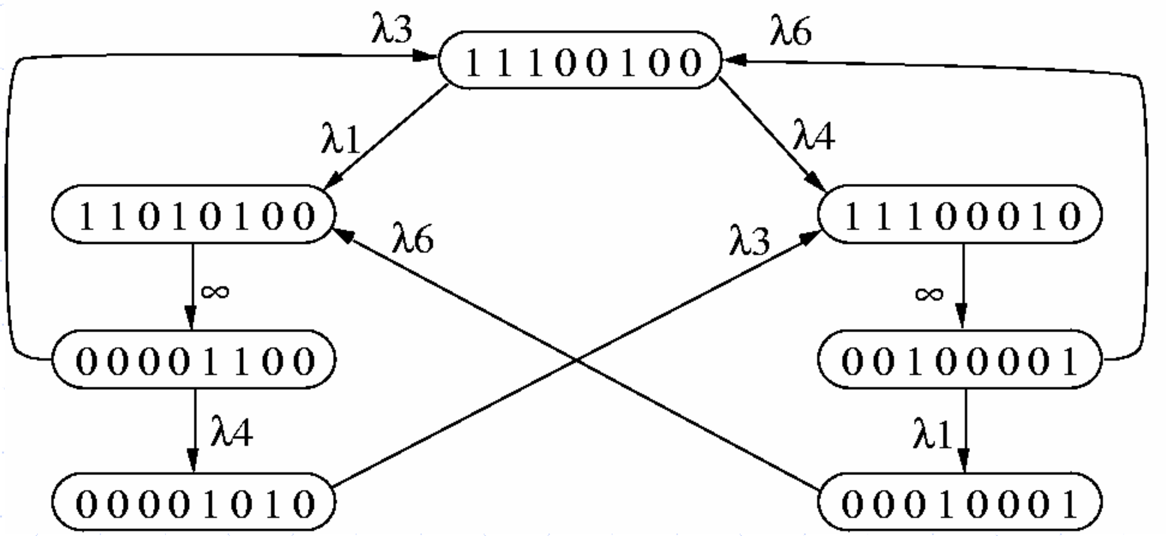
- All transitions have exponentially distributed durations, except for t2 and t5,
  access request to the shared memory (immediate)

  → GSPN



Basic exact analysis of SPN

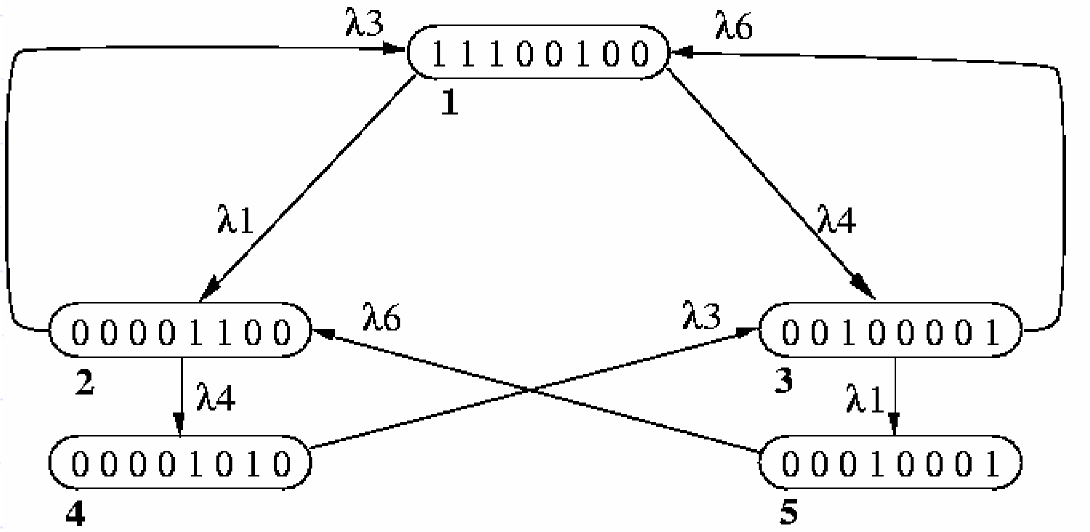
- Reachability graph



It is not isomorphic to a Continuous Time Markov Chain (infinite rates are not allowed in CTMCs)

Basic exact analysis of SPN

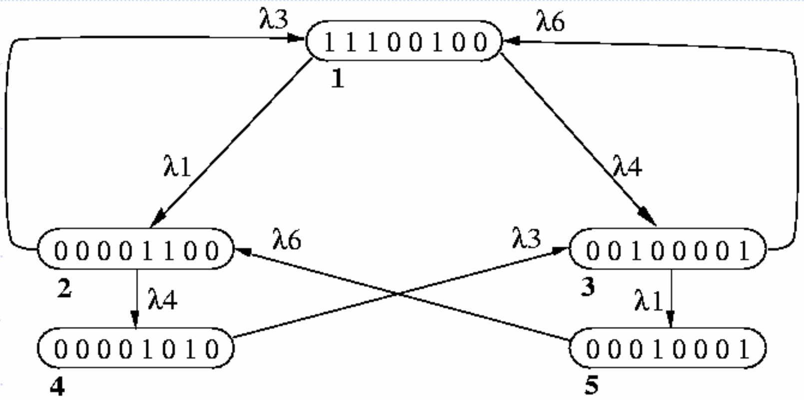
- Tangible reachability graph



It is isomorphic to a Continuous Time Markov Chain

Basic exact analysis of SPN

- Infinitesimal generator matrix of the CTMC



$$\begin{bmatrix} -(\lambda_1+\lambda_4) & \lambda_1 & \lambda_4 & 0 & 0 \\ \lambda_3 & -(\lambda_3+\lambda_4) & 0 & \lambda_4 & 0 \\ \lambda_6 & 0 & -(\lambda_1+\lambda_6) & 0 & \lambda_1 \\ 0 & 0 & \lambda_3 & -\lambda_3 & 0 \\ 0 & \lambda_6 & 0 & 0 & -\lambda_6 \end{bmatrix}$$

Basic exact analysis of SPN

- The stationary distribution can be computed (steady state probability of each state)

$$(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \boldsymbol{\pi}_3, \boldsymbol{\pi}_4, \boldsymbol{\pi}_5) \cdot \begin{bmatrix} -(\lambda_1+\lambda_4) & \lambda_1 & \lambda_4 & 0 & 0 \\ \lambda_3 & -(\lambda_3+\lambda_4) & 0 & \lambda_4 & 0 \\ \lambda_6 & 0 & -(\lambda_1+\lambda_6) & 0 & \lambda_1 \\ 0 & 0 & \lambda_3 & -\lambda_3 & 0 \\ 0 & \lambda_6 & 0 & 0 & -\lambda_6 \end{bmatrix} = \mathbf{0}$$

$$\boldsymbol{\pi}_1 + \boldsymbol{\pi}_2 + \boldsymbol{\pi}_3 + \boldsymbol{\pi}_4 + \boldsymbol{\pi}_5 = 1$$

- And from here, compute performance index:
  - Processing power = average number of processors effectively (locally) working = $2\pi_1+\pi_2+\pi_3$

Performance modelling and evaluation

12. Performance bounds for SPN

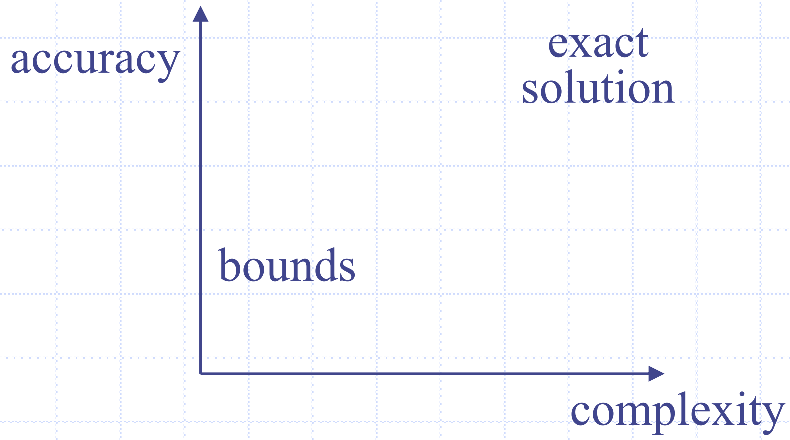
Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Preliminary comments
- Introducing the ideas: Marked Graphs case
- Generalization: use of visit ratios
- Improvements of the bounds
- A general linear programming statement

Preliminary comments

- Interest of bounding techniques
  - preliminary phases of design
    - many parameters are not known accurately
    - quick evaluation and rejection of those clearly bad



Preliminary comments

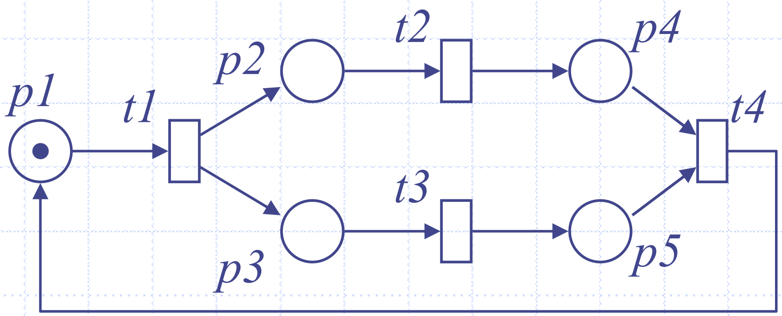
- Net-driven solution technique
  - stressing the intimate relationship between qualitative and quantitative aspects of PN's
  - structure theory of net models

→ efficient computation techniques

Outline

- Preliminary comments
- Introducing the ideas: Marked Graphs case
- Generalization: use of visit ratios
- Improvements of the bounds
- A general linear programming statement

Introducing ideas: Marked Graph case



generally distributed service times (random variables X_i with mean $\bar{\mathbf{s}}[t_j]$)

we assume **infinite-server semantics**

exact cycle time (random variable): $X = X_1 + \max\{X_2, X_3\} + X_4$

average cycle time: $\Gamma = \bar{\mathbf{s}}[t_1] + \mathrm{E}[\max\{X_2, X_3\}] + \bar{\mathbf{s}}[t_4]$

but (non-negative variables): $X_2, X_3 \le \max\{X_2, X_3\} \le X_2 + X_3$

therefore:

$$\boxed{\bar{\mathbf{s}}[t_1] + \max\{\bar{\mathbf{s}}[t_2], \bar{\mathbf{s}}[t_3]\} + \bar{\mathbf{s}}[t_4] \le \Gamma \le \bar{\mathbf{s}}[t_1] + \bar{\mathbf{s}}[t_2] + \bar{\mathbf{s}}[t_3] + \bar{\mathbf{s}}[t_4]}$$

Introducing ideas: Marked Graph case

Thus, the lower bound for the average cycle time is computed looking for the slowest circuit

$$\Gamma \geq \max_{C \in \{\text{circuits of the net}\}} \left(\sum_{t_i \in C} \bar{\mathbf{s}}[t_i] \Big/ \#\text{tokens in } C \right)$$

Interpretation:

an MG may be built synchronising circuits, so we look for the bottleneck

Introducing ideas: Marked Graph case

- Computation:

$$\begin{aligned}\Gamma \geq \ & \text{maximum } \mathbf{y} \cdot \mathbf{Pre} \cdot \bar{\mathbf{s}} \\ & \text{subject to } \ \mathbf{y} \cdot \mathbf{C} = \mathbf{0} \\ & \qquad\qquad\quad \mathbf{y} \cdot \mathbf{m_0} = 1 \\ & \qquad\qquad\quad \mathbf{y} \geq \mathbf{0}\end{aligned}$$

($\bar{\mathbf{s}}$ is the vector of average service times)

(the proof of this comes later for a more general case)

solving a linear programming problem
(**polynomial complexity** on the net size)

Introducing ideas: Marked Graph case

- Even if naïf, the bounds are tight!
- Lower bound for the average cycle time

$$\max\{\bar{\mathbf{s}}[t_2], \bar{\mathbf{s}}[t_3]\} \leq \mathrm{E}[\max\{X_2, X_3\}]$$

  - it is exact for deterministic timing
  - it cannot be improved using only mean values of r.v. (it is reached in a limit case for a family of random variables with arbitrary means and variances)

Introducing ideas: Marked Graph case

$$X_{\mu,\sigma}(\alpha) = \begin{cases} \mu\alpha & \text{with probability } 1-\varepsilon \\ \mu\left(\alpha + \dfrac{1-\alpha}{\varepsilon}\right) & \text{with probability } \varepsilon \end{cases} \qquad \varepsilon = \frac{\mu^2(1-\alpha)^2}{\mu^2(1-\alpha)^2+\sigma^2}$$

$(0 \le \alpha \le 1)$

$$\mathrm{E}\left[X_{\mu,\sigma}(\alpha)\right] = \mu \ ; \ \mathrm{Var}\left[X_{\mu,\sigma}(\alpha)\right] = \sigma^2$$

$$\lim_{\alpha\to 1} \mathrm{E}\left[\max\left(X_{\mu,\sigma}(\alpha), X_{\mu',\sigma'}(\alpha)\right)\right] = \max(\mu,\mu')$$

$$\mathrm{E}\left[X_{\mu,\sigma}(\alpha) + X_{\mu',\sigma'}(\alpha)\right] = \mu + \mu', \ \forall \ 0 \le \alpha < 1$$

they behave "as deterministic" for the 'max' and '+' operators in the limit ($\alpha \to 1$)

Introducing ideas: Marked Graph case

- Upper bound for the average cycle time

$$\boxed{\Gamma \leq \sum_{t \in T} \bar{\mathbf{s}}[t]}$$

- it cannot be improved for 1-live MG's using only mean values of r.v. (it is reached in a limit case for a family of random variables with arbitrary means)

Introducing ideas: Marked Graph case

$$X^i_\mu(\varepsilon) = \begin{cases} 0 & \text{with probability} \quad 1-\varepsilon^i \\ \dfrac{\mu}{\varepsilon^i} & \text{with probability} \quad \varepsilon^i \end{cases}$$

$(0 < \varepsilon < 1)$

$$\mathrm{E}\left[X^i_\mu(\varepsilon)\right] = \mu \; ; \; \mathrm{E}\left[X^i_\mu(\varepsilon)^2\right] = \frac{\mu^2}{\varepsilon^i}$$

If $X_j = X^{j-1}_{\bar{\mathbf{s}}[t_j]}(\varepsilon), \; \forall t_j \in T,$ then for varying (decreasing) values of ε:

$$\boxed{\mathrm{E}[\max(X_i, X_j)] = \bar{\mathbf{s}}[t_i] + \bar{\mathbf{s}}[t_j] + o(\varepsilon)}$$

Outline

- Preliminary comments
- Introducing the ideas: Marked Graphs case
- Generalization: use of visit ratios
- Improvements of the bounds
- A general linear programming statement

Generalization: use of visit ratios

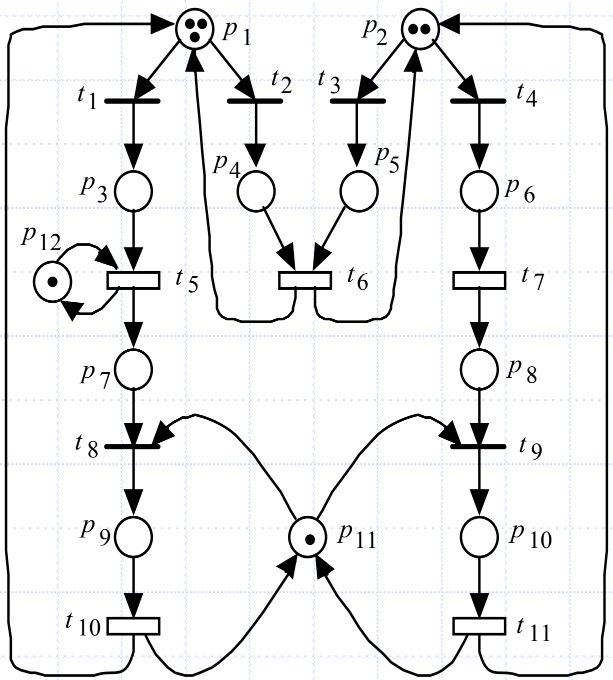
- Visit ratios = relative throughput
  (number of visits to t_i per each visit to t_1)

$$\mathbf{v}[t] = \frac{\chi[t]}{\chi[t_1]} = \Gamma[t_1]\ \chi[t]$$

average interfiring time of t_1

Generalization: use of visit ratios

- For some net classes **v** can be computed as:



$$\mathbf{C} \cdot \mathbf{v} = \mathbf{0};$$
$$r_1 \mathbf{v}[t_2] = r_2 \mathbf{v}[t_1];$$
$$r_3 \mathbf{v}[t_4] = r_4 \mathbf{v}[t_3];$$
$$\mathbf{v}[t_1] = 1$$

Generalization: use of visit ratios

- Little's law ($L=\lambda W$) applied to a place p:

$$\bar{\mu}[p] = (\mathbf{Pre}[p,T] \cdot \chi)\ \bar{\mathbf{r}}[p]$$

Assume that timed transitions are never in conflict (**conflicts are modelled with immediate transitions**), then either all output transitions of p are immediate or p has a unique output transition, say t_1, and t_1 is timed, thus:

$$\bar{\mu}[p] = (\mathbf{Pre}[p,T] \cdot \chi)\ \bar{\mathbf{r}}[p] = \mathbf{Pre}[p,t_1]\ \chi[t_1]\ \bar{\mathbf{r}}[p]$$

$$\geq \mathbf{Pre}[p,t_1]\ \chi[t_1]\ \bar{\mathbf{s}}[t_1] = \sum_{j=1}^{m} \mathbf{Pre}[p,t_j]\ \chi[t_j]\ \bar{\mathbf{s}}[t_j]$$

Generalization: use of visit ratios

Then: $\Gamma[t_1]\ \overline{\mu}[p] \geq \sum_{j=1}^{m} \mathbf{Pre}[p,t_j]\ \Gamma[t_1]\ \chi[t_j]\ \overline{\mathbf{s}}[t_j] = \sum_{j=1}^{m} \mathbf{Pre}[p,t_j]\ \mathbf{v}[t_j]\ \overline{\mathbf{s}}[t_j]$

Hence: $\Gamma[t_1]\ \overline{\mu} \geq \mathbf{Pre} \cdot \overline{\mathbf{D}}$ where $\overline{\mathbf{D}}[t] = \overline{\mathbf{s}}[t]\mathbf{v}[t]$ is the average service demand of t

Premultiplying by a *P*-semiflow **y** $(\mathbf{y} \cdot \mathbf{C} = \mathbf{0},\ \mathbf{y} \geq \mathbf{0},\ \text{thus}\ \mathbf{y} \cdot \overline{\mu} = \mathbf{y} \cdot \mathbf{m_0})$,

$$\Gamma[t_1] \geq \text{maximum} \quad \frac{\mathbf{y} \cdot \mathbf{Pre} \cdot \overline{\mathbf{D}}}{\mathbf{y} \cdot \mathbf{m_0}} \quad \text{subject to} \quad \mathbf{y} \cdot \mathbf{C} = \mathbf{0},\ \mathbf{1} \cdot \mathbf{y} > 0,\ \mathbf{y} \geq \mathbf{0}$$

$\Longleftrightarrow$

$$\Gamma[t_1] \geq \text{maximum} \quad \frac{\mathbf{y} \cdot \mathbf{Pre} \cdot \overline{\mathbf{D}}}{q} \quad \text{subject to} \quad \mathbf{y} \cdot \mathbf{C} = \mathbf{0},\ \mathbf{1} \cdot \mathbf{y} > 0,\ q = \mathbf{y} \cdot \mathbf{m_0},\ \mathbf{y} \geq \mathbf{0}$$

Generalization: use of visit ratios

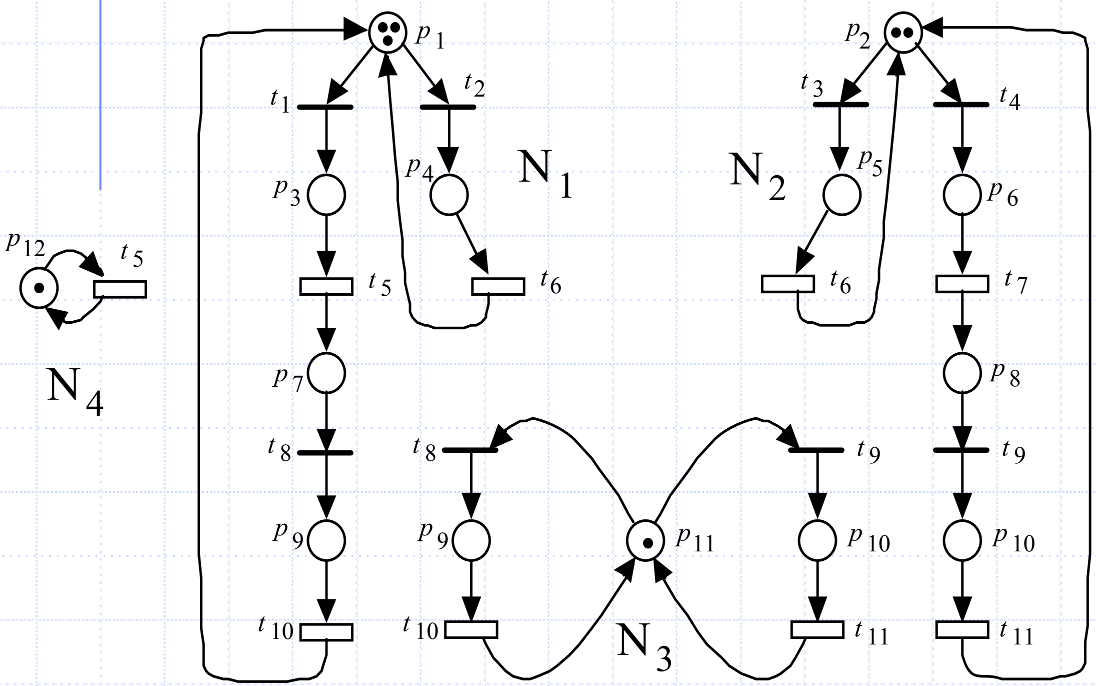
Since $\mathbf{y}\cdot\mathbf{m}_0 > 0$ (live system), we change $\mathbf{y}/q$ to $\mathbf{y}$ and we obtain ($\mathbf{1}\cdot\mathbf{y} > 0$ is removed because $\mathbf{y}\cdot\mathbf{m}_0 = 1$ implies $\mathbf{1}\cdot\mathbf{y} > 0$):

$$\boxed{\begin{array}{lll} \Gamma[t_1] \geq & \text{maximum} & \mathbf{y}\cdot\mathbf{Pre}\cdot\overline{\mathbf{D}} \\ & \text{subject to} & \mathbf{y}\cdot\mathbf{C} = \mathbf{0} \\ & & \mathbf{y}\cdot\mathbf{m_0} = 1 \\ & & \mathbf{y} \geq \mathbf{0} \end{array}}$$

again, a linear programming problem
(polynomial complexity on the net size)

Generalization: use of visit ratios

Interpretation: **slowest subsystem generated by *P*-semiflows, in isolation**



minimal *P*–semiflows

$\mathbf{y_1} = (1,0,1,1,0,0,1,0,1,0,0,0)$

$\mathbf{y_2} = (0,1,0,0,1,1,0,1,0.1,0,0)$

$\mathbf{y_3} = (0,0,0,0,0,0,0,0,1,1,1,0)$

$\mathbf{y_4} = (0,0,0,0,0,0,0,0,0,0,0,1)$

$$\Gamma[t_1] \geq \max \left\{ \begin{array}{l} (\bar{\mathbf{s}}[t_5]+\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_{10}])/3, \\ (\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_7]+\bar{\mathbf{s}}[t_{11}])/2, \\ \bar{\mathbf{s}}[t_{10}]+\bar{\mathbf{s}}[t_{11}], \\ \bar{\mathbf{s}}[t_5] \end{array} \right\}$$

Generalization: use of visit ratios

- Upper bound for the average interfiring time

$$\boxed{\Gamma[t_1] \leq \sum_{t \in T} \mathbf{v}[t]\ \bar{\mathbf{s}}[t] = \sum_{t \in T} \overline{\mathbf{D}}[t]}$$

remember the marked graphs case (**v** = **1**): $\Gamma \leq \sum_{t \in T} \bar{\mathbf{s}}[t]$

Outline

- Preliminary comments
- Introducing the ideas: Marked Graphs case
- Generalization: use of visit ratios
- Improvements of the bounds
- A general linear programming statement

Improvements of the bounds

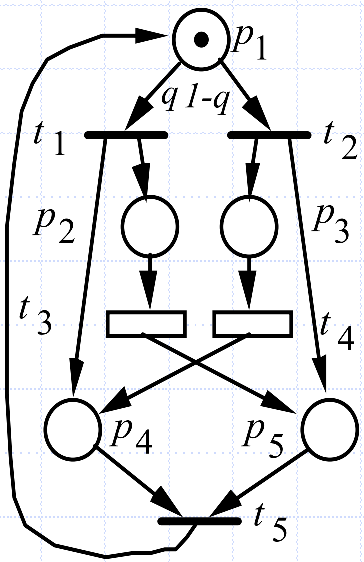
- **Structural improvements**

  bounds still based only on the mean values (not on higher moments of r.v., **insensitive** bounds)

  - lower bound for the average interfiring time: use of **implicit places** to increase the number of minimal *P*-semiflows
  - upper bound for the average interfiring time: use of **liveness bound of transitions** to improve the bound for some net subclasses

Improvements of the bounds

- Use of implicit places



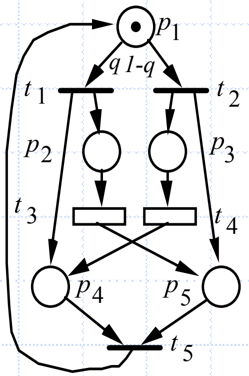
$$\Gamma[t_5] = q\bar{\mathbf{s}}[t_3] + (1-q)\bar{\mathbf{s}}[t_4]$$

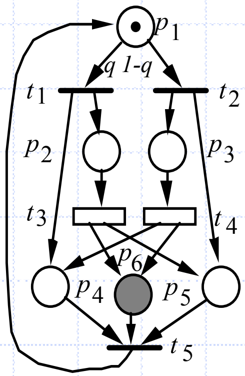
$$\begin{array}{lll} \Gamma[t_1] \geq & \text{maximum} & \mathbf{y}\cdot\mathbf{Pre}\cdot\overline{\mathbf{D}} \\ & \text{subject to} & \mathbf{y}\cdot\mathbf{C} = \mathbf{0} \\ & & \mathbf{y}\cdot\mathbf{m_0} = 1 \\ & & \mathbf{y} \geq \mathbf{0} \end{array}$$

$$\Downarrow$$

$$\Gamma[t_5] \geq \max\left\{q\bar{\mathbf{s}}[t_3], (1-q)\bar{\mathbf{s}}[t_4]\right\}$$

Improvements of the bounds





$$\Gamma[t_1] \geq \quad \text{maximum} \quad \mathbf{y}\cdot\mathbf{Pre}\cdot\overline{\mathbf{D}}$$
$$\text{subject to} \quad \mathbf{y}\cdot\mathbf{C} = \mathbf{0}$$
$$\mathbf{y}\cdot\mathbf{m_0} = 1$$
$$\mathbf{y} \geq \mathbf{0}$$

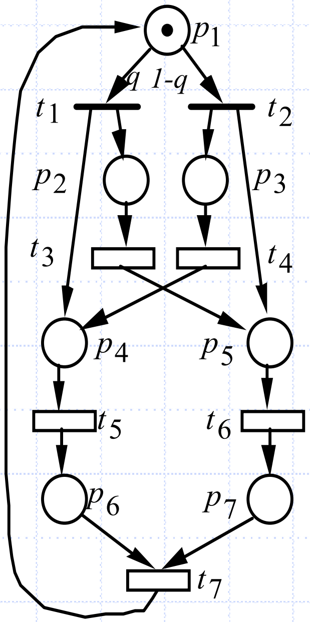
$$\Gamma[t_5] = q\bar{\mathbf{s}}[t_3] + (1-q)\bar{\mathbf{s}}[t_4]$$

$$\Gamma[t_5] \geq \max\left\{ q\bar{\mathbf{s}}[t_3],\ (1-q)\bar{\mathbf{s}}[t_4],\ q\bar{\mathbf{s}}[t_3] + (1-q)\bar{\mathbf{s}}[t_4] \right\}$$

in this case, we get the exact value!

Improvements of the bounds

in general...



$$\Gamma[t_1] \geq \begin{array}{ll} \text{maximum} & \mathbf{y}\cdot\mathbf{Pre}\cdot\overline{\mathbf{D}} \\ \text{subject to} & \mathbf{y}\cdot\mathbf{C}=\mathbf{0} \\ & \mathbf{y}\cdot\mathbf{m_0}=1 \\ & \mathbf{y}\geq\mathbf{0} \end{array}$$

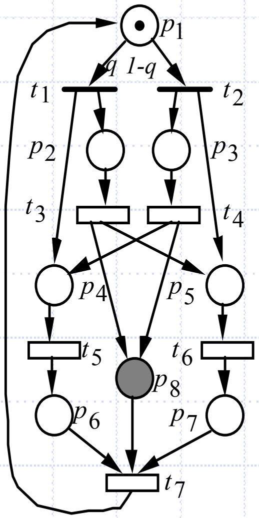
$$\Downarrow$$

$$\Gamma[t_7] \geq \max \{\ q\bar{\mathbf{s}}[t_3]+\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_7],$$

$$(1-q)\bar{\mathbf{s}}[t_4]+\bar{\mathbf{s}}[t_5]+\bar{\mathbf{s}}[t_7]\ \}$$

Improvements of the bounds

in general, the bound is non-reachable



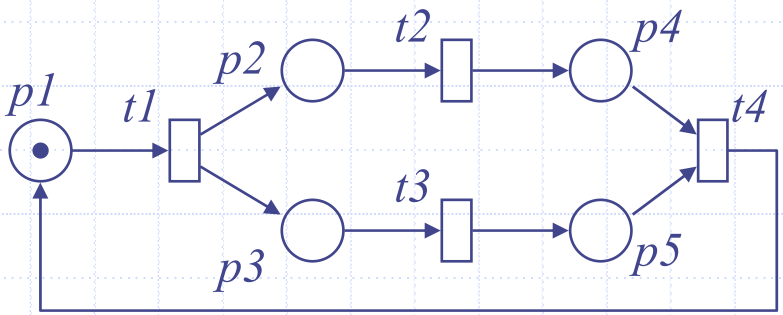
$$\begin{aligned}\Gamma[t_7] \geq \max \{\ & q\bar{\mathbf{s}}[t_3]+\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_7],\\ & (1-q)\bar{\mathbf{s}}[t_4]+\bar{\mathbf{s}}[t_5]+\bar{\mathbf{s}}[t_7],\\ & q\bar{\mathbf{s}}[t_3]+(1-q)\bar{\mathbf{s}}[t_4]+\bar{\mathbf{s}}[t_7]\ \}\end{aligned}$$

$$\begin{aligned}\Gamma[t_7] &= q\max\{\bar{\mathbf{s}}[t_5],\bar{\mathbf{s}}[t_3]+\bar{\mathbf{s}}[t_6]\}+(1-q)\max\{\bar{\mathbf{s}}[t_4]+\bar{\mathbf{s}}[t_5],\bar{\mathbf{s}}[t_6]\}+\bar{\mathbf{s}}[t_7]\\ &= \max \{\ q\bar{\mathbf{s}}[t_3]+\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_7],\\ &\qquad (1-q)\bar{\mathbf{s}}[t_4]+\bar{\mathbf{s}}[t_5]+\bar{\mathbf{s}}[t_7],\\ &\qquad q\bar{\mathbf{s}}[t_3]+(1-q)\bar{\mathbf{s}}[t_4]+(1-q)\bar{\mathbf{s}}[t_5]+q\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_7],\\ &\qquad q\bar{\mathbf{s}}[t_5]+(1-q)\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_7]\ \}\end{aligned}$$

(deterministic timing)

Improvements of the bounds

- Use of liveness bounds
  - upper bound for the average interfiring time:

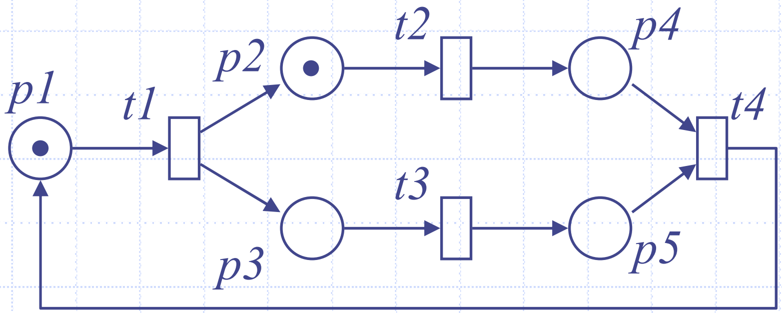


$$\Gamma \leq \sum_{t \in T} \bar{\mathbf{s}}[t]$$

reachable for 1-live marked graphs, but…

Improvements of the bounds

it can be improved for *k*-live marked graphs



$$\Gamma \le \bar{\mathbf{s}}[t_1] + \frac{\bar{\mathbf{s}}[t_2]}{2} + \bar{\mathbf{s}}[t_3] + \bar{\mathbf{s}}[t_4]$$

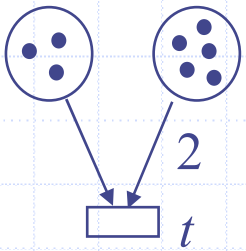
liveness bound of t_2

Improvements of the bounds

- Definitions of enabling degree, enabling bound, structural enabling bound, and liveness bound

  - instantaneous enabling degree of a transition at a given marking

$$\mathbf{e}[t](\mathbf{m}) = \sup\left\{k \in \mathrm{N} : \forall p \in {}^{\bullet}t,\ \mathbf{m}[p] \geq k\ \mathbf{Pre}[p,t]\right\}$$

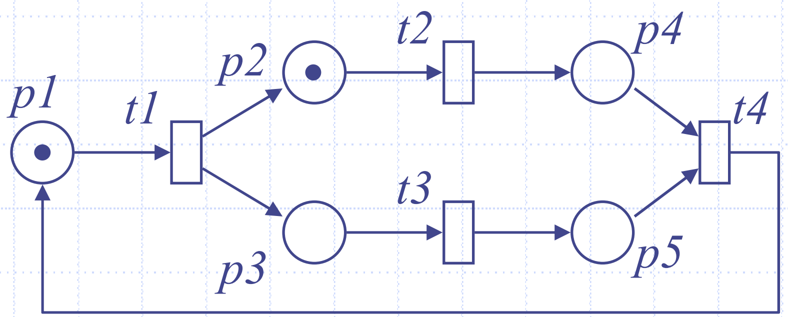


$\mathbf{e}[t](\mathbf{m}) = 2$

Improvements of the bounds

- enabling bound of a transition in a given system: maximum among the instantaneous enabling degree at all reachable markings

$$\mathbf{eb}[t] = \sup\left\{ k \in \mathrm{N} : \ \exists \mathbf{m_0} \xrightarrow{\sigma} \mathbf{m}, \ \forall p \in {}^{\bullet}t, \ \mathbf{m}[p] \geq k \ \mathbf{Pre}[p,t] \right\}$$

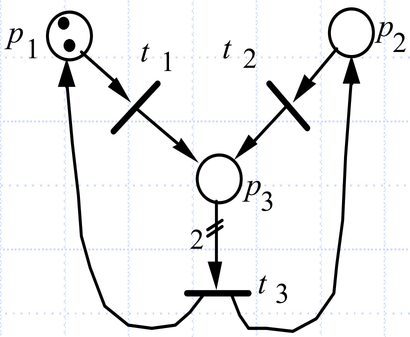


$$\mathbf{eb}[t_2] = 2$$

Improvements of the bounds

- liveness bound of a transition in a given system: number of servers available in *t* in steady state

$$\mathbf{lb}[t] = \sup\left\{ k \in \mathrm{N} : \ \forall \mathbf{m}', \mathbf{m_0} \xrightarrow{\sigma} \mathbf{m}', \exists \mathbf{m}, \mathbf{m}' \xrightarrow{\sigma'} \mathbf{m} \wedge \forall p \in {}^{\bullet}t, \mathbf{m}[p] \geq k \ \mathbf{Pre}[p,t] \right\}$$



$$\mathbf{lb}[t_1] = 1 < 2 = \mathbf{eb}[t_1]$$

Improvements of the bounds

- structural enabling bound of a transition in a given system: structural counterpart of the enabling bound (substitute reachability condition by

$$\mathbf{m} = \mathbf{m_0} + \mathbf{C} \cdot \sigma;\ \mathbf{m}, \sigma \geq \mathbf{0})$$

$$\begin{aligned}
\mathbf{seb}[t] = \ & \text{maximum } k \\
& \text{subject to } \ \mathbf{m_0}[p] + \mathbf{C}[p,T] \cdot \boldsymbol{\sigma} \geq k\ \mathbf{Pre}[p,t], \ \forall p \in P \\
& \qquad\qquad\quad \boldsymbol{\sigma} \geq 0
\end{aligned}$$

Property: For any net system $\mathbf{seb}[t] \geq \mathbf{eb}[t] \geq \mathbf{lb}[t], \ \forall \ t.$

Property: For live and bounded free choice systems,

$$\mathbf{seb}[t] = \mathbf{eb}[t] = \mathbf{lb}[t], \quad \forall t.$$

Improvements of the bounds

improvement of the bound for live and bounded free choice systems:

$$\Gamma[t_1] \leq \sum_{t \in T} \frac{\mathbf{v}[t]\ \bar{\mathbf{s}}[t]}{\mathbf{seb}[t]} = \sum_{t \in T} \frac{\overline{\mathbf{D}}[t]}{\mathbf{seb}[t]}$$

this bound cannot be improved for marked graphs (using only the mean values of service times)

Outline

- Preliminary comments
- Introducing the ideas: Marked Graphs case
- Generalization: use of visit ratios
- Improvements of the bounds
- A general linear programming statement

A general linear programming statement

- The idea

a linear function

maximize [or minimize] $f(\bar{\mu}, \chi)$

subject to any linear constraint that we are able to state for $\bar{\mu}$, χ, and other needed additional variables

linear operational laws

A general linear programming statement

- A set of linear constraints:

$$\bar{\mu} = \mathbf{m_0} + \mathbf{C} \cdot \boldsymbol{\sigma}$$ (state equation)

$$\sum_{t \in {}^\bullet p} \chi[t]\ \mathbf{Post}[p,t] \geq \sum_{t \in p^\bullet} \chi[t]\ \mathbf{Pre}[p,t], \quad \forall p \in P$$

$$\sum_{t \in {}^\bullet p} \chi[t]\ \mathbf{Post}[p,t] = \sum_{t \in p^\bullet} \chi[t]\ \mathbf{Pre}[p,t], \quad \forall p \in P \text{ bounded}$$

(flow balance equation)

$$\frac{\chi[t_i]}{r_i} = \frac{\chi[t_j]}{r_j}, \quad \forall t_i, t_j \in T: \text{ behavioural free choice}$$

$$(\text{e.g. } \mathbf{Pre}[P,t_i] = \mathbf{Pre}[P,t_j])$$

$\ldots \qquad \ldots$

A general linear programming statement

$$\chi[t]\ \bar{\mathbf{s}}[t] \leq \frac{\bar{\mu}[p]}{\mathbf{Pre}[p,t]}, \qquad \forall t \in T,\ \forall p \in {}^{\bullet}t \qquad \text{(maximum throughput law)}$$

$$\chi[t]\ \bar{\mathbf{s}}[t] \geq \frac{\bar{\mu}[p] - \mathbf{Pre}[p,t] + 1}{\mathbf{Pre}[p,t]}, \qquad \forall t \in T \text{ persistent, age memory or immediate: } {}^{\bullet}t = \{p\} \qquad \text{(minimum throughput law)}$$

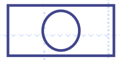
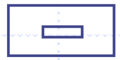
$$\ldots \qquad \ldots$$

$$\bar{\mu},\ \chi,\ \boldsymbol{\sigma} \geq \mathbf{0}$$

A general linear programming statement

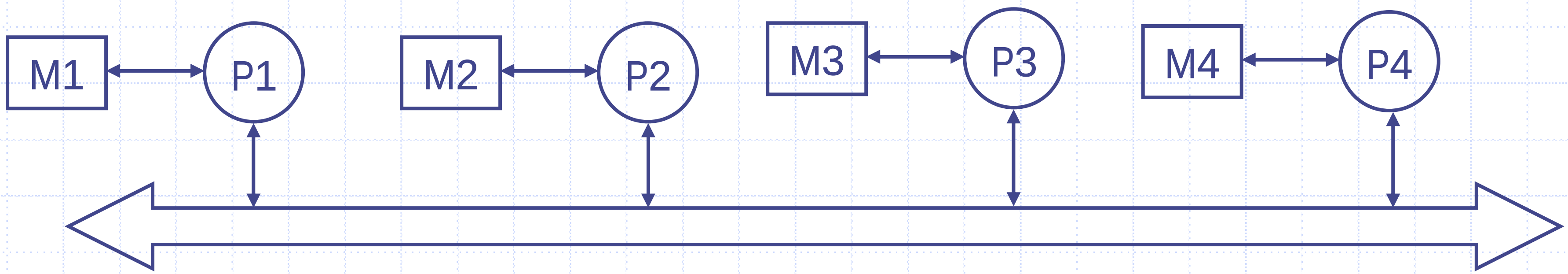
- It can be improved using second order moments
- It can be extended to well-formed coloured nets
- It has been recently extended to Time Petri Nets (timing based on intervals, usefull for the modelling and analysis of real-time systems)

A general linear programming statement

- It is implemented in *GreatSPN*
  - select place (transition) object  ()
  - click right mouse button and select “show”
  - click again right mouse button and select “Average M.B.” (“LP Throughput Bounds”)
  - click left mouse button for upper bound
  - click middle mouse button for lower bound

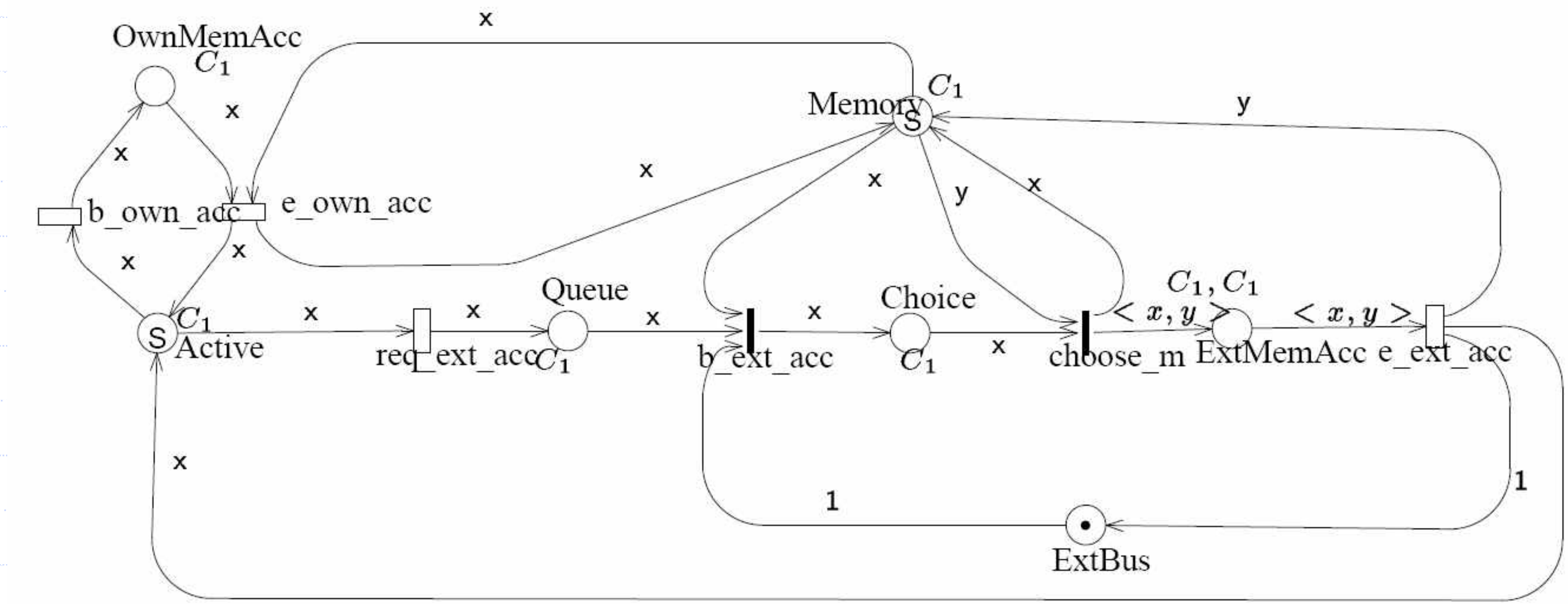
A general linear programming statement

- Example: a shared-memory multiprocessor
  - set of processing modules (with local memory) interconnected by a common bus called the "external bus"
  - a processor can access its own memory module directly from its private bus through one port, or it can access non-local shared-memory modules by means of the external bus
  - priority is given to external access through the external bus with respect to the accesses from the local processor



A general linear programming statement

- Timed Well-Formed Coloured Net (TWN) model of the shared-memory multiprocessor



Average service time of timed transitions equal to 0.5

A general linear programming statement

- The linear constraints for the LPP

$$
\begin{aligned}
&\overline{\boldsymbol{\mu}}[Active] = 4 + \boldsymbol{\sigma}[e\_e\_a] + \boldsymbol{\sigma}[e\_o\_a] - \boldsymbol{\sigma}[r\_e\_a] - \boldsymbol{\sigma}[b\_o\_a];\\
&\overline{\boldsymbol{\mu}}[Memory] = 4 + \boldsymbol{\sigma}[e\_e\_a] - \boldsymbol{\sigma}[b\_e\_a];\\
&\overline{\boldsymbol{\mu}}[OwnMemAcc] = \boldsymbol{\sigma}[b\_o\_a] - \boldsymbol{\sigma}[e\_o\_a];\\
&\overline{\boldsymbol{\mu}}[Queue] = \boldsymbol{\sigma}[r\_e\_a] - \boldsymbol{\sigma}[b\_e\_a];\\
&\overline{\boldsymbol{\mu}}[Choice] = \boldsymbol{\sigma}[b\_e\_a] - \boldsymbol{\sigma}[c\_m];\\
&\overline{\boldsymbol{\mu}}[ExtMemAcc] = \boldsymbol{\sigma}[c\_m] - \boldsymbol{\sigma}[e\_e\_a];\\
&\overline{\boldsymbol{\mu}}[ExtBus] = 1 + \boldsymbol{\sigma}[e\_e\_a] - \boldsymbol{\sigma}[b\_e\_a];\\
&\boldsymbol{\chi}[e\_e\_a] + \boldsymbol{\chi}[e\_o\_a] = \boldsymbol{\chi}[r\_e\_a] + \boldsymbol{\chi}[b\_o\_a];\\
&\boldsymbol{\chi}[b\_e\_a] = \boldsymbol{\chi}[c\_m] = \boldsymbol{\chi}[e\_e\_a] = \boldsymbol{\chi}[r\_e\_a];\\
&\boldsymbol{\chi}[b\_o\_a] = \boldsymbol{\chi}[r\_e\_a];\\
&\boldsymbol{\chi}[b\_o\_a]\ \overline{\mathrm{s}}[b\_o\_a] = \overline{\boldsymbol{\mu}}[Active]/2;\\
&\boldsymbol{\chi}[r\_e\_a]\ \overline{\mathrm{s}}[r\_e\_a] = \overline{\boldsymbol{\mu}}[Active]/2;\\
&\boldsymbol{\chi}[e\_e\_a]\ \overline{\mathrm{s}}[e\_e\_a] = \overline{\boldsymbol{\mu}}[ExtMemAcc];\\
&\boldsymbol{\chi}[e\_o\_a]\ \overline{\mathrm{s}}[e\_o\_a] \le \overline{\boldsymbol{\mu}}[OwnMemAcc];\\
&\boldsymbol{\chi}[e\_o\_a]\ \overline{\mathrm{s}}[e\_o\_a] \le \overline{\boldsymbol{\mu}}[Memory];\\
&\boldsymbol{\chi}[e\_o\_a]\ \overline{\mathrm{s}}[e\_o\_a] \ge \overline{\boldsymbol{\mu}}[OwnMemAcc] + \tfrac{\mathrm{b}[OwnMemAcc]}{\mathrm{b}[Memory]}\overline{\boldsymbol{\mu}}[Memory]\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad -\mathrm{b}[Memory];\\
&4\left(\overline{\boldsymbol{\mu}}[ExtBus] - \mathrm{b}[ExtBus]\left(1 - \frac{\overline{\boldsymbol{\mu}}[Memory]}{\mathrm{b}[Memory]}\right)\right) \le 0;\\
&4\left(\overline{\boldsymbol{\mu}}[ExtBus] - \mathrm{b}[ExtBus]\left(1 - \frac{\overline{\boldsymbol{\mu}}[Queue]}{\mathrm{b}[Queue]}\right)\right) \le 0;
\end{aligned}
$$

A general linear programming statement

- The "automatic" results:

$$\frac{8}{11} \leq \chi[e\_e\_a] \leq 2$$

The exact solution with exponential distribution would be

$$\chi[e\_e\_a] = 1.71999$$

Improving of lower bound with more "ad hoc" constraints:

$$\overline{\mu}[Choice] = 0; \ \mathrm{b}[Choice] = 0; \ \mathrm{b}[Queue] = 3$$

$$4\left(\overline{\boldsymbol{\mu}}[ExtBus] + \frac{\mathbf{b}[ExtBus]}{\mathbf{b}[Queue]}\overline{\boldsymbol{\mu}}[Queue] - \mathbf{b}[ExtBus]\right) \leq 0$$

The improved bound:

$$1 \leq \chi[e\_e\_a] \leq 2$$

Performance modelling and evaluation

13. Approximate analysis of models

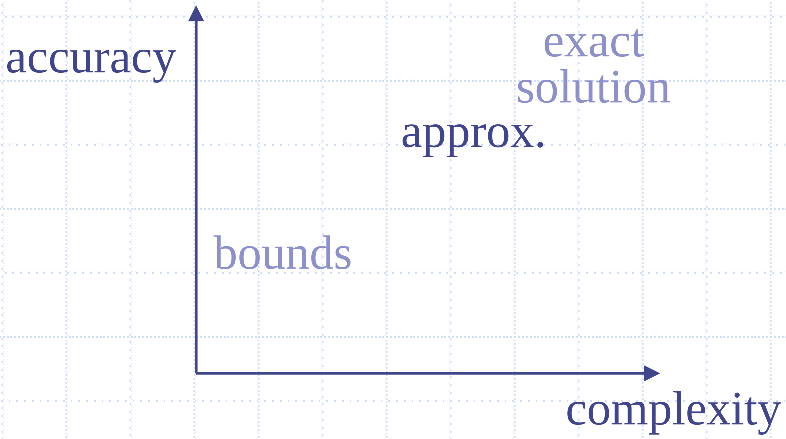
Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España
jcampos@unizar.es

Outline

- Decomposition of models
- Flow equivalent aggregation
- Iterative algorithm: marked graphs case
- Iterative algorithm: general case

Decomposition of models

- Interest of approximation techniques



Decomposition of models

- **Basic idea:**

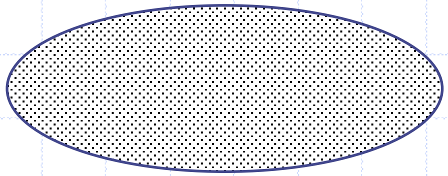
  reduce the complexity of the analysis of a complex system

- **when**

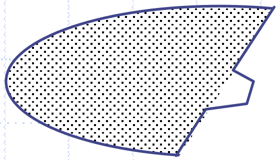
  - the system is too complex/big to be solved by any exact analytical technique
  - a simulation is too long (essentially if many different configurations must be tested or it must be included in some optimization procedure)
  - some insights about the internal behaviour of subsystems are wanted (writing equations might help)

Decomposition of models

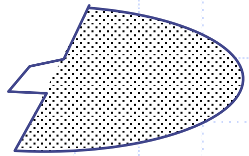
- Principle:
  - decompose the system into some subsystems



original system
state space size: n



two subsystems
state space size of each: $n/10$
(for example)
(i.e., one order of magnitud less)



  - reduce the analysis of the whole system by those of the subsystems in isolation

    if the solution technique was, e.g., $O(n^3)$ on the state space size n, the cost of solving the isolated subsystems would be $O(n^3/1000)$, i.e. three orders of magnitud less…

Decomposition of models

- Advantages:
  - drastical reduction of complexity and computational requirements
  - enables to extend the class of system that can be solved by analytical techniques
- Problems and limitations
  - Decomposition is not easy!
    - "net-driven" means to use structural information of the net model to assure that "good" qualitative properties are preserved in the isolated subsystems (e.g., liveness, boundedness…)
  - Approximation is not exact!
    - problem of error estimation or at least bounding the error
  - Accurate techniques are usually very especific to particular problems ➔ need of expertise to select the adequate technique…

Decomposition of models

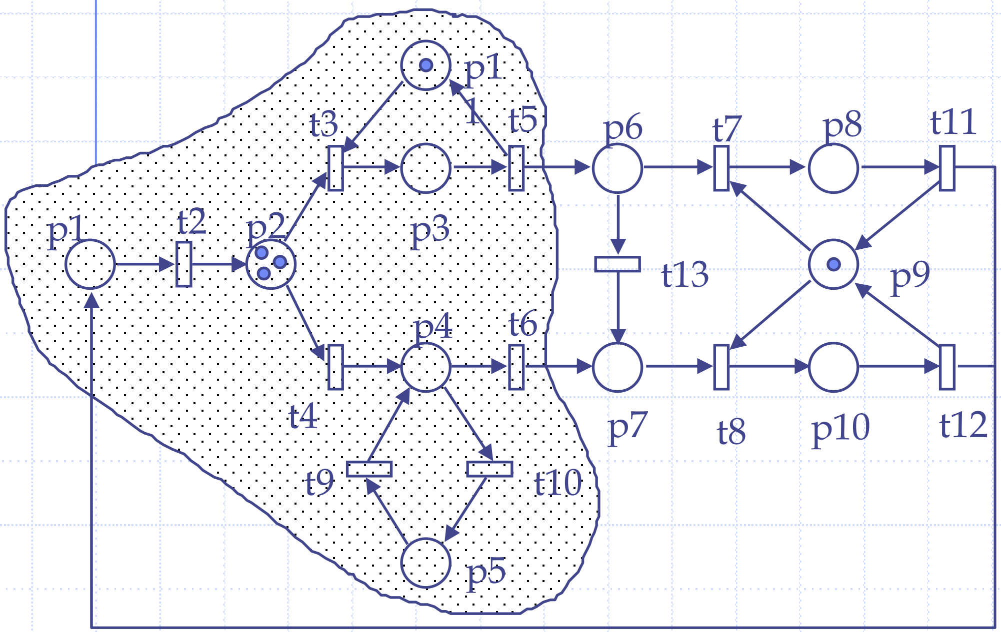
- **Steps** in an approximation technique based on decomposition:
  - Partition of the system into subsystems:
    - definition of rules for decomposition
    - consideration of functional properties that must/can be preserved
  - Characterization of subsystems in isolation:
    - definition of unknowns and variables
    - decisions related with consideration of mean variables or higher order moments of involved random variables
    - consideration or not of the "outside world"
    - need of a skeleton (high level view of the model) and characteristics considered in it
  - Estimation of the unknown parameters:
    - writing equations among unknowns
    - direct or iterative technique (in this case, definition of fixed point equations)
    - considerations on existence and uniqueness of solution
    - computational algorithm for solving the fixed point equation (implementation aspects, convergence aspects)

Outline

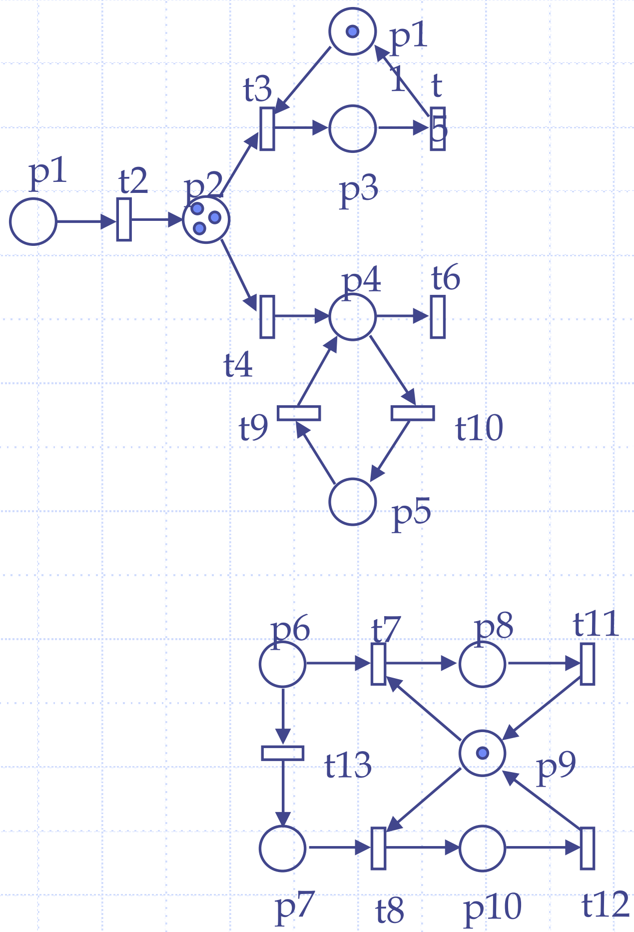
- Decomposition of models
- Flow equivalent aggregation
- Iterative algorithm: marked graphs case
- Iterative algorithm: general case

Flow equivalent aggregation

- The system:



- Partition:

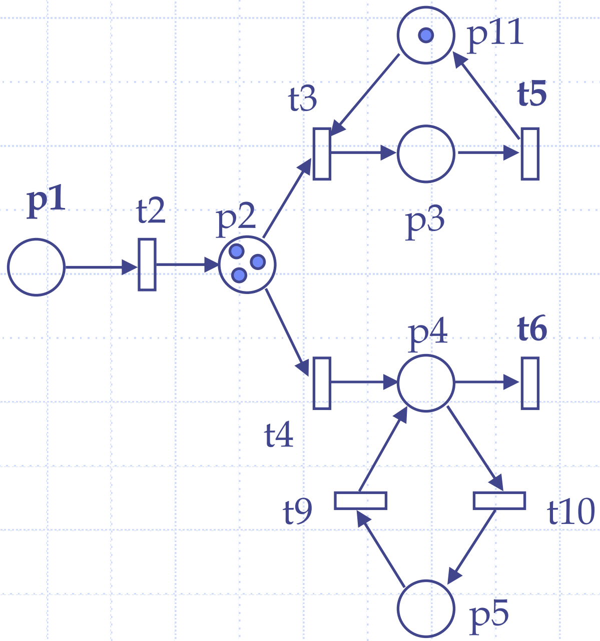


Flow equivalent aggregation

- Characterization of subsystems. Behaviour is characterized by:
  - path a token takes in the PN (what percetage leave through t5 and t6)
  - time it takes a token to be discharged

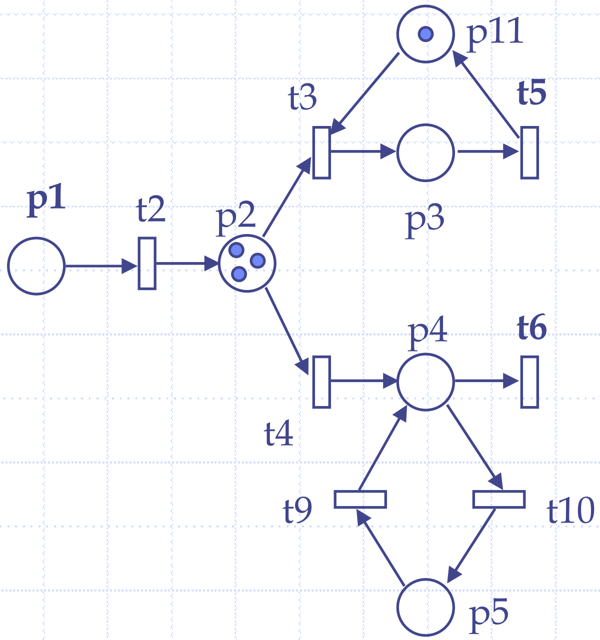
•way-in places: p1

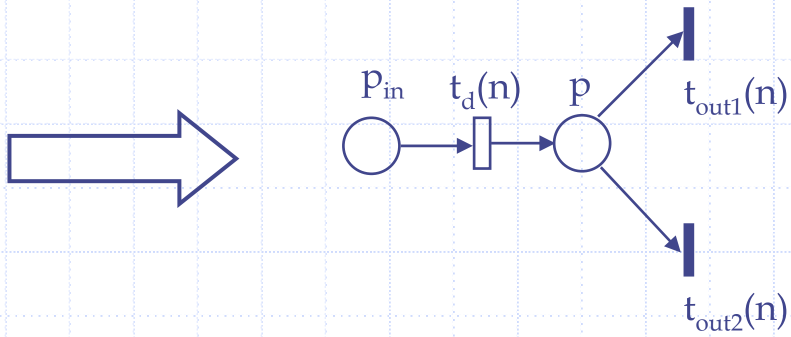
•sink transitions: t5, t6



Flow equivalent aggregation

- Reduction of the subsystem:



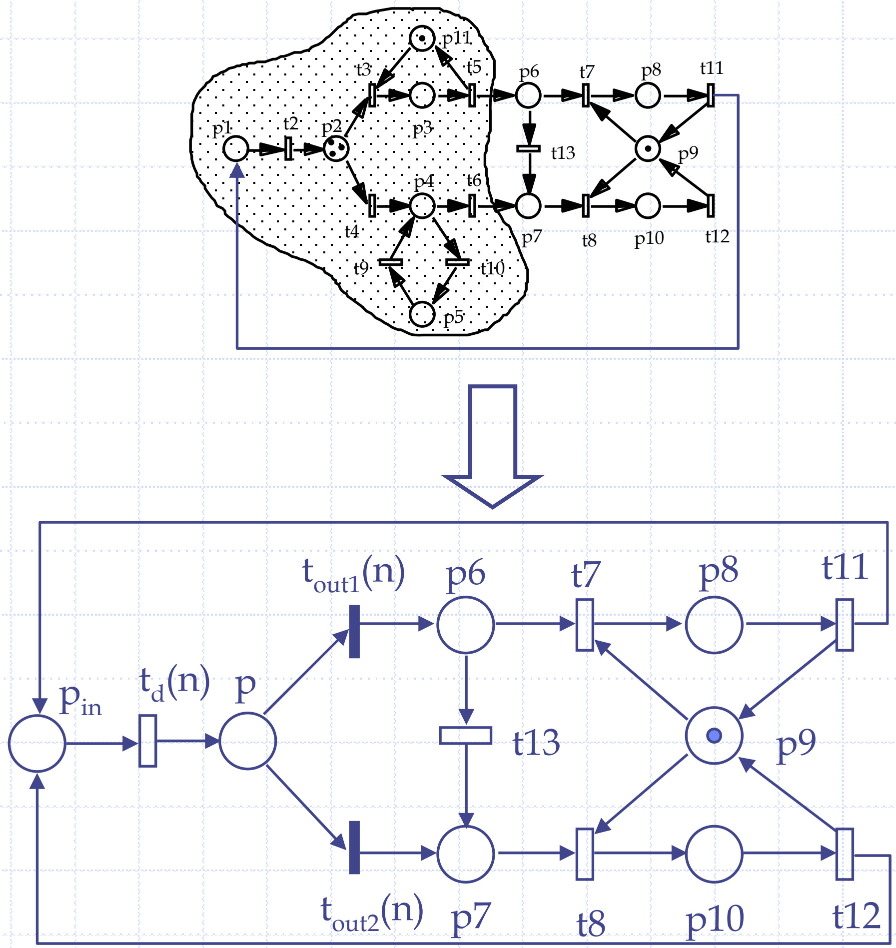


- routing rates of $t_{out1}(n)$ and $t_{out2}(n)$?
- service rate of $t_d(n)$?

(marking dependent: $n=M(p_{in})$)

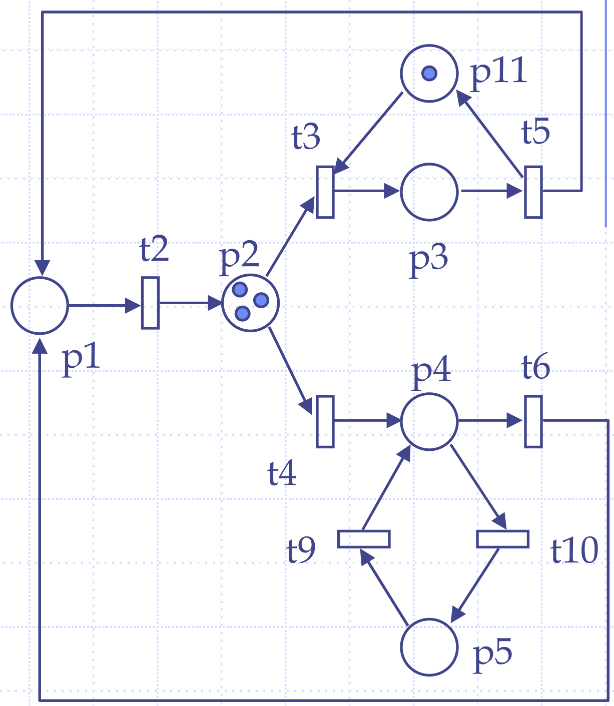
Flow equivalent aggregation

- Aggregated system:



Flow equivalent aggregation

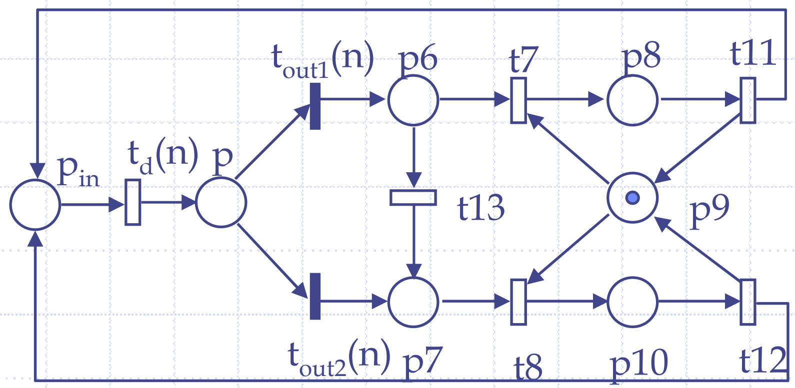
- Estimation of the unknown parameters:
  - Analyze the subnet in isolation with constant number of tokens
    - delay and routing are dependent on the number of tokens in the system
    - compute delay and routing for all possible populations



Parameters of the subsystem in isolation			
# tokens	v_5	v_6	thrput
1	0.500	0.500	0.400
2	0.431	0.569	0.640
3	0.403	0.597	0.780
4	0.389	0.611	0.863
5	0.382	0.618	0.914

Flow equivalent aggregation

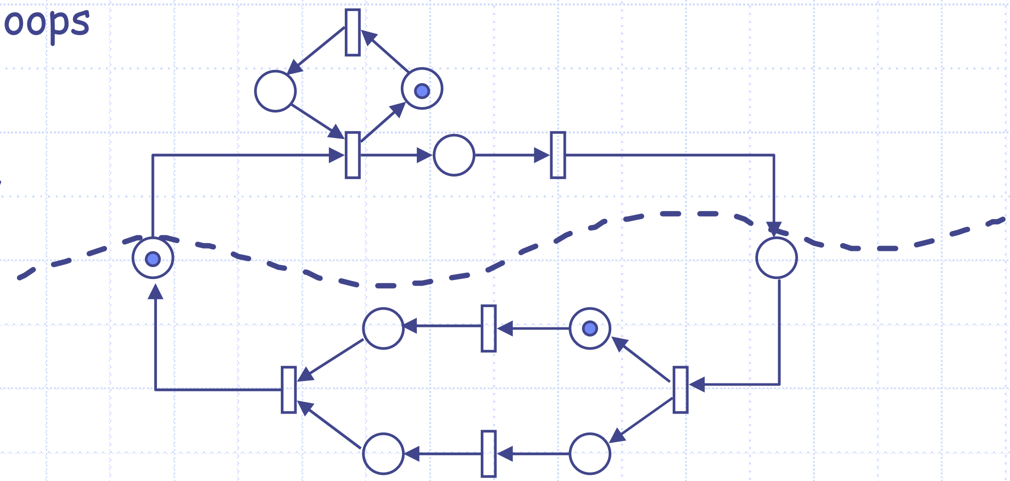
- When the subnet is substituted back, routing and delay are going to be state dependent ($n=M(p_{in})$)



Comparison of State Spaces & throughput					
#tokens	# states		throughput		%error
	aggregat	original	aggregat	original	
1	5	9	0.232	0.232	0.00
2	12	41	0.381	0.384	0.78
3	22	131	0.470	0.474	0.84
4	35	336	0.521	0.523	0.38
5	51	742	0.548	0.547	<0.10

Flow equivalent aggregation

- Limitations:
  - **Assumption**: the service time depends only on the number of customers which are currently present in the subsystem.
    - The behaviour of the subsystem is assumed independent of the arrival process
  - It is exact for product-form queueing networks.
  - The error is small if in the original model:
    - the arrivals to the subsystem are "close" to Poisson arrivals and
    - the processing times are approximately exponential
  - On the other hand, the error can be very large if
    - there exist internal loops in a subnet, or
    - there exist trapped tokens in a fork-join, or...



Outline

- Decomposition of models
- Flow equivalent aggregation
- Iterative algorithm: marked graphs case
- Iterative algorithm: general case

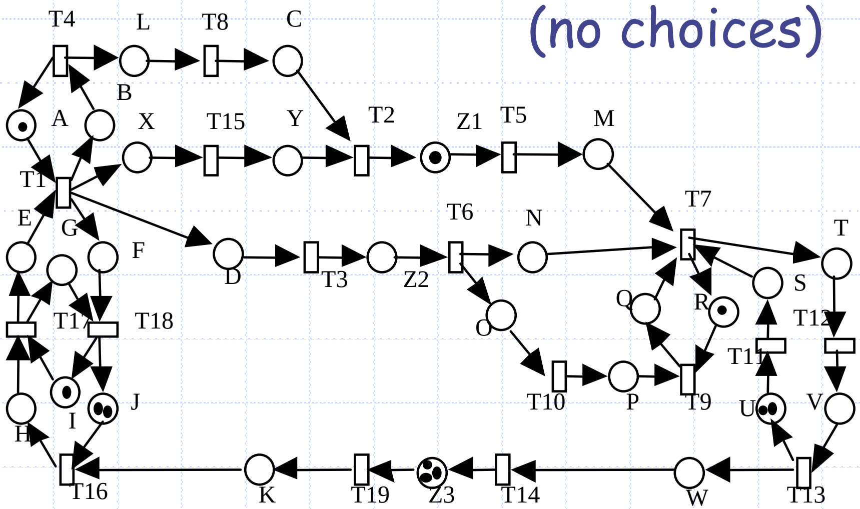
Iterative algorithm: marked graphs case

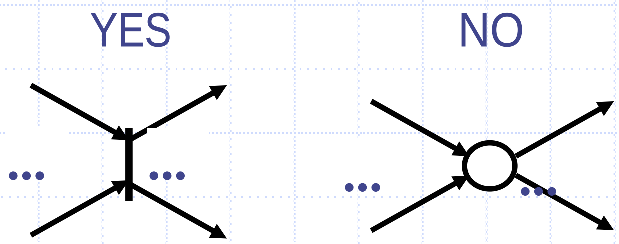
- Net-driven solution techniques
  - stressing the intimate relationship between qualitative and quantitative aspects of PN's
  - structure theory of net models

    → efficient computation techniques

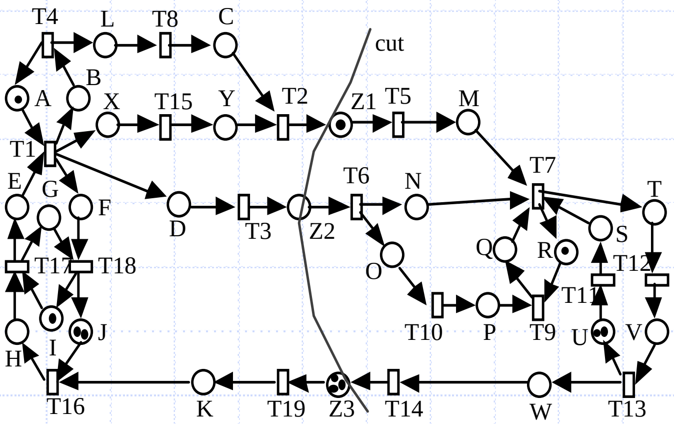
- Marked graphs: subclass of *ordinary* nets

  (no choices) (no weights)

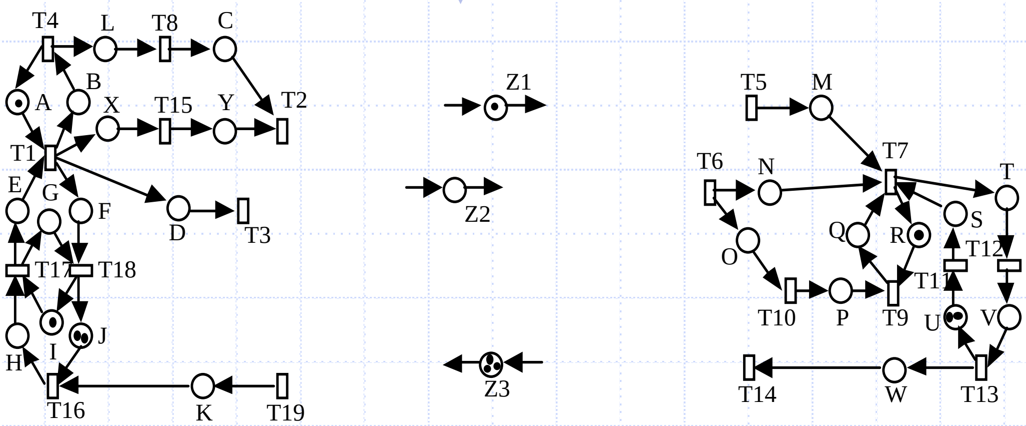




Iterative algorithm: marked graphs case

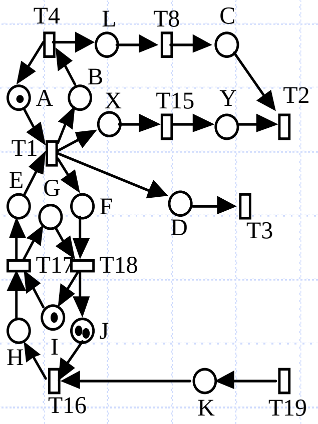


original model + definition of cut



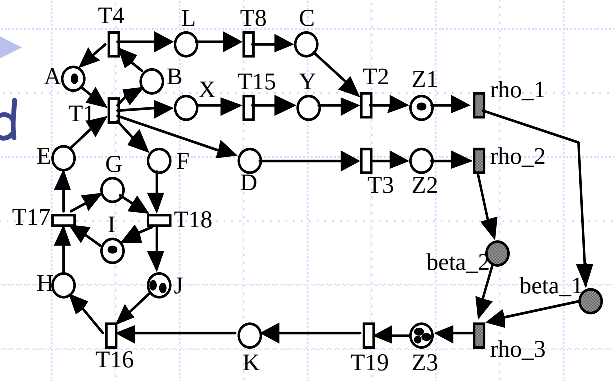
partition of the model into **modules** (subnets) connected through **buffers** (places)

Iterative algorithm: marked graphs case

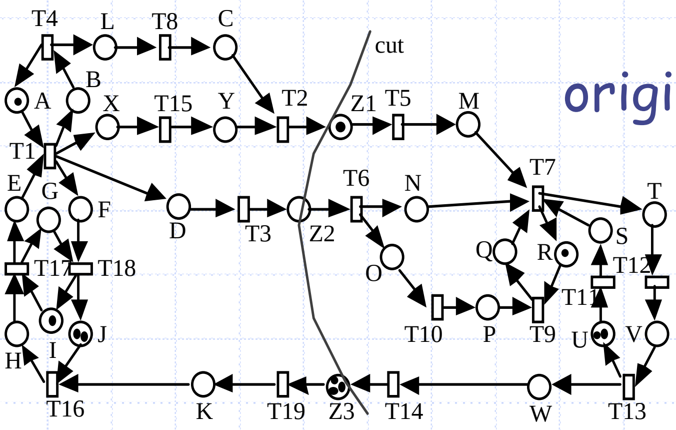


the solution of isolated modules is difficult and useless: (in this case) they are unbounded!

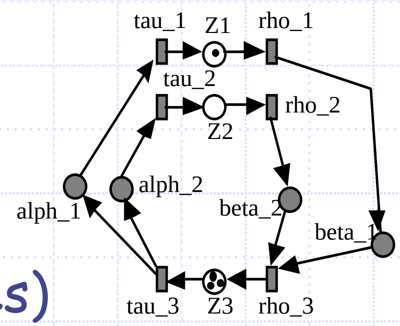
the modules must be complemented with an abstract view of the rest; **components** are obtained



Iterative algorithm: marked graphs case

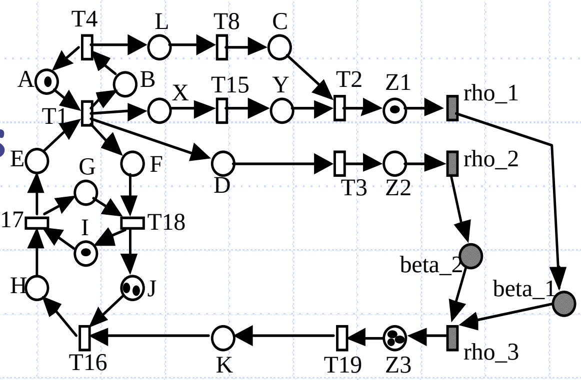


original model (89358 states)

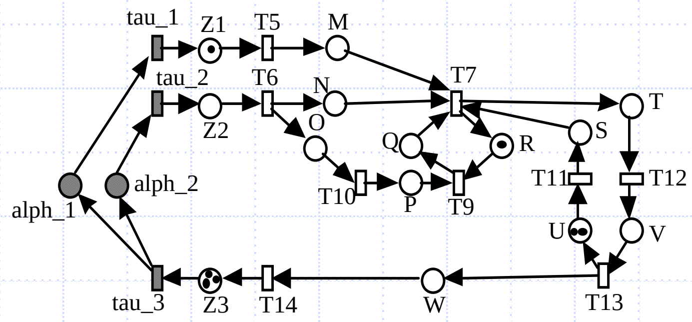


BS (231 states)

AS_1 (8288 states)



AS_2 (3440 states)



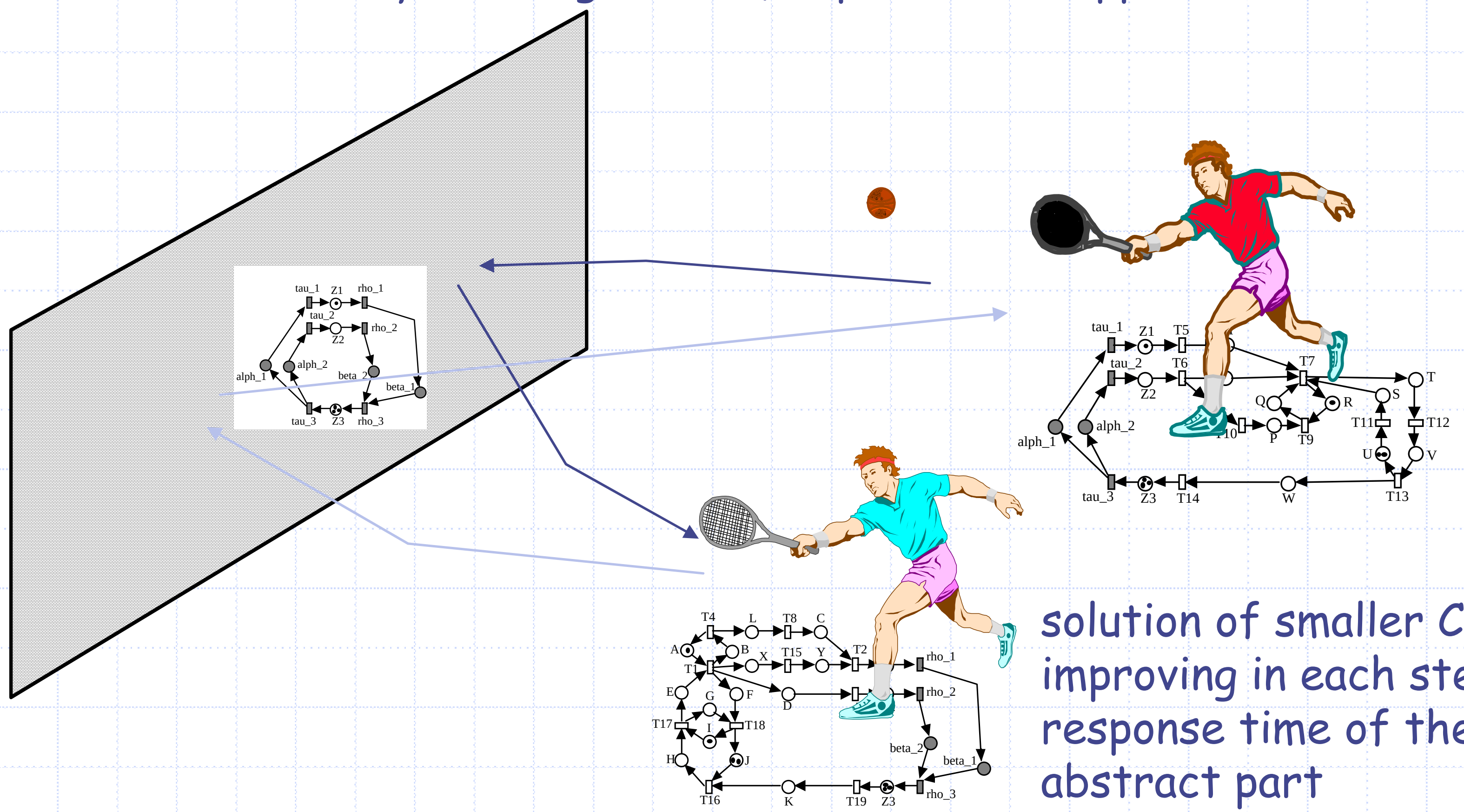
three components:

aggregated systems (low level views)

and **basic skeleton** (high level view)

Iterative algorithm: marked graphs case

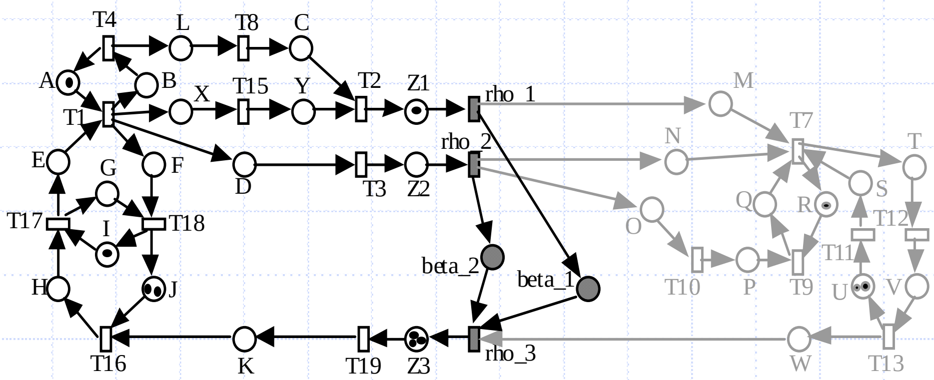
iterative solution: *pelota* algorithm (response time approximation technique)



solution of smaller CTMC's, improving in each step the response time of the abstract part

Iterative algorithm: marked graphs case

- Substitute a subnet by a set of places



  - interface transitions (input/ouput of buffers) are preserved
  - add one place from each input to each output transition
  - the set of new places can be superposed in the original model preserving the behaviour: **implicit places**

Iterative algorithm: marked graphs case

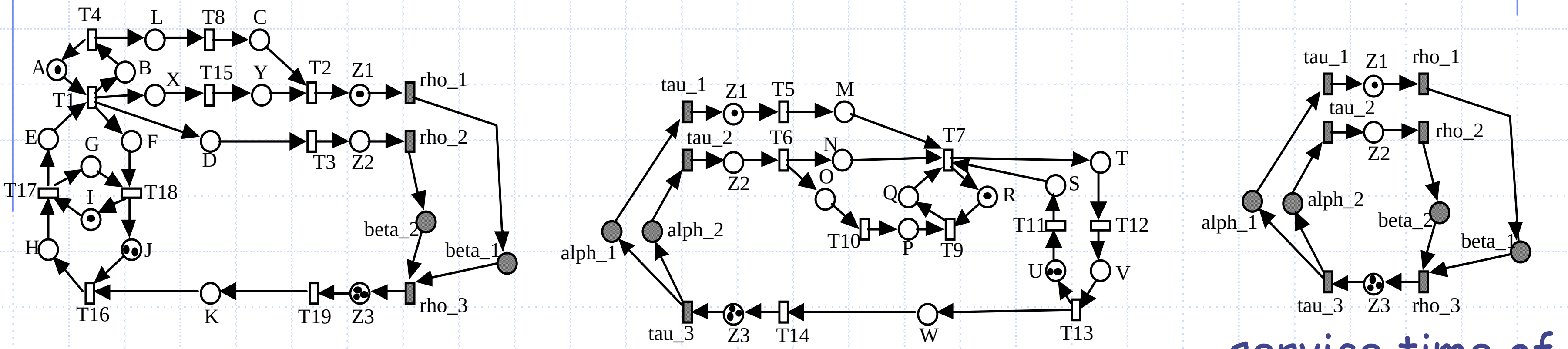
- Compute the initial marking of new places
  - minimum initial marking to make them implicit
  - computed using Floyd's *all-pairs shortest paths algorithm*:
    - the MG is considered as a weighted graph (transitions are vertices and the initial marking of places are the weigths of the arcs)

Iterative algorithm: marked graphs case

- The abstract view has "very good quality":
  - the language of firing sequences of the aggregated system is equal to that of the original system projected on the preserved transitions
  - the reachability graph of the aggregated system is isomorphous to that of the original system projected on the preserved places

Iterative algorithm: marked graphs case

❑ Definition of unknowns:

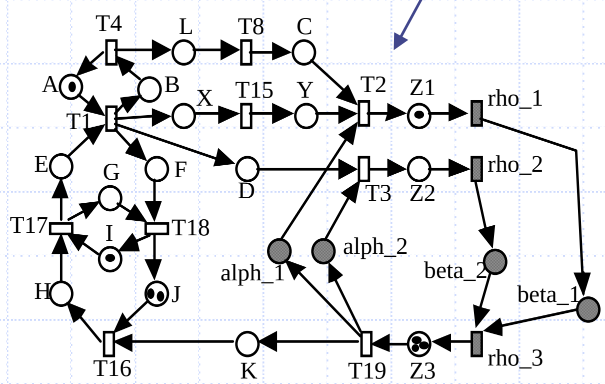


service time of *rho_i* | service time of *tau_j* | service time of *rho_i* and *tau_j*

+ throughput of each system
+ response time of interface transitions at each system

Iterative algorithm: marked graphs case

first aggregated system



response time approximation of the **left hand subnet** for a token that

exits through T2: $R_2 = \overline{\mu}[alph\_1] / \chi[t_2]$ (Little's law)

exits through T3: $R_3 = \overline{\mu}[alph\_2] / \chi[t_3]$

$(\chi[t_2] = \chi[t_3] = \chi)$

thus, solve the CTMC and compute: R_2, R_3 and also χ

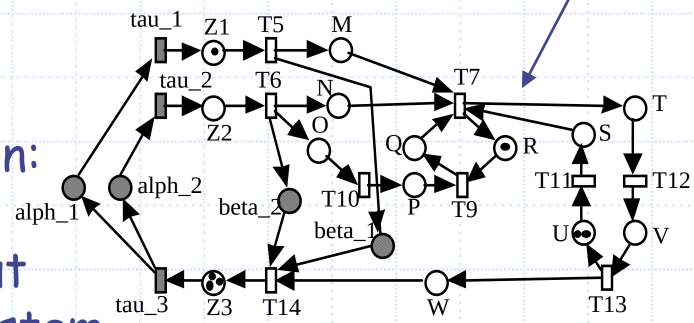
Iterative algorithm: marked graphs case

select *tau_1* and *tau_2* as:

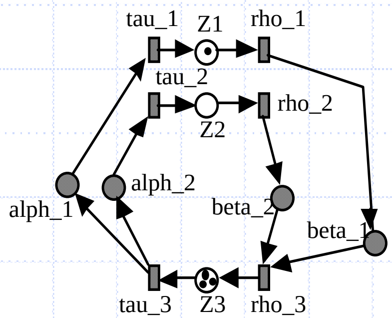
$$tau\_1 = f.R_2$$
$$tau\_2 = f.R_3$$

where *f* is computed using the skeleton: linear search until the throughput of the skeleton is equal to the throughput computed for the first aggregated system

second aggregated system



$$tau\_1 = f.R_2$$
$$tau\_2 = f.R_3$$



skeleton

Iterative algorithm: marked graphs case

The algorithm:

$$
\begin{array}{l}
\texttt{select a cut Q;}\\
\texttt{derive aggregated systems } AS_1, AS_2 \texttt{ and skeleton BS;}\\
\texttt{give initial value } \mu_t^{(0)} \texttt{ for each } t \in T_{I2};\\
\texttt{k:=0; \{counter for iteration steps\}}\\
\texttt{repeat}\\
\quad \texttt{k:=k+1;}\\
\quad \texttt{solve aggregated system } AS_1 \texttt{ with}\\
\qquad \texttt{input: } \mu_t^{(k-1)} \texttt{ for each } t \in T_{I2},\\
\qquad \texttt{output: ratios among } \mu_t^{(k)} \texttt{ of } t \in T_{I1}, \texttt{ and } X_1^{(k)};\\
\quad \texttt{solve basic skeleton BS with}\\
\qquad \texttt{input: } \mu_t^{(k-1)} \texttt{ for each } t \in T_{I2},\\
\qquad \qquad \texttt{ratios among } \mu_t^{(k)} \texttt{ of } t \in T_{I1}, \texttt{ and } X_1^{(k)},\\
\qquad \texttt{output: scale factor of } \mu_t^{(k)} \texttt{ of } t \in T_{I1};\\
\quad \texttt{solve aggregated system } AS_2 \texttt{ with}\\
\qquad \texttt{input: } \mu_t^{(k-1)} \texttt{ for each } t \in T_{I1},\\
\qquad \texttt{output: ratios among } \mu_t^{(k)} \texttt{ of } t \in T_{I2}, \texttt{ and } X_2^{(k)};\\
\quad \texttt{solve basic skeleton BS with}\\
\qquad \texttt{input: } \mu_t^{(k)} \texttt{ for each } t \in T_{I1},\\
\qquad \qquad \texttt{ratios among } \mu_t^{(k)} \texttt{ of } t \in T_{I2}, \texttt{ and } X_2^{(k)},\\
\qquad \texttt{output: scale factor of } \mu_t^{(k)} \texttt{ of } t \in T_{I2};\\
\texttt{until convergence of } X_1^{(k)} \texttt{ and } X_2^{(k)};
\end{array}
$$

Iterative algorithm: marked graphs case

Service rates (arbitrary):
T2=0.2; T4=0.7; T6=0.3; T8=0.8; T9=0.6; T10=0.5;
Ti=1.0, i=1,3,5,7,11,12,13,14,15,16,17,18,19

Throughput of the original system: 0.138341
State space of the original system: 89358

Results using the approximation technique:
State space AS1: 8288; State space AS2: 3440; State space BS: 231

AS1				AS2			
X1	tau_1	tau_2	tau_3	X2	rho_1	rho_2	rho_3
0.17352	0.05170	0.16810	0.88873	0.12714	0.89026	0.21861	0.14354
0.14093	0.06265	0.19707	0.91895	0.13795	0.88267	0.21363	0.13509
0.13856	0.06325	0.19821	0.92054	0.13841	0.88239	0.21343	0.13467
0.13844	0.06328	0.19827	0.92062	0.13843	0.88237	0.21342	0.13465
0.13843	0.06328	0.19827	0.92064	0.13843	0.88238	0.21342	0.13465

Outline

- Decomposition of models
- Flow equivalent aggregation
- Iterative algorithm: marked graphs case
- Iterative algorithm: general case

Iterative algorithm: general case

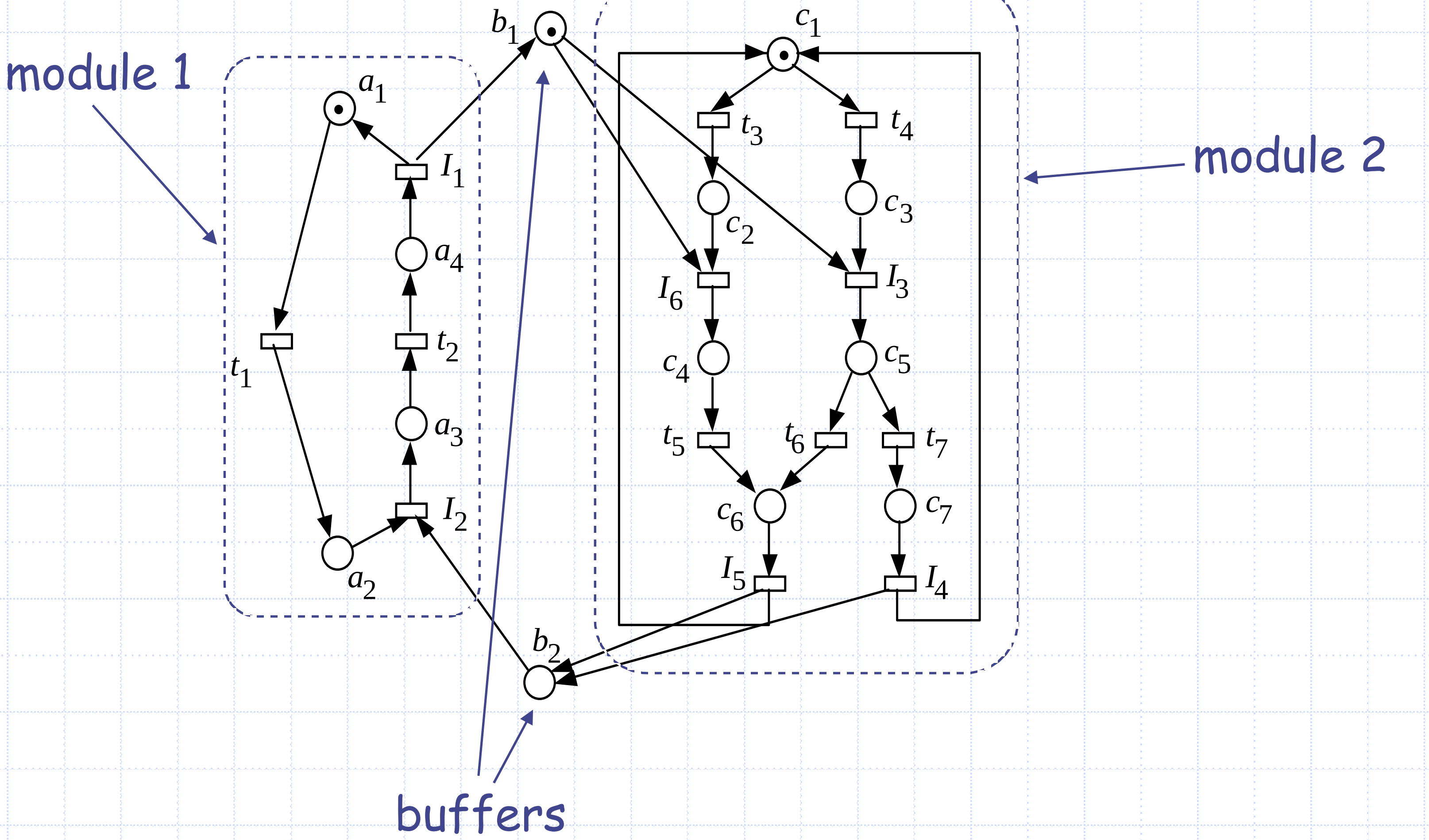
- Arbitrary *P/ T* system + **structured view**

partition into **modules** (functional units)
connected through places (**buffers**)

Iterative algorithm: general case

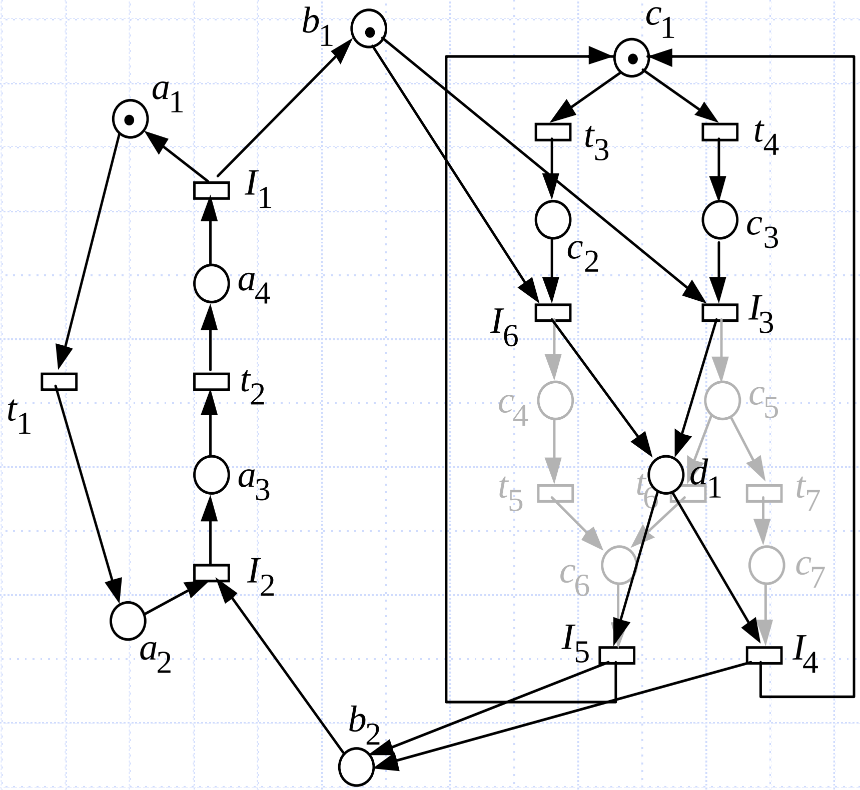
- All *P/T* systems have serveral structured views, varying between:
  - a single module (empty set of buffers)
  - as many modules as transitions (all places are considered as buffers)

Iterative algorithm: general case

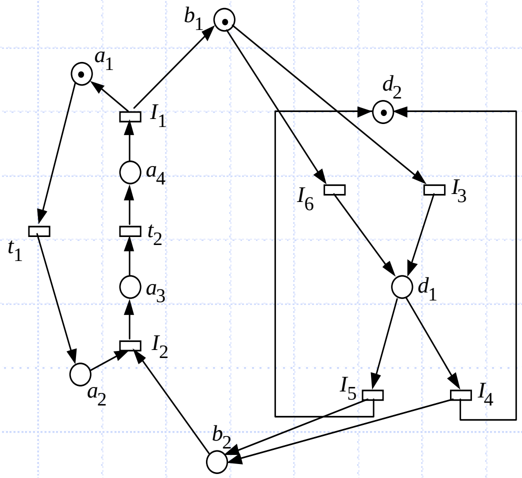


Iterative algorithm: general case

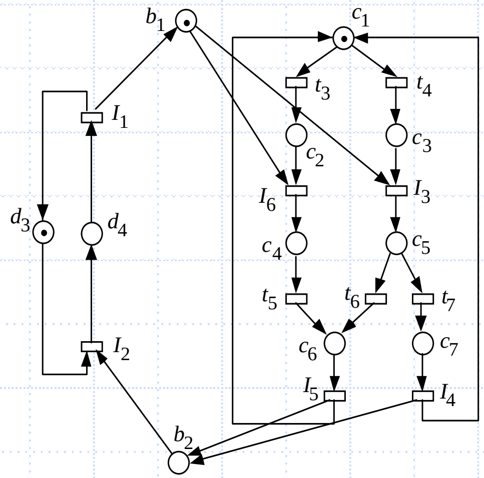
- Substitute a subnet by a set of implicit places
  - derived from minimal *P*-semiflows of the subnet (sum of the incidence rows of places)



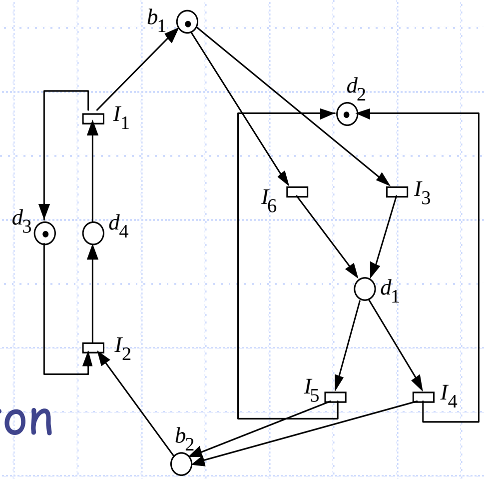
Iterative algorithm: general case



first aggregated system



second aggregated system



skeleton

Iterative algorithm: general case

- The quality of the abstract view is “not as good as” in the MG’s case
  - the language of firing sequences of the aggregated system **includes** that of the original system projected on the preserved transitions
  - the reachability graph of the aggregated system **includes** that of the original system projected on the preserved nodes

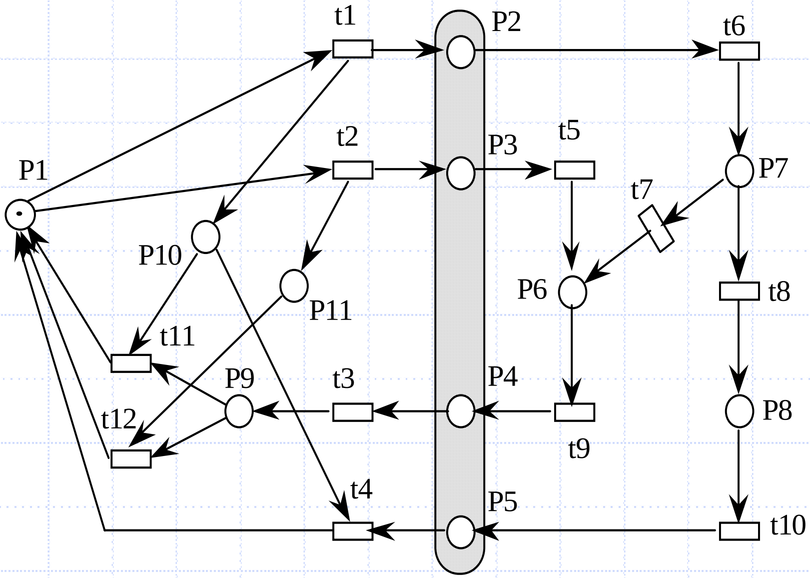
Iterative algorithm: general case

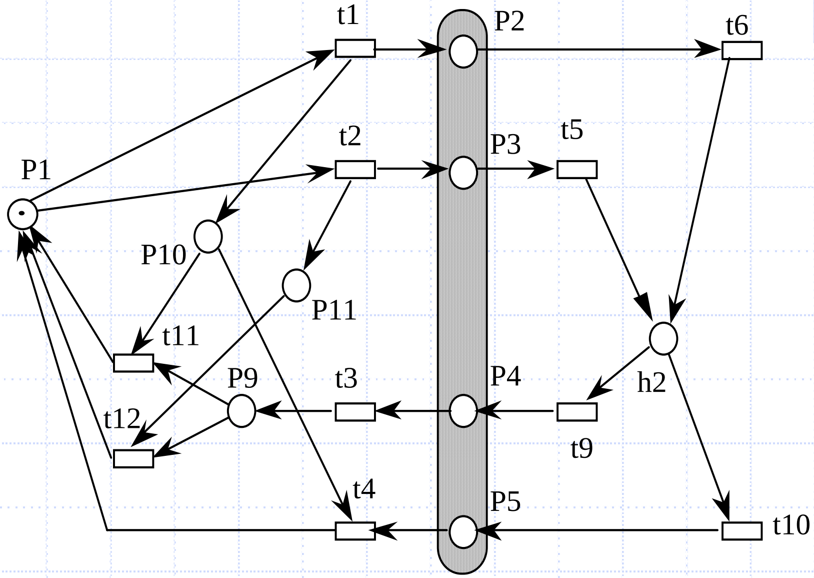
- Problems in the composition:

The RG of an aggregated system may include *spurious* markings and firing sequences that do not correspond to actual markings and firing sequences of the original system

we can obtain even **non-ergodic** systems (CTMC cannot be solved)

Iterative algorithm: general case





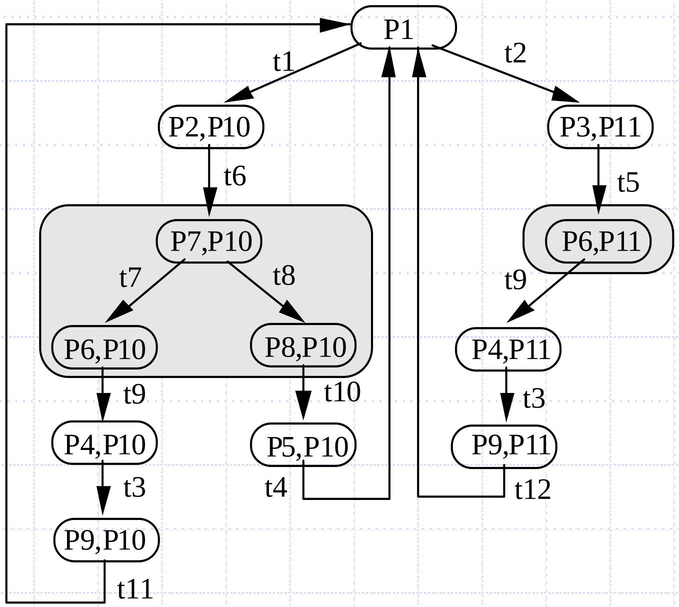
original system:
limited and reversible, thus ergodic

aggregated system:
it has a total deadlock

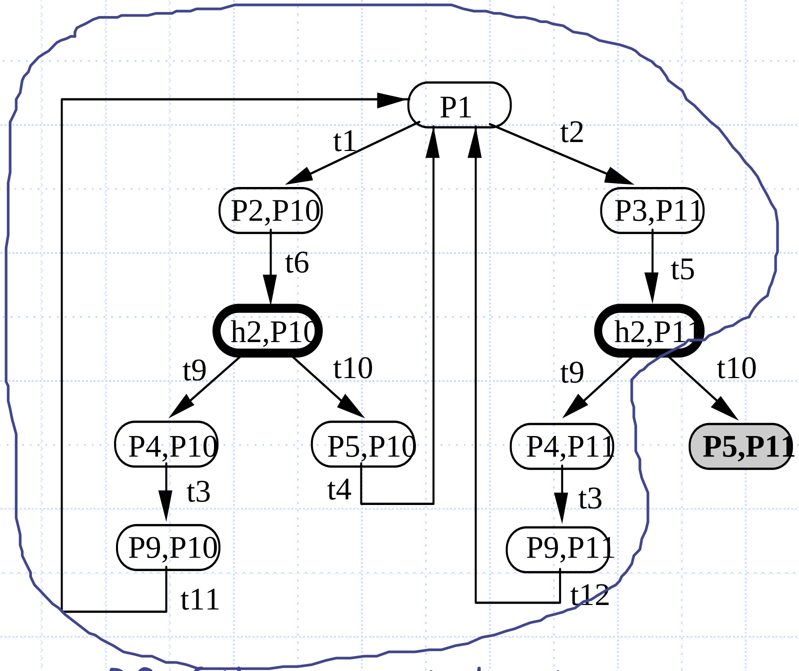
Iterative algorithm: general case

- Solution for the problem:

  select only the strongly connected component of the RG that includes the projection of the initial marking



RG of the original system



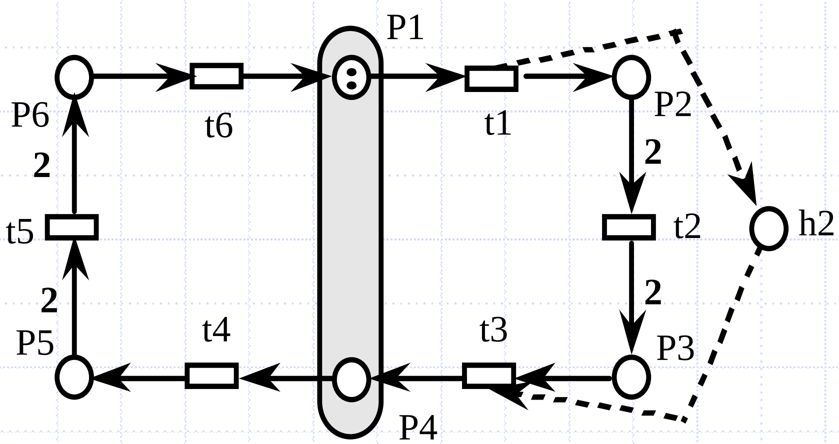
RG of the aggregated system

Iterative algorithm: general case

- More problems:

  Spurious markings (and/or firing seq.) may still be present,

  but the solution is possible!



Iterative algorithm: general case

- It is possible to eliminate all the spurious markings with additional computational effort
  - use a *Kronecker* expression of the infinitesimal generator of the original system
    - implement a depth-first search to build the RS
    - reduce the infinitesimal generators of the aggregated systems, using the information about reachability in the original system

- The whole reachability set must be derived but the CTMC is not solved (throughput is approximated from the solution of CTMC of subsystems)