Simulation and Modelling

Tony Field and Jeremy Bradley

`Room 372.  Email: jb@doc.ic.ac.uk`

Department of Computing, Imperial College London

Produced with prosper and LaTeX

Queues at Keil Ferry Terminal

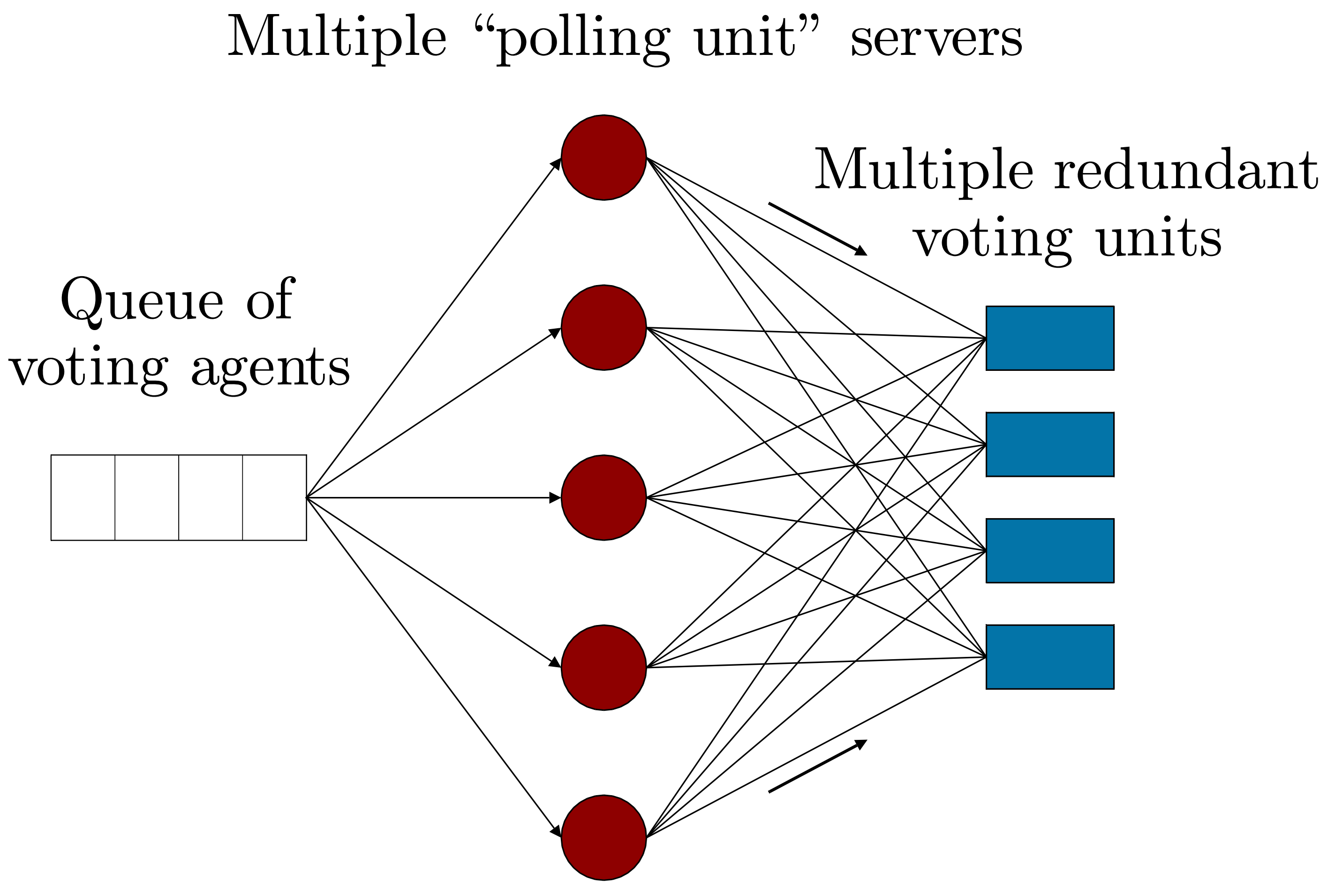
- 1 week time-lapse CCTV of the Keil ferry terminal (http://www.kielmonitor.de/)
- Multi-server queue with vacations and batch services

Quantitative modelling

- How many processors will we need to achieve throughput of 300Mbit s^{-1}?
- What is the percentage utilisatation of the upstream network link?
- What is the probability that a text message sent from mobile A to mobile B will take less than 5 seconds
- At time $t = 4$, what is the probability that the software is in a mutual exclusion lock?

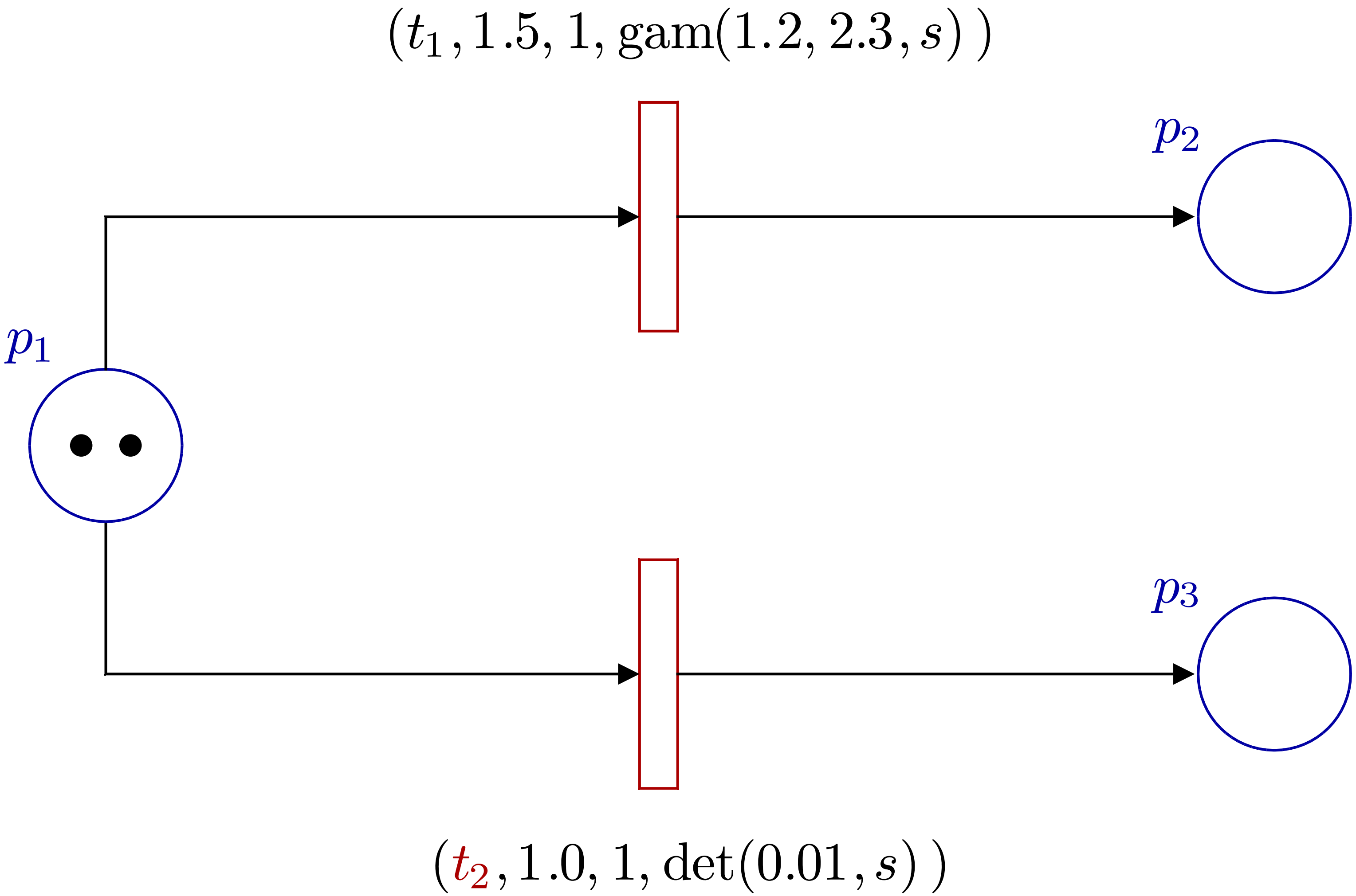
Available modelling languages

- Queueing networks:



Available modelling languages

- Stochastic Petri nets:



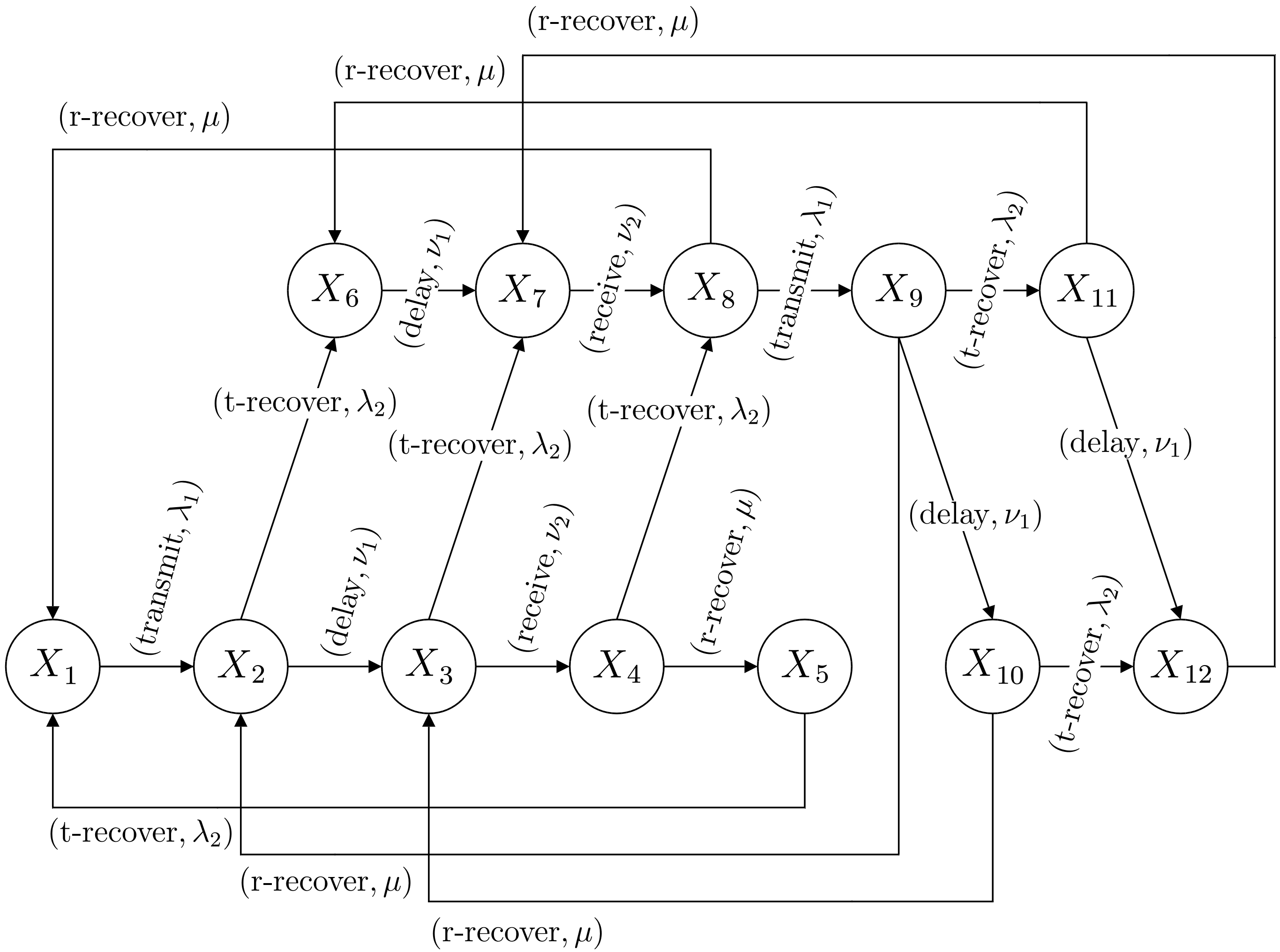
Available modelling languages

- Stochastic process algebras:

$$
\begin{aligned}
\mathrm{A1} &\stackrel{\text{def}}{=} (\mathrm{start}, r_1).\mathrm{A2} + (\mathrm{pause}, r_2).\mathrm{A3} \\
\mathrm{A2} &\stackrel{\text{def}}{=} (\mathrm{run}, r_3).\mathrm{A1} + (\mathrm{fail}, r_4).\mathrm{A3} \\
\mathrm{A3} &\stackrel{\text{def}}{=} (\mathrm{recover}, r_1).\mathrm{A1} \\
\mathrm{AA} &\stackrel{\text{def}}{=} (\mathrm{run}, \top).(\mathrm{alert}, r_5).\mathrm{AA} \\
\mathrm{Sys} &\stackrel{\text{def}}{=} \mathrm{AA} \underset{\{run\}}{\bowtie} \mathrm{A1}
\end{aligned}
$$

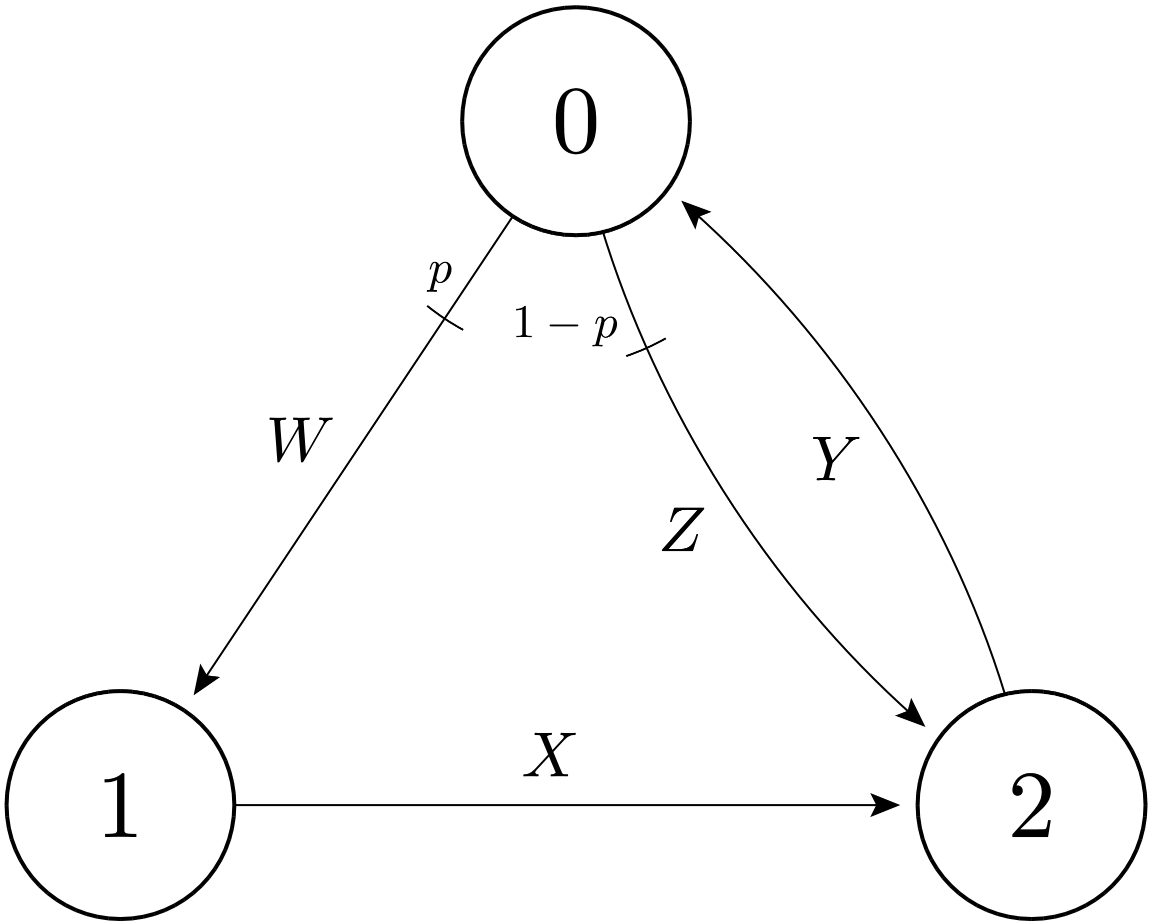
Available mathematical models

- Markov Chains:

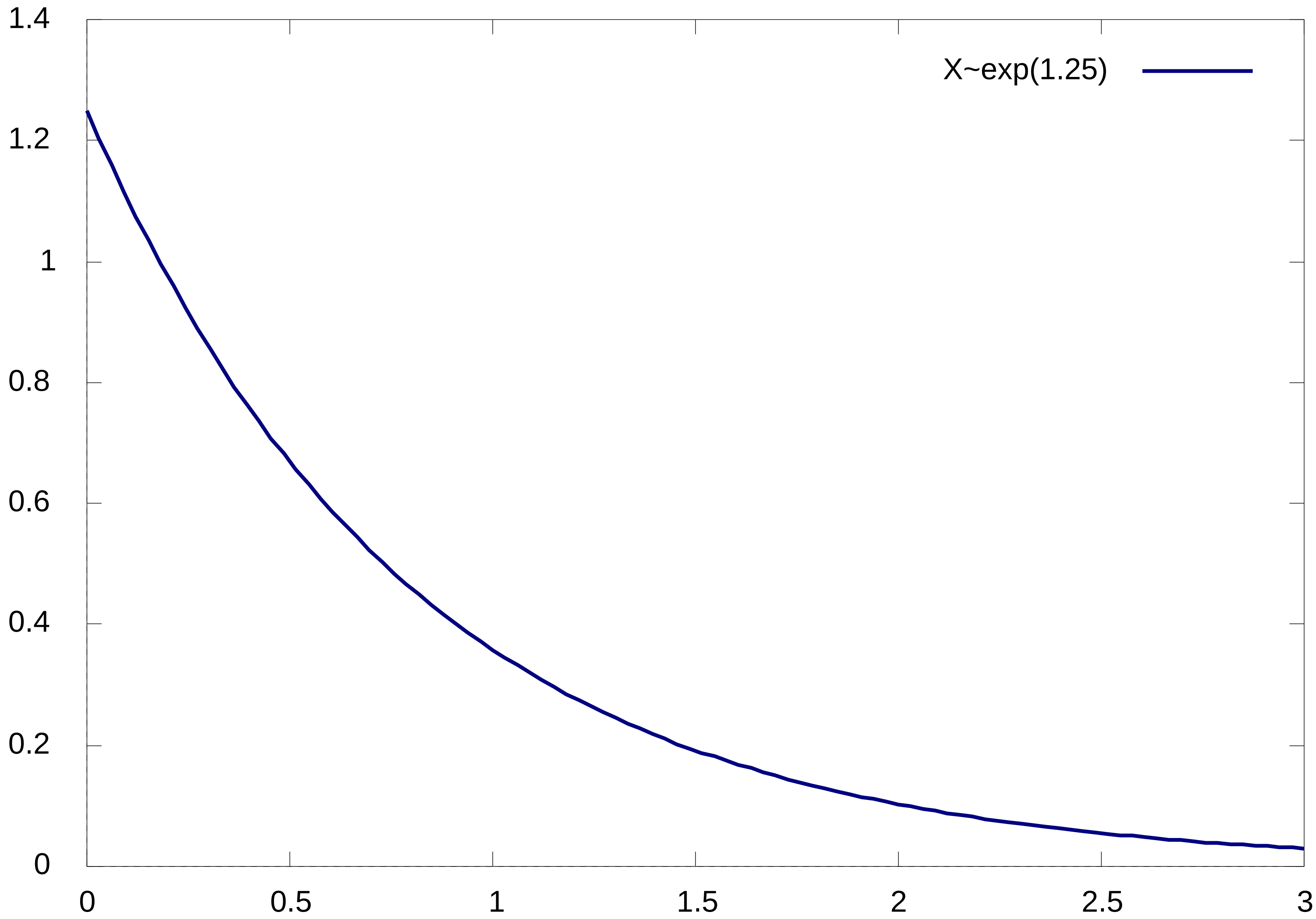


Available mathematical models

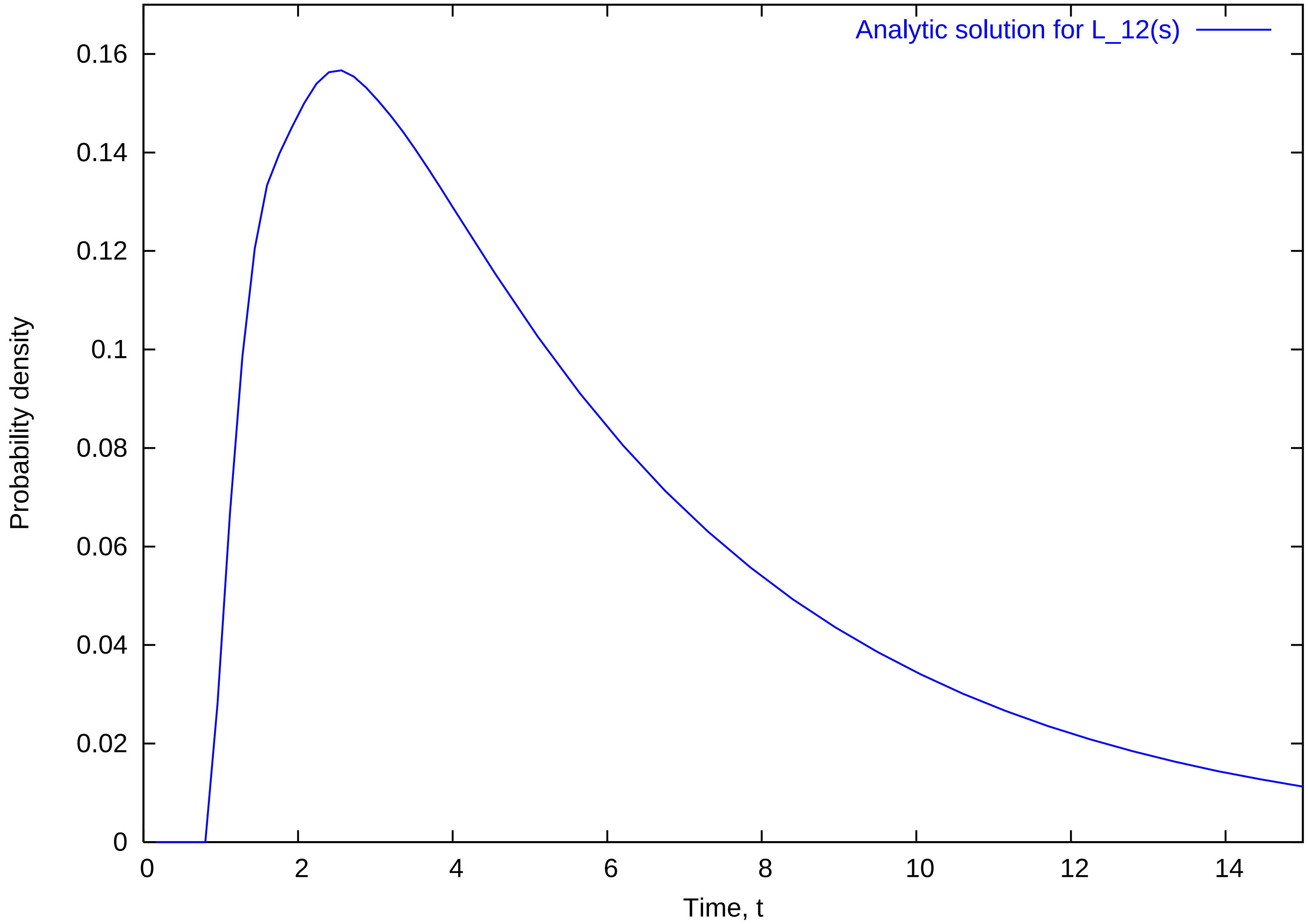
- Semi-Markov Chains:



An exponential distribution



A non-exponential distribution



An exponential distribution

- If $X \sim \exp(\lambda)$ then:
  - Probability density function (PDF)

$$f_X(t) = \lambda e^{-\lambda t}$$

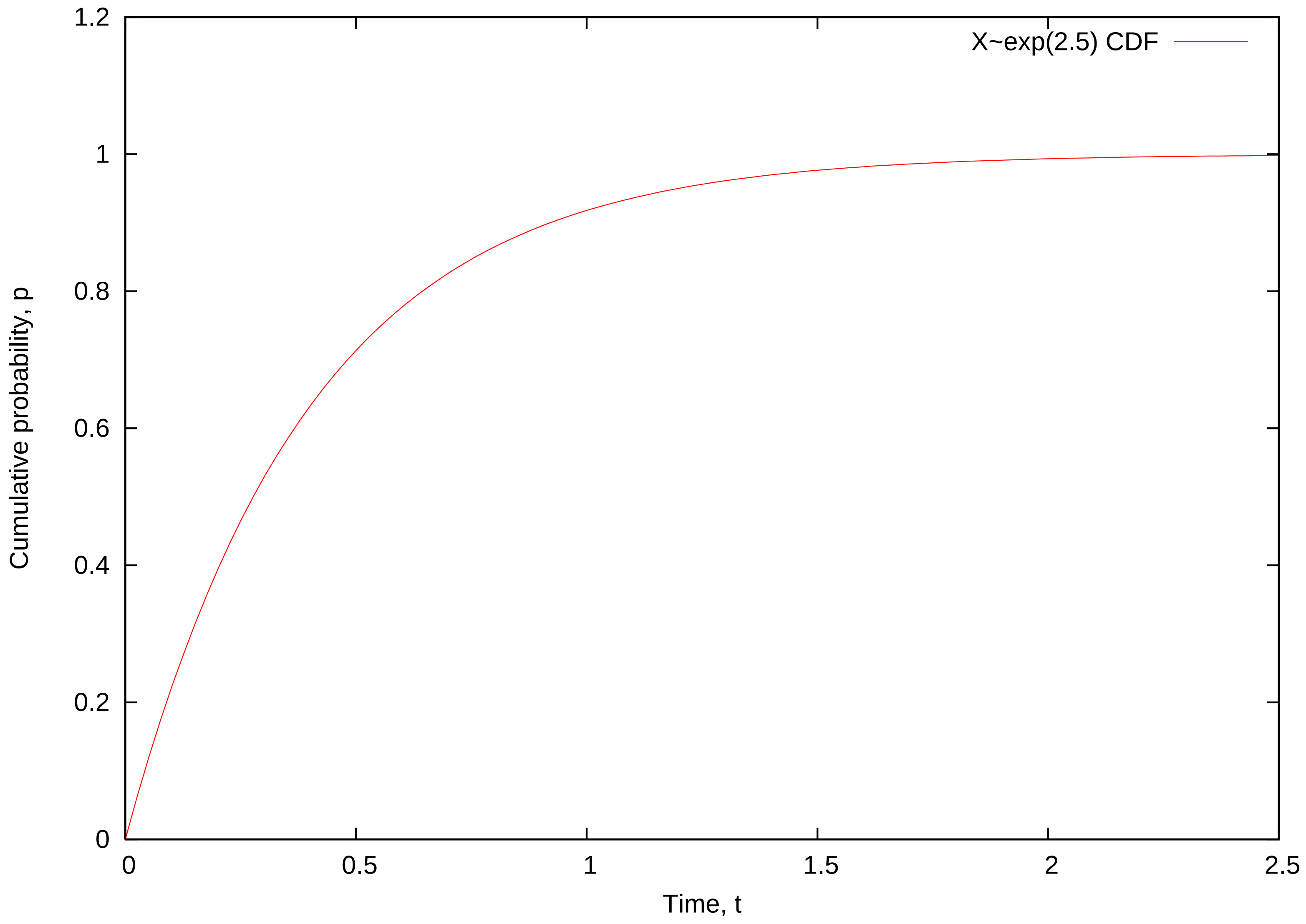
  - Cumulative density function (CDF)

$$F_X(t) = \mathbb{P}(X \leq t) = \int_0^t f_X(u)\,\mathrm{d}u = 1 - e^{-\lambda t}$$

  - Laplace transform of PDF

$$L_X(s) = \frac{\lambda}{\lambda + s}$$

An exponential CDF

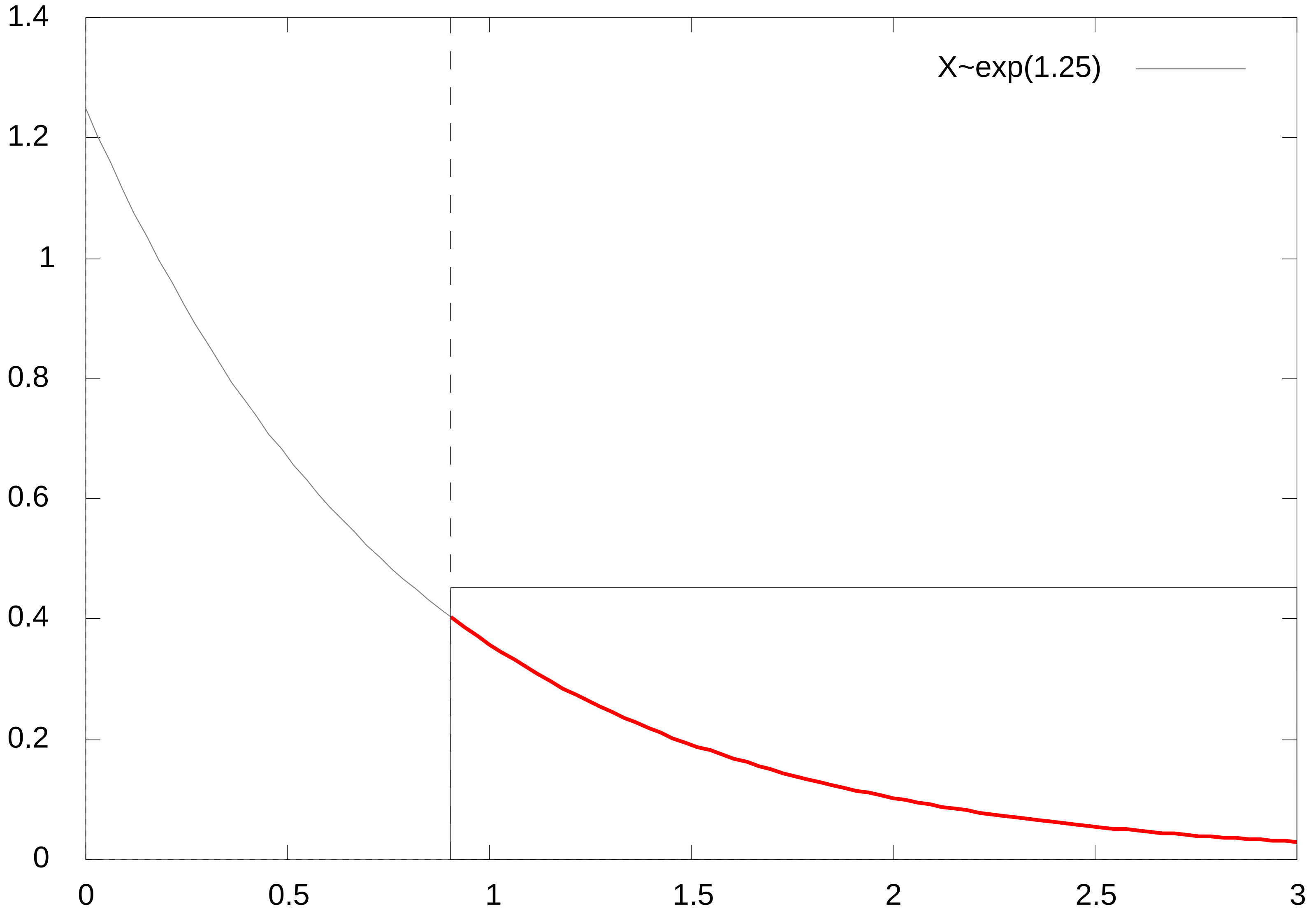


Memoryless property

- The exponential distribution is unique by being *memoryless*
  - i.e. if you interrupt an exponential event, the remaining time is also exponential
  - Mathematically we would say – let $X \sim \exp(\lambda)$ and at time, $t + u$, where $X > u$, let $Y = X - u$ be the distribution of the *remaining time*:

$$f_{(Y|X>u)}(t) = f_X(t)$$

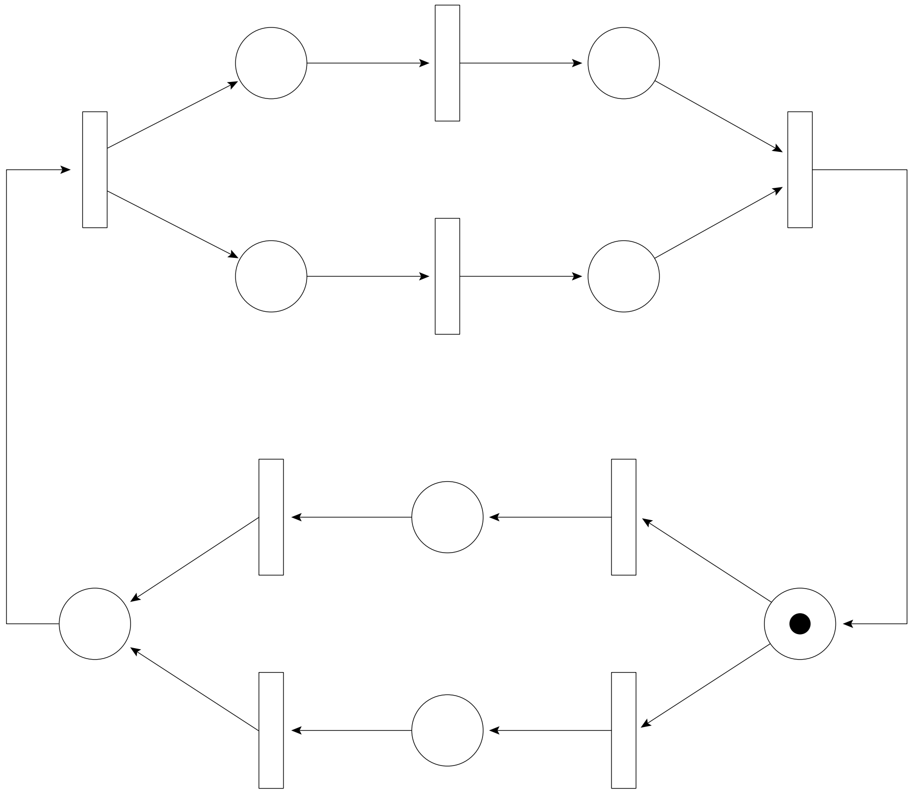
Markov property



Stochastic Petri nets

- Stochastic Petri nets (SPNs) are based on untimed Petri nets (PNs)
- SPNs have exponential firing delays
- Generalised stochastic Petri nets (GSPNs) have exponential and immediate firing delays
- PNs/SPNs/GSPNs good at capturing resource usage, functional dependency
- No syntax for composition – although easy to compose Petri nets by eye!

Petri nets: summary



- Circles are **places**, solid discs are **tokens**, rectangles are **transitions**
- Arrows indicate flow of tokens/execution

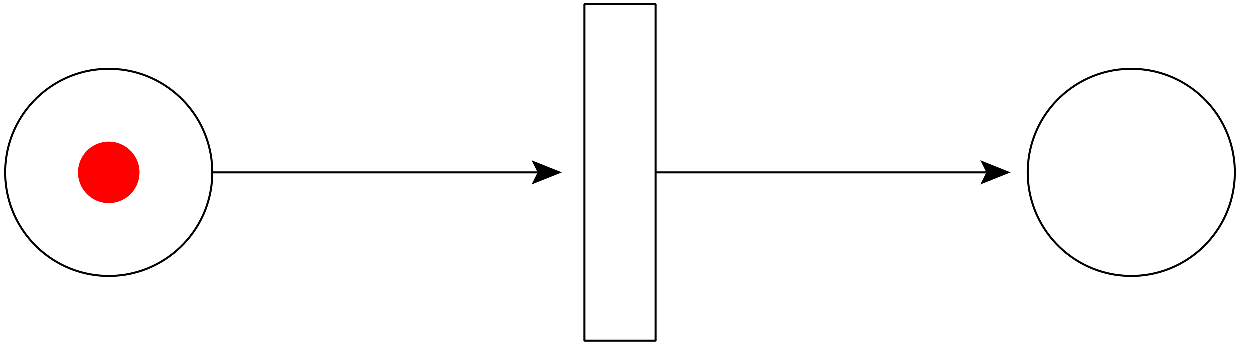
Petri nets: definitions

- Petri nets consist of *places*, *transitions* and *tokens*
- Places are connected to other places via transitions
- Tokens move from place to place by *enabling* and then *firing* transitions according to rules
- The configuration of tokens in a Petri net is known as the *marking* or state of the Petri net

Petri nets: enabling and firing

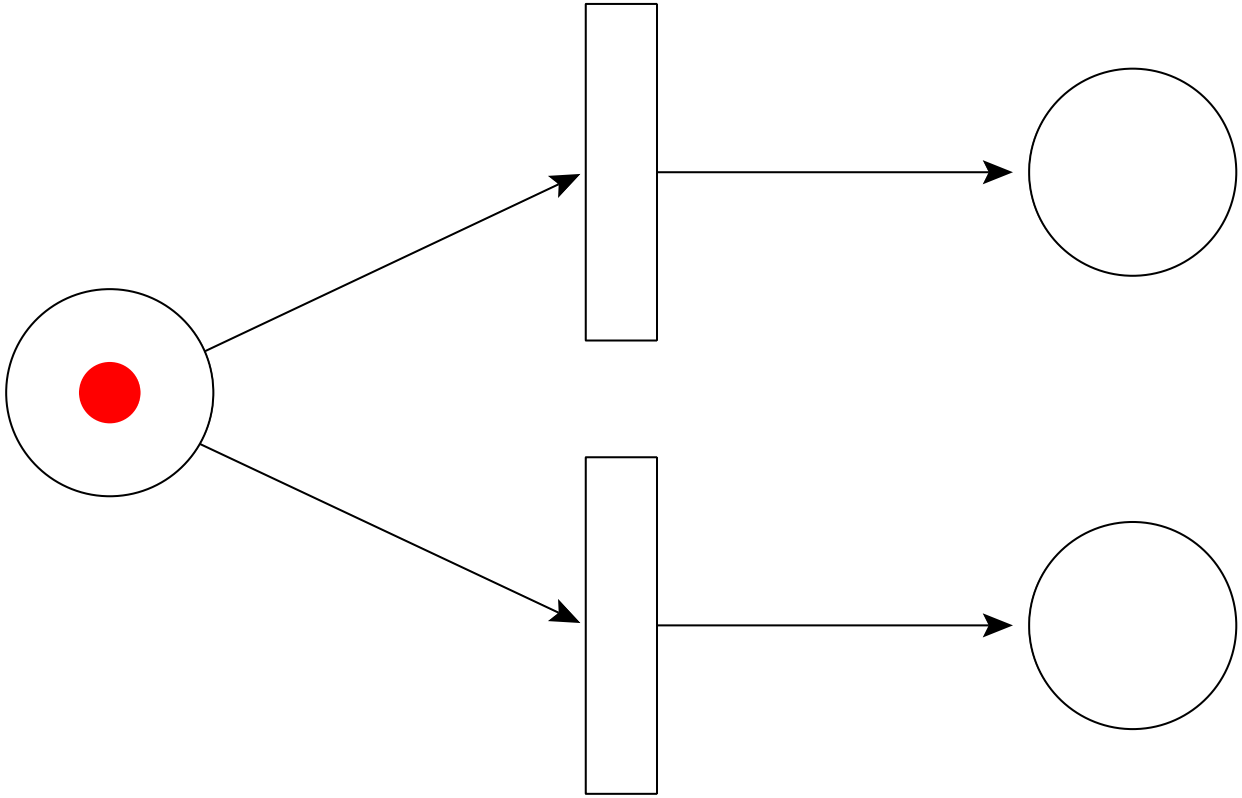
- An *in-place* for a transition is a place which points to that transition; an *out-place* for a transition is a place which is pointed to by that transition
- A transition is *enabled* if all the in-places contain tokens
- A transition *fires* by taking 1 token from each in-place and putting 1 token in each out-place for that transition
- A transition firing does not necessarily preserve the token count

Simple process transition



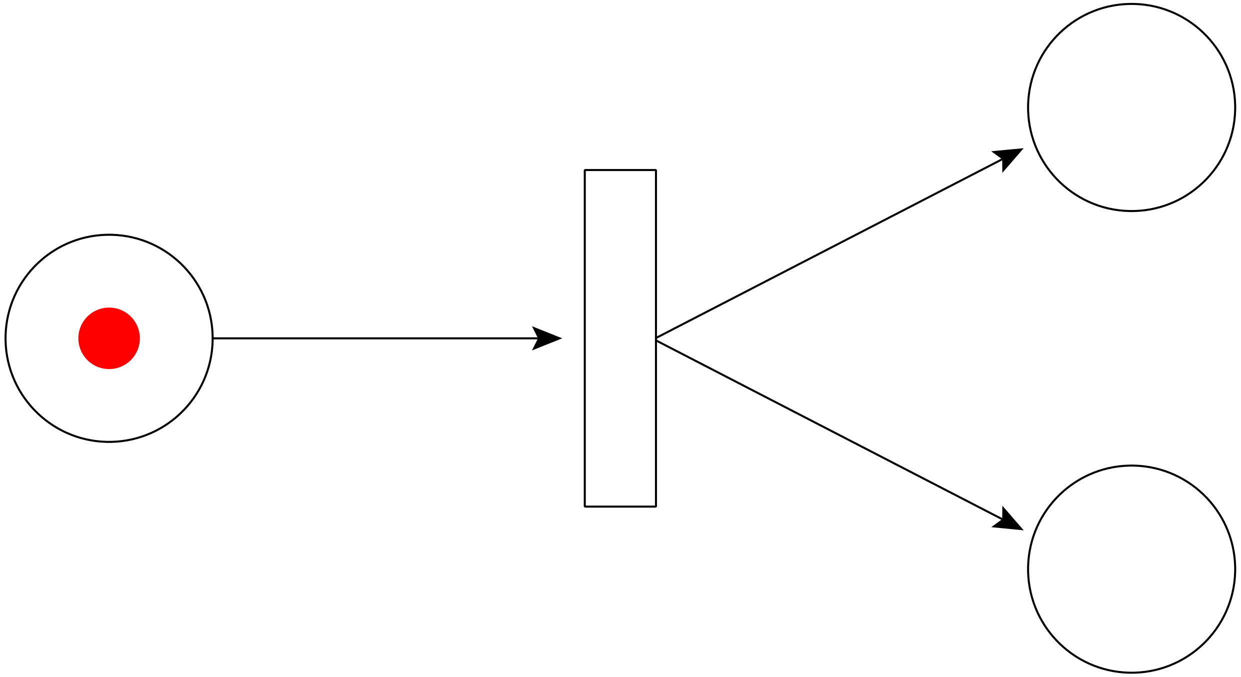
- A single token enables the transition, fires the transition and transits to the out-place

Process choice



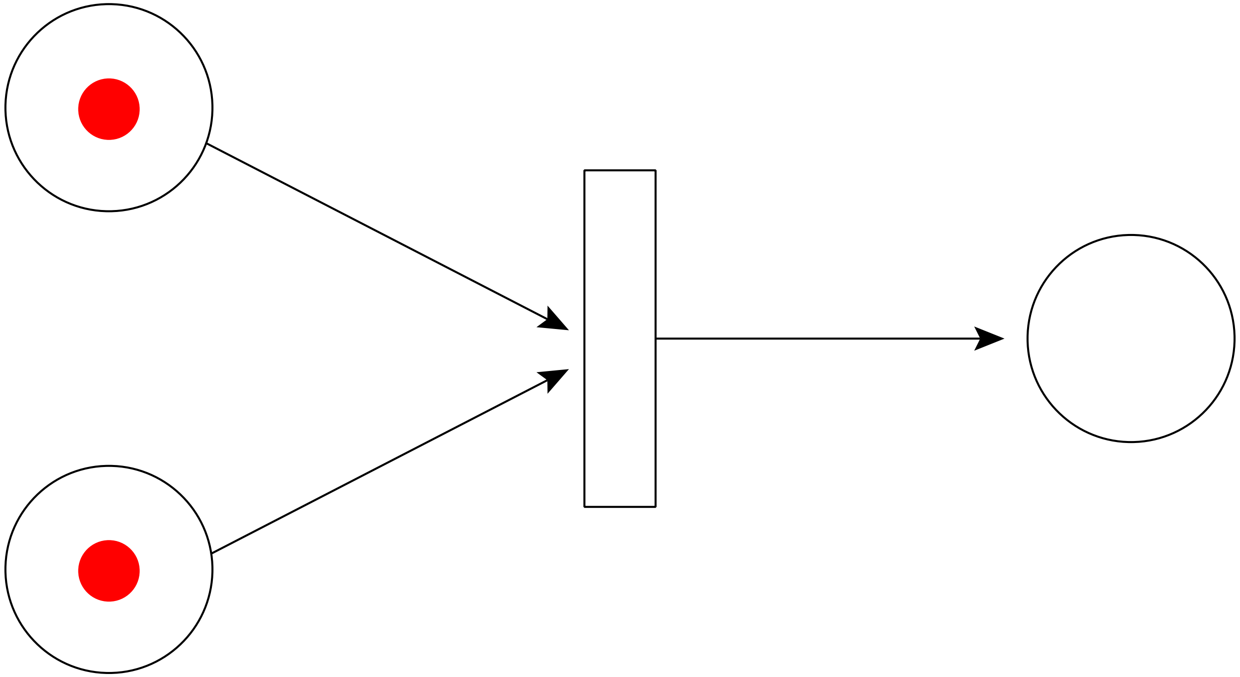
- A token can progress to either one of its out-places, but not both

Process forking



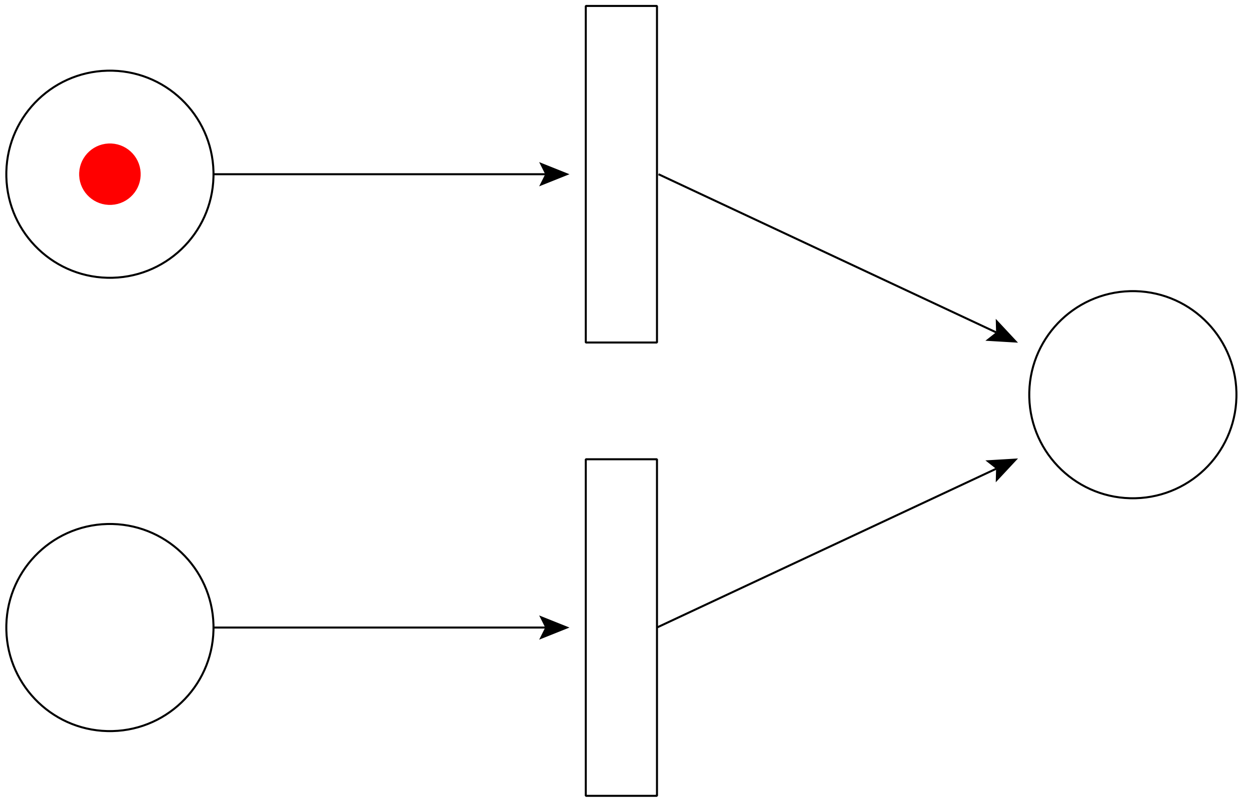
- A process can fork two independent processes with distinct behaviour, that operate in parallel with each other

Process joining



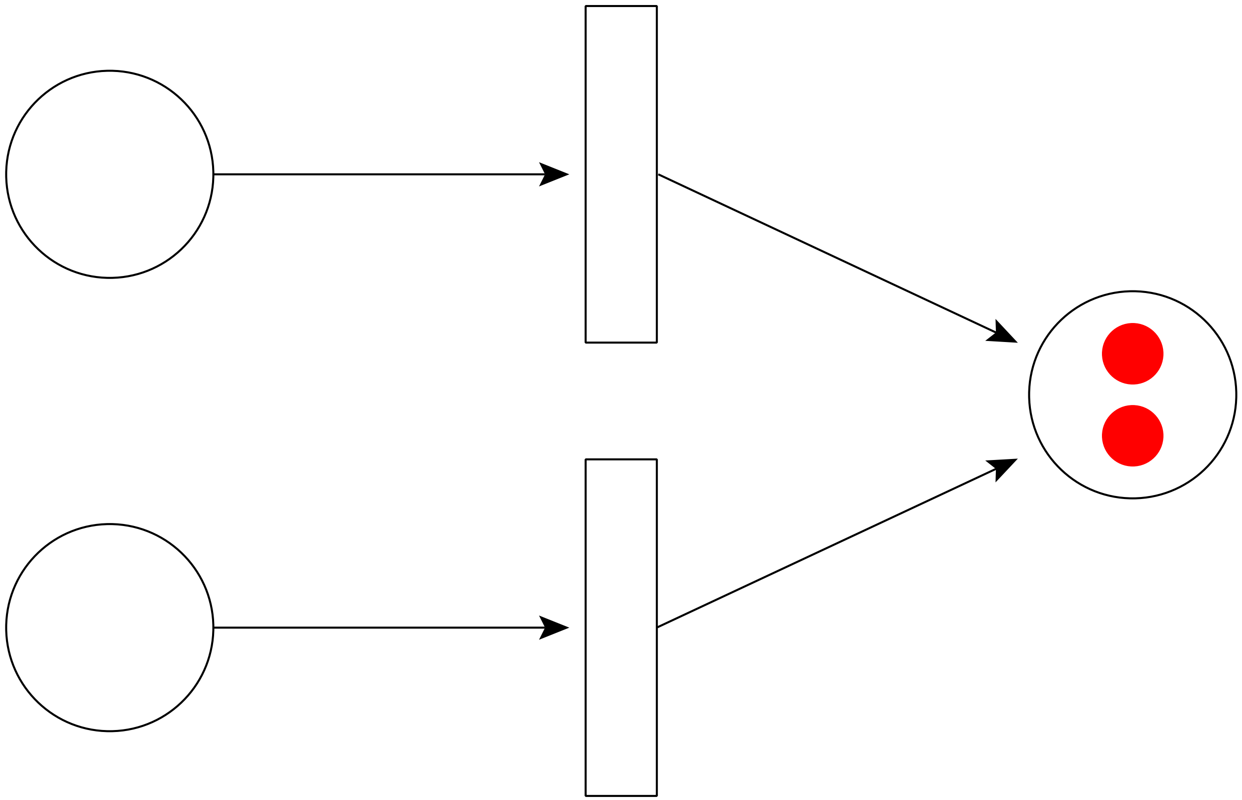
- Two independent parallel processes can be joined to form a single execution behaviour. Both (all) in-places need to be occupied before the transition is enabled.

Duplicate behaviour



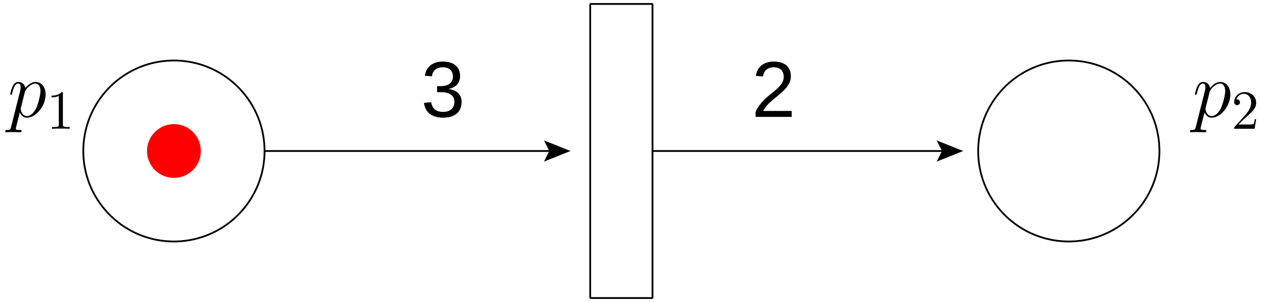
- Independent processes with duplicate behaviours can make use of the same Petri net structure

Duplicate behaviour



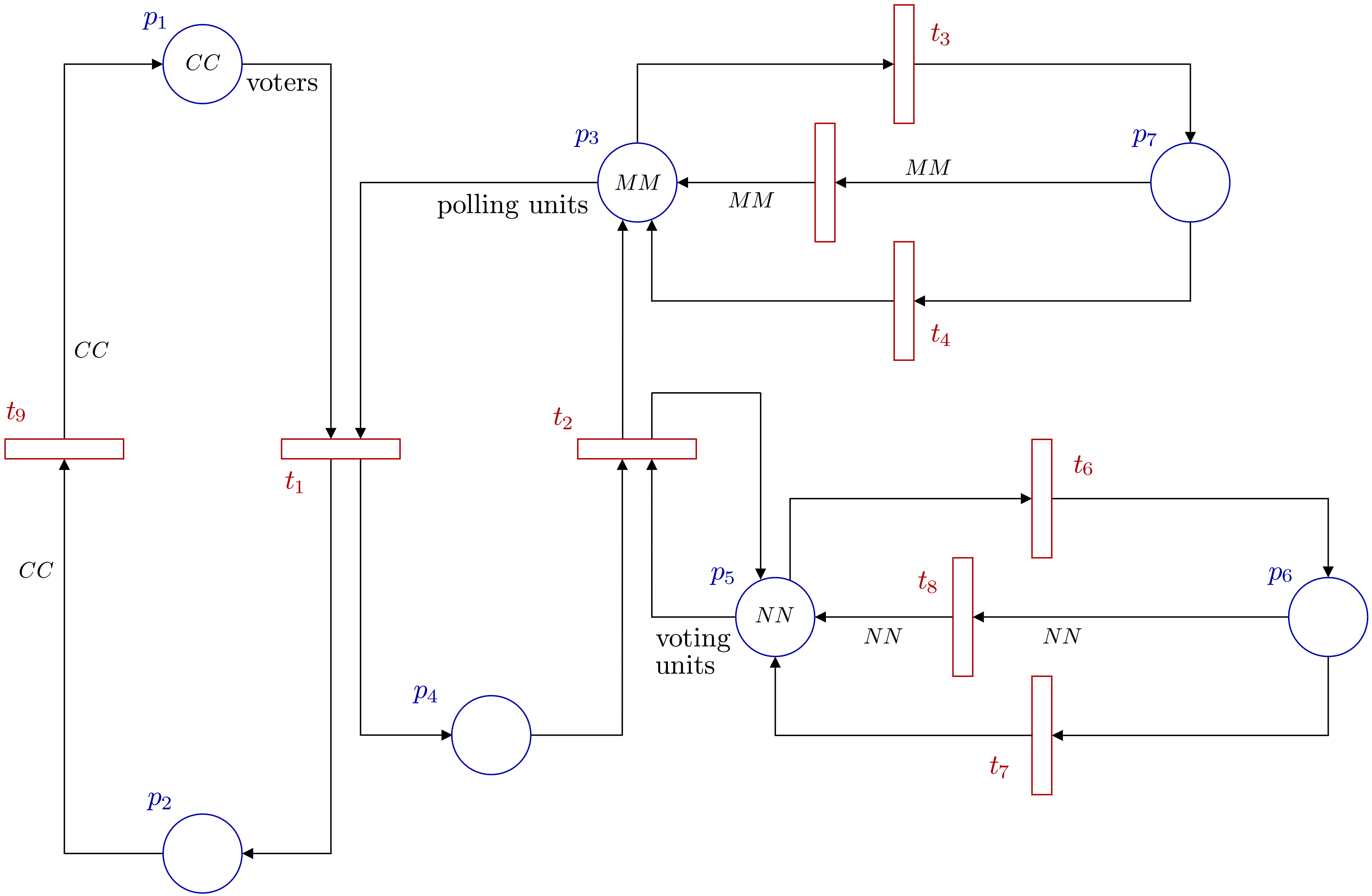
- Places can have multiple tokens representing independent processes

Multiple token enabling

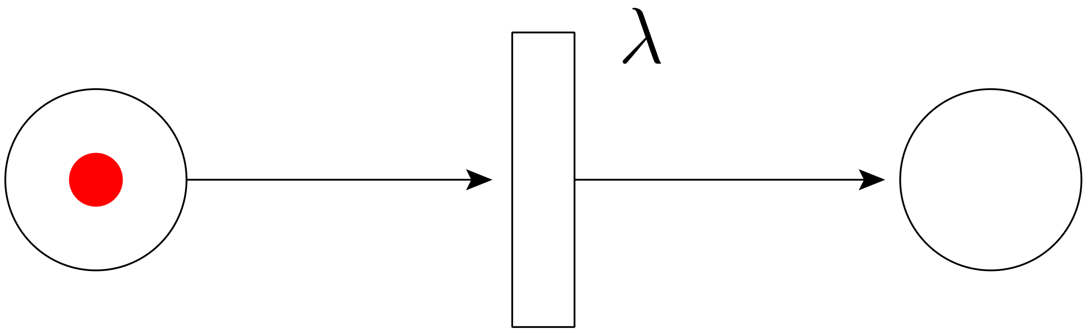


- If edges annotated with numbers, as above: it takes 3 tokens to enable the transition
- On firing, 3 tokens removed from place p_1 and 2 put into place p_2
- An unannotated Petri net implicitly has 1s on all its edges

SPN Example: Voting model

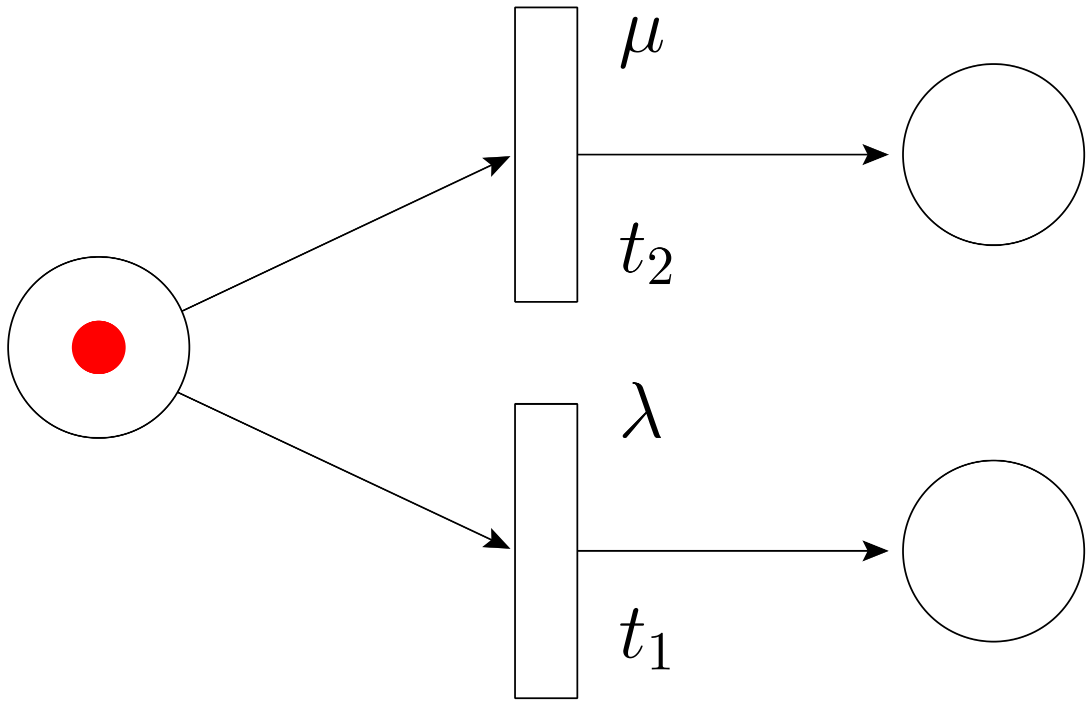


Stochastic Petri nets



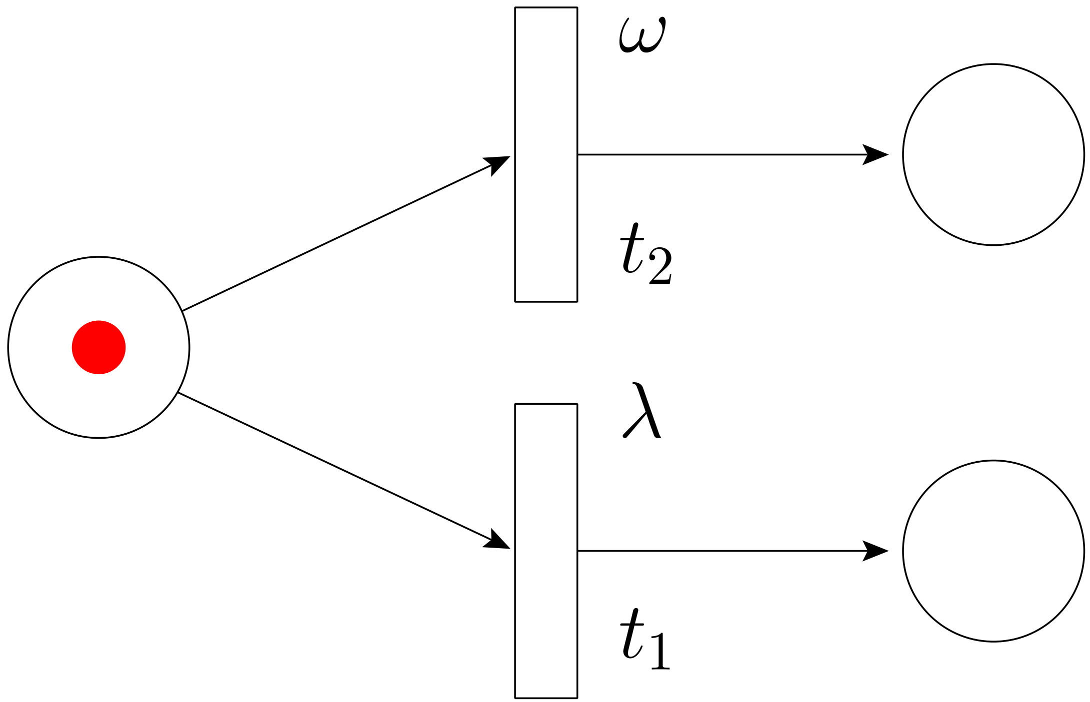
- SPNs have same functional behaviour as Petri nets; it now takes time to fire a transition
- Each transition has an exponential rate associated with it
- Let X be the time-to-fire-once-enabled of the transition above, $X \sim \exp(\lambda)$

Stochastic Petri nets: racing



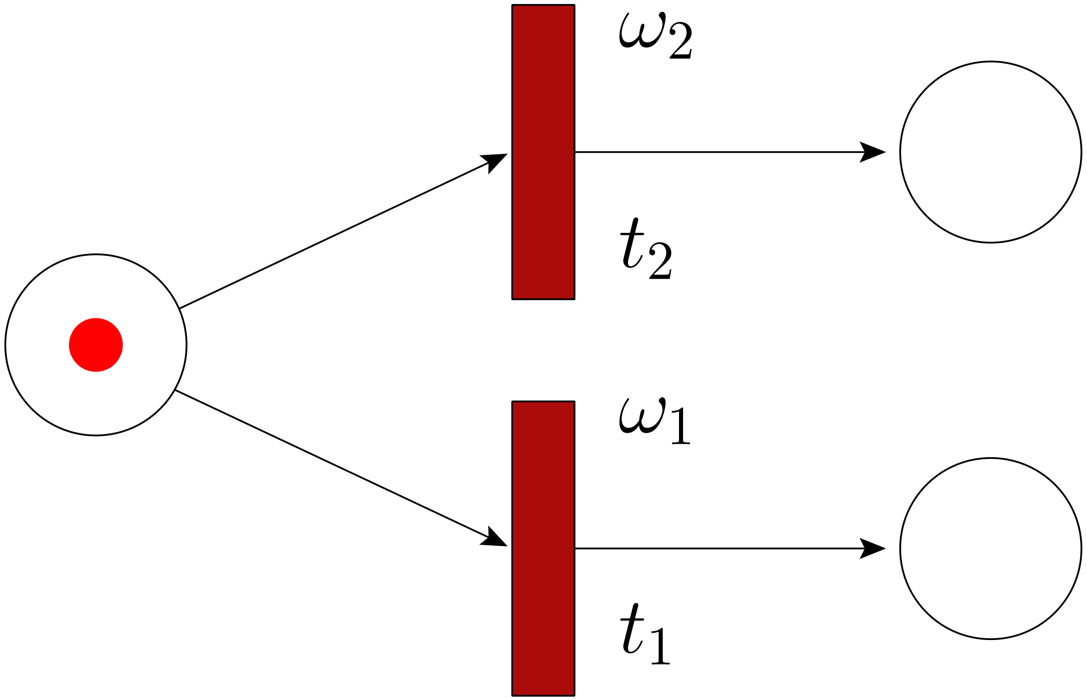
- What happens when we have two enabled transitions?
- If t_1 fires first what delay is left on t_2?

Generalised Stochastic Petri nets



- t_1 is timed transition with exponential delay λ; t_2 is an immediate transition with *weight*, ω
- Immediate transitions are always enabled before timed transitions

Generalised Stochastic Petri nets



- If multiple immediate transitions are enabled then weights are used to choose which fires
- Transition t_i fires with probability $\frac{\omega_i}{\sum_j \omega_j}$

Continuous Time Markov Chains

- The mathematical model underlying SPNs (and GSPNs after removing vanishing states) is a continuous time Markov chain (CTMC)
- A CTMC is formally described as a sequence of states from time $t \geq 0$ or:

$$\{X(t)\ :\ t \geq 0\}$$

  where $X(t)$ is a random variable representing the state of the chain at time t
- A CTMC is usually represented, in practice, by a generator matrix of rates, Q

Continuous Time Markov Chains

- A CTMC has the property:

$$\mathbb{P}(X(t) = i \mid X(t_n) = j_n, X(t_{n-1}) = j_{n-1}, \ldots, X(t_0) = j_0)$$
$$= \mathbb{P}(X(t) = i \mid X(t_n) = j_n)$$

for any sequence of times

$$t_0 < t_1 < \cdots < t_{n-1} < t_n$$

- This means that the probability of progressing to any state i depends only on the current state, and not on any prior trace history.

Generating a CTMC from an SPN

1. Label the places of the SPN
2. Create a tuple representing the current marking of the SPN, e.g. $(1, 0, 0, 1, 0)$
3. Find all possible transitions out of that marking
   (a) For each transition, write down the new tuple that is created
   (b) an arrow leading from first tuple to the second annotated with the rate of the firing transition
4. Repeat from (2) until all markings discovered

Creating a Generator Matrix

Need to create the *generator matrix* for the CTMC, Q:

1. Number markings as generated on previous slide
2. Set $q_{ij} =$ sum of rates from marking i to j
3. Ignore any transitions from marking i to itself
4. Set $q_{ii} = -\sum_{j \neq i} q_{ij}$
5. Now sum of all rows of Q should be 0

Steady state analysis of a CTMC

- To solve for the steady-state of CTMC with generator matrix, Q:

$$\vec{\pi} Q = \vec{0}$$

  find the elements of $\vec{\pi} = (\pi_1, \pi_2, \ldots \pi_n)$ using the additional constraint that:

$$\pi_1 + \pi_2 + \cdots + \pi_n = 1$$

- π_i represents the steady state probability of being in marking i

Steady state

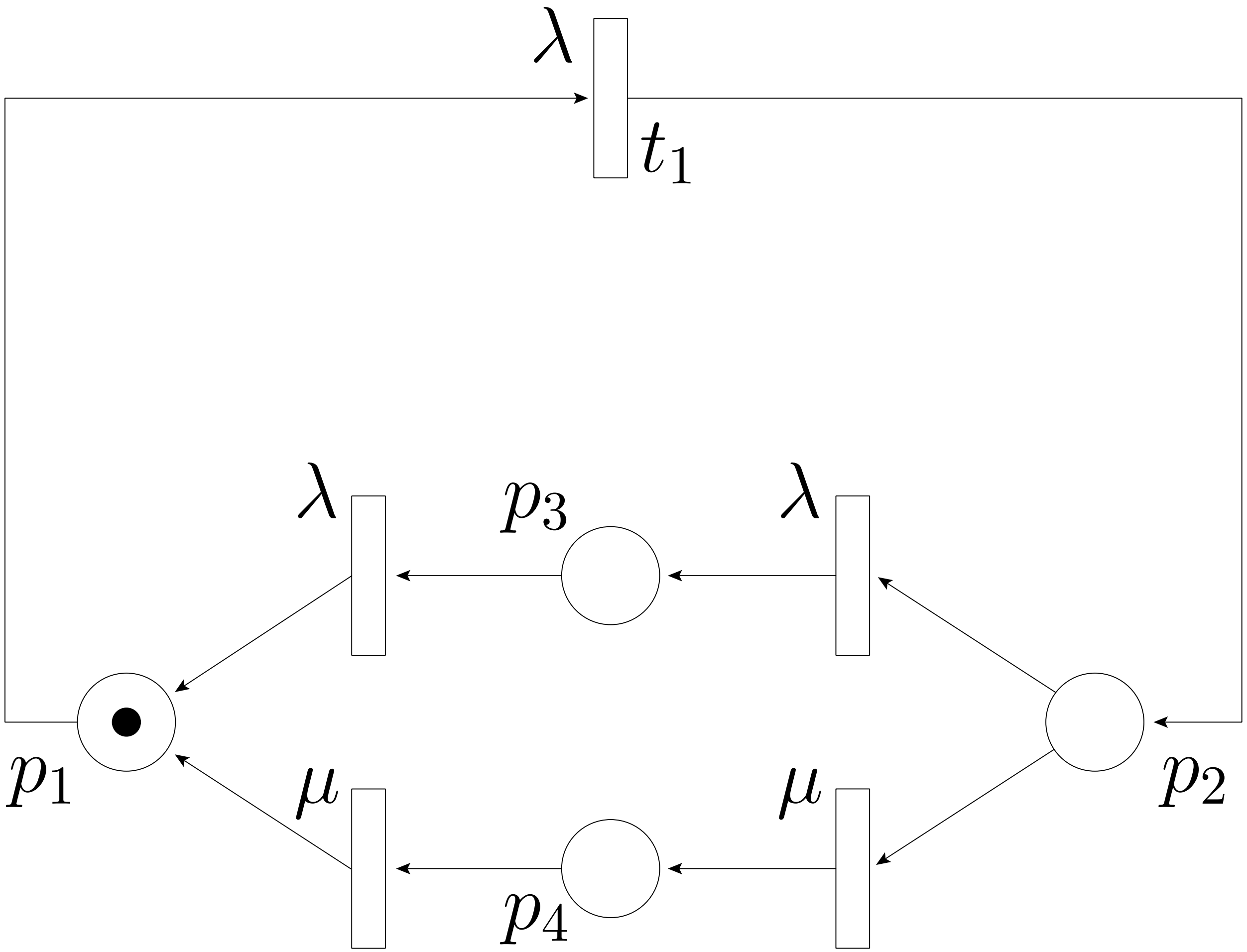
- Steady state probability of their being m tokens in place p_n is:

$$\sum_{i:M(i,m,n)} \pi_i$$

- where $M(i, m, n) =$
  marking i has m tokens in place n
- This means that if you were to let your SPN run for a long time and glimpse it's marking, it would have this probability of being in this state

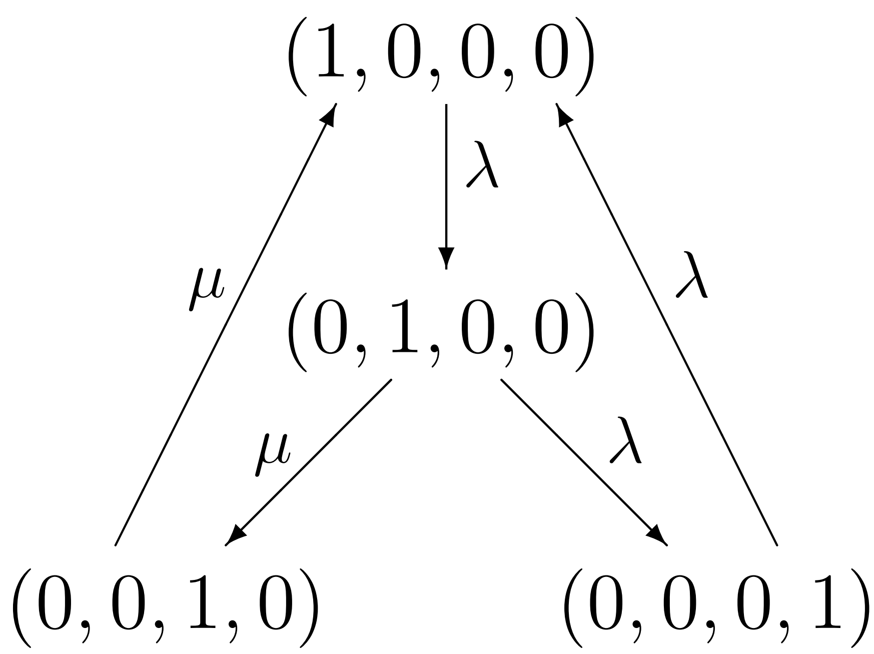
Steady-state example [1]

- Given a stochastic Petri net:



Steady-state example [2]

- Using a tuple representation of marking, (p_1, p_2, p_3, p_4), construct the underlying Markov chain



Steady-state example [3]

- By enumerating the states of the SPN, we can write down the CTMC genrator matrix, Q
- State enumeration:
  1. $(1, 0, 0, 0)$
  2. $(0, 1, 0, 0)$
  3. $(0, 0, 1, 0)$
  4. $(0, 0, 0, 1)$
- Gives the following transition matrix:

$$Q = \begin{pmatrix} -\lambda & \lambda & 0 & 0 \\ 0 & -(\lambda + \mu) & \mu & \lambda \\ \mu & 0 & -\mu & 0 \\ \lambda & 0 & 0 & -\lambda \end{pmatrix}$$

Steady-state example [4]

- Solving $\vec{\pi}Q = \vec{0}$ for a specific $\vec{\pi} = (\pi_1, \ldots, \pi_n)$ gives a steady state probablity, π_i, for being in marking i
- The equations from $\vec{\pi}Q$ will be linearly dependent since $|Q| = 0$, so you will need $\sum_i \pi_i = 1$ to give a unique solution
- In this case we get:

$$\vec{\pi} = \frac{1}{4\lambda + \mu}(\lambda + \mu, \lambda, \lambda, \lambda)$$

Steady-state example [5]

- Example calculations:
  - So if $\lambda = 1$, $\mu = 6$:

$$\mathbb{P}(\text{1 token in } p_1) = \pi_1 = 0.7$$

$$\mathbb{P}(\text{1 token in either } p_3 \text{ or } p_4) = \pi_3 + \pi_4 = 0.2$$

  - Firing rate of a transition, t, is $\sum_{i\,:\,E(i,t)} \pi_i\, r(t)$ where:
    - $E(i,t) =$ marking i enables t
    - $r(t) =$ rate of transition, t

$$\mathbb{P}(\text{average firing rate of } t_1) = \pi_1 \lambda = 0.7$$

PEPA: Stochastic process algebra

- PEPA is a language for describing systems which have underlying continuous time Markov chains
- PEPA is useful because:
  - it is a formal, algebraic description of a system
  - it is compositional
  - it is parsimonious (succinct)
  - it is easy to learn!
  - it is used in research and in industry

Tool Support

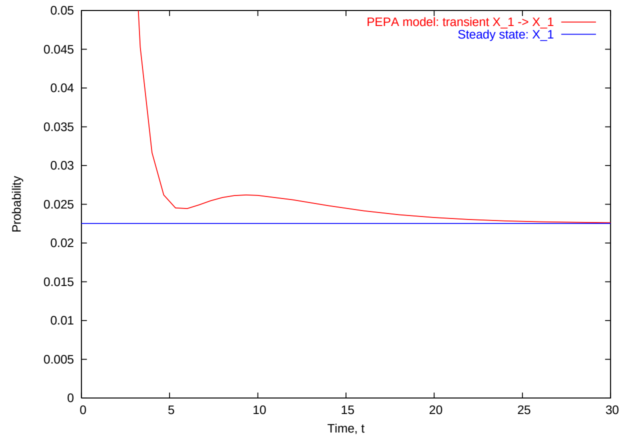
- PEPA has several methods of execution and analysis, through comprehensive tool support:
  - PEPA Workbench: Edinburgh
  - Möbius: Urbana-Champaign, Illinois
  - PRISM: Birmingham
  - ipc: Imperial College London

Types of Analysis

Steady-state and transient analysis in PEPA:

$$
\begin{aligned}
A1 &\stackrel{def}{=} (start, r_1).A2 + (pause, r_2).A3 \\
A2 &\stackrel{def}{=} (run, r_3).A1 + (fail, r_4).A3 \\
A3 &\stackrel{def}{=} (recover, r_1).A1 \\
AA &\stackrel{def}{=} (run, \top).(alert, r_5).AA \\
Sys &\stackrel{def}{=} AA \underset{\{run\}}{\bowtie} A1
\end{aligned}
$$

$\Rightarrow$

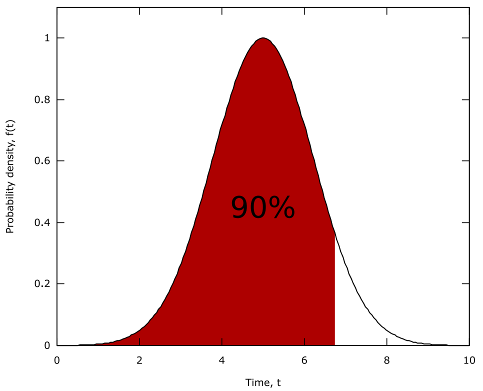


Passage-time Quantiles

Extract a passage-time density from a PEPA model:

$$
\begin{aligned}
\mathrm{A1} &\stackrel{\text{def}}{=} (\mathrm{start}, r_1).\mathrm{A2} + (\mathrm{pause}, r_2).\mathrm{A3} \\
\mathrm{A2} &\stackrel{\text{def}}{=} (\mathrm{run}, r_3).\mathrm{A1} + (\mathrm{fail}, r_4).\mathrm{A3} \\
\mathrm{A3} &\stackrel{\text{def}}{=} (\mathrm{recover}, r_1).\mathrm{A1} \\
\mathrm{AA} &\stackrel{\text{def}}{=} (\mathrm{run}, \top).(\mathrm{alert}, r_5).\mathrm{AA} \\
\mathrm{Sys} &\stackrel{\text{def}}{=} \mathrm{AA} \underset{\{run\}}{\bowtie} \mathrm{A1}
\end{aligned}
$$

$\Rightarrow$



PEPA Syntax

Syntax:

$$P \;::=\; (a,\lambda).P \;\Big|\; P+P \;\Big|\; P \underset{L}{\bowtie} P \;\Big|\; P/L \;\Big|\; A$$

- Action prefix: $(a,\lambda).P$
- Competitive choice: $P_1 + P_2$
- Cooperation: $P_1 \underset{L}{\bowtie} P_2$
- Action hiding: P/L
- Constant label: A

Prefix: $(a, \lambda).A$

- Prefix is used to describe a process that evolves from one state to another by *emitting* or *performing* an action
- Example:

$$P \stackrel{\text{def}}{=} (a, \lambda).A$$

...means that the process P evolves with rate λ to become process A, by emitting an a-action

- λ is an exponential rate parameter
- This is also be written:

$$P \xrightarrow{(a,\lambda)} A$$

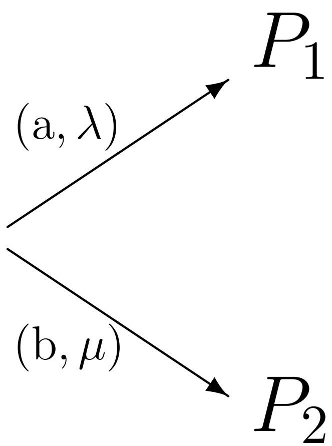
Choice: $P_1 + P_2$

- PEPA uses a type of choice known as *competitive choice*
- Example:

$$P \stackrel{\text{def}}{=} (a, \lambda).P_1 + (b, \mu).P_2$$

...means that P can evolve *either* to produce an a-action with rate λ *or* to produce a b-action with rate μ

- In state-transition terms, P



Choice: $P_1 + P_2$

- $P \stackrel{\text{def}}{=} (a, \lambda).P_1 + (b, \mu).P_2$
- This is competitive choice since:
  - P_1 and P_2 are in a *race condition* – the first one to perform an a or a b will dictate the direction of choice for $P_1 + P_2$
- What is the probability that we see an a-action?

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- $\underset{L}{\bowtie}$ defines concurrency and communication within PEPA
- The L in $P_1 \underset{L}{\bowtie} P_2$ defines the set of actions over which two components are to cooperate
- Any other actions that P_1 and P_2 can do, not mentioned in L, can happen independently
- If $a \in L$ and P_1 enables an a, then P_1 has to wait for P_2 to enable an a before the cooperation can proceed
- Easy source of deadlock!

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- If $P_1 \xrightarrow{(a,\lambda)} P_1'$ and $P_2 \xrightarrow{(a,\top)} P_2'$ then:

$$P_1 \underset{\{a\}}{\bowtie} P_2 \xrightarrow{(a,\lambda)} P_1' \underset{\{a\}}{\bowtie} P_2'$$

- $\top$ represents a passive rate which, in the cooperation, inherits the λ-rate of from P_1
- If both rates are specified and the only a-evolutions allowed from P_1 and P_2 are, $P_1 \xrightarrow{(a,\lambda)} P_1'$ and $P_2 \xrightarrow{(a,\mu)} P_2'$ then:

$$P_1 \underset{\{a\}}{\bowtie} P_2 \xrightarrow{(a,\min(\lambda,\mu))} P_1' \underset{\{a\}}{\bowtie} P_2'$$

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- The general cooperation case is where:
  - P_1 enables m a-actions
  - P_2 enables n a-actions

  at the moment of cooperation
- ...in which case there are mn possible transitions for $P_1 \underset{\{a\}}{\bowtie} P_2$
- $P_1 \underset{\{a\}}{\bowtie} P_2 \xrightarrow{(a,R)}$ where
  $$R = \frac{\lambda}{r_a(P_1)} \frac{\mu}{r_a(P_2)} \min(r_a(P_1), r_a(P_2))$$
- $r_a(P) = \sum_{i:P \xrightarrow{(a,r_i)}} r_i$ is the apparent rate of an action a – the total rate at which P can do a

Simplified Cooperation: $P_1 \underset{L}{\bowtie} P_2$

An approximation to pairwise cooperation:

- $P_1 \xrightarrow{(a,\top)} P_1'$ and $P_2 \xrightarrow{(a,\top)} P_2'$
  - $P_1 \underset{L}{\bowtie} P_2 \xrightarrow{(a,\top)} P_1' \underset{L}{\bowtie} P_2'$
- $P_1 \xrightarrow{(a,\lambda)} P_1'$ and $P_2 \xrightarrow{(a,\top)} P_2'$
- $P_1 \xrightarrow{(a,\top)} P_1'$ and $P_2 \xrightarrow{(a,\lambda)} P_2'$
  - Both give: $P_1 \underset{L}{\bowtie} P_2 \xrightarrow{(a,\lambda)} P_1' \underset{L}{\bowtie} P_2'$
- $P_1 \xrightarrow{(a,\lambda)} P_1'$ and $P_2 \xrightarrow{(a,\mu)} P_2'$
  - $P_1 \underset{L}{\bowtie} P_2 \xrightarrow{(a,\min(\lambda,\mu))} P_1' \underset{L}{\bowtie} P_2'$

Hiding: P/L

- Used to turn observable actions in P into hidden or silent actions in P/L
- L defines the set of actions to hide
- If $P \xrightarrow{(a,\lambda)} P'$:

$$P/\{a\} \xrightarrow{(\tau,\lambda)} P'/\{a\}$$

- τ is the *silent* action
- Used to hide complexity and create a component interface
- Cooperation on τ not allowed

Constant: A

- Used to define components labels, as in:
  - $P \stackrel{\text{def}}{=} (a, \lambda).P'$
  - $Q \stackrel{\text{def}}{=} (q, \mu).W$
- P,P', Q and W are all constants

PEPA: A Transmitter-Receiver

$$\text{System} \stackrel{\text{def}}{=} (\text{Transmitter} \underset{\emptyset}{\bowtie} \text{Receiver}) \underset{\{\text{transmit},\text{receive}\}}{\bowtie} \text{Network}$$

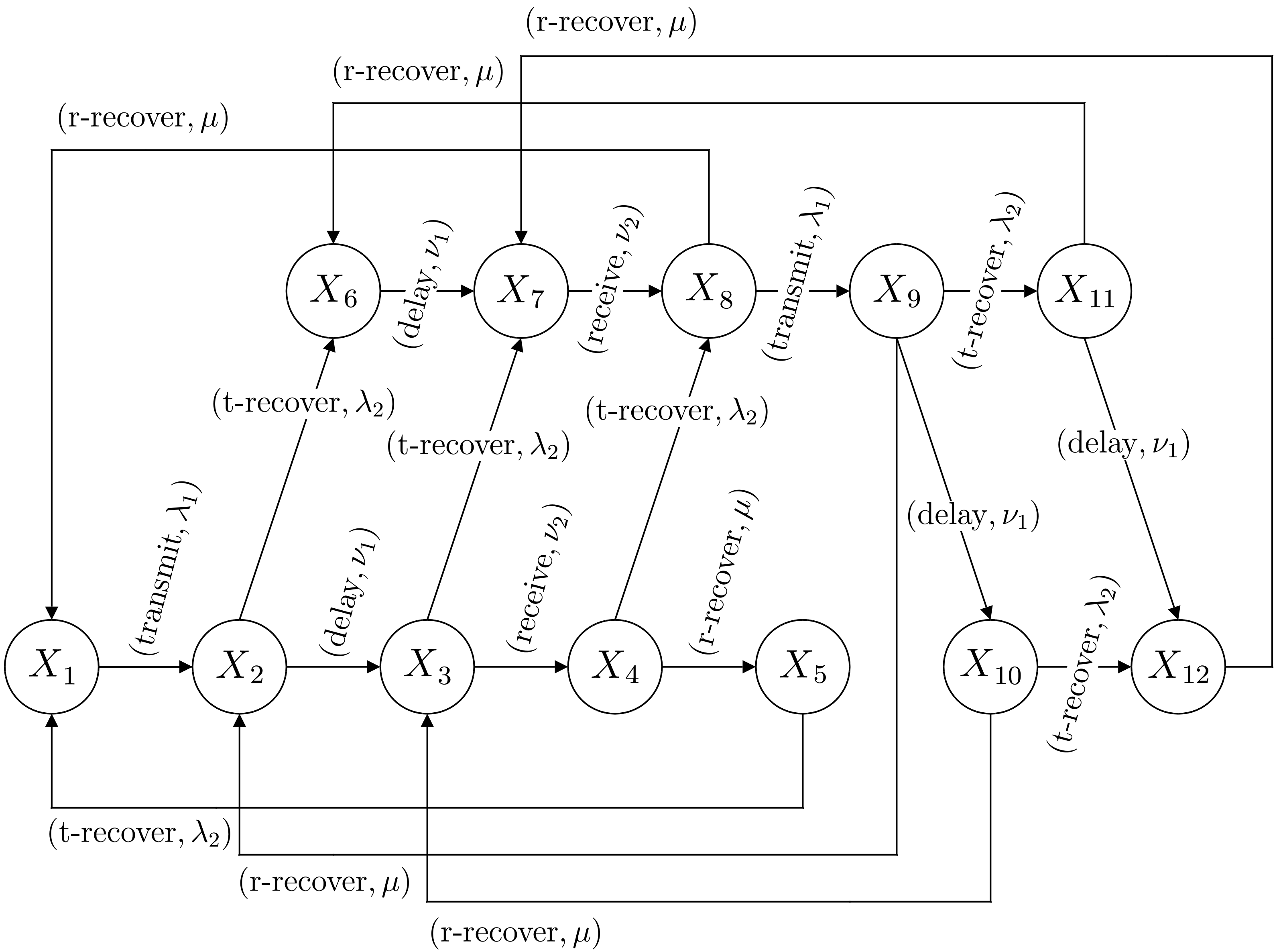
$$\text{Transmitter} \stackrel{\text{def}}{=} (\text{transmit}, \lambda_1).(\text{t\_recover}, \lambda_2).\text{Transmitter}$$

$$\text{Receiver} \stackrel{\text{def}}{=} (\text{receive}, \top).(\text{r\_recover}, \mu).\text{Receiver}$$

$$\text{Network} \stackrel{\text{def}}{=} (\text{transmit}, \top).(\text{delay}, \nu_1).(\text{receive}, \nu_2).\text{Network}$$

- A simple transmitter-receiver over a network

T-R: Global state space



Expansion law for 2 Components

- $P_1 \bowtie_L P_2$ where $P_1 \xrightarrow{(a_1,r_1)} P_1'$ and $P_2 \xrightarrow{(a_2,r_2)} P_2'$
- There are four cases: $a_1, a_2 \notin L$, $a_1 = a_2 \in L$, $a_1 \in L, a_2 \notin L$ and $a_1 \notin L, a_2 \in L$:
  - $P_1 \bowtie_L P_2 = (a_1, r_1).(P_1' \bowtie_L P_2) + (a_2, r_2).(P_1 \bowtie_L P_2')$
    if $a_1, a_2 \notin L$
  - $P_1 \bowtie_L P_2 = (a_1, \min(r_1, r_2)).(P_1' \bowtie_L P_2')$
    if $a_1 = a_2 \in L$
  - $P_1 \bowtie_L P_2 = (a_1, r_1).(P_1' \bowtie_L P_2)$
    if $a_1 \notin L, a_2 \in L$
  - ...

Possible Evolutions of 2 Cpts

- $P_1 \underset{L}{\bowtie} P_2$ where $P_1 \xrightarrow{(a_1,r_1)} P_1'$ and $P_2 \xrightarrow{(a_2,r_2)} P_2'$.
  - $a_1, a_2 \notin L$: $P_1 \underset{L}{\bowtie} P_2 \xrightarrow{(a_1,r_1)} P_1' \underset{L}{\bowtie} P_2$
  - $a_1, a_2 \notin L$: $P_1 \underset{L}{\bowtie} P_2 \xrightarrow{(a_2,r_2)} P_1 \underset{L}{\bowtie} P_2'$
  - $a_1 \notin L, a_2 \in L$: $P_1 \underset{L}{\bowtie} P_2 \xrightarrow{(a_1,r_1)} P_1' \underset{L}{\bowtie} P_2$
  - $a_1 \in L, a_2 \notin L$: $P_1 \underset{L}{\bowtie} P_2 \xrightarrow{(a_2,r_2)} P_1 \underset{L}{\bowtie} P_2'$
  - $a_1 = a_2 \in L$: $P_1 \underset{L}{\bowtie} P_2 \xrightarrow{(a_1,\min(r_1,r_2))} P_1' \underset{L}{\bowtie} P_2'$
  - $a_1 \neq a_2, a_1, a_2 \in L$: $P_1 \underset{L}{\bowtie} P_2 \not\longrightarrow$

Extracting the CTMC

- So how do we get a Markov chain from this
  - Once we have enumerated the global states, we map each PEPA state onto a CTMC state
  - The transitions of the global state space become transitions of the CTMC generator matrix
  - Any self loops are ignored in the generator matrix – why?
  - Any multiple transitions have their rate summed in the generator matrix - why?

Extracting the CTMC (2)

For example if: $P_1 \underset{L}{\bowtie} P_2 \xrightarrow{(a,\lambda)} P \underset{L}{\bowtie} P_2'$

1. Enumerate all the states and assign them numbers:
   - $\ldots$
   - 3: $P_1 \underset{L}{\bowtie} P_2$
   - 4: $P_1 \underset{L}{\bowtie} P_2'$
   - $\ldots$
2. Construct Q by setting $q_{34} = \lambda$ in this case
3. If another transition with rate μ is discovered for states 3 to 4 then q_{34} becomes $\lambda + \mu$

Extracting the CTMC (3)

4. Ignore any transitions from state i to state i
5. Finally set $q_{ii} = -\sum_{j \neq i} q_{ij}$
6. Now sum of all rows of Q should be 0
7. To solve for the steady-state of Q:

$$\vec{\pi} Q = \vec{0}$$

find the elements of $\vec{\pi} = (\pi_1, \pi_2, \ldots \pi_n)$ using the additional constraint that:

$$\pi_1 + \pi_2 + \cdots + \pi_n = 1$$

Voting Example I

$$\text{System} \stackrel{\text{def}}{=} (\text{Voter} \mid\mid \text{Voter} \mid\mid \text{Voter}) \underset{\{\text{vote}\}}{\bowtie} ((\text{Poler} \underset{L}{\bowtie} \text{Poler}) \underset{L'}{\bowtie} \text{Poler\_group\_0})$$

where

- $L = \{\text{recover\_all}\}$
- $L' = \{\text{recover}, \text{break}, \text{recover\_all}\}$

Voting Example II

$$
\begin{aligned}
\text{Voter} &\stackrel{\text{def}}{=} (\text{vote}, \lambda).(\text{pause}, \mu).\text{Voter} \\
\text{Poler} &\stackrel{\text{def}}{=} (\text{vote}, \top).(\text{register}, \gamma).\text{Poler} \\
&\quad + (\text{break}, \nu).\text{Poler\_broken} \\
\text{Poler\_broken} &\stackrel{\text{def}}{=} (\text{recover}, \tau).\text{Poler} \\
&\quad + (\text{recover\_all}, \top).\text{Poler}
\end{aligned}
$$

Voting Example III

$$\text{Poler\_group\_0} \stackrel{\text{def}}{=} (\text{break}, \top).\text{Poler\_group\_1}$$

$$\text{Poler\_group\_1} \stackrel{\text{def}}{=} (\text{break}, \top).\text{Poler\_group\_2} + (\text{recover}, \top).\text{Poler\_group\_0}$$

$$\text{Poler\_group\_2} \stackrel{\text{def}}{=} (\text{recover\_all}, \delta).\text{Poler\_group\_0}$$

M/M/2/3 Queue

- From tutorial sheet, asked to design a M/M/2/3 queue in PEPA
- There are two possible architectures depending on the type of M/M/2 queue
  - fully parallel client – client can be processed by as many servers as are available concurrently
  - fully serial client – client is allocated to a particular server and dealt with solely by that server until complete

M/M/2/3 Queue: Parallel Client

$$\begin{aligned}
\text{Arrival} &\stackrel{\text{def}}{=} (\mathit{arrive}, \lambda).\text{Arrival} \\
\text{Server}_1 &\stackrel{\text{def}}{=} (\mathit{service}, \mu).\text{Server}_1 \\
\text{Server}_2 &\stackrel{\text{def}}{=} (\mathit{service}, \mu).\text{Server}_2 \\
\text{Buff}_0 &\stackrel{\text{def}}{=} (\mathit{arrive}, \top).\text{Buff}_1 \\
\text{Buff}_1 &\stackrel{\text{def}}{=} (\mathit{arrive}, \top).\text{Buff}_2 + (\mathit{service}, \top).\text{Buff}_0 \\
\text{Buff}_2 &\stackrel{\text{def}}{=} (\mathit{arrive}, \top).\text{Buff}_3 + (\mathit{service}, \top).\text{Buff}_1 \\
\text{Buff}_3 &\stackrel{\text{def}}{=} (\mathit{service}, \top).\text{Buff}_2
\end{aligned}$$

$$\text{Sys}_\text{p} \stackrel{\text{def}}{=} \text{Arrival} \underset{L}{\bowtie} (\text{Buff}_0 \underset{M}{\bowtie} (\text{Server}_1 \parallel \text{Server}_2))$$

where $L = \{\mathit{arrive}\}, M = \{\mathit{service}\}$

M/M/2/3 Queue: Parallel Client

- $(\text{Server}_1 \parallel \text{Server}_2)$ is shorthand notation for $(\text{Server}_1 \underset{\emptyset}{\bowtie} \text{Server}_2)$
- The model Sys_p would be analogous to having a single server serving at twice the rate, i.e. 2μ
- ...so why not have a single server serve at rate 2μ?
- ...because it allows us to model breakdowns or hetereogeneous servers i.e. in some way give each server individual behaviour
- $Server_i = (service, \mu).Service_i + (break, \gamma).(recover, \chi).Server_i$

M/M/2/3 Queue: Serial Client

- Client is allocated to a particular server (first one free) e.g. post-office counters

$$\begin{aligned}
\mathrm{Arrival} &\stackrel{\mathrm{def}}{=} (arrive, \lambda).\mathrm{Arrival} \\[1em]
\mathrm{Server}_1 &\stackrel{\mathrm{def}}{=} (to_server, \top).(service, \mu).\mathrm{Server}_1 \\
\mathrm{Server}_2 &\stackrel{\mathrm{def}}{=} (to_server, \mu).(to_server, \top).\mathrm{Server}_2 \\[1em]
\mathrm{B}_0 &\stackrel{\mathrm{def}}{=} (arrive, \top).\mathrm{B}_1 \\
\mathrm{B}_1 &\stackrel{\mathrm{def}}{=} (arrive, \top).\mathrm{B}_2 + (to_server, \rho).\mathrm{B}_0 \\
\mathrm{B}_2 &\stackrel{\mathrm{def}}{=} (arrive, \top).\mathrm{B}_3 + (to_server, \rho).\mathrm{B}_1 \\
\mathrm{B}_3 &\stackrel{\mathrm{def}}{=} (to_server, \rho).\mathrm{B}_2
\end{aligned}$$

M/M/2/3 Queue: Serial Client

- Compose servers with B components

$$\mathrm{B}_0 \underset{\{to\_server\}}{\bowtie} (\mathrm{Server}_1 \parallel \mathrm{Server}_2)$$

- Now 3 customers can arrive in succession initially but while 2 are being serviced, a further 2 customers could arrive – making 5 i.e. not strictly a M/M/2/3 queue
- Need to have a further counting process, Buff, to check buffer not exceeded

M/M/2/3 Queue: Serial Client

$$\begin{aligned}
\text{Buff}_0 &\stackrel{\text{def}}{=} (\mathit{arrive}, \top).\text{Buff}_1 \\
\text{Buff}_1 &\stackrel{\text{def}}{=} (\mathit{arrive}, \top).\text{Buff}_2 + (\mathit{service}, \top).\text{Buff}_0 \\
\text{Buff}_2 &\stackrel{\text{def}}{=} (\mathit{arrive}, \top).\text{Buff}_3 + (\mathit{service}, \top).\text{Buff}_1 \\
\text{Buff}_3 &\stackrel{\text{def}}{=} (\mathit{service}, \top).\text{Buff}_2
\end{aligned}$$

- Now overall composed process looks like:

$$\text{Arrival} \underset{\{\mathit{arrive}\}}{\bowtie} (\text{Buff}_0 \underset{\{\mathit{arrive},\mathit{service}\}}{\bowtie} (\text{B}_0 \underset{\{\mathit{to\_server}\}}{\bowtie} (\text{Server}_1 \parallel \text{Server}_2)))$$

Steady-state reward vectors

- Reward vectors are a way of relating the analysis of the CTMC back to the PEPA model
- A reward vector is a vector, $\vec{r}$, which expresses a looked-for property in the system:
  - e.g. utilisation, loss, delay, mean buffer length
- To find the reward value of this property at steady state – need to calculate:

$$\text{reward} = \vec{\pi} \cdot \vec{r}$$

Constructing reward vectors

- Typically reward vectors match the states where particular actions are enabled in the PEPA model

$$
\begin{aligned}
Client &= (use, \top).(think, \mu).Client \\
Server &= (use, \lambda).(swap, \gamma).Server \\
Sys &= Client \underset{use}{\bowtie} Server
\end{aligned}
$$

- There are 4 states – enumerated as $1 : (C, S)$, $2 : (C', S')$, $3 : (C, S')$ and $4 : (C', S)$

Constructing reward vectors

- If we want to measure *server usage* in the system, we would reward states in the global state space where the action *use* is enabled or active
- Only the state $1 : (C, S)$ enables *use*
- So we set $r_1 = 1$ and $r_i = 0$ for $2 \leq i \leq 4$, giving:

$$\vec{r} = (1, 0, 0, 0)$$

- These are typical *action-enabled* rewards, where the result of $\vec{r} \cdot \vec{\pi}$ is a probability

Mean Occupation as a Reward

- Quantities such as mean buffer size can also be expressed as rewards

$$
\begin{aligned}
B_0 &= (arrive, \lambda).B_1 \\
B_1 &= (arrive, \lambda).B_2 + (service, \mu).B_0 \\
B_2 &= (arrive, \lambda).B_3 + (service, \mu).B_1 \\
B_3 &= (service, \mu).B_2
\end{aligned}
$$

- For this M/M/1/3 queue, number of states is 4

Mean Occupation as a Reward

- Having a reward vector which reflects the number of elements in the queue will give the mean buffer occupation for M/M/1/3
- i.e. set $\vec{r} = (0, 1, 2, 3)$ such that:

$$\text{mean buffer size} = \vec{\pi} \cdot \vec{r} = \sum_{i=0}^{3} \pi_i r_i$$

Useful facts about queues

- Little's Law: $N = \tau W$
  - N – mean buffer length; τ – arrival rate; W – mean waiting time/passage time
  - only applies to system in steady-state; no creating/destroying of jobs
- For M/M/1 queue:
  - λ – arrival rate, μ – service rate
  - Stability condition, $\rho = \lambda/\mu < 1$ for steady state to exist
  - Mean queue length $= \frac{\rho}{1-\rho}$
  - $\mathbb{P}(n \text{ jobs in queue at s-s}) = \rho^n(1-\rho)$

Small bit of queueing theory

- Going to show for M/M/1 queue, that:
  1. steady-state probability for buffer having k customers is:

$$\pi_k = (1-\rho)\rho^k$$

  2. mean queue length, N, at steady-state is:

$$\frac{\rho}{1-\rho}$$

Small bit of queueing theory

- As $N = \sum_{k=0}^{\infty} k\pi_k$, we need to find π_k:
  - Derive steady-state equations from time-varying equations
  - Solve steady-state equations to get π_k
  - Calculate M/M/1 mean queue length, N
- (In what follows, remember $\rho = \lambda/\mu$)

Small bit of queueing theory

- Write down time-varying equations for M/M/1 queue:
  - At time t, in state $k = 0$:

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

  - At time, t, in state $k \geq 1$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\pi_k(t) = -(\lambda+\mu)\pi_k(t)+\lambda\pi_{k-1}(t)+\mu\pi_{k+1}(t)$$

Steady-state for M/M/1

- At steady-state, $\pi_k(t)$ are constant (i.e. π_k) and $\frac{\mathrm{d}}{\mathrm{d}t}\pi_k(t) = 0$ for all k

$\Rightarrow$ Balance equations:

  - $-\lambda\pi_0 + \mu\pi_1 = 0$
  - $-(\lambda + \mu)\pi_k + \lambda\pi_{k-1} + \mu\pi_{k+1} = 0 \quad : k \geq 1$

- Rearrange balance equations to give:
  - $\pi_1 = \frac{\lambda}{\mu}\pi_0 = \rho\pi_0$
  - $\pi_{k+1} = \frac{\lambda+\mu}{\mu}\pi_k - \frac{\lambda}{\mu}\pi_{k-1} \quad : k \geq 1$

- Solution: $\pi_k = \rho^k\pi_0$ (proof by induction)

Normalising to find π_0

- As these π_k are probabilities which sum to 1:

$$\sum_{k=0}^{\infty} \pi_k = 1$$

- i.e. $\sum_{k=0}^{\infty} \pi_k = \sum_{k=0}^{\infty} \rho^k \pi_0 = \frac{\pi_0}{1-\rho} = 1$

$\Rightarrow$ $\pi_0 = 1 - \rho$ as long as $\rho < 1$

- So overall steady-state formula for M/M/1 queue is:

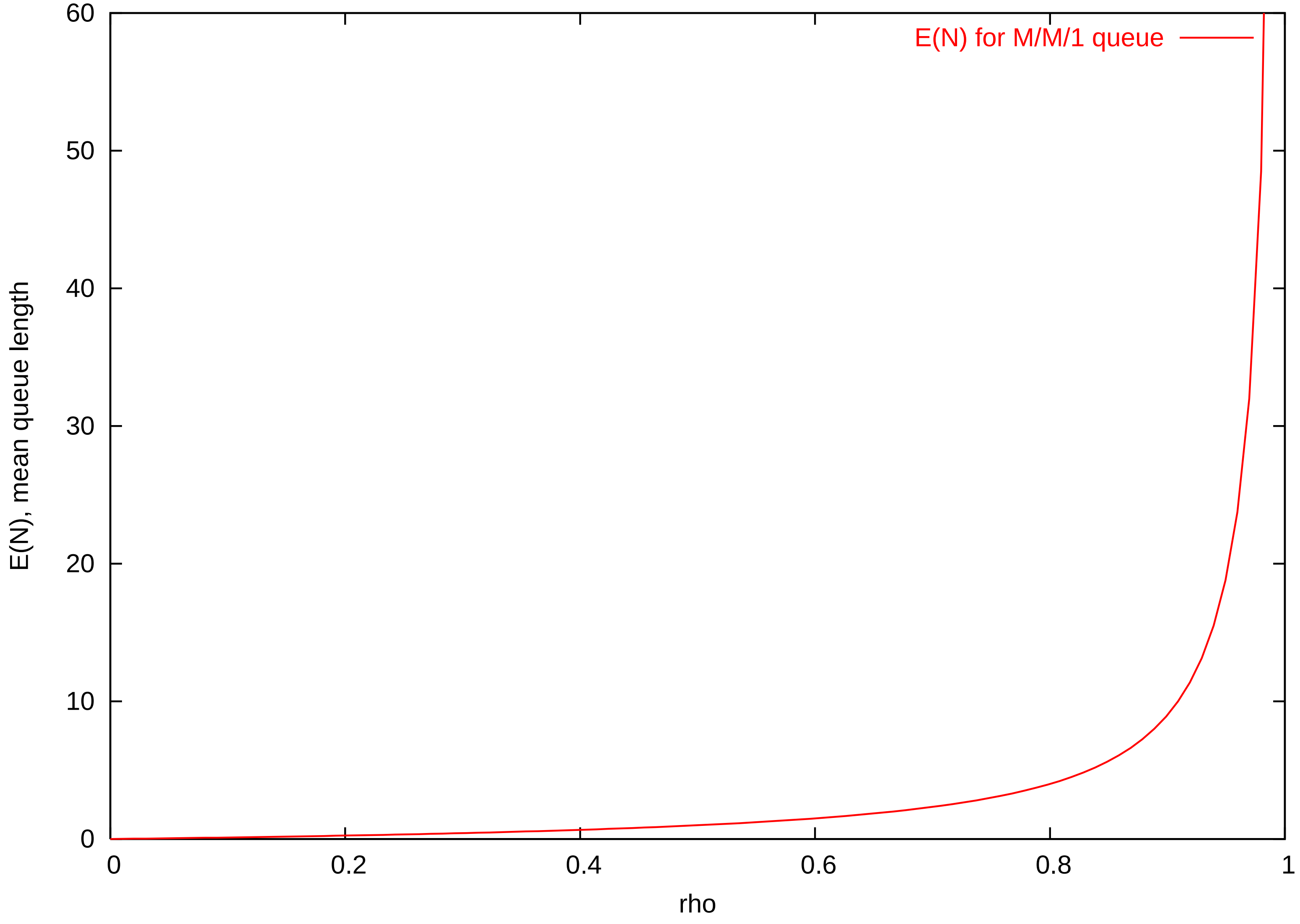
$$\pi_k = (1 - \rho)\rho^k$$

M/M/1 Mean Queue Length

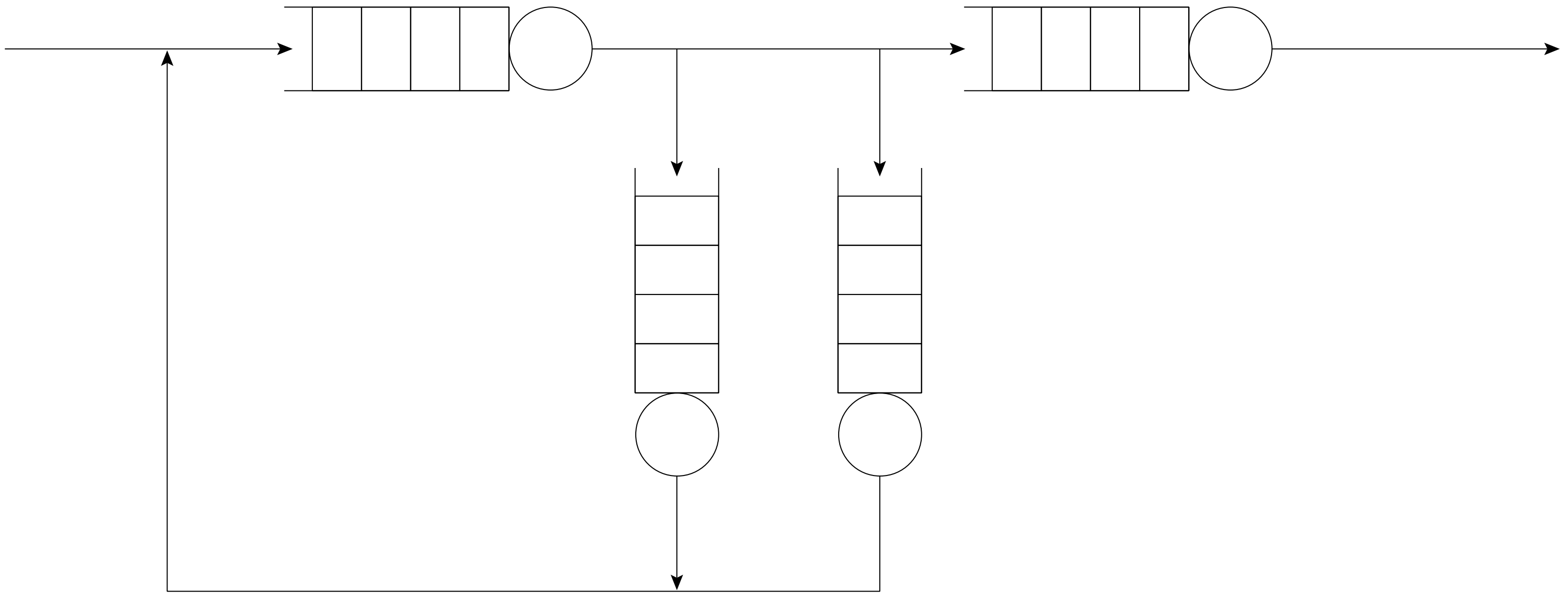
- N is queue length random variable
- N could be 0 or 1 or 2 or 3 ...
- Mean queue length is written N:

$$
\begin{aligned}
N &= 0.\mathbb{P}(\text{in state } 0) + 1.\mathbb{P}(\text{in state } 1) + 2.\mathbb{P}(\text{in state } 2) + \cdots \\
&= \sum_{k=0}^{\infty} k\pi_k \\
&= \pi_0 \sum_{k=0}^{\infty} k\rho^k = \pi_0\rho \sum_{k=0}^{\infty} k\rho^{k-1} = \pi_0\rho \sum_{k=0}^{\infty} \frac{\mathrm{d}}{\mathrm{d}\rho}\rho^k \\
&= \pi_0\rho \frac{\mathrm{d}}{\mathrm{d}\rho} \sum_{k=0}^{\infty} \rho^k = \pi_0\rho \frac{\mathrm{d}}{\mathrm{d}\rho}\left(\frac{1}{1-\rho}\right) \\
&= \frac{\pi_0\rho}{(1-\rho)^2} = \frac{\rho}{1-\rho} \qquad \square
\end{aligned}
$$

M/M/1 Mean Queue Length



Queueing Networks



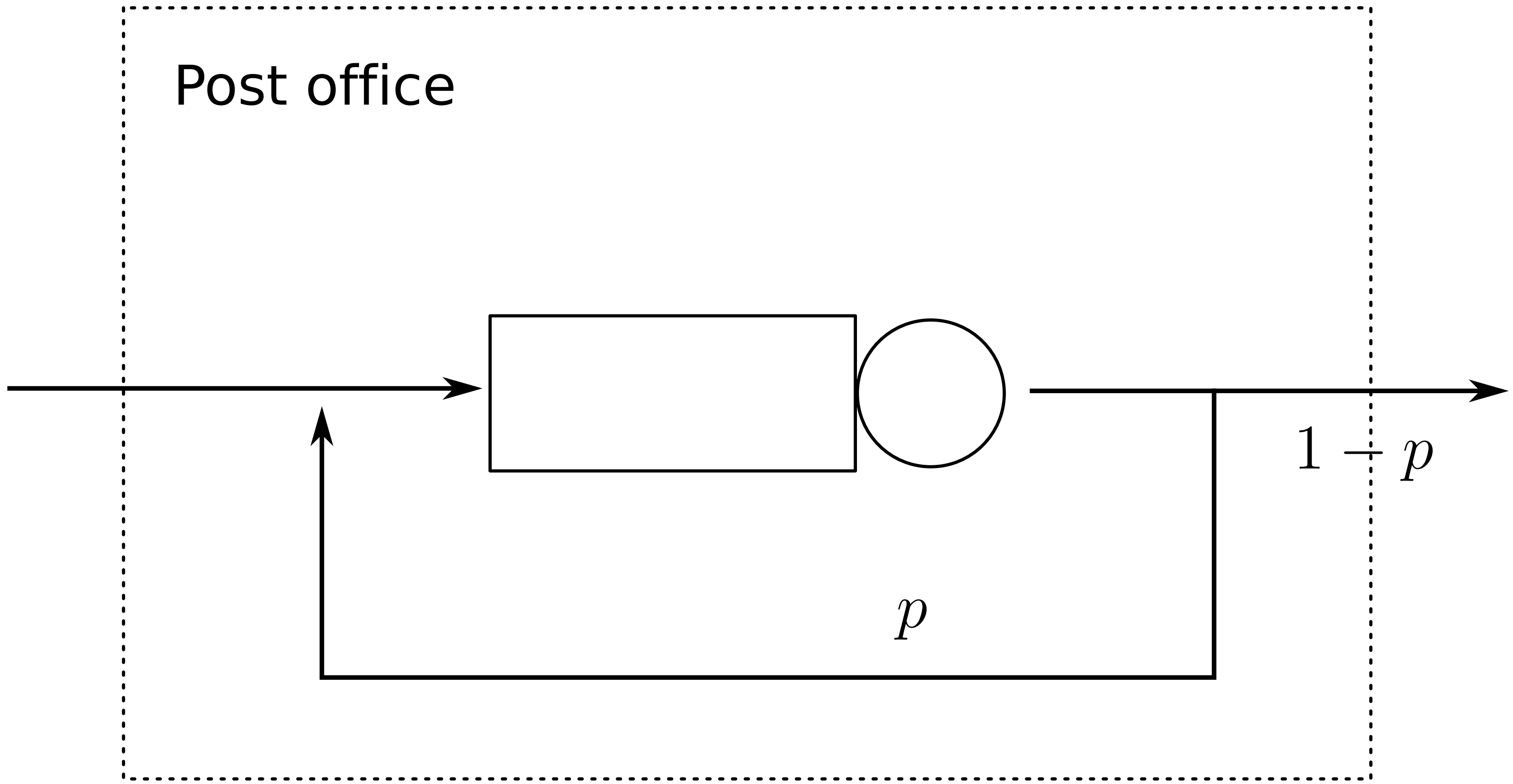
- Individual queue nodes represent contention for single resources
- A system consists of many inter-dependent resources – hence we need to reason about a *network* of queues to represent a system

Example: Post office queueing

Exam 2006:

A customer enters a Post Office and queues for service. After being served by a cashier, the customer either requires further service and returns to the back of the queue with probability p*, or departs the Post Office with probability* $(1 - p)$*.*

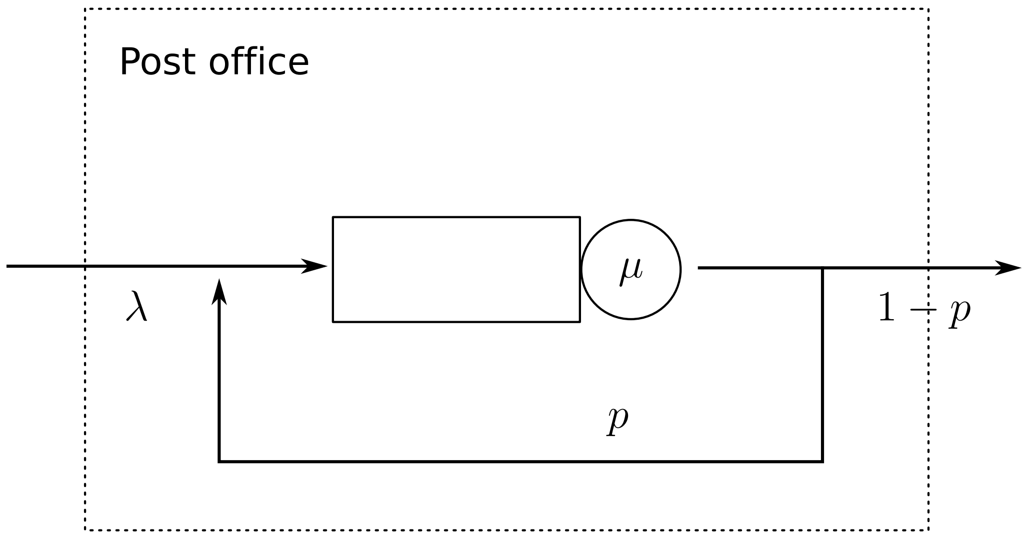
Example: Post office queueing



1. Find the mean number of times that the customer has to enter the queue.

Example: Post office queueing

2. Assuming that the queue in the Post Office is an M/M/1 queue with service rate μ, and that customers arrive at the Post Office with rate λ. Find the mean number of customers in the queue **and** the mean time spent in the Post Office.

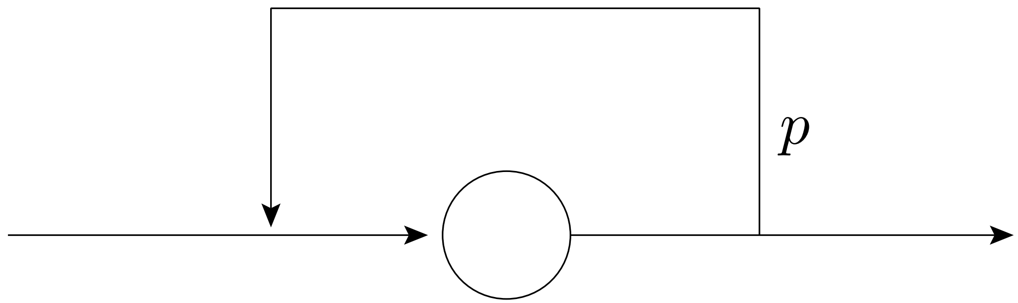


Open Queueing Networks

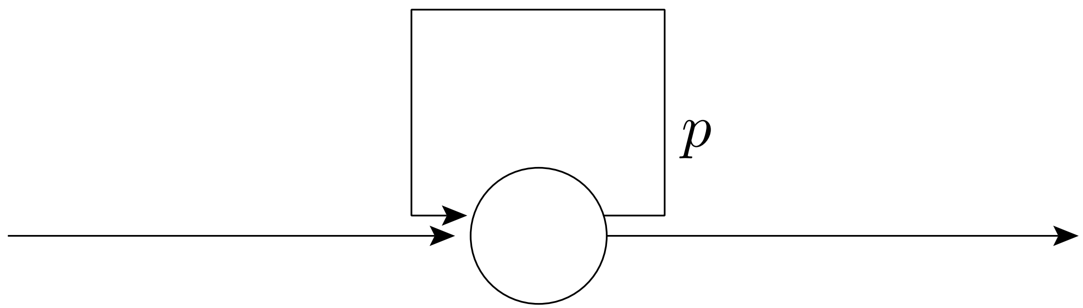
- A network of queueing nodes with inputs/outputs connected to each other
- Called an *open* queueing network (or OQN) because, traffic may enter (or leave) one or more of the nodes in the system from an external source (to an external sink)
- An open network is defined by:
  - γ_i, the exponential arrival rate from an external source
  - q_{ij}, the probability that traffic leaving node i will be routed to node j
  - μ_i exponential service rate at node i

OQN: Notation

- A node whose output can be probabilistically redirected into its input is represented as:



- or...



- probability p of being rerouted back into buffer

OQN: Network assumptions

In the following analysis, we assume:

- Exponential arrivals to network
- Exponential service at queueing nodes
- FIFO service at queueing nodes
- A network may be stable (be capable of reaching steady-state) or it may be unstable (have unbounded buffer growth)
- If a network reaches steady-state (becomes stationary), a single rate, λ_i, may be used to represent the throughput (both arrivals and departure rate) at node i

OQN: Traffic Equations

- The traffic equations for a queueing network are a linear system in λ_i
- λ_i represents the aggregate arrival rate at node i (taking into account any traffic feedback from other nodes)
- For a given node i, in an open network:

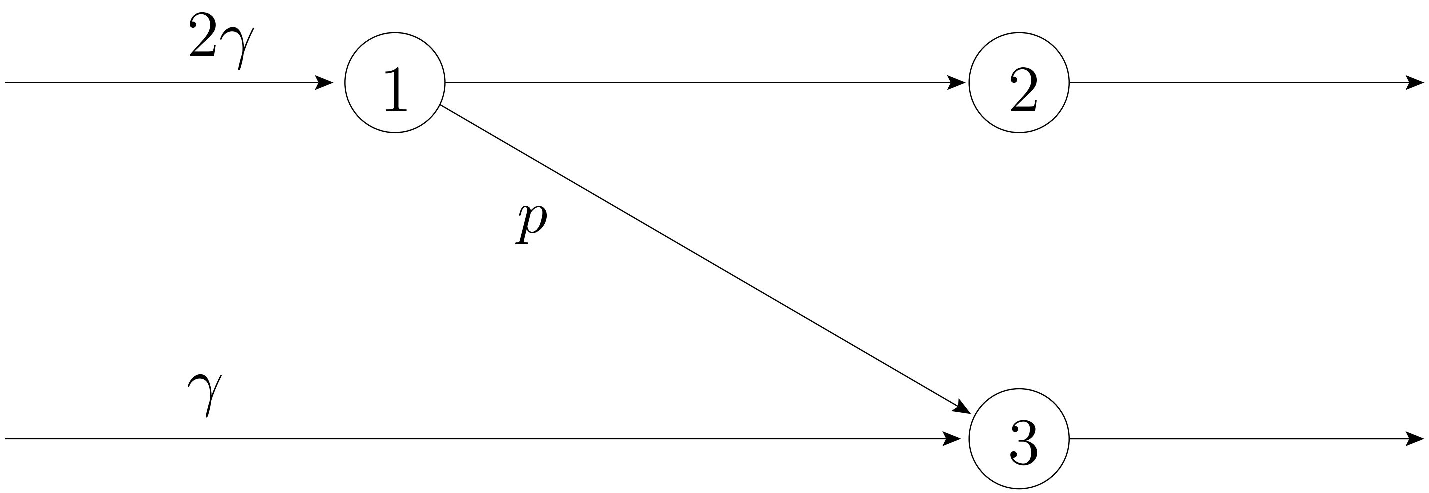
$$\lambda_i = \gamma_i + \sum_{j=1}^{n} \lambda_j q_{ji} \quad : i = 1, 2, \ldots, n$$

OQN: Traffic Equations

- Define:
  - the vector of aggregate arrival rates $\vec{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n)$
  - the vector of external arrival rates $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_n)$
  - the matrix of routeing probabilities $Q = (q_{ij})$
- In matrix form, traffic equations become:

$$\begin{aligned} \vec{\lambda} &= \vec{\gamma} + \vec{\lambda} Q \\ &= \vec{\gamma}(I - Q)^{-1} \end{aligned}$$

OQN: Traffic Equations: example 1

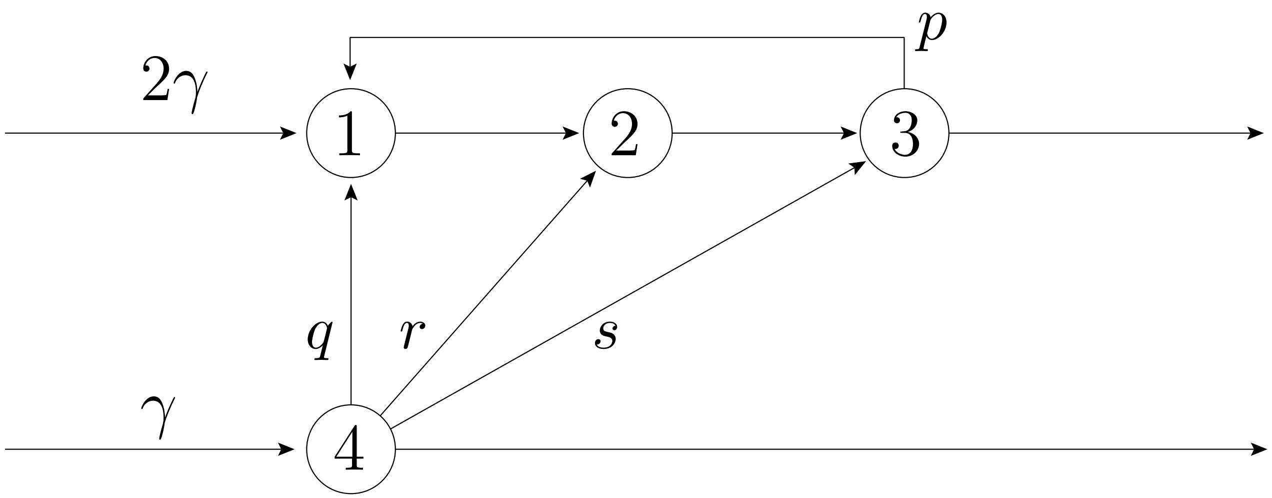


- Set up and solve traffic equations to find λ_i:

$$\vec{\lambda} = \begin{pmatrix} 2\gamma \\ 0 \\ \gamma \end{pmatrix} + \vec{\lambda} \begin{pmatrix} 0 & 1-p & p \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- i.e. $\lambda_1 = 2\gamma$, $\lambda_2 = (1-p)\lambda_1$, $\lambda_3 = \gamma + p\lambda_1$

OQN: Traffic Equations: example 2



- Set up and solve traffic equations to find λ_i:

$$\vec{\lambda} = \begin{pmatrix} 2\gamma \\ 0 \\ 0 \\ \gamma \end{pmatrix} + \vec{\lambda} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 0 \\ q & r & s & 0 \end{pmatrix}$$

OQN: Network stability

- Stability of network (whether it achieves steady-state) is determined by utilisation, $\rho_i < 1$ at every node i
- After solving traffic equations for λ_i, need to check that:

$$\rho_i = \frac{\lambda_i}{\mu_i} < 1 \quad : \forall i$$

Recall facts about M/M/1

- If λ is arrival rate, μ service rate then $\rho = \lambda/\mu$ is utilisation
- If $\rho < 1$, then steady state solution exists
- Average buffer length:

$$N = \frac{\rho}{1-\rho}$$

- Distribution of jobs in queue is:

$$\mathbb{P}(k \text{ jobs is queue at steady-state}) = (1-\rho)\rho^k$$

OQN: Jackson's Theorem

- Where node i has a service rate of μ_i, define $\rho_i = \lambda_i / \mu_i$
- If the arrival rates from the traffic equations are such that $\rho_i < 1$ for all $i = 1, 2, \ldots, n$, then the steady-state exists and:

$$\pi(r_1, r_2, \ldots, r_n) = \prod_{i=1}^{n} (1 - \rho_i)\rho_i^{r_i}$$

- This is a *product form* result!

OQN: Jackson's Theorem Results

- The marginal distribution of no. of jobs at node i is same as for isolated M/M/1 queue: $(1-\rho)\rho^k$
- Number of jobs at any node is independent of jobs at any other node – hence *product form* solution
- Powerful since queues can be reasoned about separately for queue length – summing to give overall network queue occupancy

OQN: Mean Jobs in System

- If only need mean results, we can use Little's law to derive mean performance measures
- Product form result implies that each node can be reasoned about as separate M/M/1 queue in isolation, hence:

$$\text{Av. no. of jobs at node } i = N_i = \frac{\rho_i}{1-\rho_i}$$

- Thus total av. number of jobs in system is:

$$N = \sum_{i=1}^{n} \frac{\rho_i}{1-\rho_i}$$

OQN: Mean Total Waiting Time

- Applying Little's law to whole network gives:

$$N = \tau W$$

  where τ is total external arrival rate, W is mean response time.

- So mean response time from entering to leaving system:

$$W = \frac{1}{\tau} \sum_{i=1}^{n} \frac{\rho_i}{1 - \rho_i}$$

OQN: Intermediate Waiting Times

- r_i represents the the average waiting time from arriving at node i to leaving the system
- w_i represents average response time at node i, then:

$$r_i = w_i + \sum_{j=1}^{n} q_{ij} r_j$$

- which as before gives a vector equation:

$$\begin{aligned} \vec{r} &= \vec{w} + Q\vec{r} \\ &= (I - Q)^{-1}\vec{w} \end{aligned}$$

OQN: Average node visit count

- v_i represents the average number of times that a job visits node i while in the network
- If τ represents the total arrival rate into the network, $\tau = \sum_i \gamma_i$:

$$v_i = \frac{\gamma_i}{\tau} + \sum_{j=1}^{n} v_j q_{ji}$$

- so for $\vec{\gamma}' = \vec{\gamma}/\tau$:

$$\begin{aligned} \vec{v} &= \vec{\gamma}' + \vec{v}Q \\ &= \vec{\gamma}'(I - Q)^{-1} \end{aligned}$$

OQN: Average node visit count

- Compare average visit count equations with traffic equations:

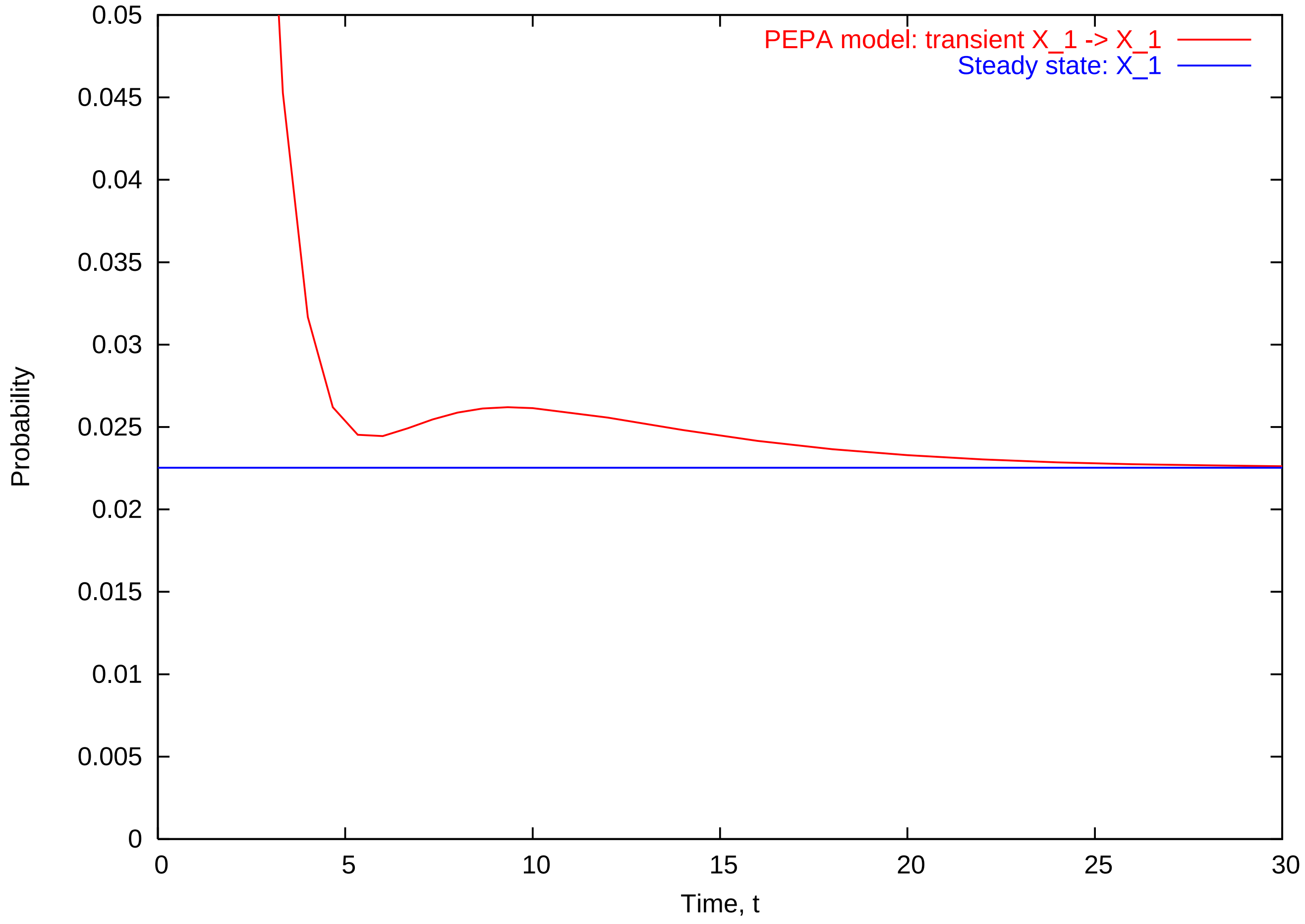
$$\begin{aligned}\vec{v} &= \vec{\gamma}'(I-Q)^{-1}\\ \vec{\lambda} &= \vec{\gamma}(I-Q)^{-1}\end{aligned}$$

- We can see that: $\vec{v} = \vec{\lambda}/\tau$, so if we have solved the traffic equations, we needn't perform a separate linear calculation

Transient Analysis of CTMCs

- What is transient analysis?
- Transient analysis finds, $\pi_i(t)$, the probability of being in a state i, at time t.
- For irreducible Markov chains, the limit of the transient probability is the steady-state probability for that state.

Transient Analysis of CTMCs



- Blue line: steady-state, π_{X_1}
- Red line: transient-state, $\pi_{X_1}(t)$

Transient Analysis: Notation

- $\{X(t) \ : \ t \geq 0\}$: the state of the MC at time t
- $p_{ij}(t) = \mathbb{P}(X(t) = j \mid X(0) = i\}$: probability of being in state j at time t, given that was in state i at time 0 (time-homogeneous)
- $\pi_j(t) = \mathbb{P}(X(t) = j)$: transient-state distn.

$$\pi_j(t) = \sum_i p_{ij}(t)\pi_i(0)$$

- π_j: steady-state probability of being in state j

$$\lim_{t\to\infty} p_{ij}(t) = \lim_{t\to\infty} \pi_j(t) = \pi_j$$

for irreducible Markov chains

Transient Analysis

- For a CTMC with generator matrix A with elements, a_{ij}
  - Transient equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{\pi}(t) = \vec{\pi}(t)A \quad (*)$$

  - At steady-state:

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{\pi}(t) = \vec{\pi}A = 0$$

where $\vec{\pi} = \{\pi_1, \pi_2, \cdots, \pi_N\}$,
$\vec{\pi}(t) = \{\pi_1(t), \pi_2(t), \cdots, \pi_N(t)\}$

Transient Analysis

- Solving equation $(*)$ gives:

$$\vec{\pi}(t) = \vec{\pi}(0)e^{At} \qquad (**)$$

where:

$$e^{At} = \sum_{k=0}^{\infty} \frac{(At)^k}{k!}$$

- Why not calculate $(**)$ directly?
  - A has negative and positive entries – numerically unstable
  - $\sum_{k=0}^{\infty}$ needs to be truncated

Transient Analysis

- Why not calculate $(**)$ directly?
  - A^k is computationally expensive and has fill-in for large k. If A is sparse, A^k will be dense!
- To get round first problem, we scale $\vec{\pi}(t)$ by $\vec{y}(t) = e^{qt}\vec{\pi}(t)$, where $q > \max_i(-a_{ii})$

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\vec{y}(t) &= e^{qt}\frac{\mathrm{d}}{\mathrm{d}t}\vec{\pi}(t) + qe^{qt}\vec{\pi}(t) \\ &= e^{qt}\vec{\pi}(t)A + qe^{qt}\vec{\pi}(t) \quad \text{:by eqn } (*) \\ &= e^{qt}\vec{\pi}(t)\underbrace{(A + qI)}_{\text{+ive diagonal elements}}\end{aligned}$$

Transient Analysis

- We get an equation analogous to $(*)$ in $\vec{y}(t)$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{y}(t) = \vec{y}(t)qA^*$$

where $A^* = A/q + I$

- for which the solution is:

$$\begin{aligned}
\vec{y}(t) &= y(0)e^{qA^*t} \\
e^{qt}\vec{\pi}(t) &= \vec{\pi}(0)e^{qA^*t} \\
\vec{\pi}(t) &= \vec{\pi}(0)\sum_{k=0}^{\infty}\frac{(qt)^k e^{-qt}}{k!}A^{*k}
\end{aligned}$$

Transient Analysis

- Now let $\vec{\theta}(k) = \vec{\theta}(k-1)A^*$ and $\vec{\theta}(0) = \vec{\pi}(0)$
- This prevents having to calculate ${A^*}^k$ directly and having fill-in
- Our final formula for the transient state probability is:

$$\vec{\pi}(t) = \sum_{k=0}^{\infty} \vec{\theta}(k) \frac{(qt)^k e^{-qt}}{k!}$$

- Summation can be truncated effectively
- Number iterations: $O(qt)$

Uniformization: Interpretation

- A^* is a DTMC transition matrix, so $\vec{\theta}(k) = \vec{\theta}(k-1)A^*$ is kth transition vector
- Constructing A^* from A can be seen as sampling the CTMC at regular intervals
- The probability of being in a given CTMC state at one of these sample times is dictated by the DTMC
- The time taken between state changes can be seen as a *uniformized* exponential distribution of rate, q

Transient Analysis

$$\vec{\pi}(t) = \sum_{k=0}^{\infty} \vec{\theta}(k) \frac{(qt)^k e^{-qt}}{k!}$$

- This can be interpreted as:

$$\begin{aligned} & \mathbb{P}(\text{in state } i \text{ at time, } t) \\ = & \sum_k \mathbb{P}(\text{in state } i \mid k \text{ transitions}) \cdot \mathbb{P}(\text{num. transitions} = k) \end{aligned}$$

- If $X \sim \mathrm{Poisson}(qt)$, number of exponential transitions of rate q in a time period, t:

$$\mathbb{P}(X = k) = \frac{(qt)^k e^{-qt}}{k!}$$