Lecture 9: Queueing networks

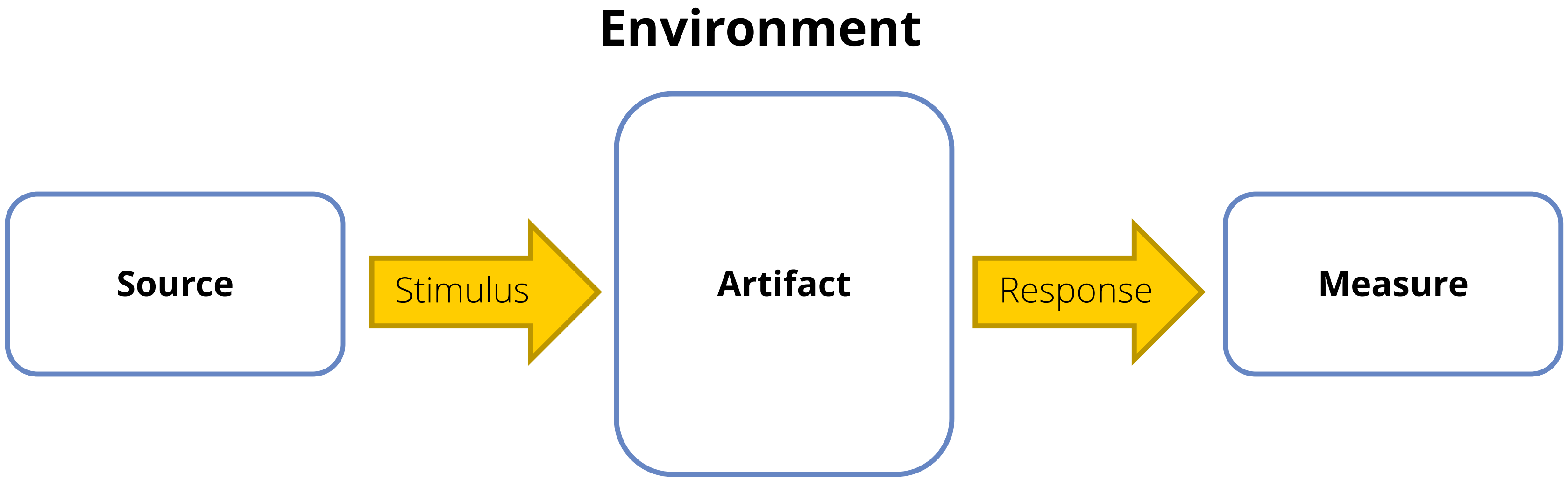
Utrecht University

For today

- 09:00 – 09:45: Queueing networks I
- 10:00 – 10:30: Queueing networks II
- 10:30 – 11:00: Exercise on queueing networks
- 11:00 – 12:35: Assignment time
- 12:35 – 12:45: Wrap Up

Addressing quality attributes

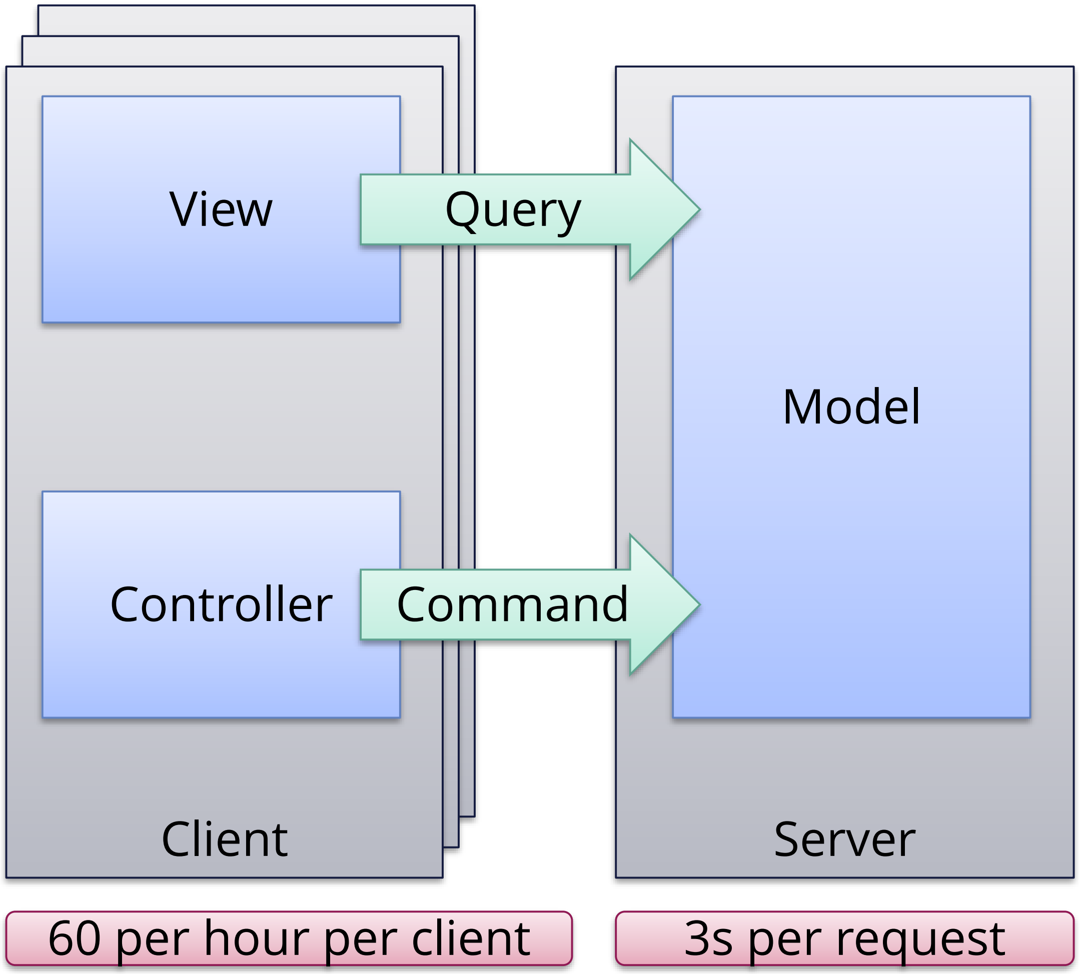
- Create general scenario as a context view:

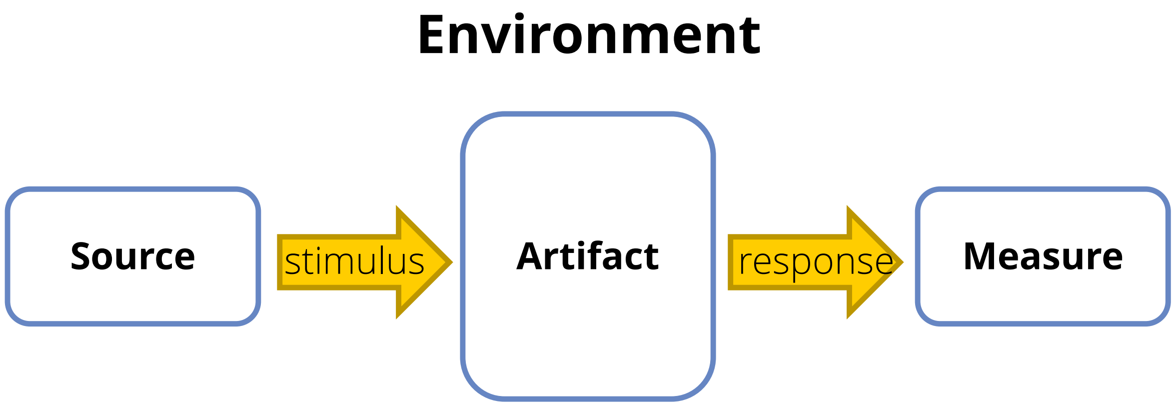


- Address in each view how the property is satisfied
- Quality attributes can **and will** change views!

Utrecht University

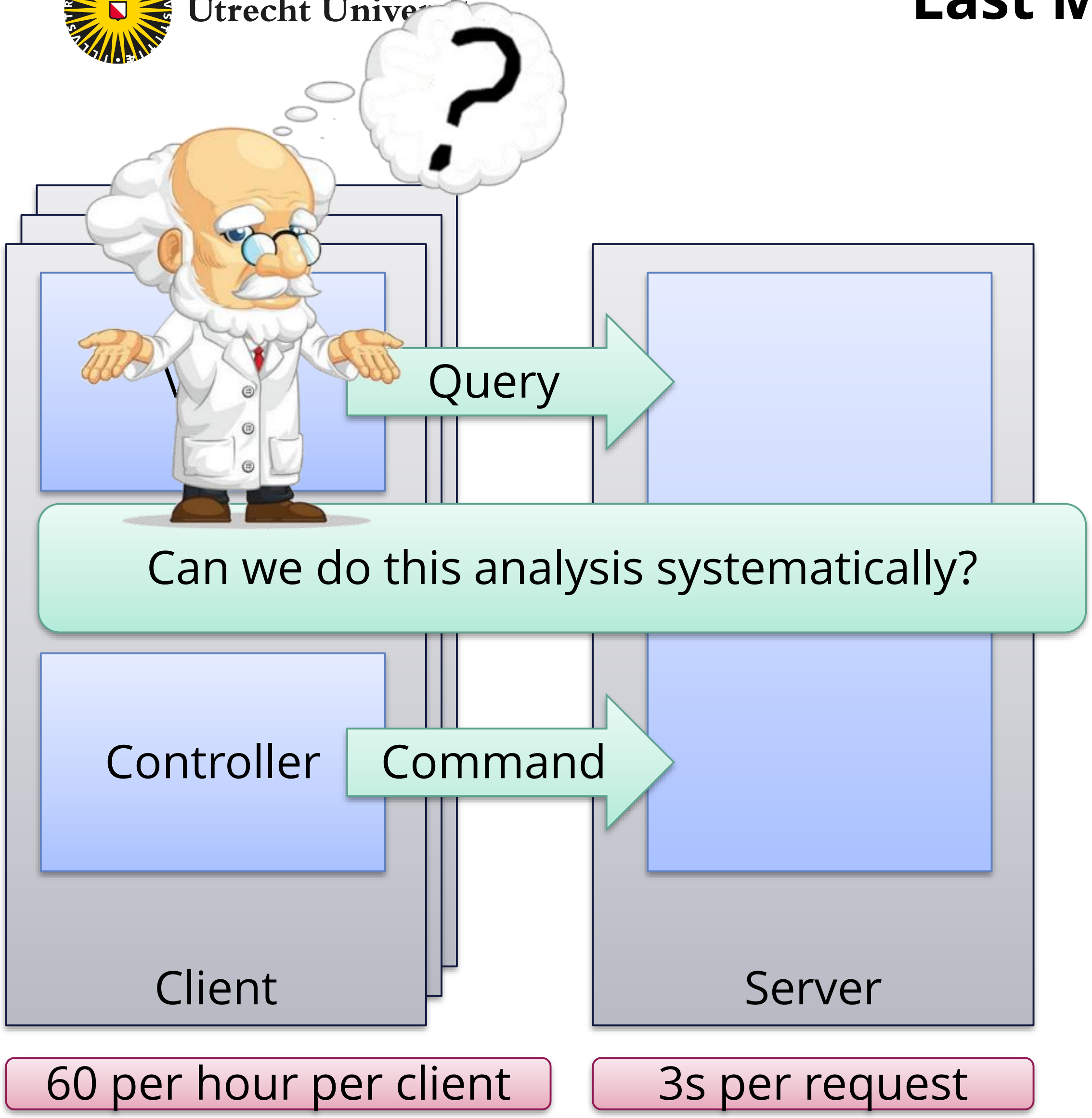
Last Monday: back-of-the-envelope analysis



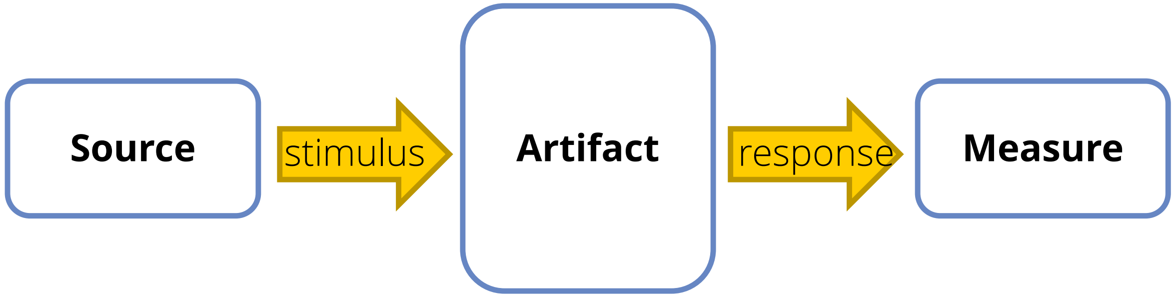


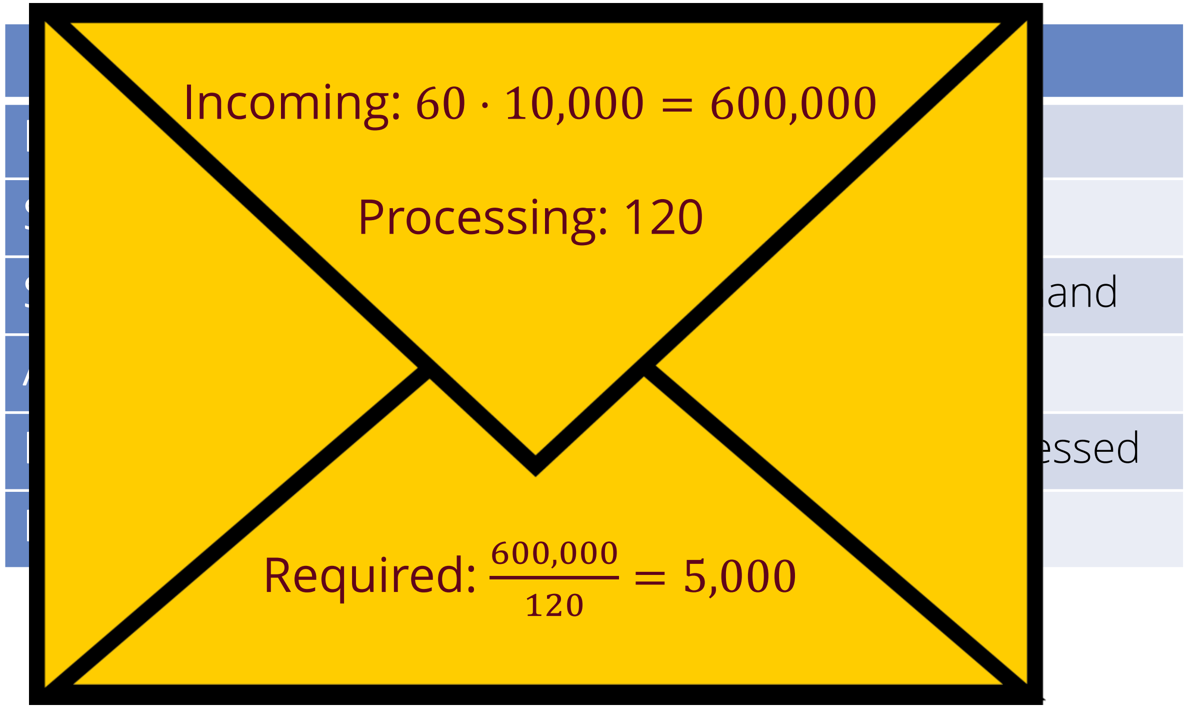
	QS 1	QS 2
Environment	Running system	Running system
Source	User	User
Stimulus	Initiates a query	Initiates a command
Artifact	System	System
Response	Gives result	Command processed
Measure	Latency	Latency

Last Monday: back-of-the-envelope analysis



Environment





Preliminaries

Utrecht University

“I will win this time!”

- “My favourite numbers have not been drawn for too long! At least two weeks!”
- “It is for the law of large numbers. I am sure!”

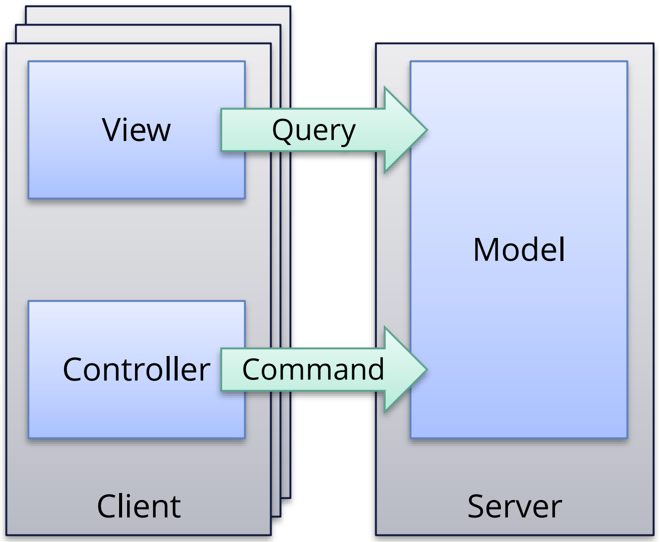
You shouldn’t be

Memorylessness!

Besides, the law of large numbers states that, given a sample of independent and identically distributed values, the empirical mean converges towards the expected value.

Queueing networks

Queueing networks



- Each element is handled on a first come, first served principle: FIFO

- Arrival rate:
  how many enter the queue per time unit?
- Service rate:
  how many can you process per time unit?

- These are probability distributions:
  What is the chance that an element arrives at moment X?
  What is the service time of a unit?

abazon

- On average, a new order is handled by Abazon in a second and a half.
- What is the probability we handle the next order somewhen between 4 and 6 seconds?
- Between 2 and 3 seconds?
- Between 8 and 9 seconds?
- Between 0.5 and 2?

Rate
$\lambda = 1.5$

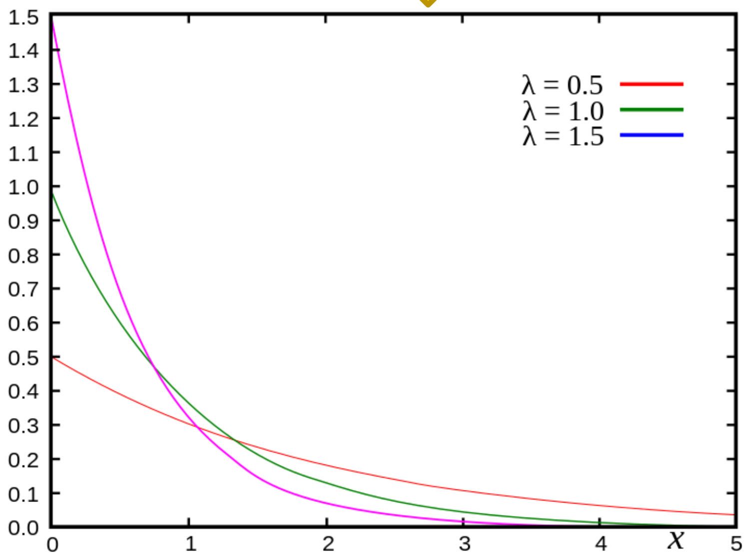
To answer these questions, we should compute the area (read: the integral of the function over an interval) underneath the curve

Probability density function

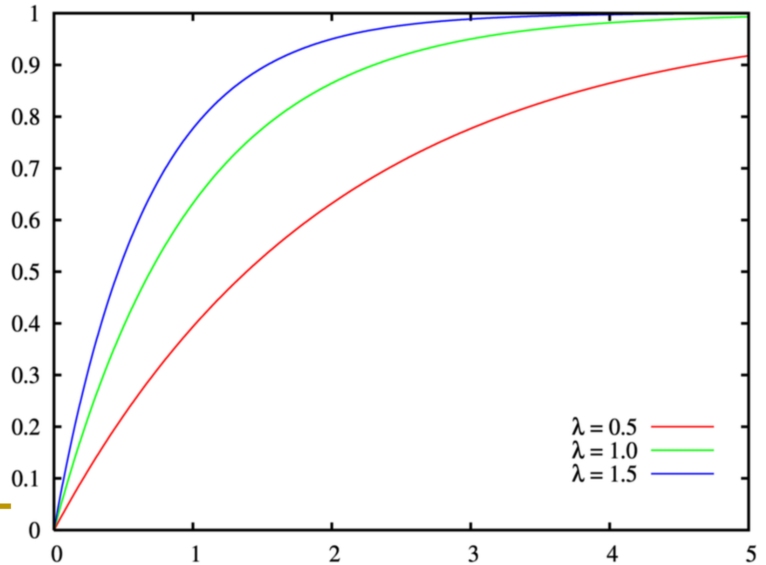
$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & \text{with } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Cumulative distribution function

$$F(x;\lambda) = \begin{cases} 1 - e^{-\lambda x} & \text{with } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

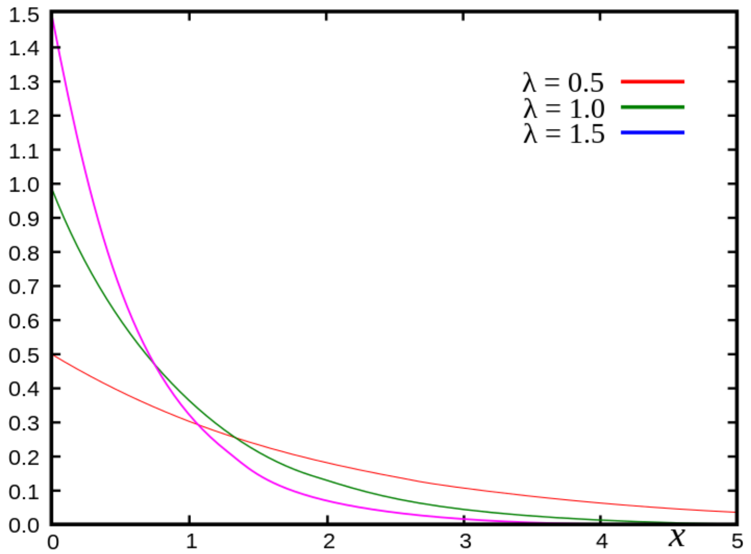


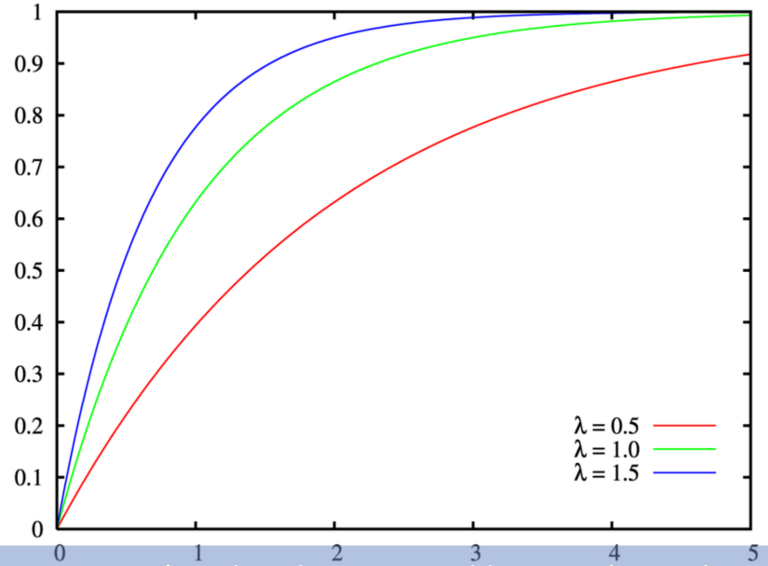
- In less than 2 seconds?



Utrecht University

Exponential distribution





- Exponential distribution
  - **Rate parameter:** λ
  - **Expected value:** $E[X] = \frac{1}{\lambda}$
  - **Variance:** $\sigma^2[X] = \frac{1}{\lambda^2}$

$$P(X > s + t \mid X > s) = P(X > t) \text{ for all } s, t \geq 0$$

We call this the **memorylessness property**:

Chance of next occurrence of an event is **independent of** previous occurrences

Image sources: (top by Cburnett and bottom by D. Ilyin, via Wikimedia Commons) https://commons.wikimedia.org/wiki/

Poisson process

- Let $N(t)$ be the amount of orders in interval $[0, t]$
  Suppose the oreder rate is **exponentially distributed**
  Parameter: λ

- Then:
  $N(t)$ is a **Poisson distribution** with **parameter:** $\boldsymbol{\lambda \cdot t}$
  Expected value: $E[N(t)] = \lambda \cdot t$
  Variance: $\sigma^2[N(t)] = \lambda \cdot t$

- By approximation:

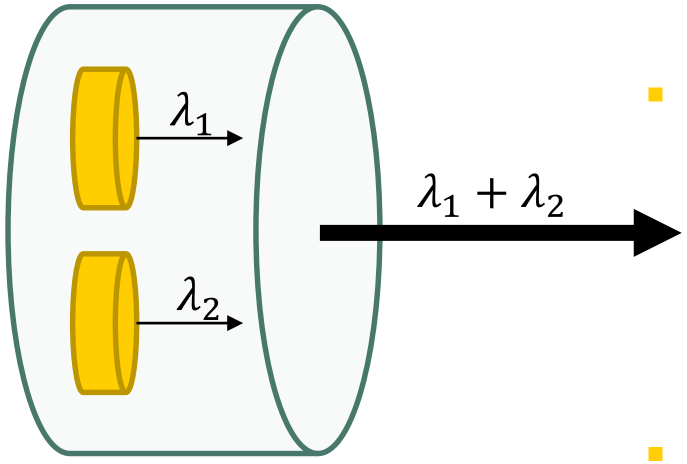
$$P[\text{Order in } (t, \Delta\text{t})] \approx \lambda \cdot \Delta\text{t}$$

Generally not true,
Only holds for exponential distribution!

Do you recognize the **memorylessness property**?

Utrecht University

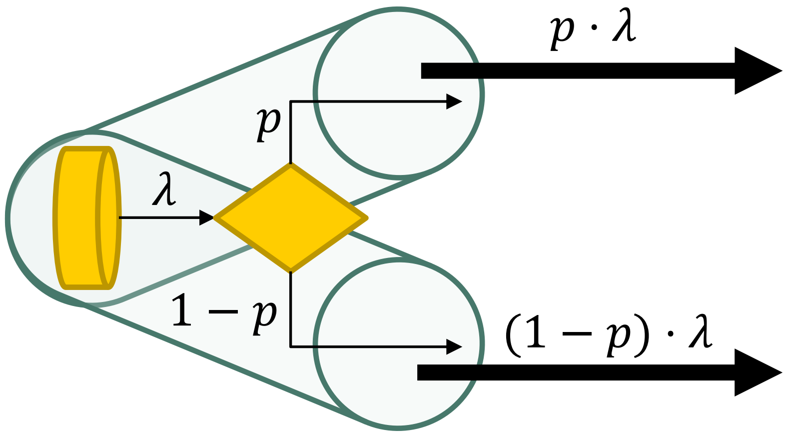
Two properties of Poisson processes



- Merging:

  Let $N_1(t)$ be a **Poisson process** with **arrival rate** λ_1
  Let $N_2(t)$ be a **Poisson process** with **arrival rate** λ_2

  Then: $N_1(t) + N_2(t)$ is a **Poisson process** with **arrival rate** $\lambda_1 + \lambda_2$

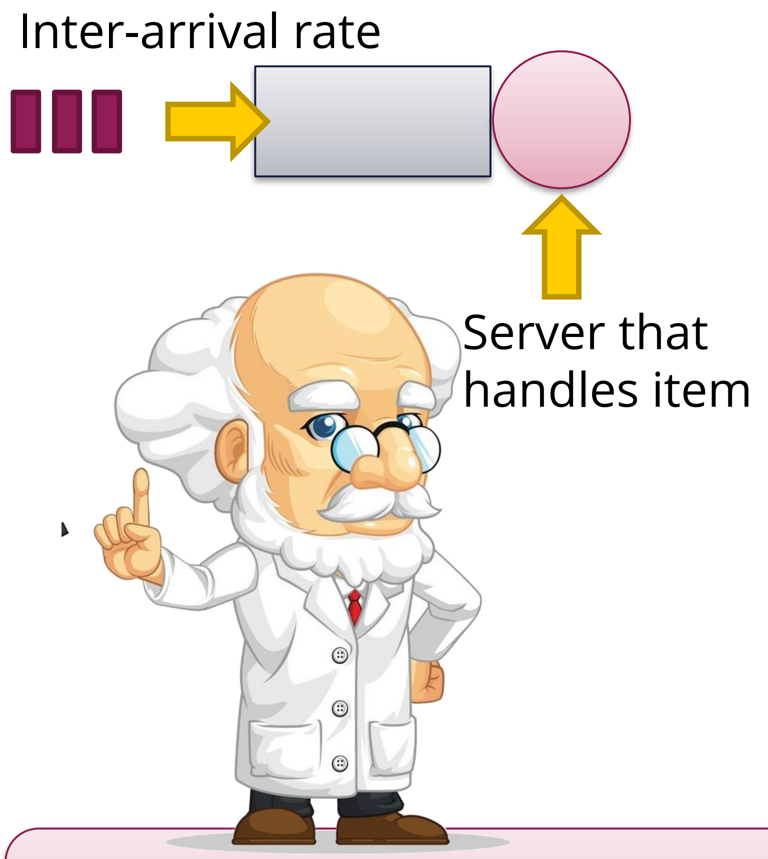


- Splitting:

  Let $N(t)$ be a Poisson process with **arrival rate λ**
  Mark every served order independently with chance p

  Then:
  Marked served orders form a **Poisson process** with **arrival rate** $p \cdot \lambda$
  Unmarked served orders form a **Poisson process** with **arrival rate** $(1 - p) \cdot \lambda$

Queues combine these ideas!



We typically use M/M/c queues

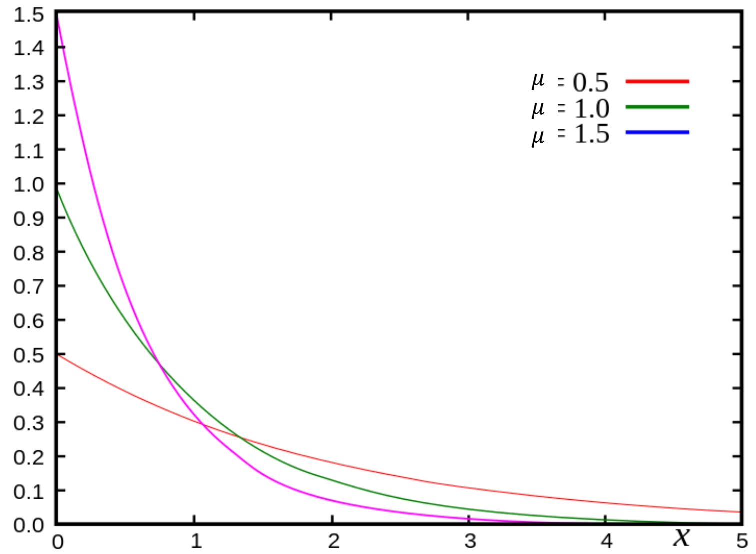
- Queues are FIFO: First in, First out
  - **Inter-arrival rate:** how many enter the queue in a time unit?
  - **Service rate:** how many can you process in a time unit?

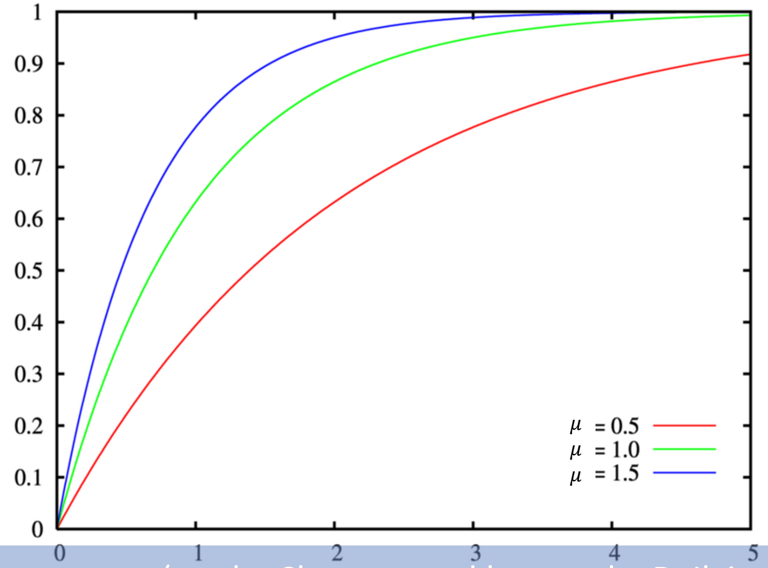
  These are probability distributions!

- Kendall's notation: A/B/c
  - **A:** type of distribution for **inter-arrival rate**
  - **B:** type of distribution **for service time**
  - **c: number of servers** (workers / handlers / ...)

- Typical values in Kendall's notation:
  - **M: memoryless** (here, exponential) or **Markovian**
  - **G: general** distribution
  - **D: deterministic**

Exponential distribution





- Exponential distribution
  - **Rate parameter:** μ
  - **Expected value:** $E[X] = \frac{1}{\mu}$
  - **Variance:** $\sigma^2[X] = \frac{1}{\mu^2}$

For forward-compatibility, let us rebrand λ with μ

$$P(X > s + t \mid X > s) = P(X > t) \text{ for all } s, t \geq 0$$

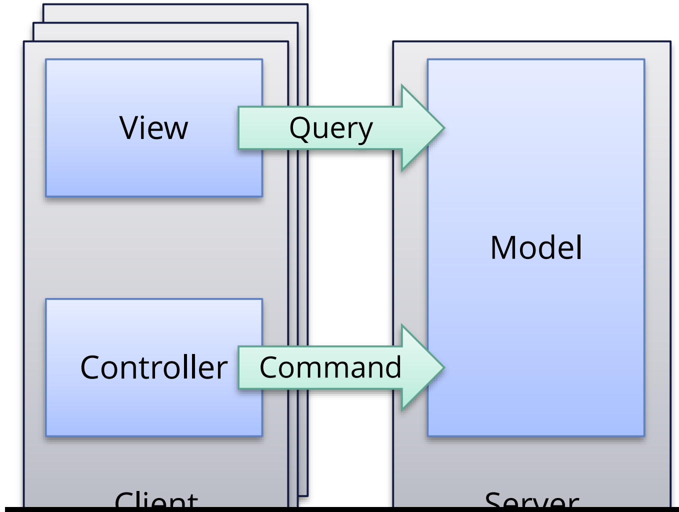
We call this the **memorylessness property**:

Chance of next occurrence of an event is **independent** of previous occurrences

Image sources: (top by Cburnett and bottom by D. Ilyin, via Wikimedia Commons) https://commons.wikimedia.org/wiki/

Utrecht University

M/M/c queues



- Notation for M/M/c queues:
  Arrival rate: λ
  Service rate: μ
- Occupation rate: ρ
  Amount of work per time unit

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

ρ should always be smaller than 1.
Try with 0.8!

Incoming: $60 \cdot 10{,}000 = 600{,}000$
Processing: 120

Our aim:
$\rho = 1$

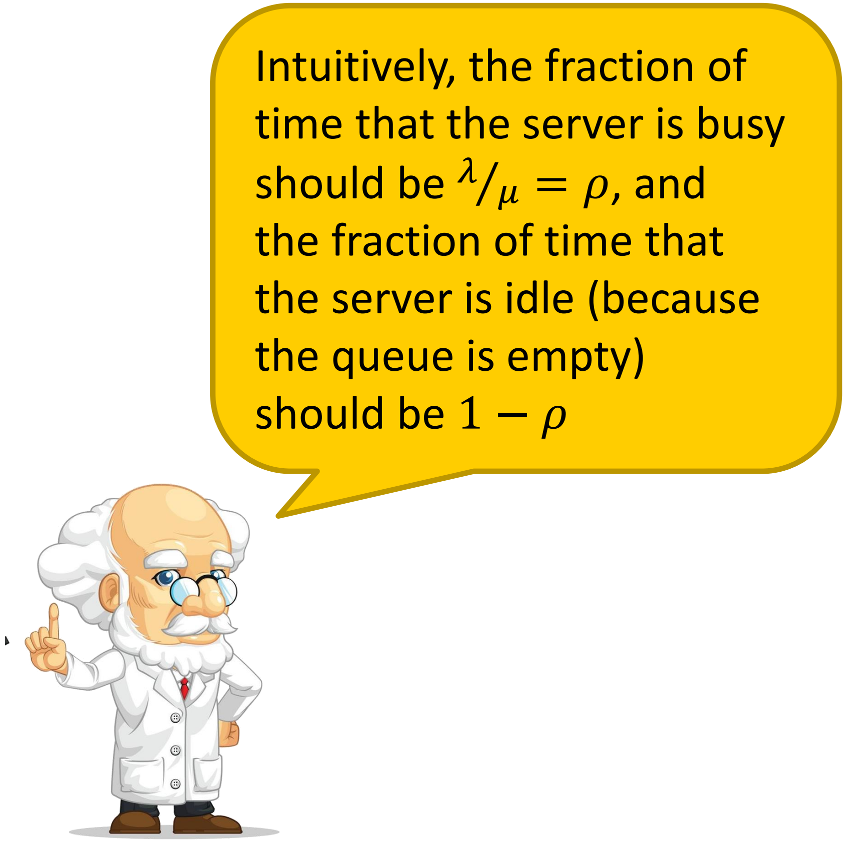
Required: $\frac{600{,}000}{120} = 5{,}000$

Arrival rate: 600,000
Service rate: 120

Occupation rate: $\rho = \frac{600{,}000}{c \cdot 120}$

$c = 5000 \Rightarrow \rho = 1$

M/M/1 queues



- Notation for M/M/1 queues:
  Arrival rate: λ
  Service rate: μ

- Occupation rate: ρ
  Amount of work per time unit

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

Probability of n **elements in the node** in steady state:

$P(N = n) = (1 - \rho) \cdot \rho^n$ for $n \in [0, \infty)$

Probability of **0 elements in the node** in steady state:

$P(N = 0) = 1 - \rho$ for $n \in [0, \infty)$

Utrecht University

M/M/1 queue performance characteristics

Notice that the average nr of elements in a node is λ times the soujourn time:
$E[N] = \lambda \cdot S$

If you do not trust the above, prove it!

- We call a queue together with a server a **node**.
- If we have **one server**, formulae are "easy"

ρ should always be smaller than 1! The model is considere d *stable* only if $\lambda < \mu$ here as $\rho = \frac{\lambda}{\mu}$ with $c = 1$

- Average nr of elements in a node: $\mathrm{E[N]} = \frac{\rho}{\boxed{1-\rho}}$
- Average nr of elements in the queue: $\frac{\rho^2}{\boxed{1-\rho}}$
- Average waiting time of an element in the queue: $\frac{\rho}{\mu-\lambda}$
- Average waiting time of an element in a node: $\mathrm{S} = \frac{1}{\mu-\lambda}$

We call this **sojourn time!**

Incoming: $60 \cdot 10{,}000 = 600{,}000$
Processing: 120

$\rho < 1$

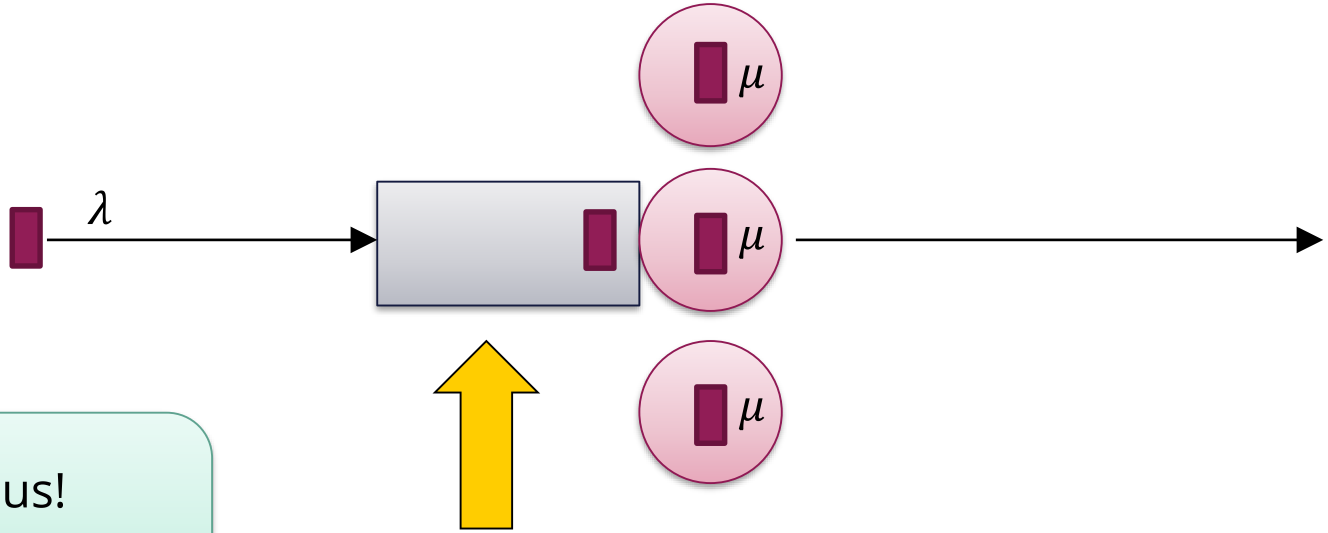
Required: $\frac{600{,}000}{120} = 5{,}000$

Queue length: measure of performance seen from the viewpoint of the system operator.
Expected waiting time in the queue; sojourn time: more relevant from the point of view of a customer.

Utrecht University

M/M/c queues

- What happens if there are c servers?



Good news is, there is a calculator for us!

www.staff.science.uu.nl/~werf0006/queues/

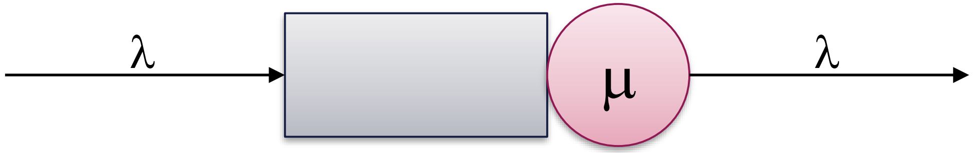
You have to wait
if all servers are “busy”

Queue length, given c servers and occupation rate ρ:

$$\frac{(c\rho)^c}{c!}\left((1-\rho)\sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!}+\frac{(c\rho)^c}{c!}\right)^{-1}\cdot\frac{\rho}{1-\rho}$$

Queuing networks (1)

- M/M/c queues have a nice property:



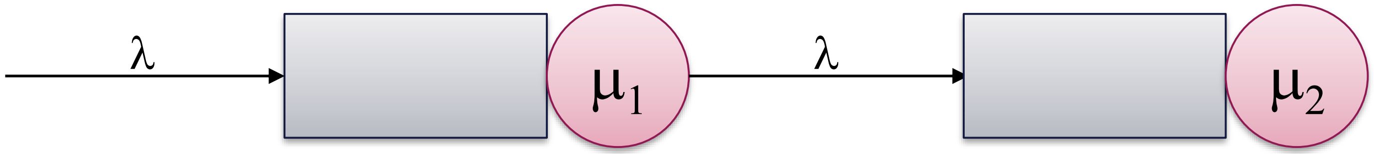
- Lemma:

If **the incoming arrival rate follows a Poisson process with rate λ**
and **the service time is exponentially distributed**
then **the outgoing arrival rate follows a Poisson process with rate λ**

Utrecht University

Queuing networks (2)

- Consequence:

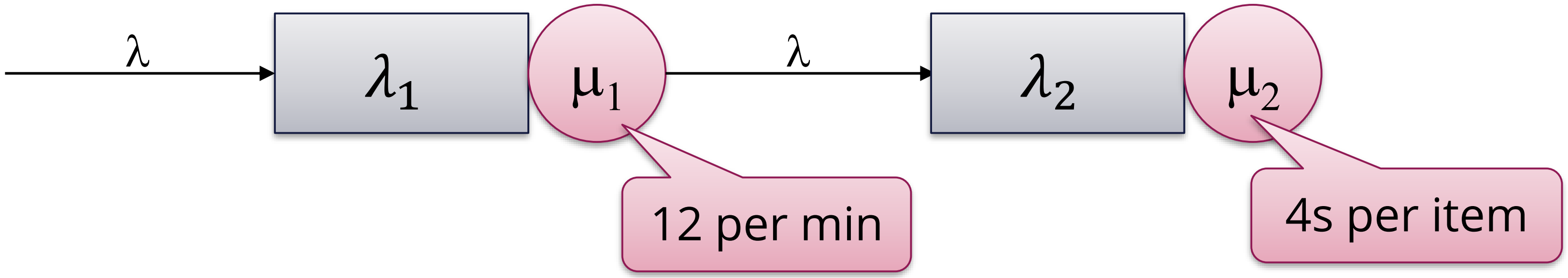


- For each node, the previous formulae hold!

Generally not true.
Only holds for the exponential distribution!

Queuing networks (2): an example

- Arrival rate: $\lambda = 10$ per minute

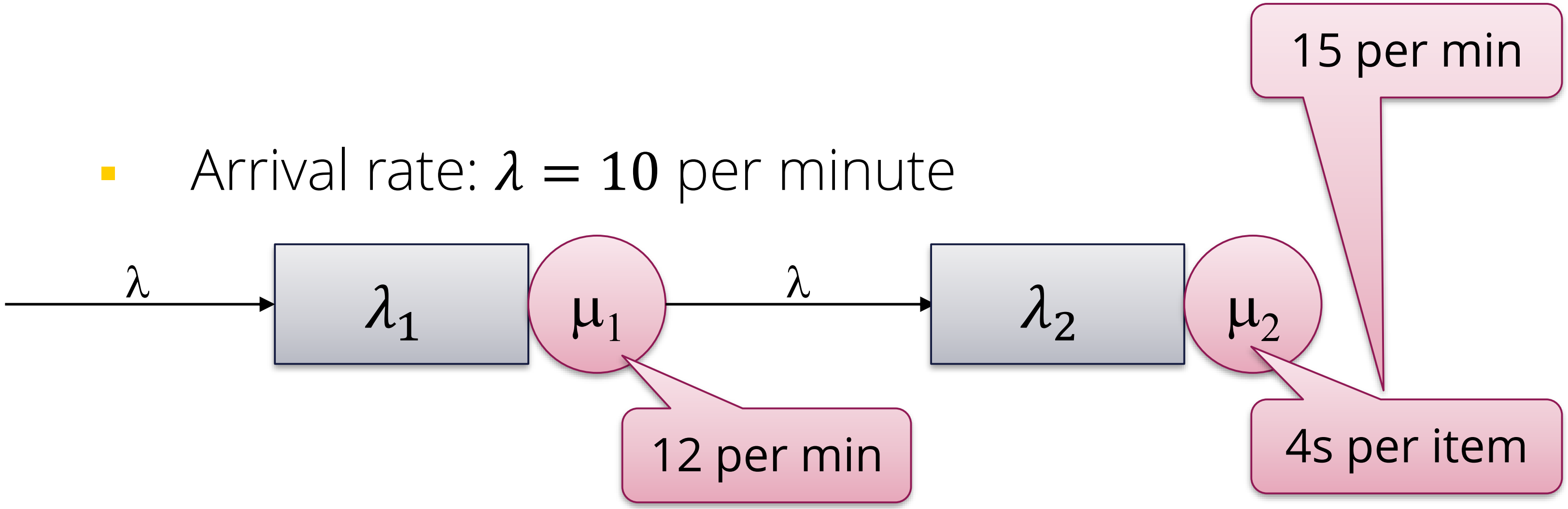


- Expected sojourn time:

- Utilisation (per node):

Queuing networks (2): an example

- Arrival rate: $\lambda = 10$ per minute



- Expected sojourn time:

$$E[S] = \sum_{i=1}^{2} \frac{1}{\mu_i - \lambda_i} = \frac{1}{12-10} + \frac{1}{15-10} = \frac{1}{2} + \frac{1}{5} = \frac{7}{10} min = 42s$$

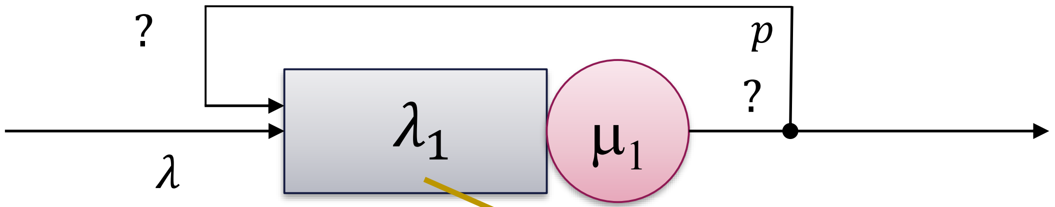
- Utilisation (per node):

$$\rho_1 = \frac{\lambda_1}{\mu_1} = \frac{10}{12} \qquad \rho_2 = \frac{\lambda_2}{\mu_2} = \frac{10}{15}$$

Utilization **only** per node!

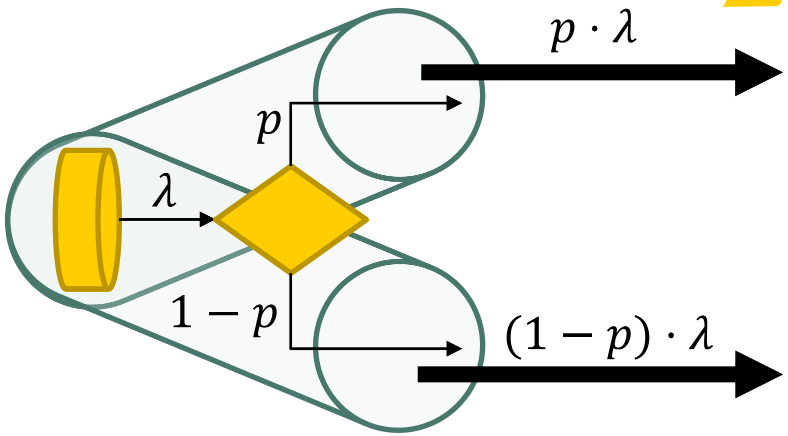
How about self loops?

- Given the following network

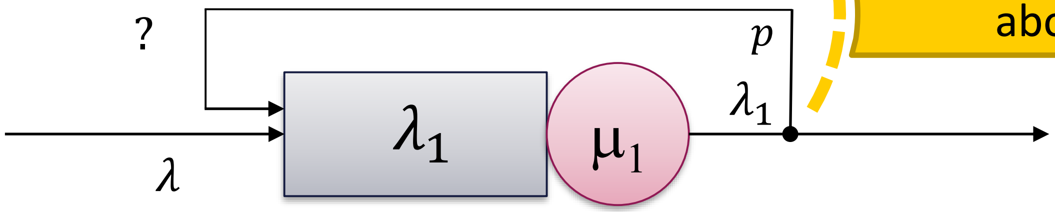


No, it is not just λ because the input is manifold!

How about self loops?

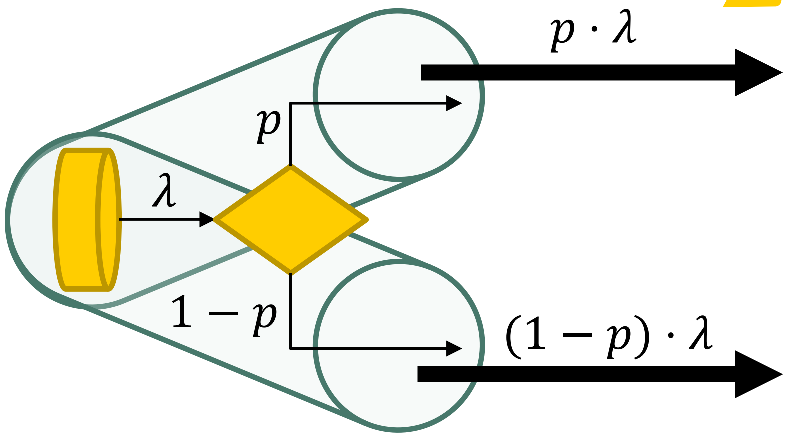


- Given the following network

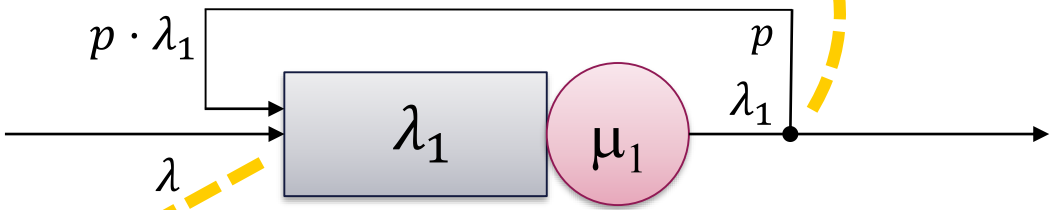


Remember the property about splitting

How about self loops?



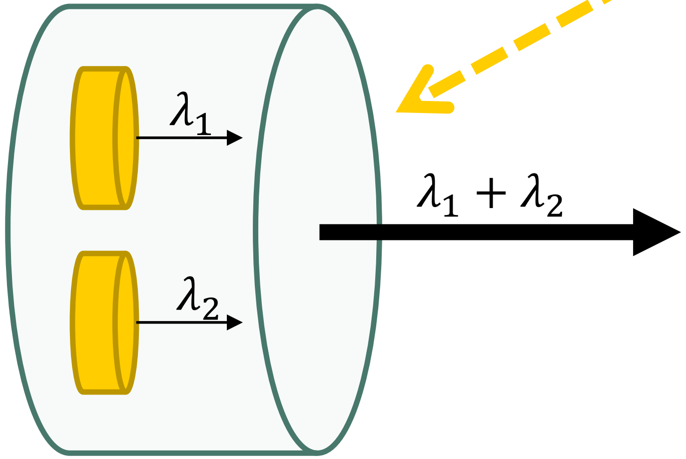
- Given the following network



Remember the property about merging

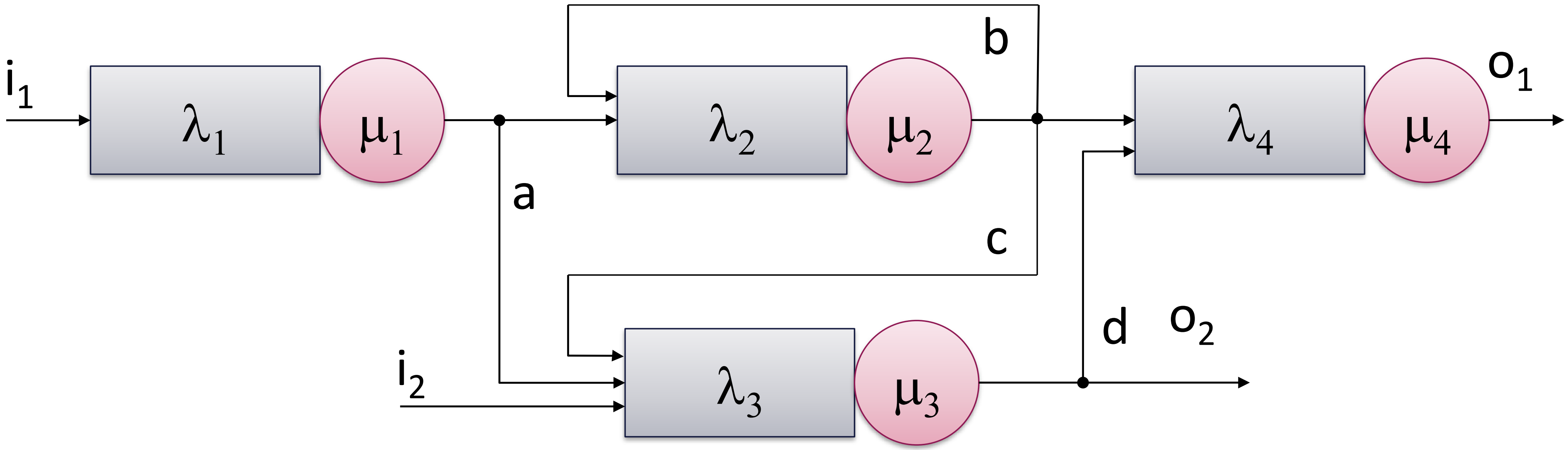
- How can we calculate the arrival rates?

$$\left.\begin{aligned} \lambda_1 &= \lambda + p \cdot \lambda_1 \\ \Rightarrow \lambda_1 - p \cdot \lambda_1 &= \lambda \\ \Rightarrow (1-p)\lambda_1 &= \lambda \end{aligned}\right\} \Rightarrow \lambda_1 = \frac{\lambda}{1-p}$$



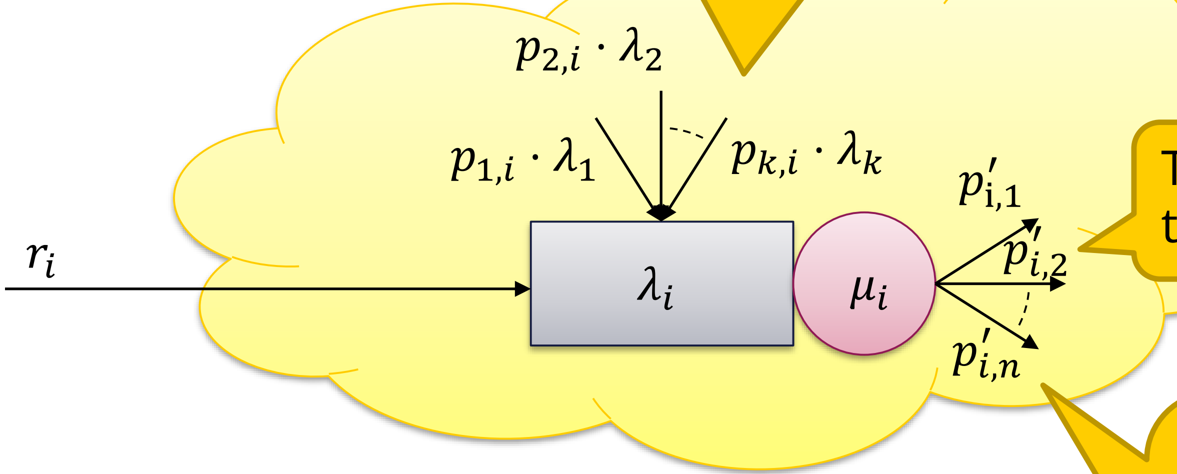
Utrecht University

How to handle this one in a structured way?



Queueing networks: let us generalize!

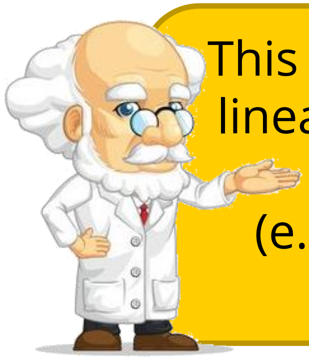
From k other nodes in the network



To n other nodes in the network

$$\sum_{j=1}^{n} p'_{i,j} = 1$$

$$\Rightarrow p'_{i,n} = 1 - \sum_{j=1}^{n-1} p'_{i,j}$$

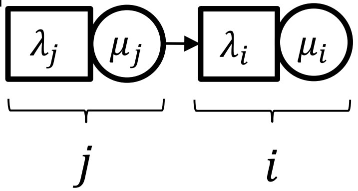


This gives us a system of linear equations we can solve (e.g., using Gaussian elimination)

For each node i: $\lambda_i = r_i + \sum_{j=1}^{k} (p_{j,i} \cdot \lambda_j)$

Arrival rate from a source outside the network

$p_{j,i}$: probability that a packet streams from node j to node i



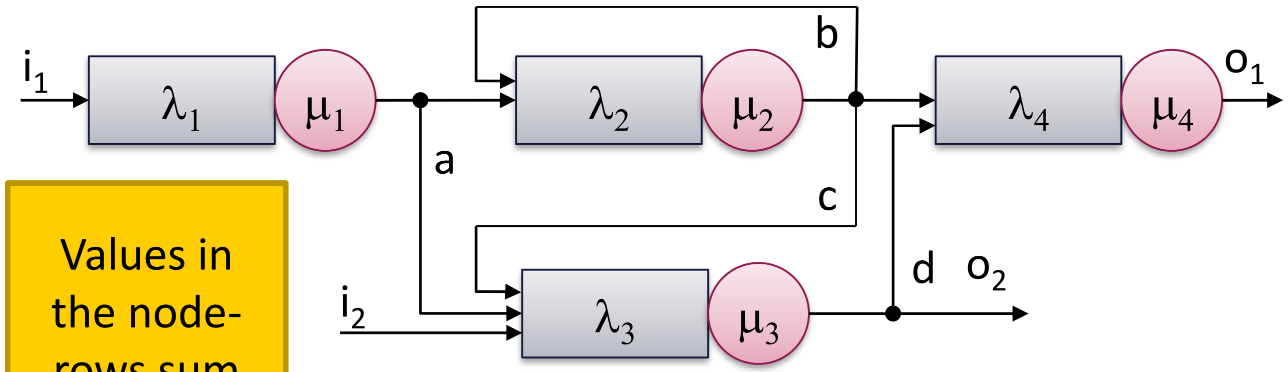
1: Create an adjacency matrix

Arc’s head

From / to	Node 1	Node 2	Node 3	Node 4	o1	o2
Input						
Node 1						
Node 2						
Node 3						
Node 4						

Arc’s tail

Values in the node-rows sum up to 1



$$\sum_{j=1}^{n} p'_{i,j} = 1$$

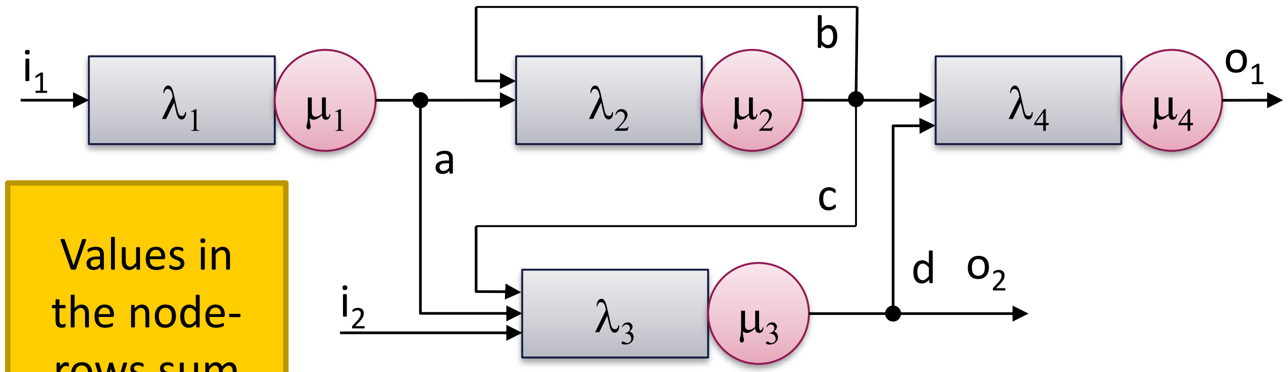
$$\Rightarrow p'_{i,n} = 1 - \sum_{j=1}^{n-1} p'_{i,j}$$

Utrecht University

1: Create the adjacency matrix

From / to	Node 1	Node 2	Node 3	Node 4	o1	o2
Input	i_1		i_2			
Node 1		(1-a)	a			
Node 2		b	c	(1-b-c)		
Node 3				d		(1-d)
Node 4					1	

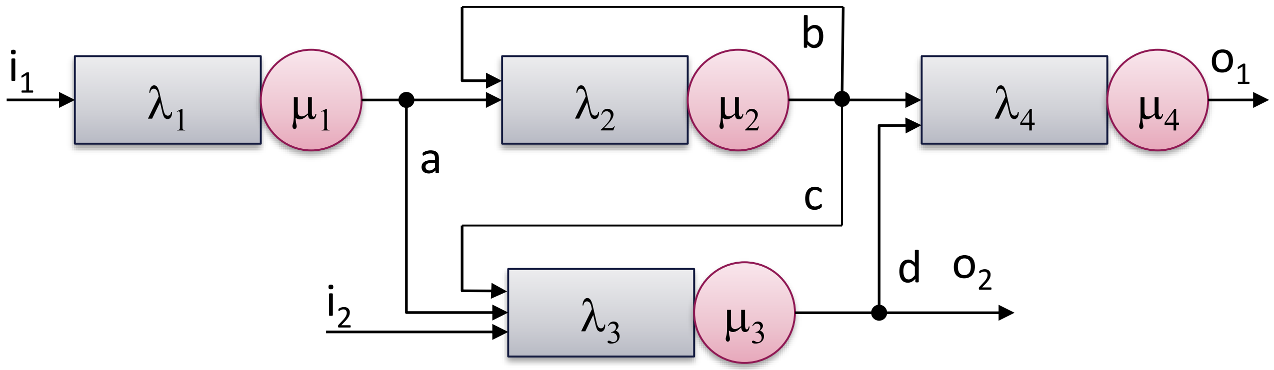
Values in the node-rows sum up to 1



Utrecht University

2: Create the equation system for arrival rates

From / to	Node 1	Node 2	Node 3	Node 4	o1	o2
Input	i_1		i_2			
Node 1		(1-a)	a			
Node 2		b	c	(1-b-c)		
Node 3				d		(1-d)
Node 4					1	



$$
\begin{cases}
\lambda_1 = i_1 \\
\lambda_2 = (1-a)\lambda_1 + b\lambda_2 \\
\lambda_3 = i_2 + a\lambda_1 + c\lambda_2 \\
\lambda_4 = (1-b-c)\lambda_2 + d\lambda_3
\end{cases}
$$

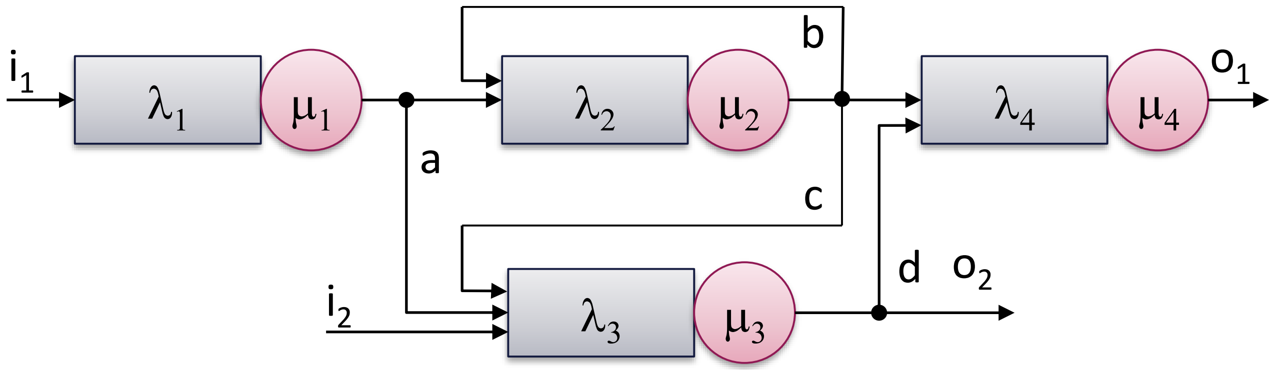
For each node i: $\lambda_i = r_i + \sum_{j=1}^{k} (p_{j,i} \cdot \lambda_j)$

We take the probabilities from the columns in the adjacency matrix!

Utrecht University

Step 2: create equation system for arrival rates

From / to	Node 1	Node 2	Node 3	Node 4	o1	o2
Input	i_1		i_2			
Node 1		(1-a)	a			
Node 2		b	c	(1-b-c)		
Node 3				d		(1-d)
Node 4					1	



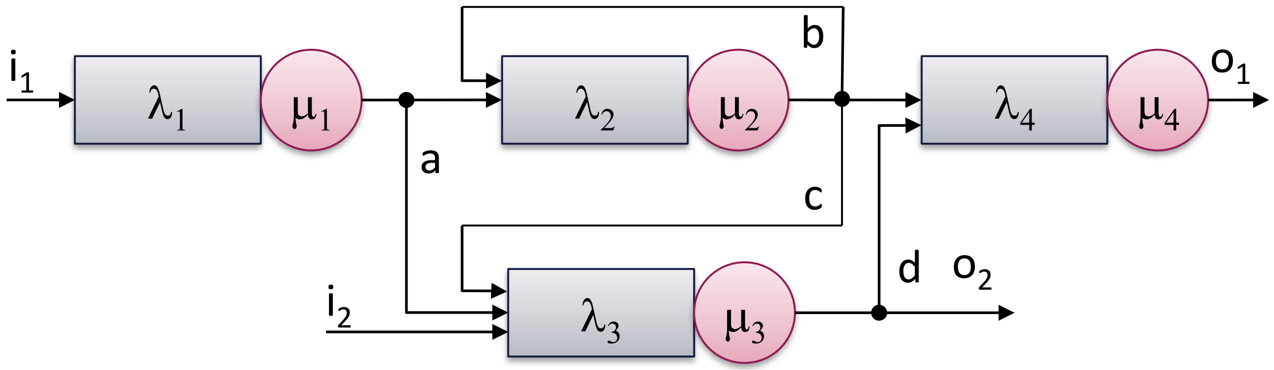
$$\begin{cases} \lambda_1 = i_1 \\ \lambda_2 = (1-a)\lambda_1 + b\lambda_2 \\ \lambda_3 = i_2 + a\lambda_1 + c\lambda_2 \\ \lambda_4 = (1-b-c)\lambda_2 + d\lambda_3 \end{cases}$$

By rewriting, we obtain:

$$\begin{cases} \lambda_1 = i_1 \\ \lambda_2 = \dfrac{(1-a)\cdot i_i}{(1-b)} \\ \lambda_3 = i_2 + a \cdot i_1 + \dfrac{c\cdot(1-a)\cdot i_i}{(1-b)} \\ \lambda_4 = \dfrac{(1-b-c)\cdot(1-a)\cdot i_i}{(1-b)} + d\cdot\left(i_2 + a\cdot i_1 + \dfrac{c\cdot(1-a)\cdot i_i}{(1-b)}\right) \end{cases}$$

Step 2: create equation system for arrival rates

From / to	Node 1	Node 2	Node 3	Node 4	o1	o2
Input	i_1		i_2			
Node 1		(1-a)	a			
Node 2		b	c	(1-b-c)		
Node 3				d		(1-d)
Node 4					1	



$$\begin{cases} \lambda_1 = i_1 \\ \lambda_2 = (1-a)\lambda_1 + b\lambda_2 \\ \lambda_3 = i_2 + a\lambda_1 + c\lambda_2 \\ \lambda_4 = (1-b-c)\lambda_2 + d\lambda_3 \end{cases}$$

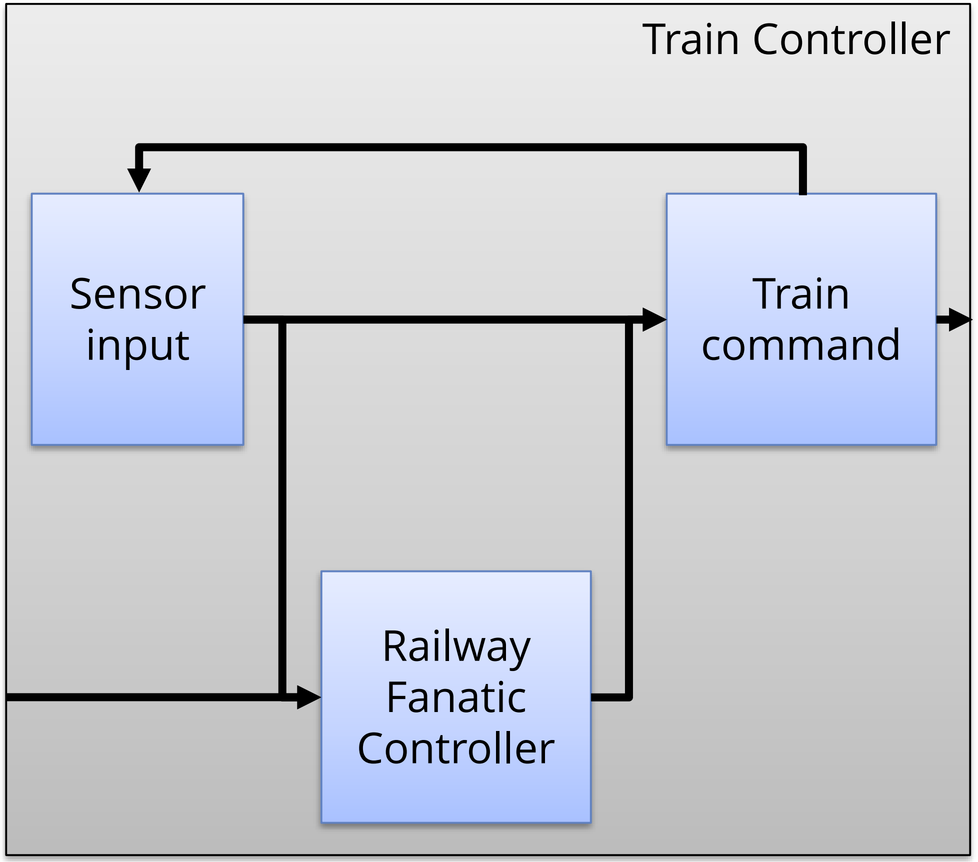
By rewriting, we obtain:

$$\begin{cases} \lambda_1 = i_1 \\ \lambda_2 = \dfrac{1-a}{1-b} \cdot i_1 \\ \lambda_3 = i_2 + \dfrac{a \cdot (1-b) + c \cdot (1-a)}{1-b} \cdot i_1 \\ \lambda_4 = \dfrac{(1-b-c+d \cdot c) \cdot (1-a) + d \cdot a \cdot (1-b)}{1-b} \cdot i_1 + d \cdot i_2 \end{cases}$$

Oh, what a beautiful *theory!*
How can we apply this in practice?

Utrecht University

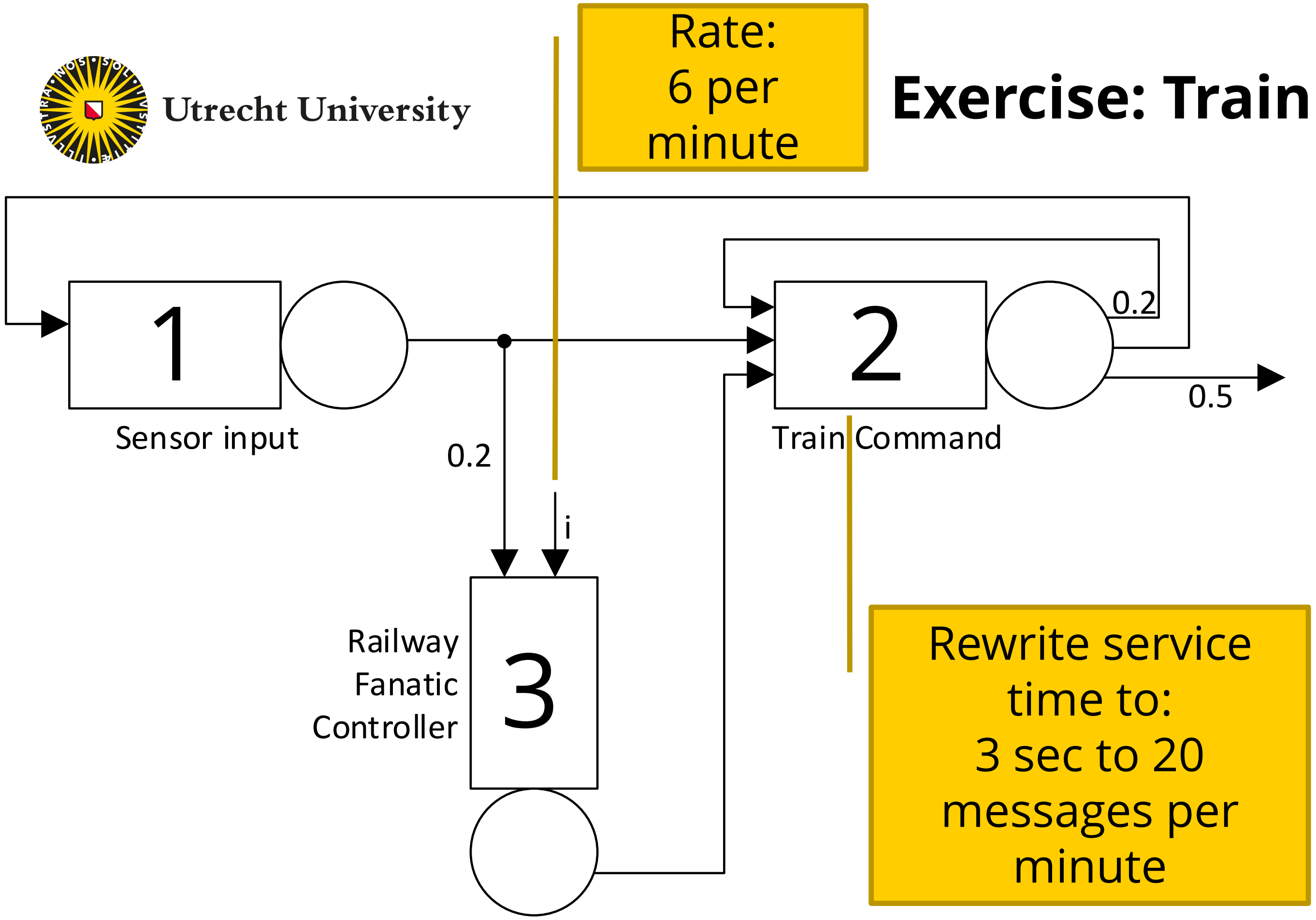
Exercise: Train Controller module



What is the utilization of the Train Command?

- The architects need to analyse the **utilisation of the Train Command module**. They assume all components to be M/M/1.
- They know that 50% of the messages to Train Command are handled in time. Of the other half, 20% is routed back to the Train Command, and the remainder is sent to the Sensor Input node.
- 20% of the Sensor Input is used by the Railway Fanatic Controller, whereas the remainder is directly fed in the Train Command.
- Each output of the Railway Fanatic Controller leads to a message to the Train Command.
- The architects assume that handling a Train Command message takes about 3 seconds. The railway fanatics are so fanatic that they send 6 commands per minute

Exercise: Train Controller module

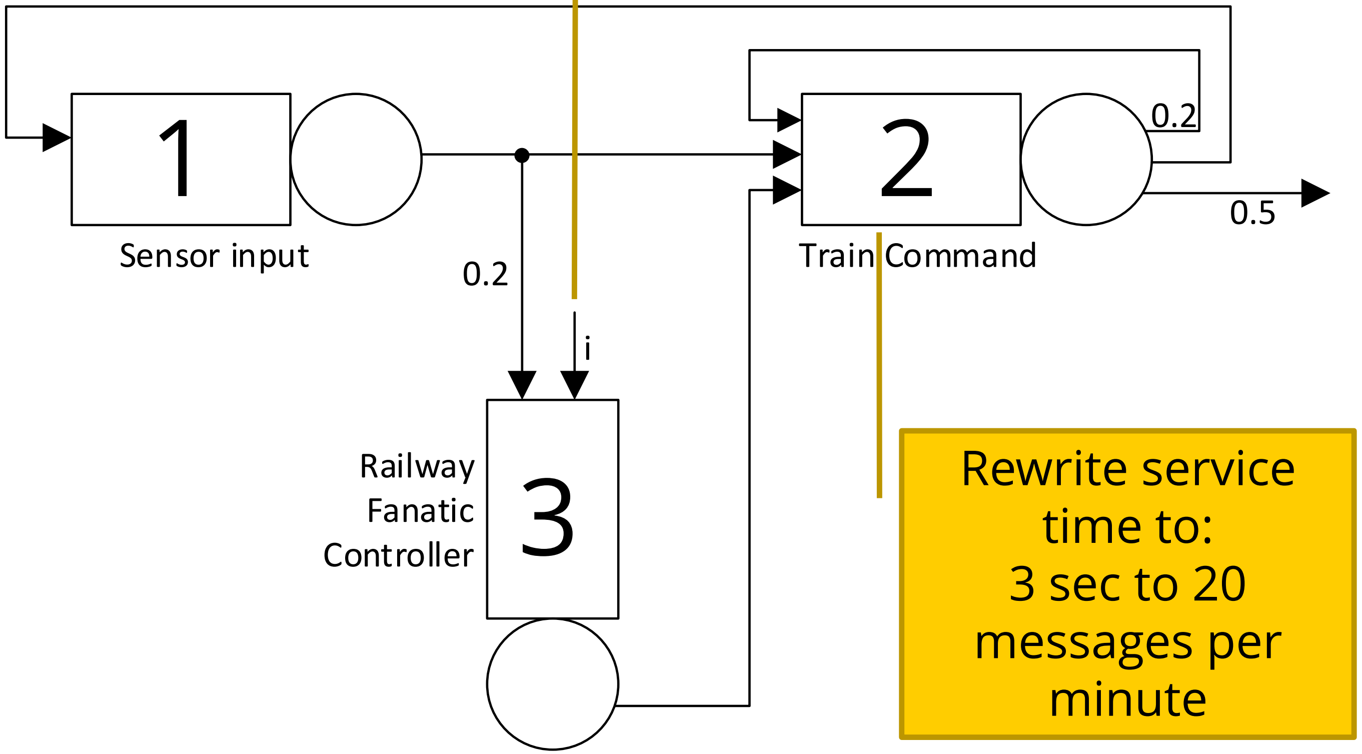


From / to	Node 1	Node 2	Node 3	Out
Input				
Node 1				
Node 2				
Node 3				

$$\rho_2 = ?$$

Exercise: Train Controller module

Rate:
6 per
minute



Rewrite service time to:
3 sec to 20 messages per minute

From / to	Node 1	Node 2	Node 3	Out
Input			i	
Node 1		0.8	0.2	
Node 2	0.3	0.2		0.5
Node 3		1		

$$\begin{cases} \lambda_1 = 0.3\lambda_2 \\ \lambda_2 = 0.8\lambda_1 + 0.2\lambda_2 + \lambda_3 \\ \lambda_3 = 0.2\lambda_1 + i \end{cases} \Rightarrow \begin{cases} \lambda_1 = 0.3\lambda_2 \\ \lambda_2 = \dfrac{i}{0.5} = 2i \\ \lambda_3 = 0.2\lambda_1 + i \end{cases}$$

$$\rho_2 = \frac{\lambda_2}{\mu_2} = \frac{2 \cdot 6}{20} = 0.6$$

What would happen if the fanatics sent **12 commands per minute?**

The model (and the system) would be unstable!

Utrecht University

Klock et al. 2017: What is the main message?

Utrecht University

For today

- 09:00 – 09:45: Queueing networks I
- 10:00 – 10:30: Queueing networks II
- 10:30 – 11:00: Exercise on queueing networks
- 11:00 – 12:35: Assignment time
- 12:35 – 12:45: Wrap Up

Utrecht University

Next Lecture: Monday

- Read papers:
  - (F) **M. Shaw, D. Garlan (1995). Formulations and formalisms in software architecture.**
  - (G) **L. de Alfaro, Th. Henzinger. Interface Automata**
  - (H) **T. Murata (1989). Petri nets: Properties, Analysis and Applications.**

The information in this presentation has been compiled with the utmost care,
but no rights can be derived from its contents.

© Utrecht University