22|11|23

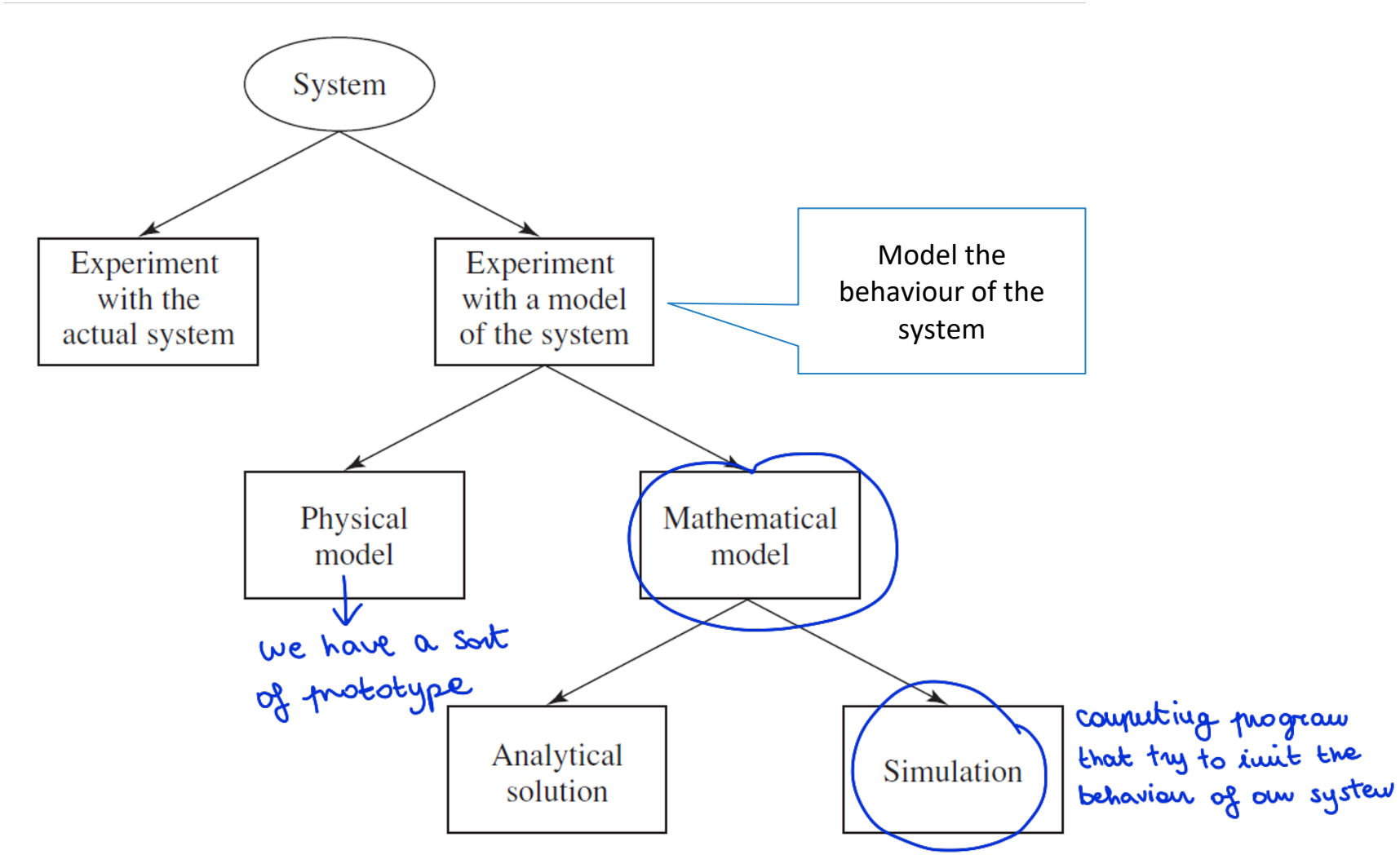
Dependable Distributed Systems Master of Science in Engineering in Computer Science

AA 2023/2024

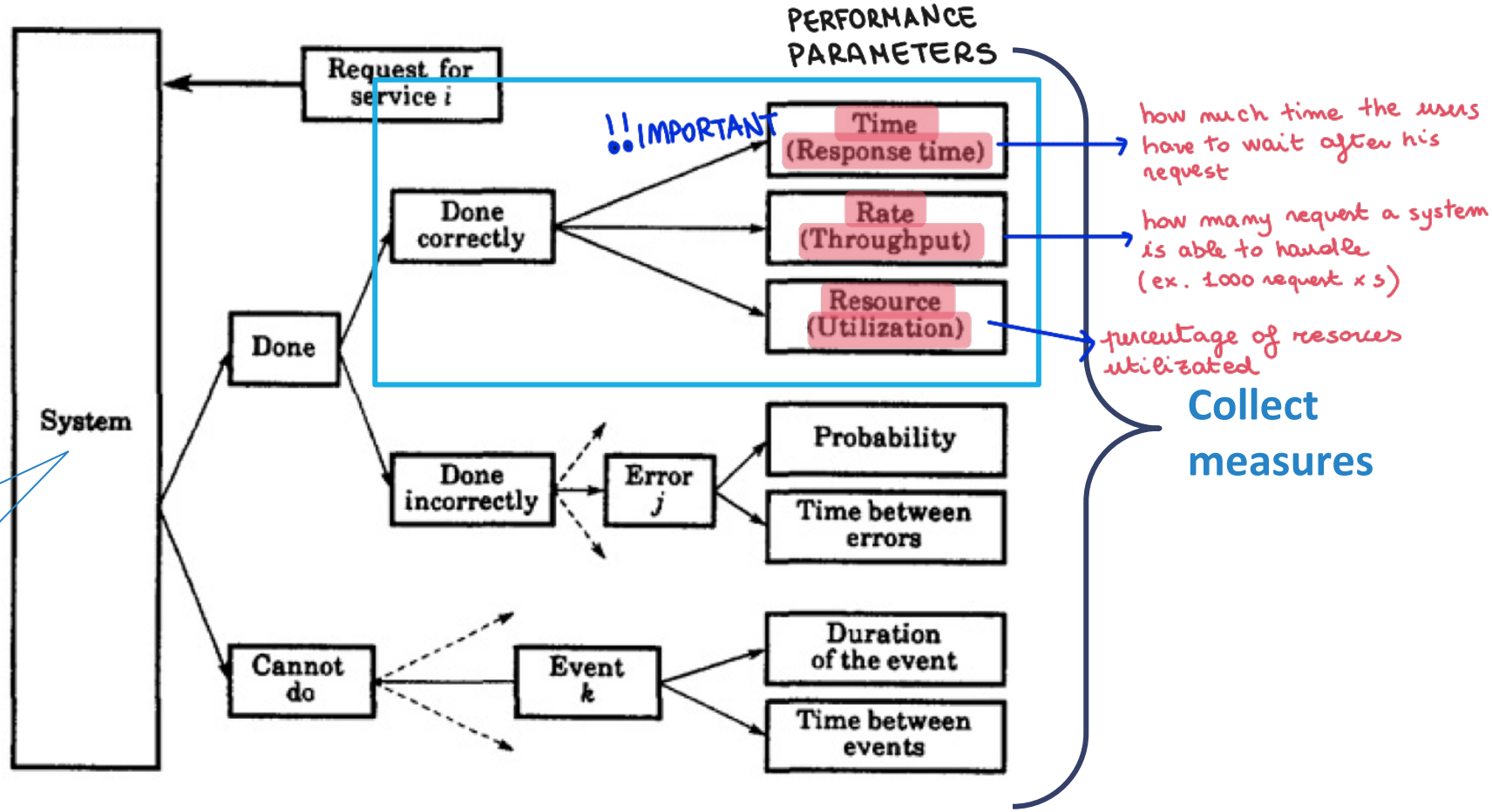
LECTURE 23 : BUILDING A PERFORMANCE MODEL 1 – OPERATIONAL LAWS AND QUEUEING THEORY

Schema

Recall: Ways to evaluate a system



Recall: Very Basics for Dependability Evaluation



Note that it is assumed that the system is available for measurements!

Introduction

Performance evaluation is required at every stage in the life cycle of a computer system:

- **Design**: compare alternatives designs and find the best design
- **Operation**: determining how well it is performing and whether any improvements need to be made
- **Maintenance**: compare systems and decide which one is best for a given set of applications

NOTE: **the types of applications of computers are so numerous** that **it is not possible to have** a standard measure of performance, a standard measurement environment (application), or a **standard technique for all cases**

To measure the performance of a computer system, you need at least two tools:

- a tool to load the system (**load generator**)
- a tool to measure the results (**monitor**)

How to do performance evaluation?

- Measurement (Benchmarking)
- Analytical modeling
- Simulation modeling

load the system and make some measurement

performance; is a function with INPUT param and you take the response time

↓
program;

Measurements are possible **only if** the **system already exists**

If it is a new concept, analytical modeling and simulation are the only techniques from which to choose

It would be **more convincing** to others **if** the analytical modeling or simulation is **based on previous measurement**

How to do performance evaluation?

- Measurement (Benchmarking)
- Analytical modeling
- Simulation modeling

Criterion	Analytical Modeling	Simulation	Measurement
1. Stage	Any	Any	Postprototype
2. Time required	Small	Medium	Varies
3. Tools	Analysts	Computer languages	Instrumentation
4. Accuracy[a]	Low	Moderate	Varies
5. Trade-off evaluation	Easy	Moderate	Difficult
6. Cost	Small	Medium	High
7. Saleability	Low	Medium	High

Performance Models

A model is **an abstraction** of a generalized overview of a real system

The level of detail of the model and the specific aspects of the real system that are considered in the model **depends on the purpose** of the model

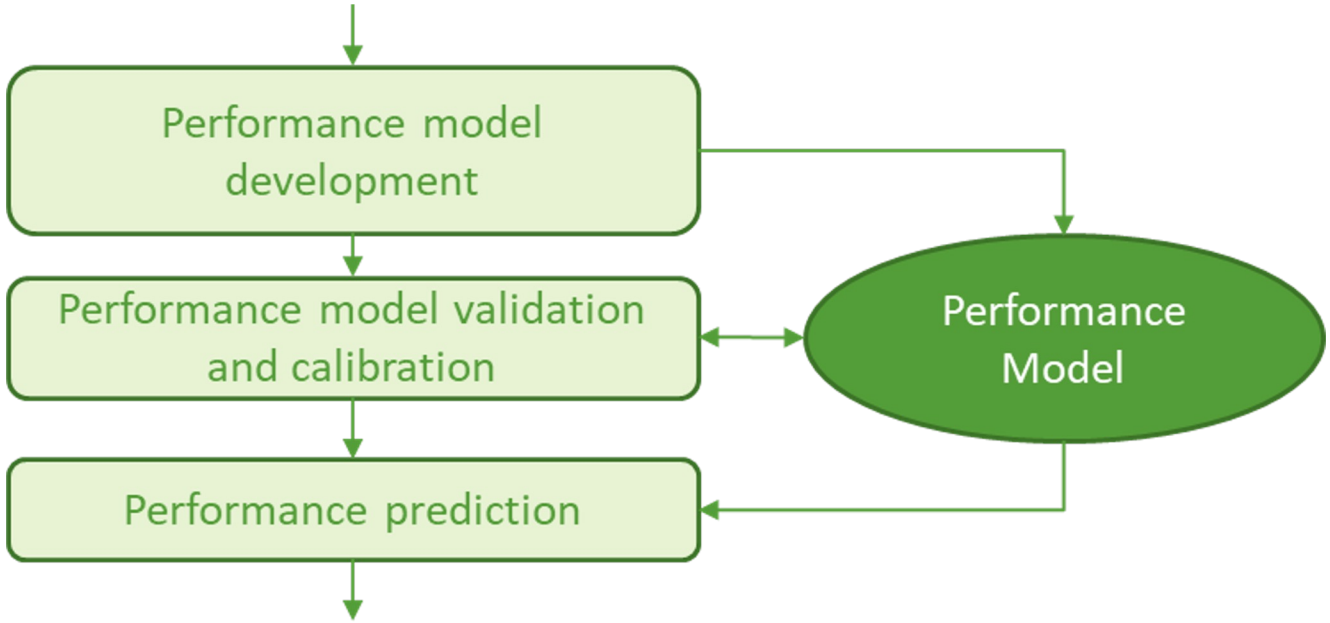
→ should not be more complex than is necessary to achieve its goal

- **Analytical** models: **set of formulas** and/or computational algorithms
  - Lower level of detail -> Less accurate but more efficient
- **Simulation** models: **computer programs**, all **resources and the dataflow are simulated**
  - Higher level of detail -> Expensive to run, develop and validate

There are some system behaviours that analytic models cannot (or very poor) capture

Performance model

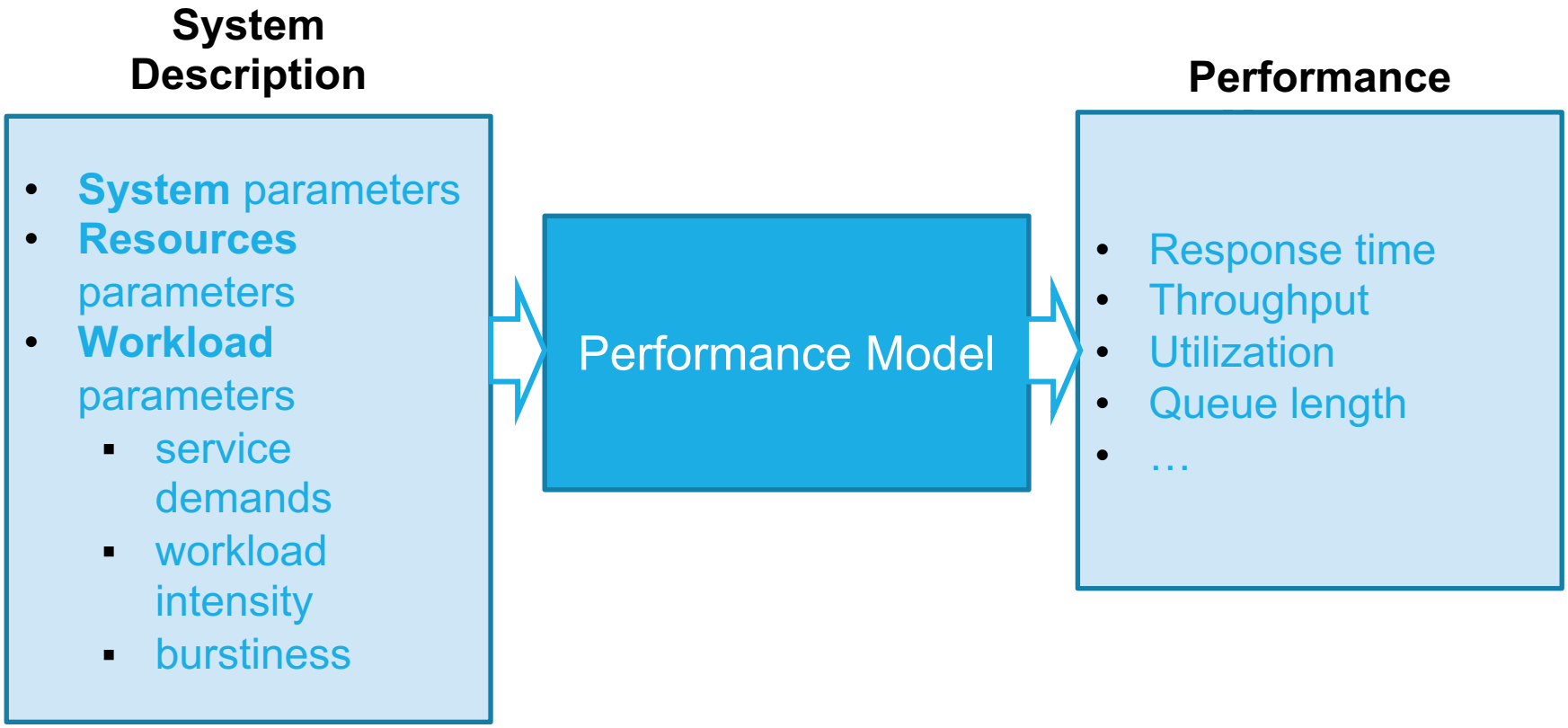
Used to predict performance as function of system description and workload parameters



Estimates performance measures of a computer system for a given set of parameters

Outputs: response times, throughputs, system resources utilizations, queue lengths, etc.

Estimating Performance

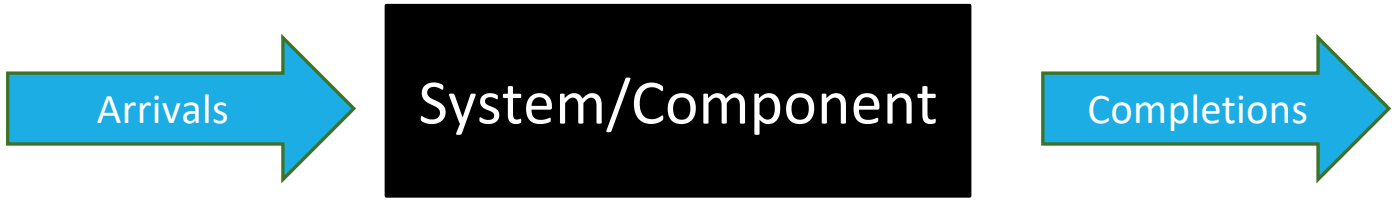


Basics for performance evaluations

The performance of a system that gives services could be seen from two **different viewpoints**:

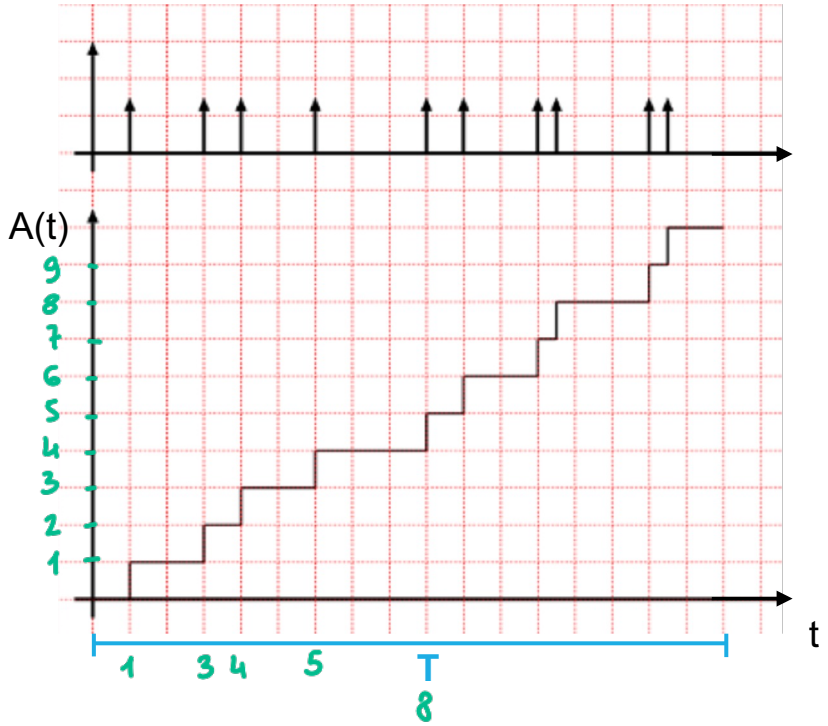
- **user**: the *time* to obtain a service or the *waiting time* before getting a service
- **system**: the *number of users* served in the unit time or the resources *utilization level*

Measurable Quantities

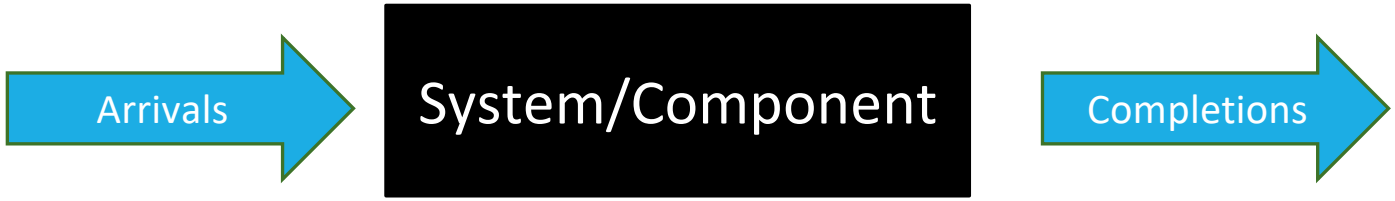


T: system *observation interval*;

A: *number of arrivals* in the period T;

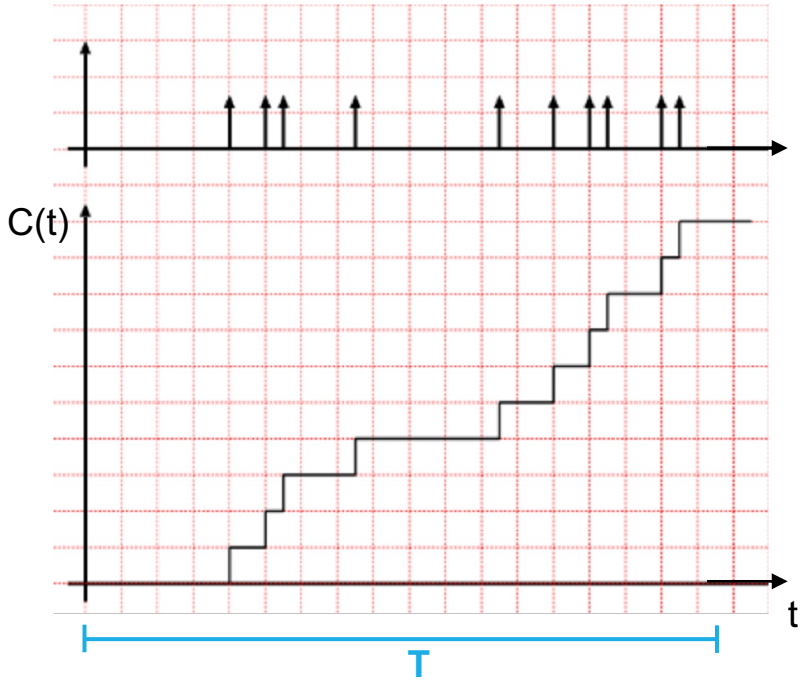


Measurable Quantities



T: system *observation interval*;

C: *number of completions* in the period T;



Basic Relations (Valid for Every Performance Model)

T: system *observation interval*

A: *number of arrivals* in the period T

C: *number of completions* in the period T

λ: *arrival rate* $\lambda = A/T$

EX. 10 request per second
one you can compute it?

X: *throughput* $X = C/T$

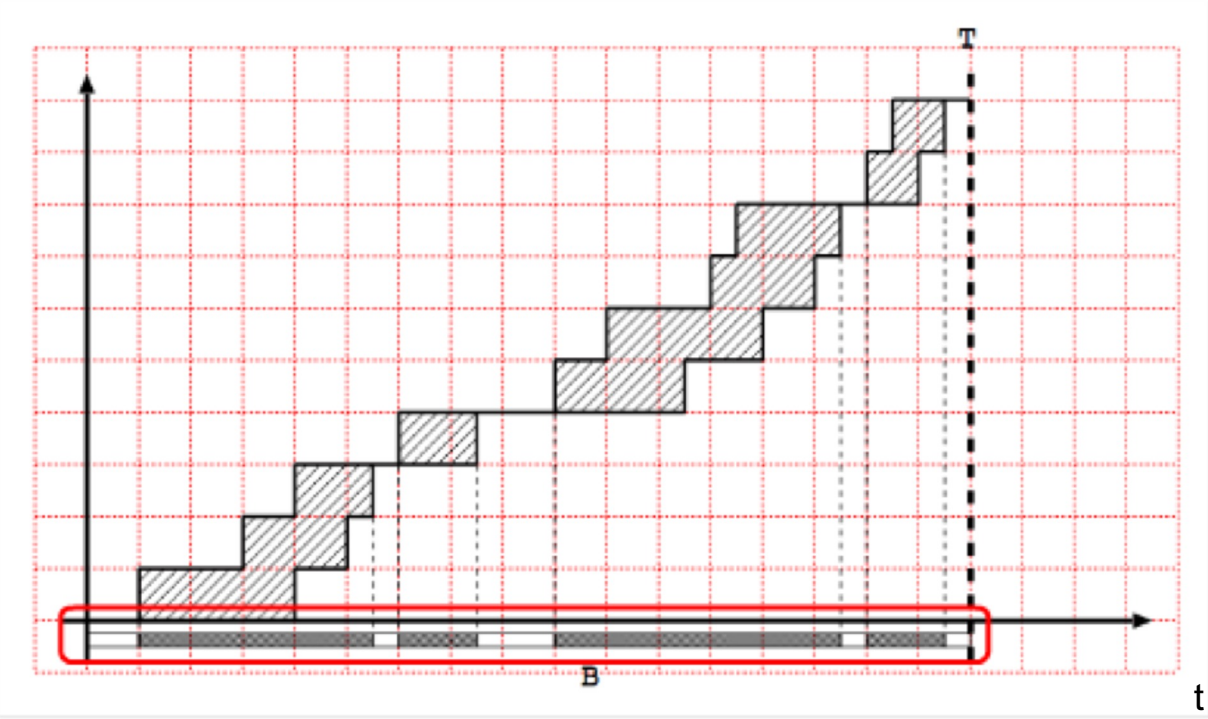
If A = C, *i.e.* the system is able to serve all its requests during the observation interval with no losses, the two quantities are equal.

$$\lambda = X$$

Utilization Law

B: *busy period* time in the period T;

amount of time in which our system is busy or not



Utilization Law

B: *busy period* time in the period T;

U: system/component *utilization* **U = B/T**

S: *average service time* per completion **S = B/C**

X: *throughput* **X = C/T**

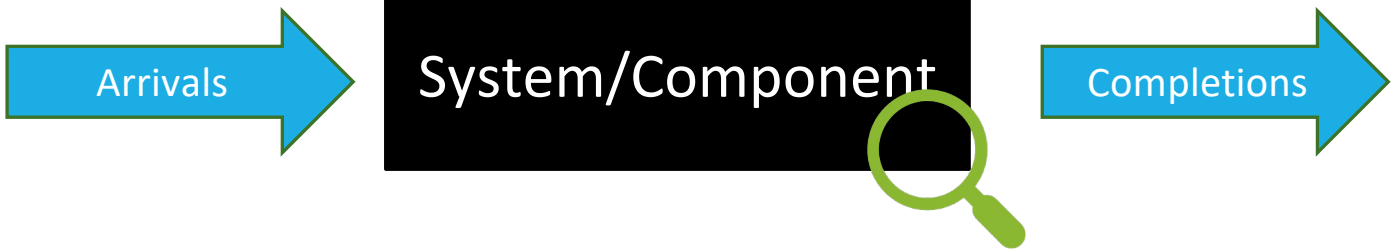
$$U = \frac{B}{T} = \frac{B/C}{T/C} = \frac{S}{1/X} = S \cdot X$$

U ↓ %

Utilization Law

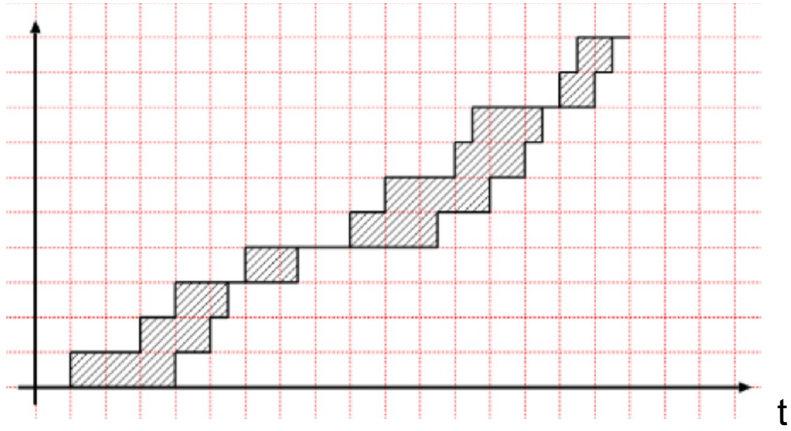
compute the utilization of a system

Little Law → used to understand how many users I'm expecting to be inside my system



How many requests are in the system?

W: "the sum of the *amount of time spent by all the request* in the system in T"

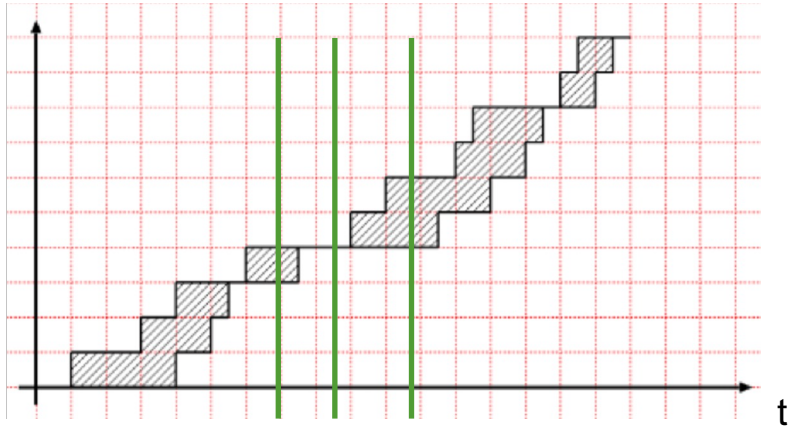


Little Law

W: “the sum of the *amount of time spent by all the requests* in the system in T”

How many requests are in the system?

N: *average number of requests* in the system
N = W/T

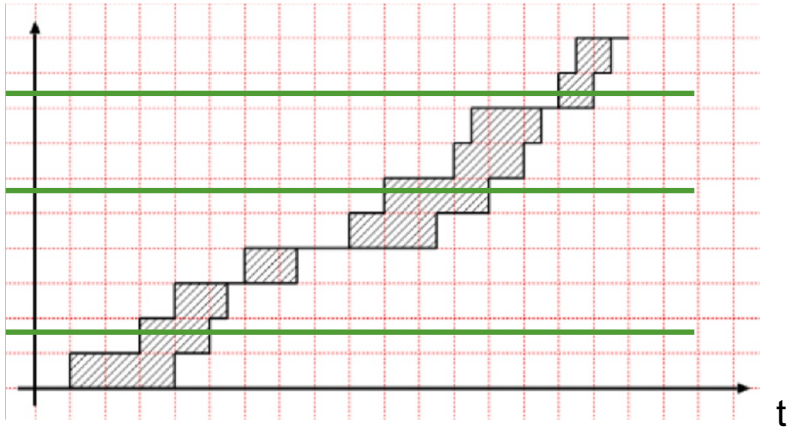


1 request in the system
0 request in the system
2 requests in the system

Little Law

W: "the sum of the *amount of time spent by all the requests* in the system in T"

How many requests are in the system?



R: average *response time* (or *residence time*),

waiting time + service time

R = W/C

time that you wait to be served

SERVICE time: time that it's required to perform the operation

RESPONSE time: all the time that you spent in order to perform an operation

Little Law

W: "the sum of the *amount of time spent by all the requests* in the system in T"

N: *average number of requests* in the system — N = W/T

R: average *response time* — R = W/C

X: *throughput* **X = C/T**

How many requests are in the system?

compute the expected number of users that are inside of the system

$$N = \frac{W}{T} = \frac{W/C}{T/C} = \frac{R}{1/X} = R \cdot X$$

Little Law

Common Performance Measures

If the system performs correctly the service, its performance is mainly measured by the **time taken to perform the service**, **the rate** at which the service is performed, **and the resources consumed** while performing the service.

The **utilization** gives an indication of the percentage of time a resource is busy for a given workload

The resource with the highest utilization is called the **bottleneck**

Stability condition

B: busy time

U: utilization $U = B/T$

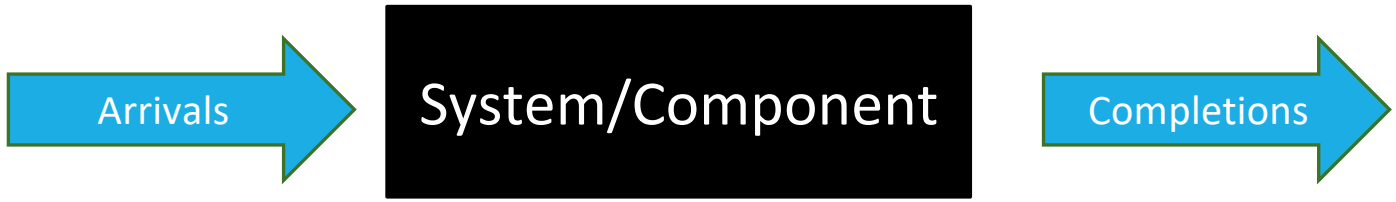
$B \leq T \implies U = B/T \leq 1$

Serve all the requests, $A = C, \implies \lambda = X$

$X \cdot S = \lambda \cdot S \leq 1$

$$\lambda \leq \frac{1}{S} \text{ or } S \leq \frac{1}{\lambda}$$

Recap Operational Laws (valid for all performance models)



λ: arrival rate, **X**: throughput

S: average **service time** per completion

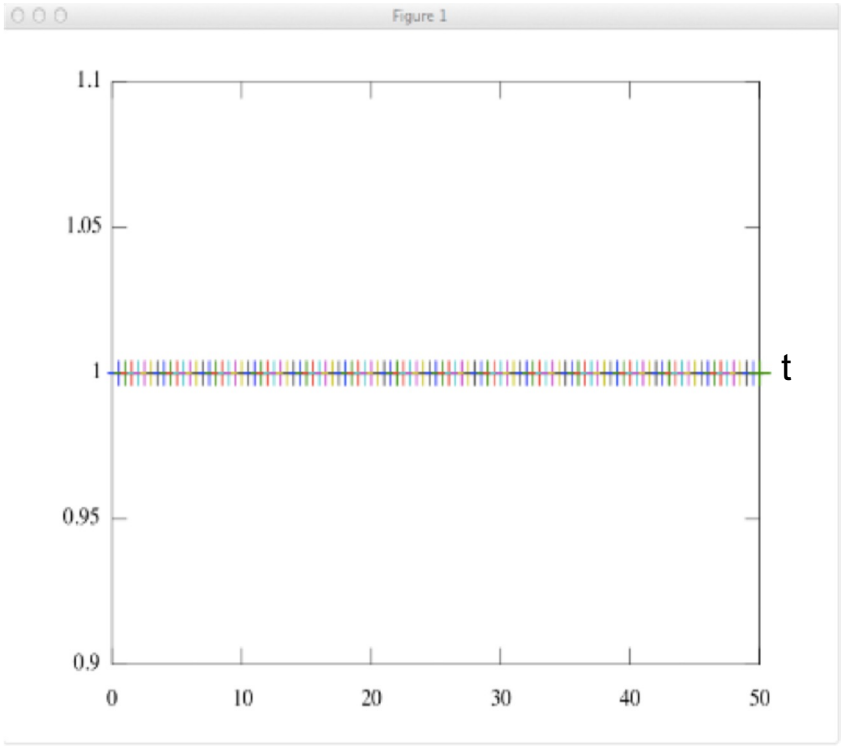
R: average **response time** (or residence time), waiting time + service time

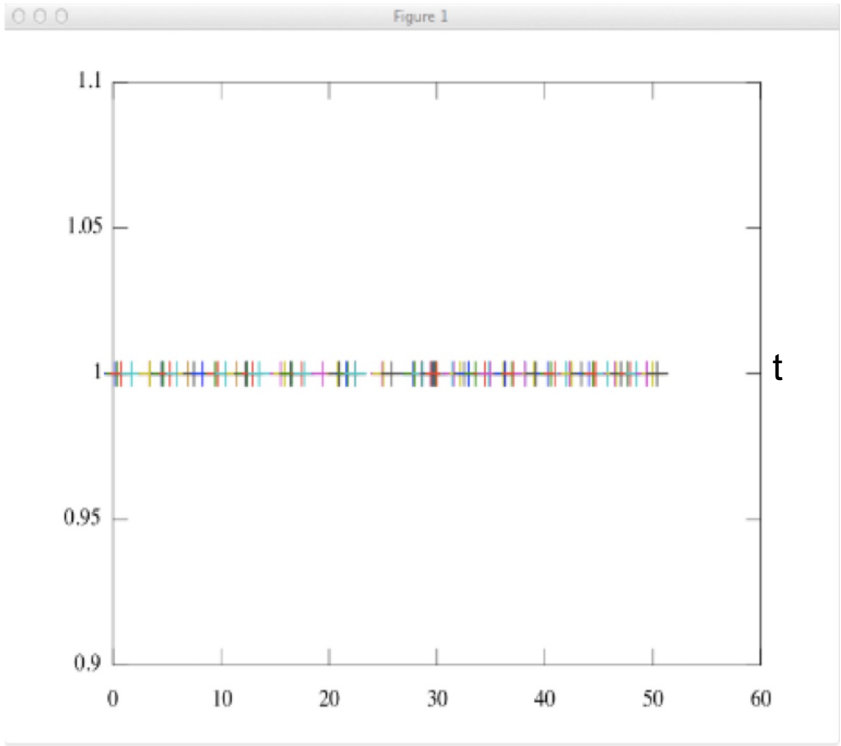
U = S · X (Utilization Law)

N = R · X (Little Law, number of requests IN the system)

Stability condition: $\lambda \leq 1/S$

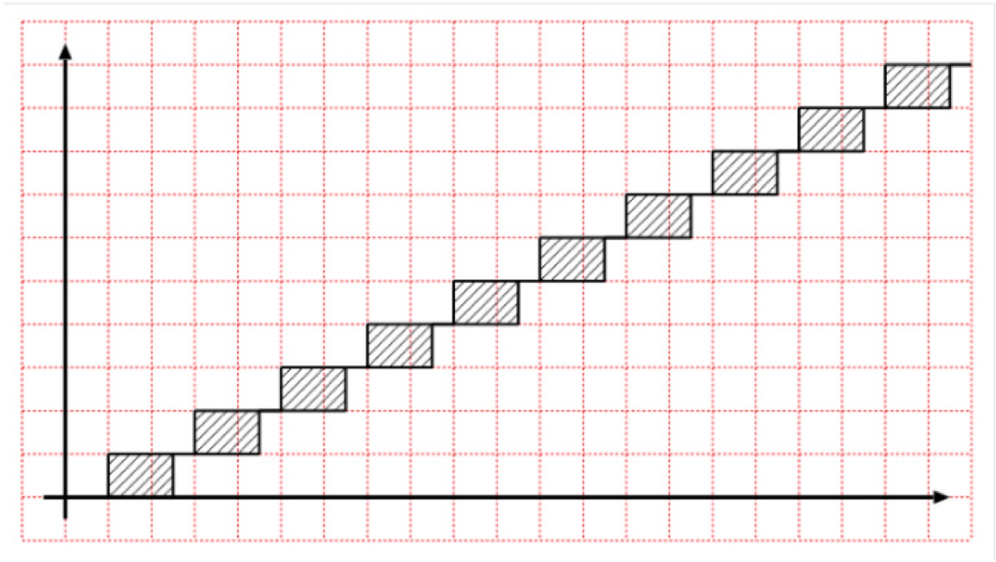
Arrivals examples





The arrival rate λ in the two workload is identical

Deterministic Inter-arrival

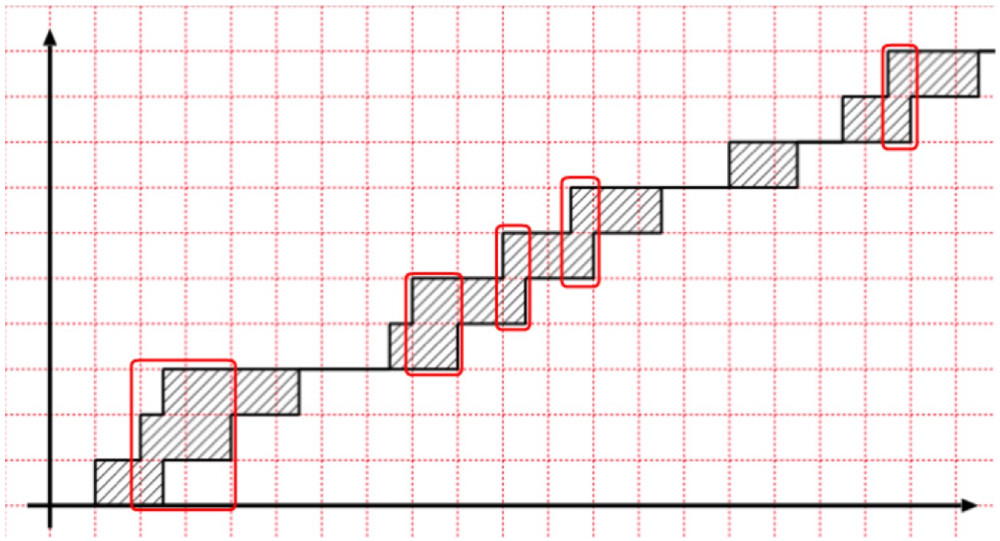


Service time is assumed constant in the figure

Jobs never queues

$U = \lambda \cdot S$
$N = \lambda \cdot S$
$X = \lambda$
$R = S$

Random Inter-arrival



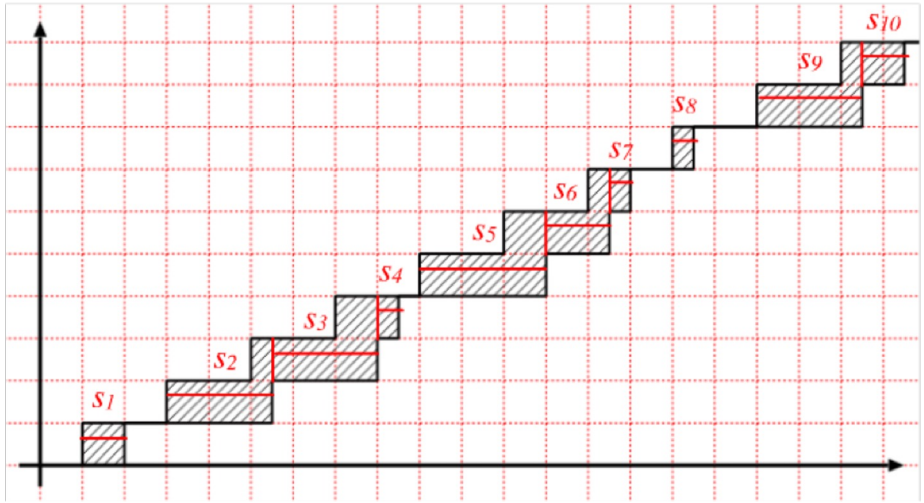
Service time is assumed constant in the figure

Non-determinism in the arrivals creates queuing

Service Features

The same as arrivals:

- Their ***rate*** (i.e. how fast requests are served)
- Their ***regularity*** (i.e. the time that passes between two served requests)
- Their ***correlation*** (informally, subsequent service time are correlate?)

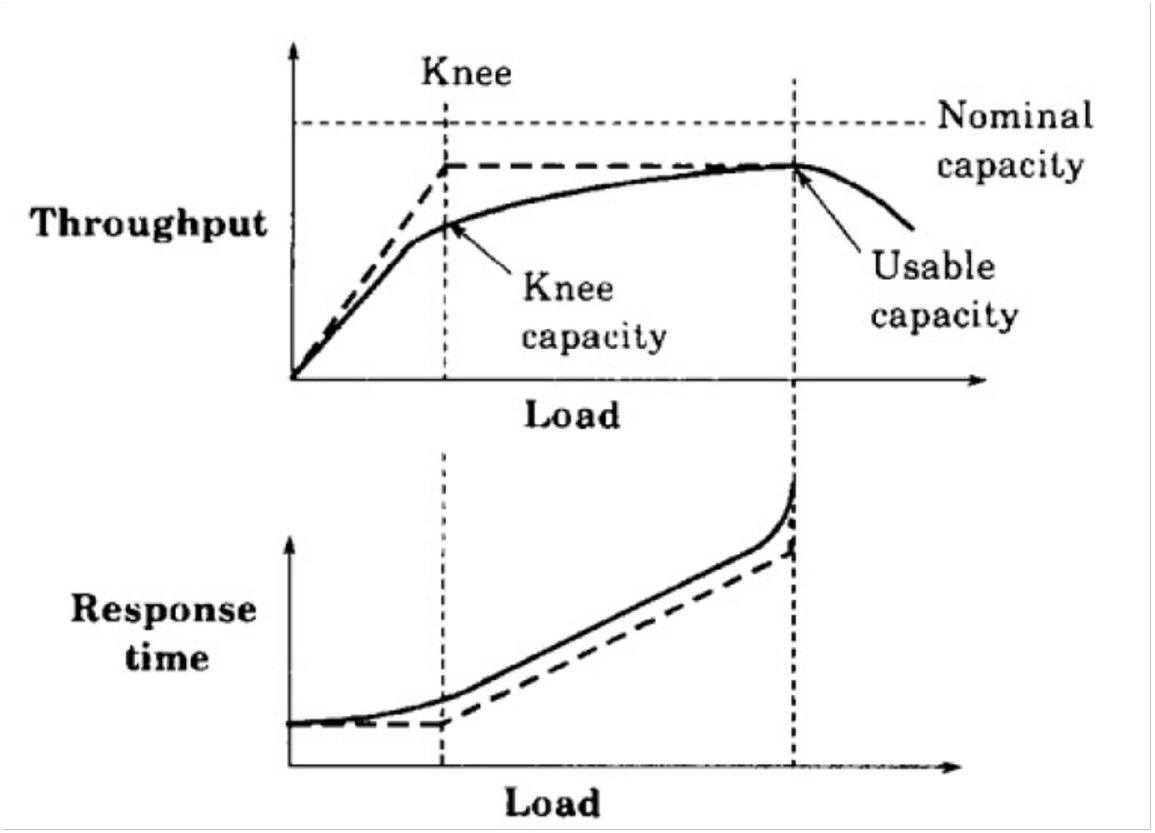


Note for Capacity Planning

The maximum achievable throughput under ideal workload conditions is called nominal capacity of the system

Often the response time at maximum throughput is too high to be acceptable.
In such cases, it is more interesting to know **the maximum throughput achievable without exceeding a prespecified response time limit.**
This may be called the usable capacity of the system.

In many applications, **the knee of the throughput or the response-time curve is considered the optimal operating point**.
This is the point beyond which the response time increases rapidly as a function of the load but the gain in throughput is small.



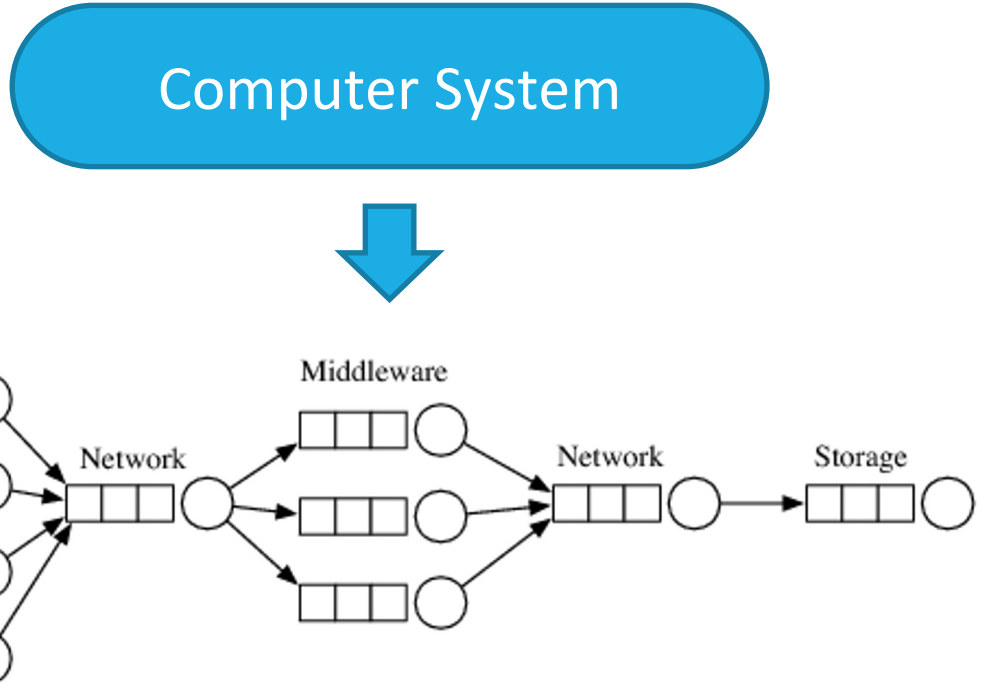
Analytical Performance Model: **Queuing Model**

Computer systems, including software systems, are composed of a **collection of resources** (e.g., processors, disks, communication links, process threads, critical sections, database locks) that are **shared by various requests** (e.g., transactions, web requests, batch processes).

Usually, there are **several requests running concurrently** and many of them may want to **access the same resource at the same time**.

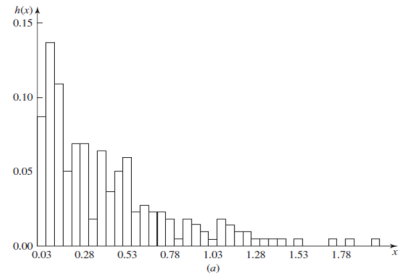
Since **resources have a finite capacity** of performing work (e.g., CPU finite number of instructions per seconds, disk transfers a certain amount of data per second, a communication link …) **waiting lines** often build up in front of these resources.

Queuing Model



Network of queues

Exponential Distribution

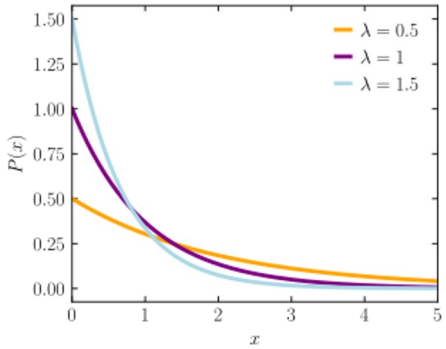


We say that a random variable X is distributed Exponentially with rate λ,

$$X \sim \text{Exp}(\lambda)$$

if X has the probability density function:

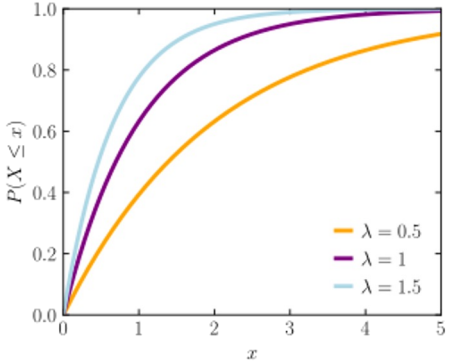
$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0. \\ 0 & x < 0. \end{cases}$$



The cumulative distribution function, $F(x) = \mathbf{P}\{X \leq x\}$, is given by

$$F(x) = \int_{-\infty}^{x} f(y)dy = \begin{cases} 1 - e^{-\lambda x} & x \geq 0. \\ 0 & x < 0. \end{cases}$$

$$\overline{F}(x) = e^{-\lambda x}, \ x \geq 0.$$



Plots taken from Wikipedia

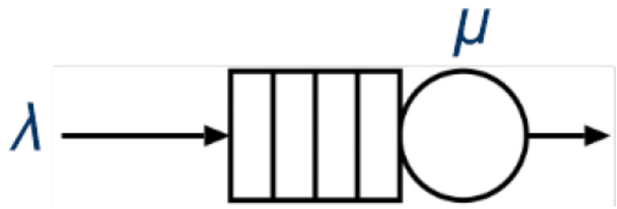
Single Open Queue – M/M/1

it is a mathematical tool

1 : SERVER

λ : ARRIVAL RATE

$\frac{1}{\lambda}$: SERVICE TIME



- The **inter-arrival time** of the request is distributed in accordance to the **exponential** distribution, with arrival rate equal to λ *(M)* (**Poisson process**)

  (Poisson process, i.e., a process in which events occur **continuously and**

  Independently at a constant average rate)

- The **service time** is distributed in accordance to the **exponential** distribution (**load independent**), with service time equals to 1/μ *(M)*
- One single server *(1)*
- The requests are statistically indistinguishable: **single class,** or homogeneous workload, or single workload component
- The server does not refuse any request: **infinite queue**

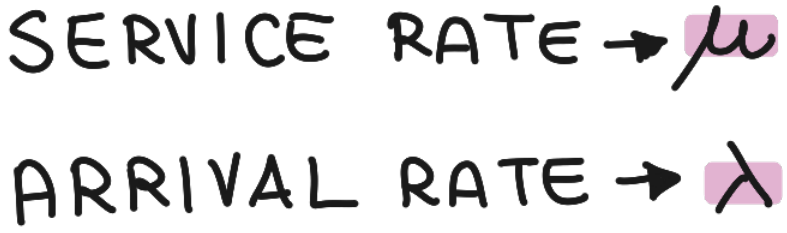
Note

λ and μ are rates (respectively, arrival rate and service rate)

e.g. λ = 10 requests/sec, μ = 15 requests/sec

The average inter-arrival time and the average service are multiplicative inverse (reciprocal)

- **S = 1/ μ**
- **Average inter-arrival time = 1/ λ**



λ = 10 requests/sec _> average inter-arrival time = 1/λ = 0,1 seconds

μ = 15 requests/sec _> average service time = 1/μ = 0,067 seconds

Single Open Queue – M/M/1

Respouse time

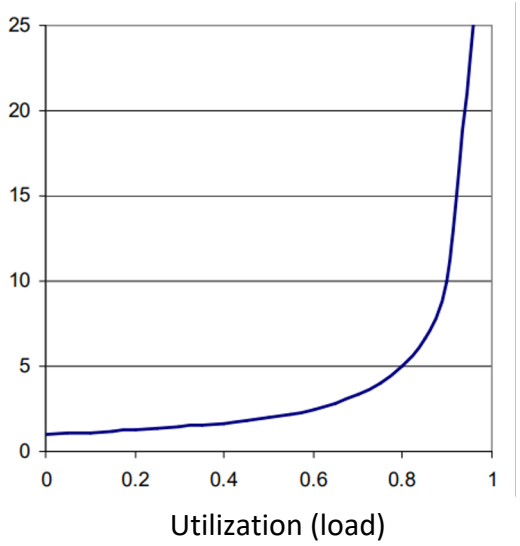
$$R = \frac{1}{\mu - \lambda}$$

$$R = \frac{1}{\mu \cdot \left(1 - \frac{\lambda}{\mu}\right)} = \frac{1}{\mu - \lambda}$$

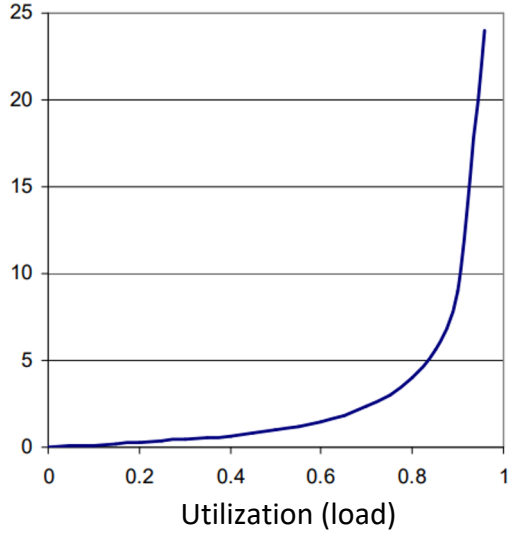
$$N = \frac{\rho}{1 - \rho}$$

$$U = \rho = 1 - p_0 = \frac{\lambda}{\mu}$$

Expected **utilization**



Expected **response time**



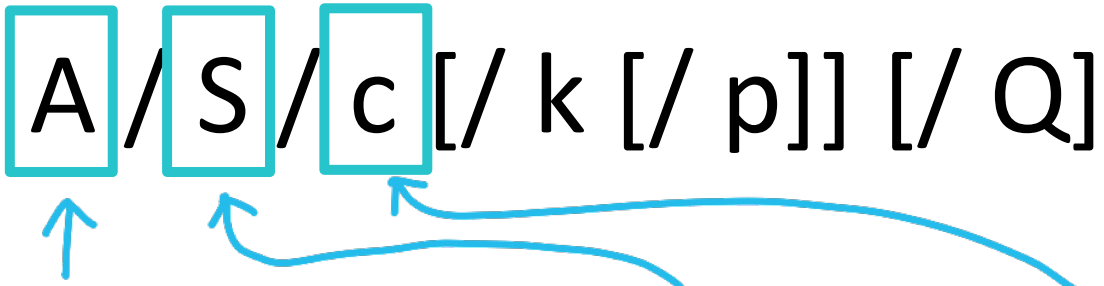
Expected **number of requests in the system**

Kendall's notation

A special notation, composed by 3 to 6 terms, is used to identify the behavior of a single queue

$$A / S / c [/ k [/ p]] [/ Q]$$

Kendall's notation



The first term defines the **arrival process**

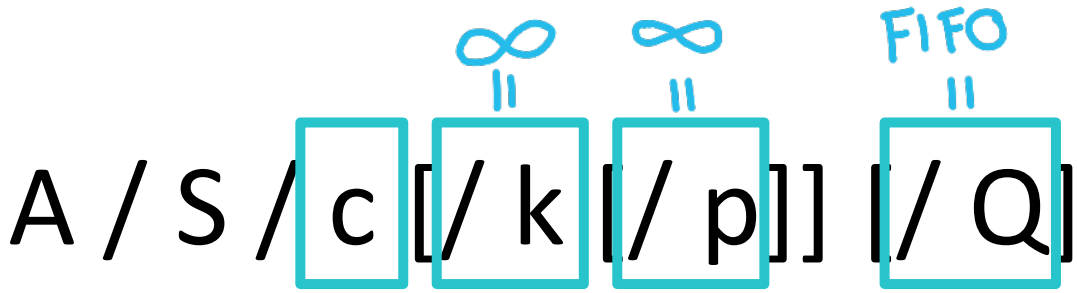
The second corresponds to the **service time distribution**.

c reports the **number of servers**

Most common acronyms for Inter-arrival and service time distributions:

- **M** • exponential / Poisson (Markov)
- **D** • Deterministic
- $\mathbf{E_k}$ • Erlang with *k* stages
- **G** • General

Kendall's notation



k the maximum capacity of the station (queues + service centers)

p If the population from which the jobs that enter the system is finite

Q The Service Discipline or Priority, order that jobs in the queue:

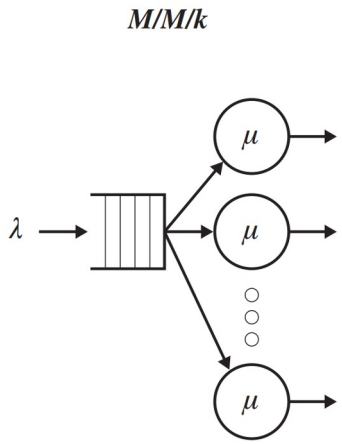
FIFO or FCFS	• First-in first-out
LIFO or LCFS	• last-in last-out
SIRO	• service in random order
PS	• processor sharing

Default values are: k = ∞, p = ∞, Q = FIFO

M/M/k

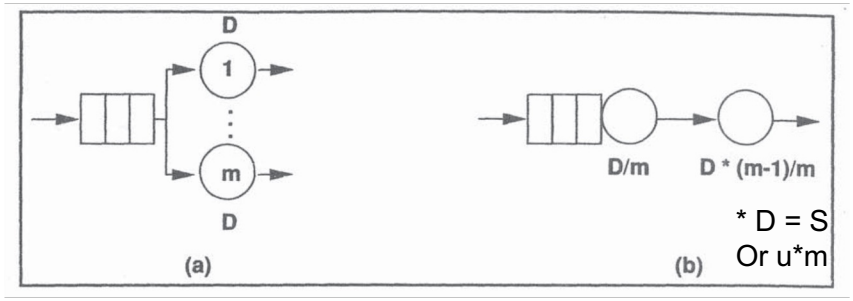
↓

k SERVERS



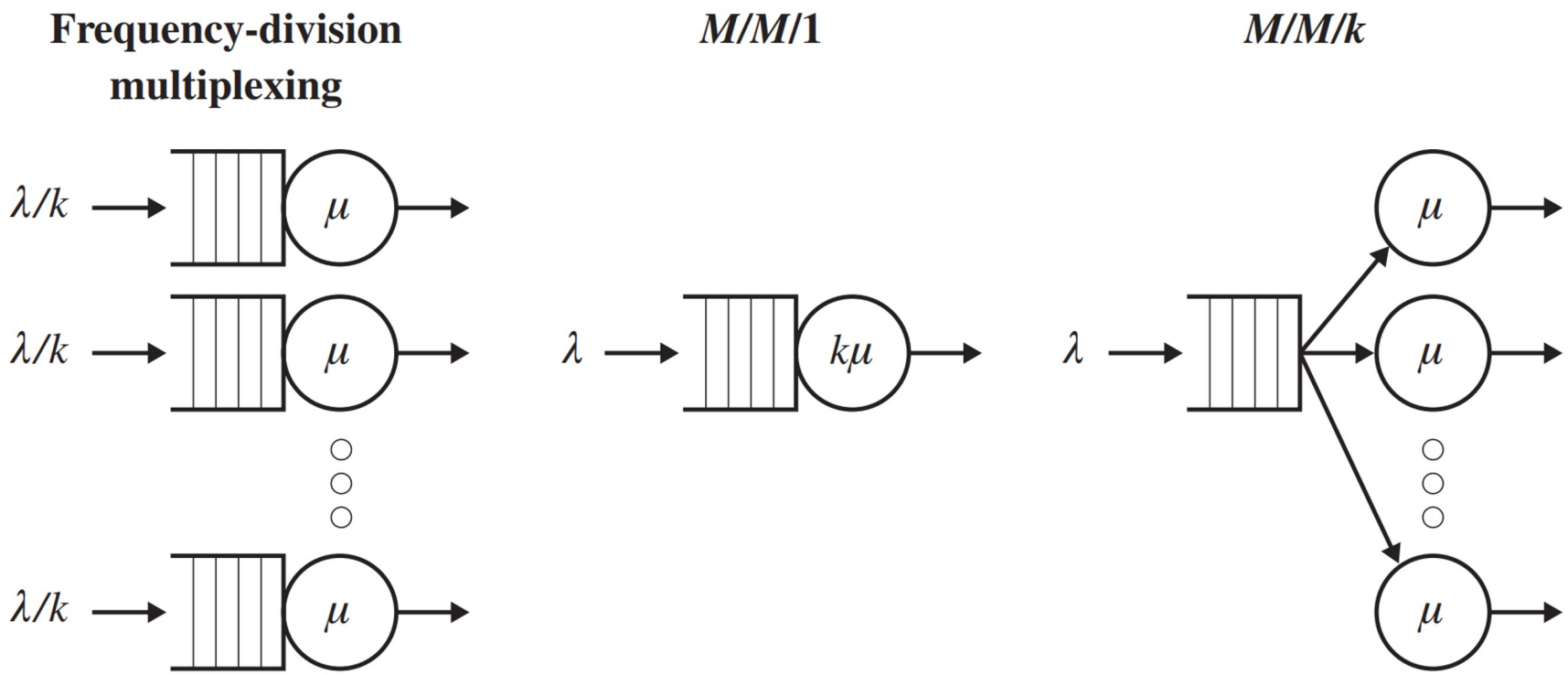
$$\mathbf{E}\,[T]^{\mathrm{M/M/k}} = \frac{1}{\lambda} \cdot P_Q \cdot \frac{\rho}{1-\rho} + \frac{1}{\mu}$$

where $\rho = \frac{\lambda}{k\mu}$, and P_Q is the probability an arrival is forced to queue.



Approximation for multiple servers:
1 M/M/1 k times faster
+ a constant value (no queue)

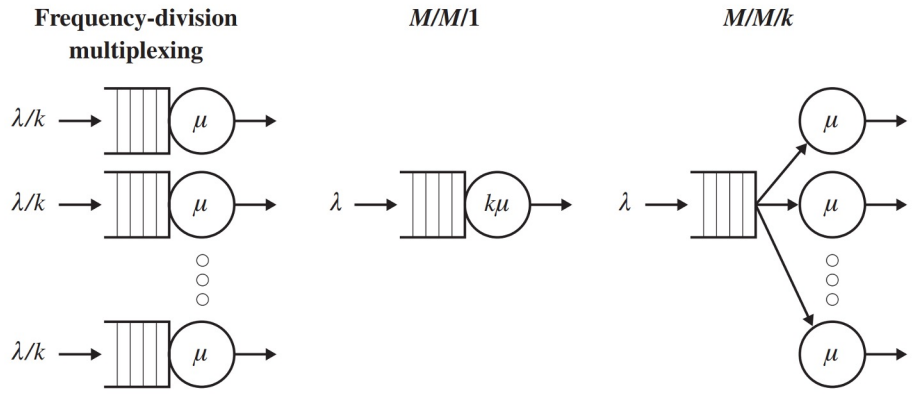
Comparing Server Organizations



$$\rho = \frac{\lambda}{k\mu}$$ **same load**

Which solution provides lower **response time** R?

Comparing Server Organizations



$$\mathbf{E}\left[T\right]^{\text{FDM}} = \frac{1}{\mu - \frac{\lambda}{k}} = \frac{k}{k\mu - \lambda}.$$

$$\mathbf{E}\left[T\right]^{\text{M/M/1}} = \frac{1}{k\mu - \lambda}.$$

$$\mathbf{E}\left[T\right]^{\text{M/M/k}} = \frac{1}{\lambda} \cdot P_Q \cdot \frac{\rho}{1-\rho} + \frac{1}{\mu}$$

Response time:

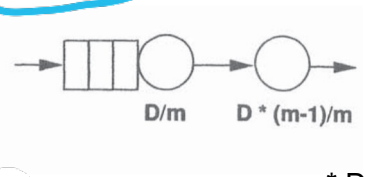
- FDM > M/M/1
- Low load, M/M/1 k times < M/M/k
- High load, M/M/1 ≈ M/M/k

Comparing Server Organizations

$$\mathbf{E}[T]^{\text{FDM}} = \frac{1}{\mu - \frac{\lambda}{k}} = \frac{k}{k\mu - \lambda}.$$

$$\mathbf{E}[T]^{\text{M/M/1}} = \frac{1}{k\mu - \lambda}.$$

$\mathbf{E}[T]^{\text{M/M/k}}$:



$$= \frac{1}{k\mu - \lambda} + \frac{1}{\mu} \cdot \rho = \frac{1}{k\mu - \lambda} + \frac{1}{\mu} \cdot \frac{\lambda}{k\mu}$$

* D = S
Or u*m

ρ : same load

Exemple 1

$\mu = 10$ req/s

$\lambda = 30$ req/s

$k = 4$

$$\rho = \frac{\lambda}{k\mu} = \frac{30}{4 \cdot 10} = \frac{30}{40} = \frac{3}{4}$$

$$FDM = \frac{k}{k\mu - \lambda} = \frac{4}{4 \cdot 10 - 30} = \frac{4}{40 - 30} = \frac{4}{10} = 0{,}4 \text{ s}$$

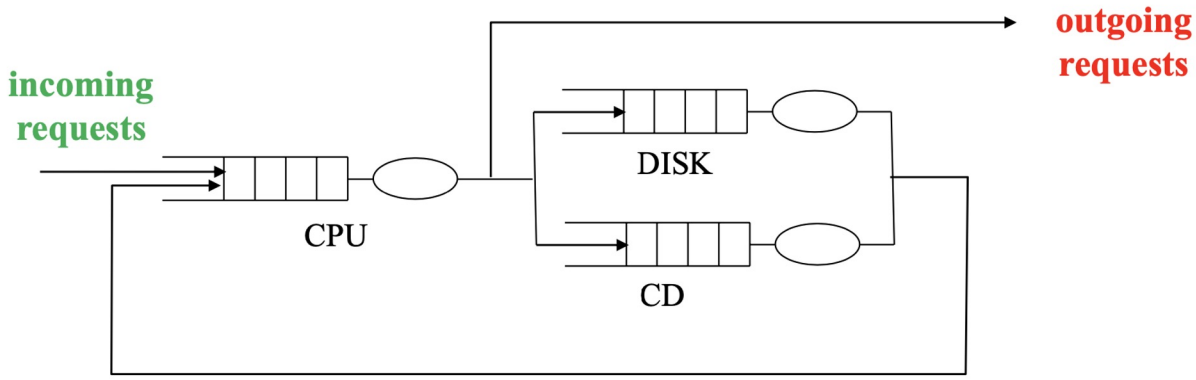
$$MM1 = \frac{1}{k\mu - \lambda} = \frac{1}{4 \cdot 10 - 30} = \frac{1}{10} = 0{,}1 \text{ s}$$

$$MMK = \frac{1}{k\mu - \lambda} + \frac{1}{\mu} \cdot \rho = 0{,}1 + \frac{1}{10} \cdot \frac{3}{4}$$

$$= 0{,}1 + 0{,}1 \cdot 0{,}75$$
$$= 0{,}1 + 0{,}075$$
$$= 0{,}175 \text{ s}$$

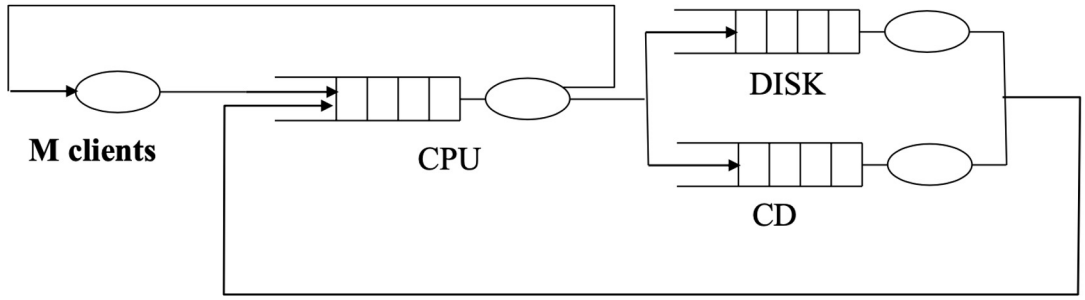
Queuing Networks

Open queuing networks



Closed queuing networks

(finite number of users)



Open Jackson Networks (simplified)

An open queueing network composed by M/M/k queues can be analyzed piecewise, namely the performance metrics of the single queues can be computed independently and the **performance metrics for the entire network can be obtained by composition**

All queues receives the same arrivals at rate λ

_> it is possible to assume that a *request passes through a queue multiple times*

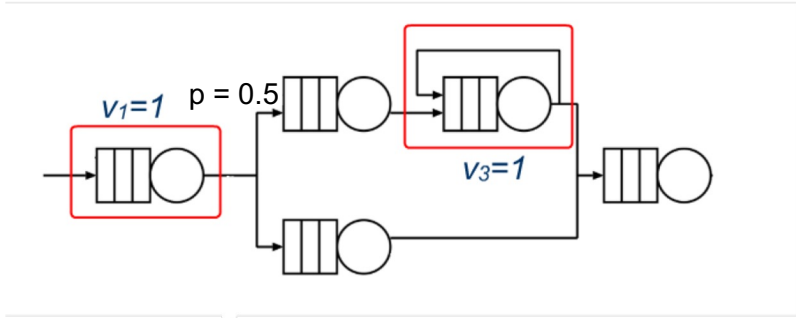
_> it is possible to *associate probabilities to the arrows* interconnecting queues

Visits

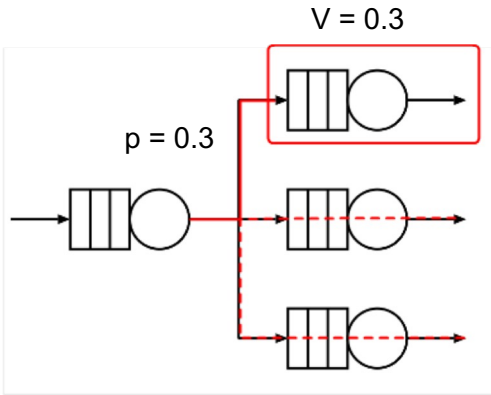
The **visits** represent the **average number of times a request pass through a station *i* for the moment it enters the system to the time it leaves**

$$V_i = \frac{C_i}{C}$$

V_i can be less than one if there could be paths that skips the considered node *k*.



Note that $V_i = 1$ **means that station *i* is visited on the average once during the service of one request: this does not exclude that the considered station can be skipped or that it can be re-entered by the job during its service.**



Service Demand

The ***service demand*** $\boldsymbol{D_i}$ of a queue *i* accounts for the **average time spent by a request** to the considered service center **during all its visits**, taking into account the fact that such resource might also be skipped.

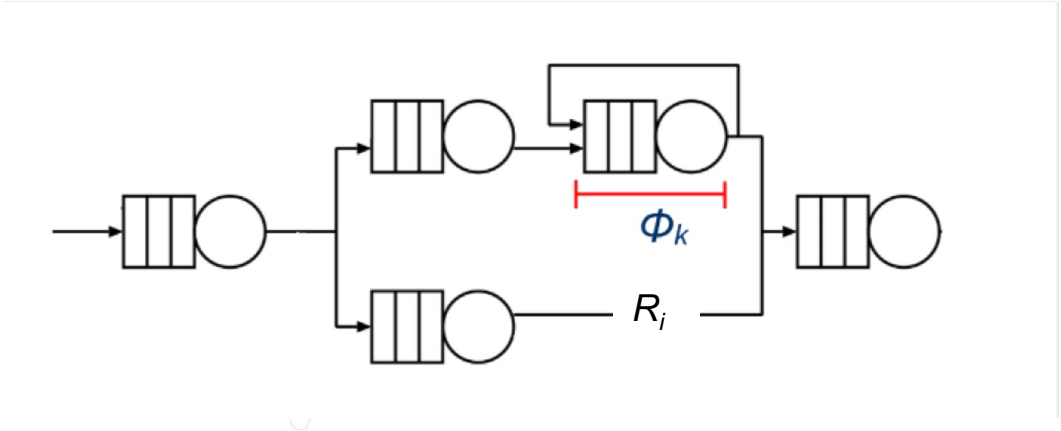
$$D_i = \frac{B}{C} = \frac{B}{C} \cdot \frac{C_i}{C_i} = V_i \cdot S_i$$

Note that:

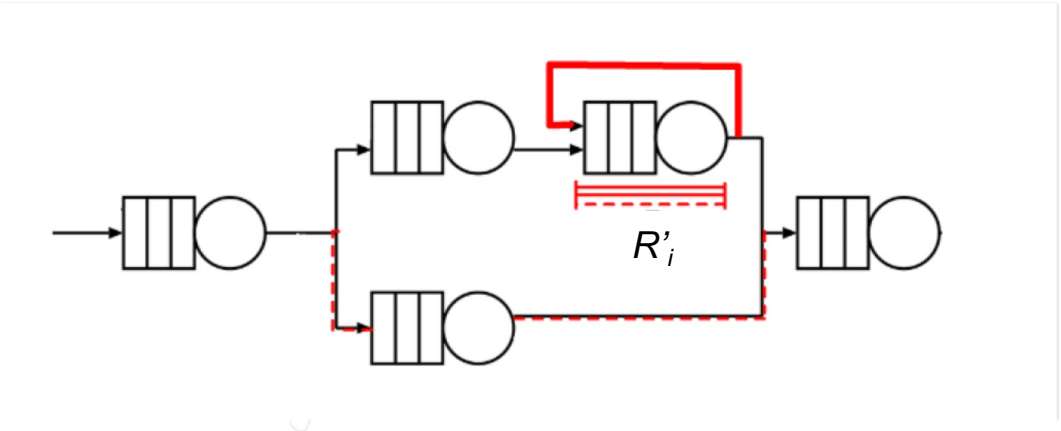
- Average **service time** S_i accounts for the **average time** that a request spends **in queue *i*** when **it is served**.
- Average **service demand** D_i accounts for the **average time** a job spends **in station *i* during its staying in the system**.

Response Time and Residence Time

R_i average ***response*** *time* is the average time spent in queue *i* when a request enters the station



R'_i average ***residence*** *time* is the average time spent by a request at queue *i* during its staying in the system



It can be greater or smaller than the response time depending on the number of visits.

Response, Residence, Service, and Demand

Note that the relation between *Residence Time* and *Response Time* is the same as the one between *Demand* and *Service*.

$$D_i = V_i \cdot S_i$$
$$R'_i = V_i \cdot R_i$$

Also note that for single queue network $V_i = 1$. This implies that the average service time equals the service demand, and the response time and the residence time are identical.

For this reason when referring to models with a single station, demand and service times, or residence and response time can be used as synonymous.

Little and Utilization Laws with Visits

$$U_i = X \cdot D_i = X \cdot V_i \cdot S_i = X_i \cdot S_i$$

$$N_i = X \cdot R'_i = X \cdot V_i \cdot R_i = X_i \cdot R_i$$

Network Performance Indices

The workload of open networks is characterized by the arrival rate λ

If the network is stable, the arrival rate corresponds to the system throughput

$$X = \lambda$$

The average total number of requests in the system can be computed as the sum of the average number of requests at each station:

$$N = \sum N_i$$

Network Performance Indices

The average *Response Time* represents the average time spent by the requests in the system: it can be computed as the *sum of the Residence Times* of the jobs at all the stations

$$R = \sum R'_i$$

Note the discrepancy: the system ***response time*** is the sum of ***residence times*** and not of the *response times* at the stations.

Note that *Little's law* continues to be valid system-wise using the network performance indices:

$$N = \sum N_i = \sum X \cdot R'_i = X \cdot \sum R'_i = X \cdot R$$

Single Queue Analysis in Open Jackson Network

$$R_i = \frac{S_i}{1-U_i}$$

Response time i

$$N_i = \frac{U_i}{1-U_i}$$

Number of requests i

$$U_i = \lambda \cdot V_i \cdot S_i$$

Utilization/load i

$$R'_i = V_i \cdot R_i = \frac{D_i}{1-U_i}$$

Residence time i

$$R = \sum R'_i$$

Response time network

NOTE: if all Vi = 1 it is possible to compute the performance metrics using M/M/k formulas, otherwise formulas are slighly different (there are valid for what ever value of Vi

Bottleneck Identification (open network)

In an open network the average frequency of users incoming into the network is fixed. For λ too much big the network will become unstable

$$U_i = X_i \cdot S_i = \lambda \cdot V_i \cdot S_i \qquad D_i = V_i \cdot S_i$$

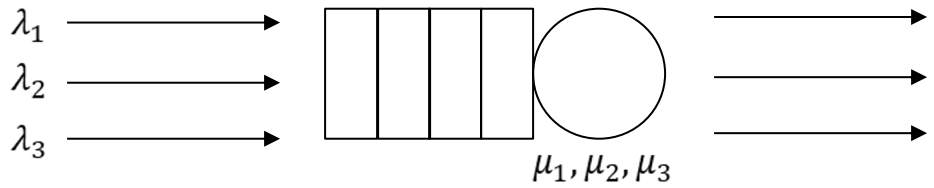
$U_i = 1$ is the greatest utilization factor of a queue $U_i \leq 1$

$$\lambda \cdot D_i \leq 1$$

stability

$$\lambda \leq \frac{1}{\max D_i}$$

Open Jackson Networks - Multiple Classes



(Multiple classes)

If the arrivals associated to each workload component (more than one) are all exponentially distributed, then the single workload components can be analyzed almost independently

It is necessary to know:

- All the arrival rates λ_r
- All the demands associated to each class $D_{i,r}$

Details available in Chapter 9 - D. A. Menascé, V. A. F. Almeida: Capacity Planning for Web Services: metrics, models and methods.

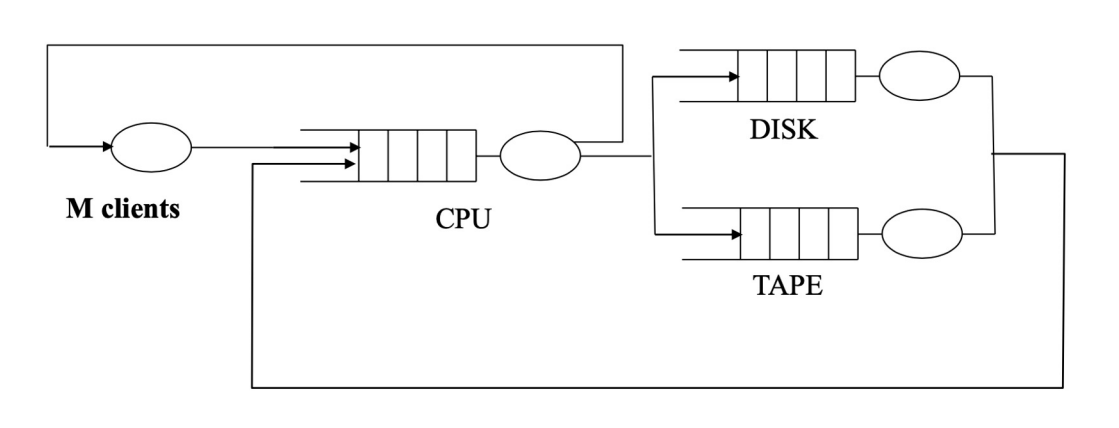
Closed Networks

Finite population: **M** source of requests

Each source submits a request, waits for the response, composes a new request (”thinking”) to be submitted to the system

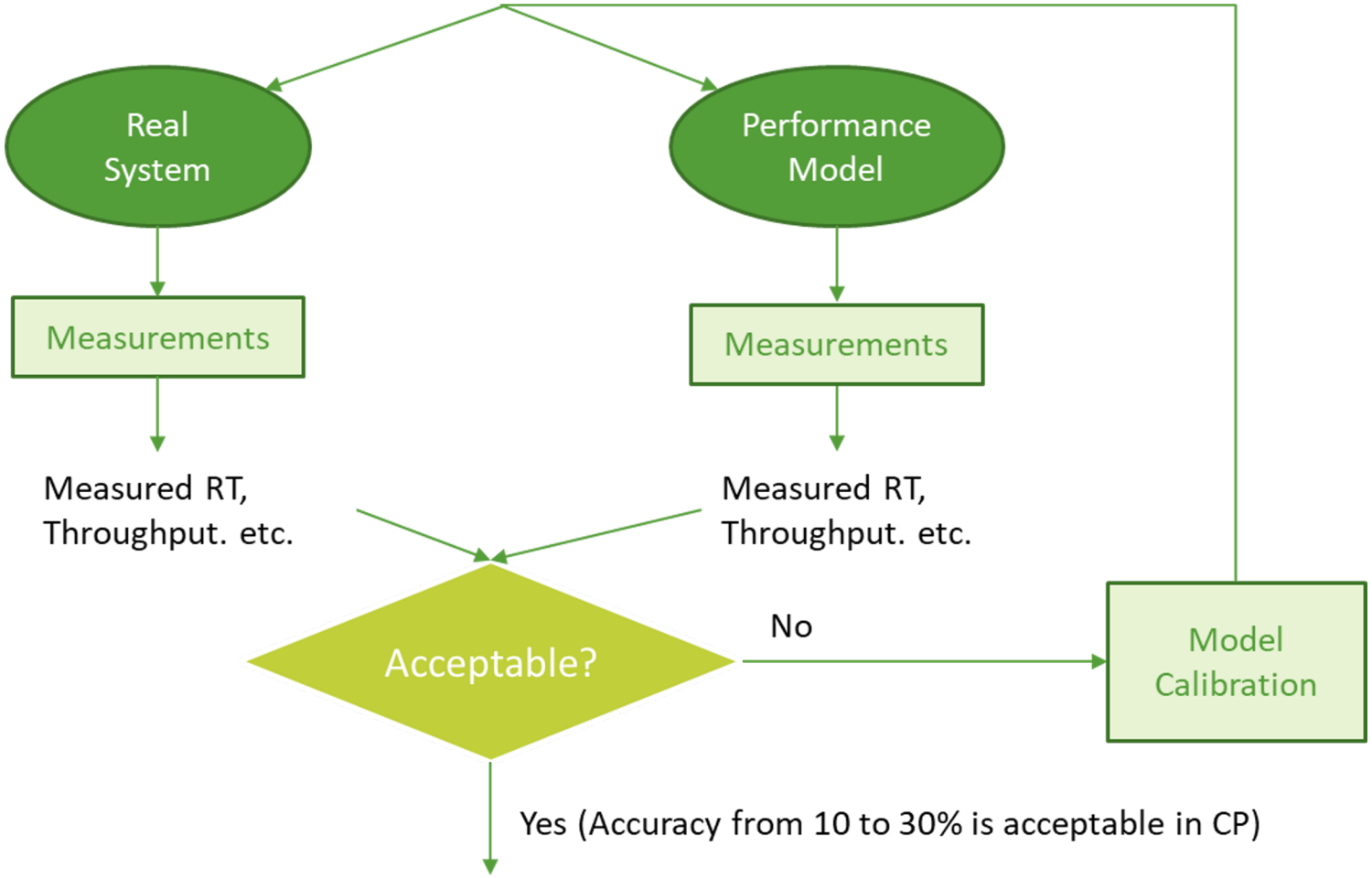
The time spent by the source from the time a response to a request is received and the next request is submitted is called **think time**

Z = average think time



Details available in Chapter 9 - D. A. Menascé, V. A. F. Almeida: Capacity Planning for Web Services: metrics, models and methods.

Performance model validation and calibration



References

- D. A. Menascé, V. A. F. Almeida: Capacity Planning for Web Services: metrics, models and methods. Prentice Hall, PTR

- Quantitative System Performance, Computer System Analysis Using Queueing Network Models. Edward D. Lazowska, John Zahorjan, G. Scott Graham, Kenneth C. Sevcik https://homes.cs.washington.edu/~lazowska/qsp/

- L. Kleinrock: Queueing Systems, Vol. 1:Theory, John Wiley & Sons

- Java Modelling Tools – JMT https://jmt.sourceforge.net/

- R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling,“ https://www.cin.ufpe.br/~rmfl/ADS_MaterialDidatico/PDFs/performanceAnalysis/Art%20of%20Computer%20Systems%20Performance%20Analysis%20Techniques.pdf

TOOL → load generator
↓ monitor

How to do performance evaluation?
- ↓ measurement
- ↓ analytical modeling
- ↓ simulation modeling

Performance models → analytical: formulas
↓ simulation: computer programs

Viewpoints → user
↓ system

STABILITY CONDITION

$\lambda \leq \frac{1}{S}$

λ: arrival rate; S: service time

$S = \frac{1}{\mu}$

$\lambda \leq \frac{1}{\frac{1}{\mu}}$

$\boxed{\lambda \leq \mu}$

$\boxed{\rho < 1}$ where $\rho = \frac{\lambda}{\mu}$