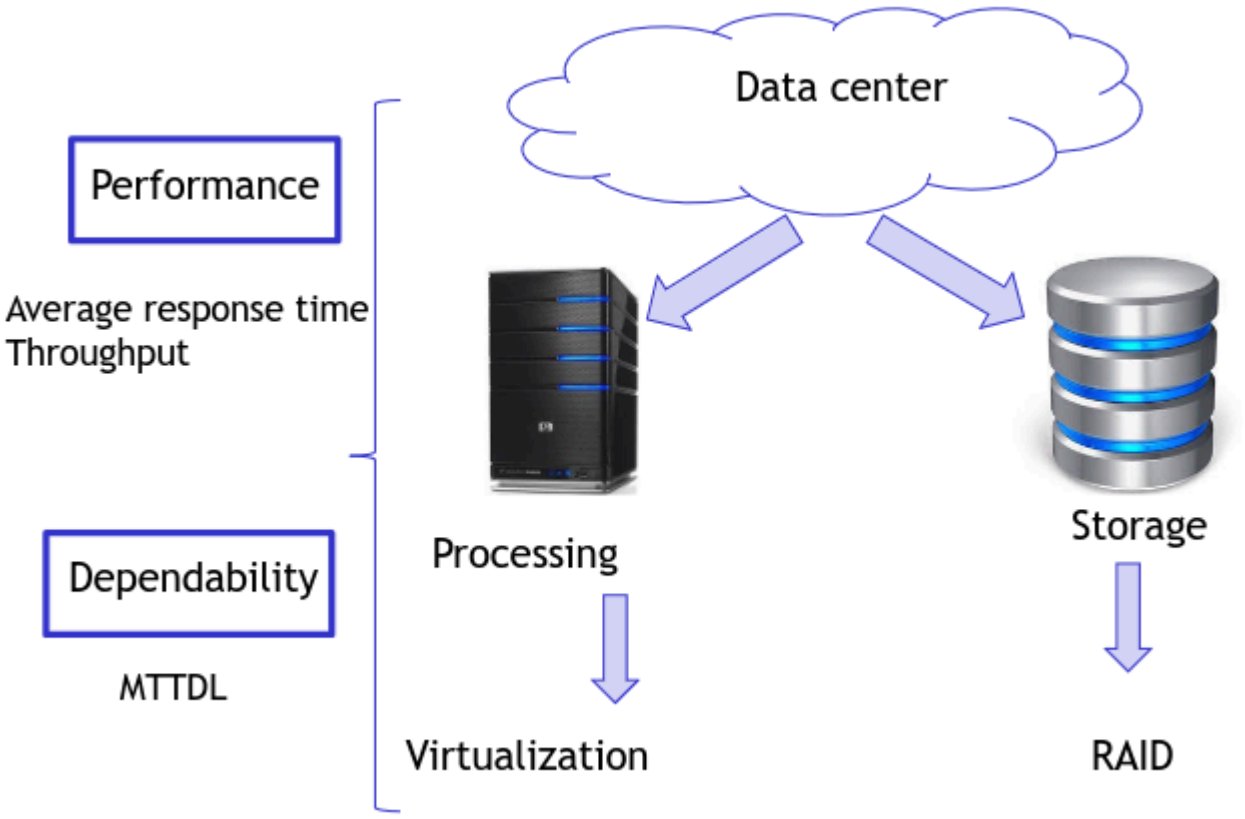
Performance



We start from the datacenter structure, we know all the procedure parts; we need to talk about the virtualization. We now want to know the response time, the system performance.
Computer performance: the total effectiveness of a computer system, including throughput, individual response time and availability. Can be characterized by the amount of useful work accomplished by a computer system or computer network compared to the time and resources used.
People not pay much attention to performance, they want to see functionality, but in some cases performance cannot be improved by only buying more hardware. You need some stratagem to improve performance without changing too much(need to use the right architecture to improve performance). Need different skill for quality verification.
Short time to market, i.e. quickly available products and infrastructures “seem” to be more attractive nowadays! Little information related to quality is usually available early in the system lifecycle but its understanding is of great importance from the cost and performance point of view.

How to evalutate system quality

- Use of intuition and trend extrapolation
  Unfortunately, those who possess these qualities in sufficient quantity are rare
  Pro: rapid and flexible
  Con: accuracy

- Experimental evaluation of alternatives
  Experimentation is always valuable, often required, and sometimes the approach of choice
  It is also expensive - often prohibitively so
  A further drawback: an experiment is likely to yield accurate knowledge of system behavior under one set of assumptions, but not any insight that would allow generalization
  Pro: excellent accuracy
  Con: laborious and inflexible
  You need to have a system in place to follow the second approach. It is also a lot costly as it need a lot of computation and testing.

Solution: Model Based Approach

As systems are so complex we use abstraction of the system called Models to investigate the best solution for the task.
We use two techniques to do quality evaluation techniques:

- Measurement-based: based on direct measurement, benchmarking and prototyping to conform to the task
- Model-based: based on analytical and numerical techniques, simulations and hybrid techniques
  A model is a representation of a system that is simpler than the actual system, captures the essential characteristics and can be evaluated to make predictions.
  We can use model in system design defining the goal and creating the model. Once we define the model we need to define the parameters and then we evaluate the model and define the performance indices.
- Analytical and Numerical techniques are based on the application of mathematical techniques, which usually exploit results coming from the theory of probability and stochastic process
  They are the most efficient and the most precise, but are available only in very limited cases
- Simulation techniques are based on the reproduction of traces of the model, you can see all the events that the system needs to handle and simulate the system to get all the statistics for the system.
  They are the most general, but might also be the less accurate, especially when considering cases in which rare events can occur
  The solution time can also become really large when high accuracy is desired
- Hybrid techniques combine analytical/numerical methods with simulation. We can use simulation for the parts of the systems that take more times to be handles with analytical/numerical methods.

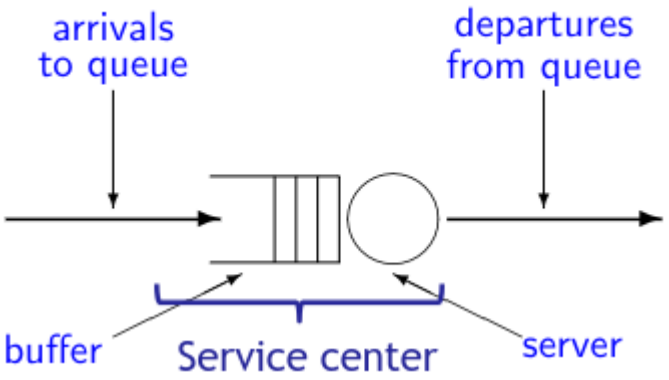
Queueing Methods

Queueing theory is the theory behind what happens when you have a lot of jobs, scarce resources, and so long queue and delays. Queueing network modelling is a particular approach

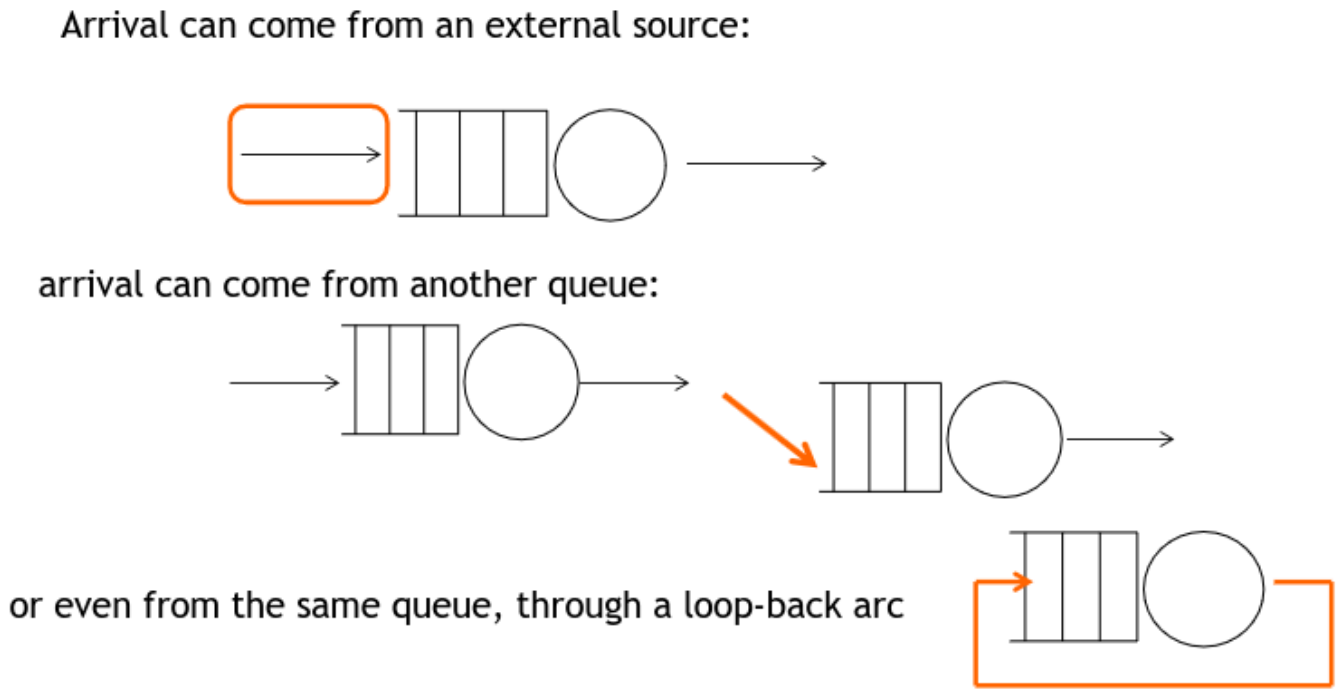
to computer system modelling in which the computer system is represented as a network of queues. A network of queues is a collection of service centers, which represent system resources, and customers, which represent users or transactions.
Success of queueing network: low-level details of a system are largely irrelevant to its high-level performance characteristics.
Queues in computer systems:

- CPU uses a time-sharing scheduler
- Disk serves a queue of requests waiting to read or write blocks
- A router in a network serves a queue of packets waiting to be routed
- Databases have lock queues, where transactions wait to acquire the lock on a record

Single queue



The basic scenario for a single queue is that customers, who belong to some population arrive at the service facility. The service facility has one or more servers which can perform the service required by customers. If a customer cannot gain access to a server it must join a queue, in a buffer, until a server is available. When service is complete the customer departs, and the server selects the next customer from the buffer according to the service discipline (queueing policy).
Arrival of customers
Arrivals represent jobs entering the system: they specify how fast, how
often and which types of jobs does the station service. We are interested in the average arrival rate λ (req/s)



Example of a loop is the CPU sharing as the CPU gives you only a part of the total capacity and you have to do more passage to the CPU to complete the task.

Service & Service time distribution

The service part represents the time a job spends being served

The service time is the time which a server spends satisfying a customer

As with the inter-arrival time, the important characteristics of this time will be its average duration (advanced: the distribution function)

If the average duration of a service interaction between a server and a customer is 1/μ then μ is the maximum service rate

We need to take care of burst of requests as they slow down our system and are rare cases.

λ needs to be strictly lower than μ to not have a queue that grown infinitely and cloggle the system.

Number of servers

Possible situations:

- single server: the service facility only has the capability to serve one customer at a time; waiting customers will stay in the buffer until chosen for service; how the next customer is chosen will depend on the service discipline
- infinite server: there are always at least as many servers as there are customers, so that each customer can have a dedicated server as soon as it arrives in the facility. There is no queueing, (and no buffer) in such facilities.
- multiple server: they have a fixed number of servers c each of which can service a customer at any time. If the number of customers in the facility is less than or equal to c there will no queueing—each customer will have direct access to a server. If there are more than c customers, the additional customers will have to wait in the buffer
  You can model slow services as infinite server to have multiple operation at the same time.
  You can model some fast actions as single server.
  Queue

If jobs exceed the capacity of parallel processing of the system, they are forced to wait queueing in a buffer

When the (one of the) job(s) currently in service leaves the system, one of the job in the queue can enter the now free service center

Service discipline/queuing policy determines which of the job in the queue will be selected to start its service

Buffer capacity

Customers who cannot receive service immediately must wait in the buffer until a server becomes available

If the buffer has finite capacity there are two alternatives for when the buffer becomes full:

- the fact that the facility is full is passed back to the arrival process and arrivals are suspended until the facility has spare capacity, i.e. a customer leaves;
- or, arrivals continue and arriving customers are lost (turned away) until the facility has spare capacity again

If the buffer capacity is so large that it never affects the behaviour of the customers it is assumed to be infinite

Service discipline

When more than one customer is waiting for service, we need a rule for selecting which of the waiting customers will be the next one to gain access to a server

The commonly used service disciplines are

- FCFS first-come-first-serve (or FIFO first-in-first-out)
- LCFS last-come-first-serve (or LIFO last-in-first-out)
- RSS random-selection-for-service
- PRI priority, the assignment of different priorities to elements of a population is one way in which classes are formed

Population

The characteristic of the population which we are interested in is usually the size

Clearly, if the size of the population is fixed, at some value N, no more than N customers will ever be requiring service at any time

When the population is finite, the arrival rate of customers will be affected by the number who are already in the service facility (e.g. zero arrivals when all N are all already in the facility)

When the size of the population is so large that there is no perceptible impact on the arrival process, we assume that the population is infinite

Ideally, members of the population are indistinguishable from each other

When this is not the case, we divide the population into classes whose members all exhibit the same behaviour

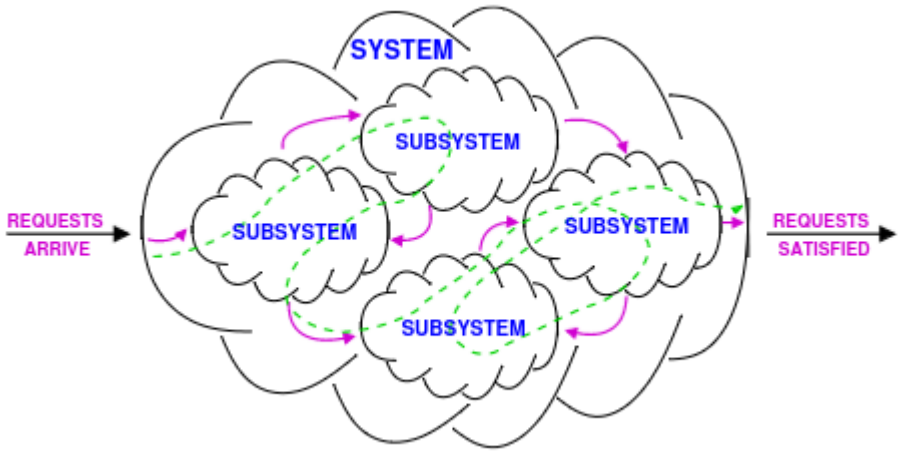
Different classes differ in one or more characteristics, for example, arrival rate, service demand

Identifying different classes is a workload characterisation task

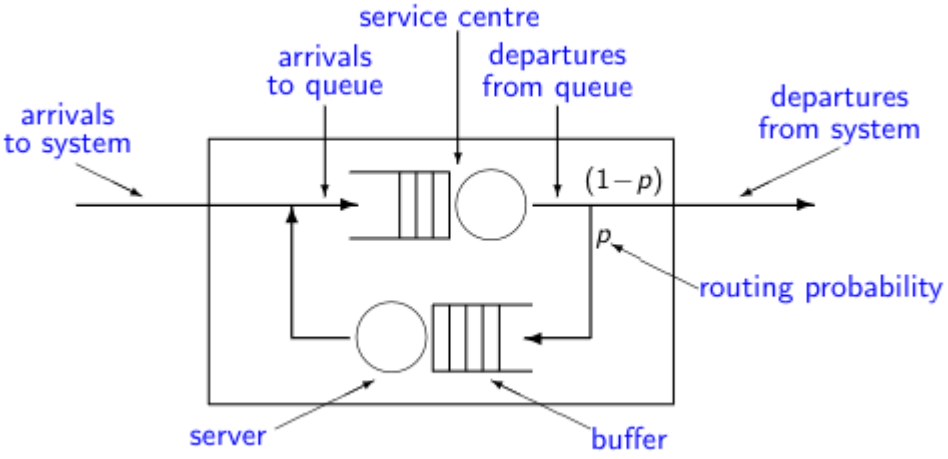
Multiclass representation is needed as average representation don't mean anything

Queueing Networks

For many systems we can adopt a view of the system as a collection of resources and devices with customers or jobs circulating between them

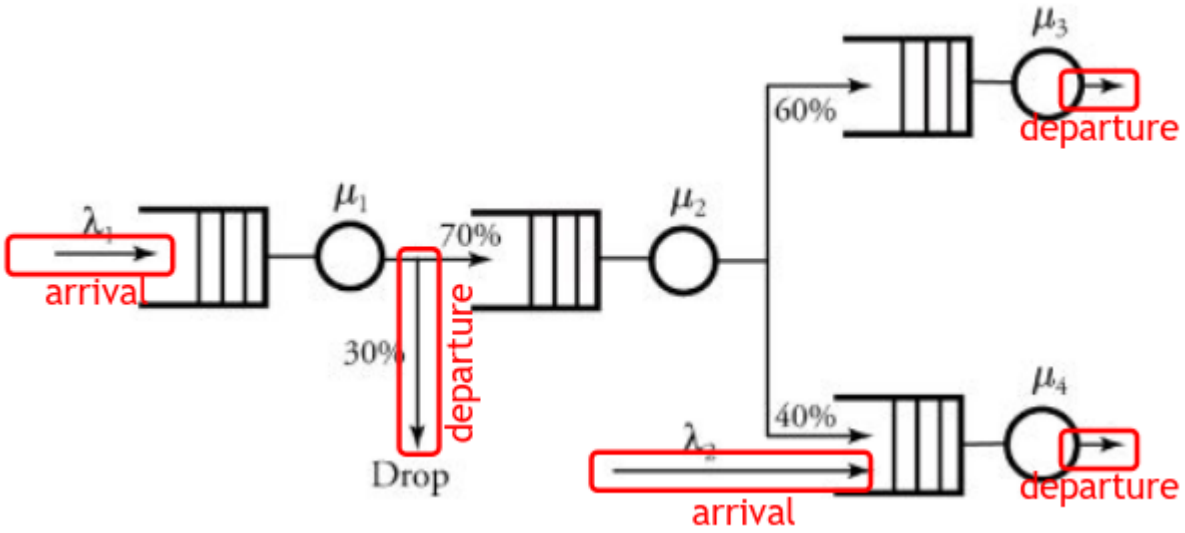


We can model the network as a graph and the nodes are connected as they are part of the same system.
A queueing network can be represented as a graph where nodes represent the service centers k and arcs the possible transitions of users from one service center to another
Nodes and arcs together define the network topology

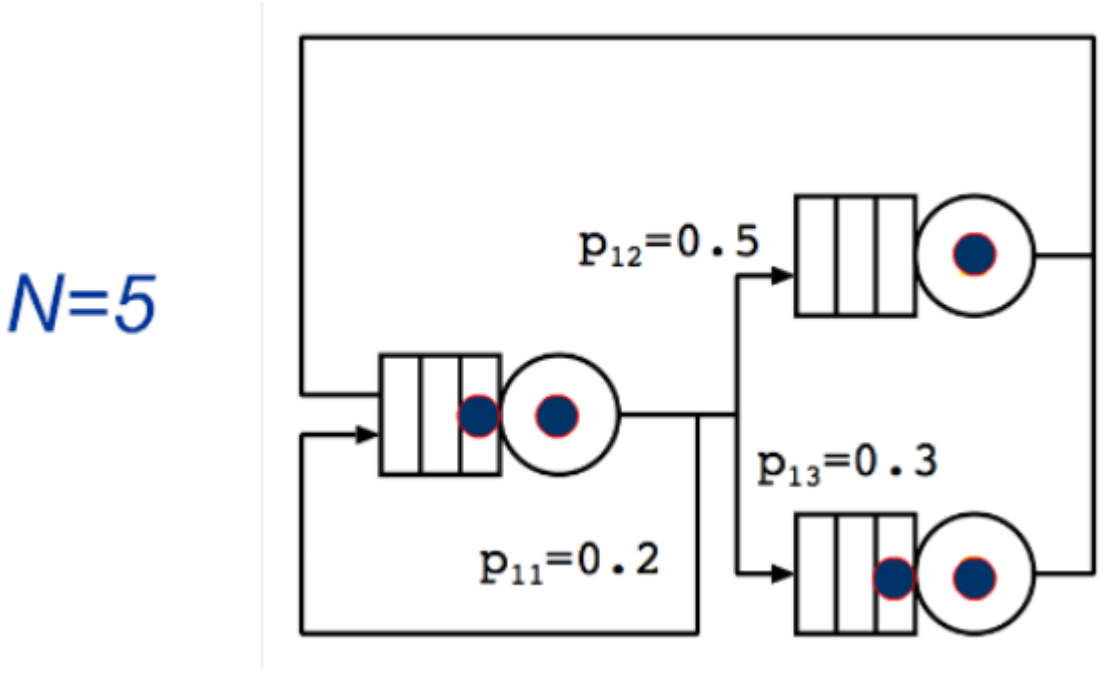


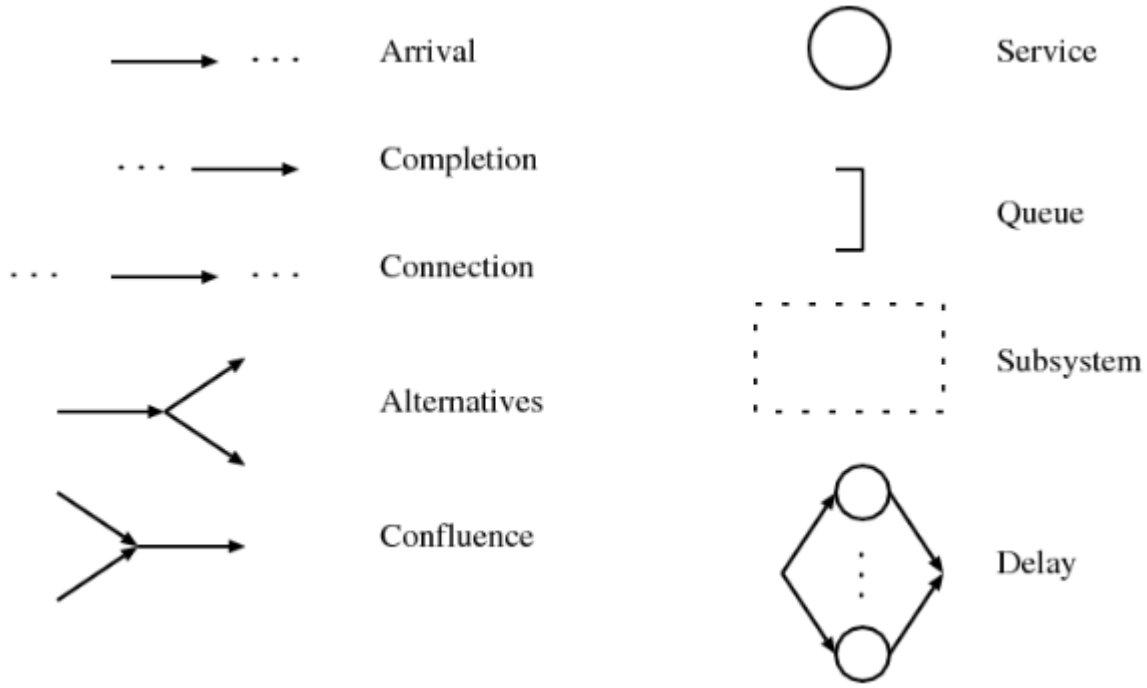
A network may be:

- Open: customers may arrive from, or depart to, some external environment(infinite size). Open Models are characterized by arrivals and departures from the system



- Closed: a fixed population of customers remain within the system. In closed models we have a parameter N that accounts for the fixed population of jobs that continuously circulate inside the system





- Mixed: there are classes of customers within the system exhibiting open and closed patterns of behaviour respectively

  Routing

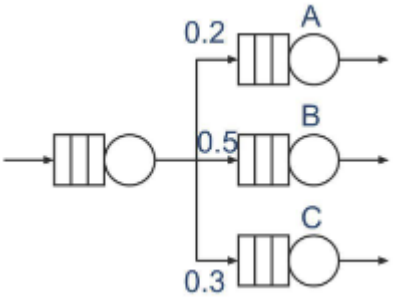
  Whenever a job, after finishing service at a station has several possible alternative routes, an appropriate selection policy must be defined

  The policy that describes how the next destination is selected is called routing

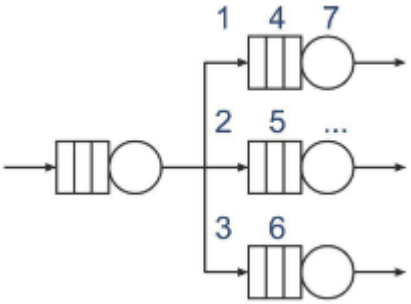
  Routing specification is required only in all the points where jobs exiting a station can have more than one destination

  The main routing algorithms that we will consider are:

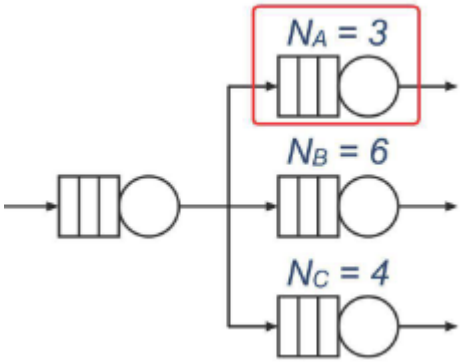
- Probabilistic: each path has assigned a probability of being chosen by the job that left the considered station



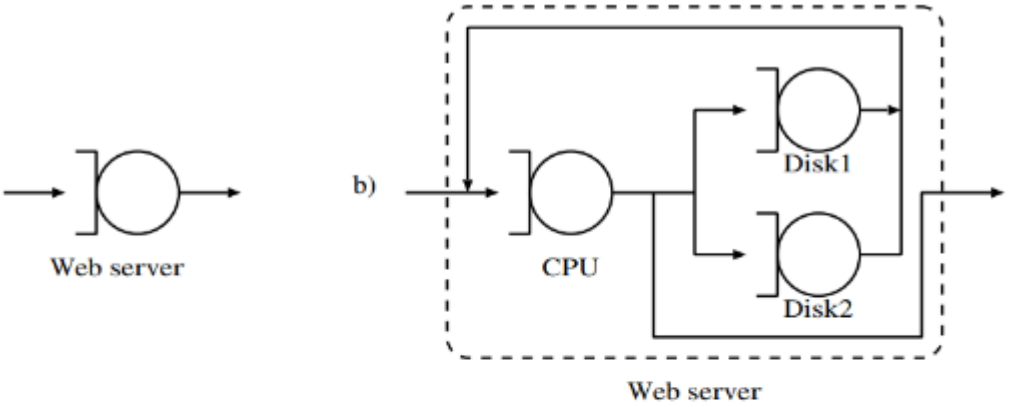
- Round robin: the destination chosen by the job rotates among all the possible exits

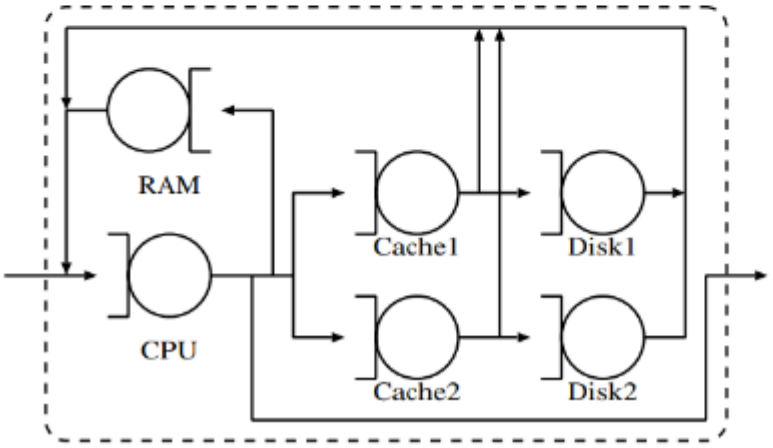


- Join the shortest queue: jobs can query the queue length of the possible destinations, and chose to move to the one with the smallest number of jobs waiting to be served



Level of Detail





In the modeling everything depends on the accuracy that you want to achieve(if not satisfied you open the box and look at the internal components).
Proceed top-down, possibly detailing service centers as need arises
Parameterizing the model is hard, so it is better to keep the complexity low