POLITECNICO DI MILANO

Performance Evaluation and Applications





Separable and Open models

Motivating example

A server for an on-line game, takes on the average $S_1 = 10\ ms$ to send the state of the scene to one client. The network to which it is connected, can handle a request with an average of $S_2 = 2\ ms$. Each client then requires an extra time $Z = 15\ ms$ to unpack and process the data received. For the game being playable, the total delay must be less than $R < 50\ ms$.

Which is the maximum global arrival rate the system can handle?

Queue policies

When a service finishes and there are more than one job waiting in the queue, the system must decide which one to serve next.

Queue policies

Several different policies to select the next job exist: each one has its goal and can work better than another for a specific application.

Queuing policies

Two types of queuing policies exists:

- **Non-preemptive**
- **Preemptive**

In *non-preemptive* policies, when a job starts it cannot be interrupted.

Preemptive policies can stop a job in service when some event occurs, and start serving some other.

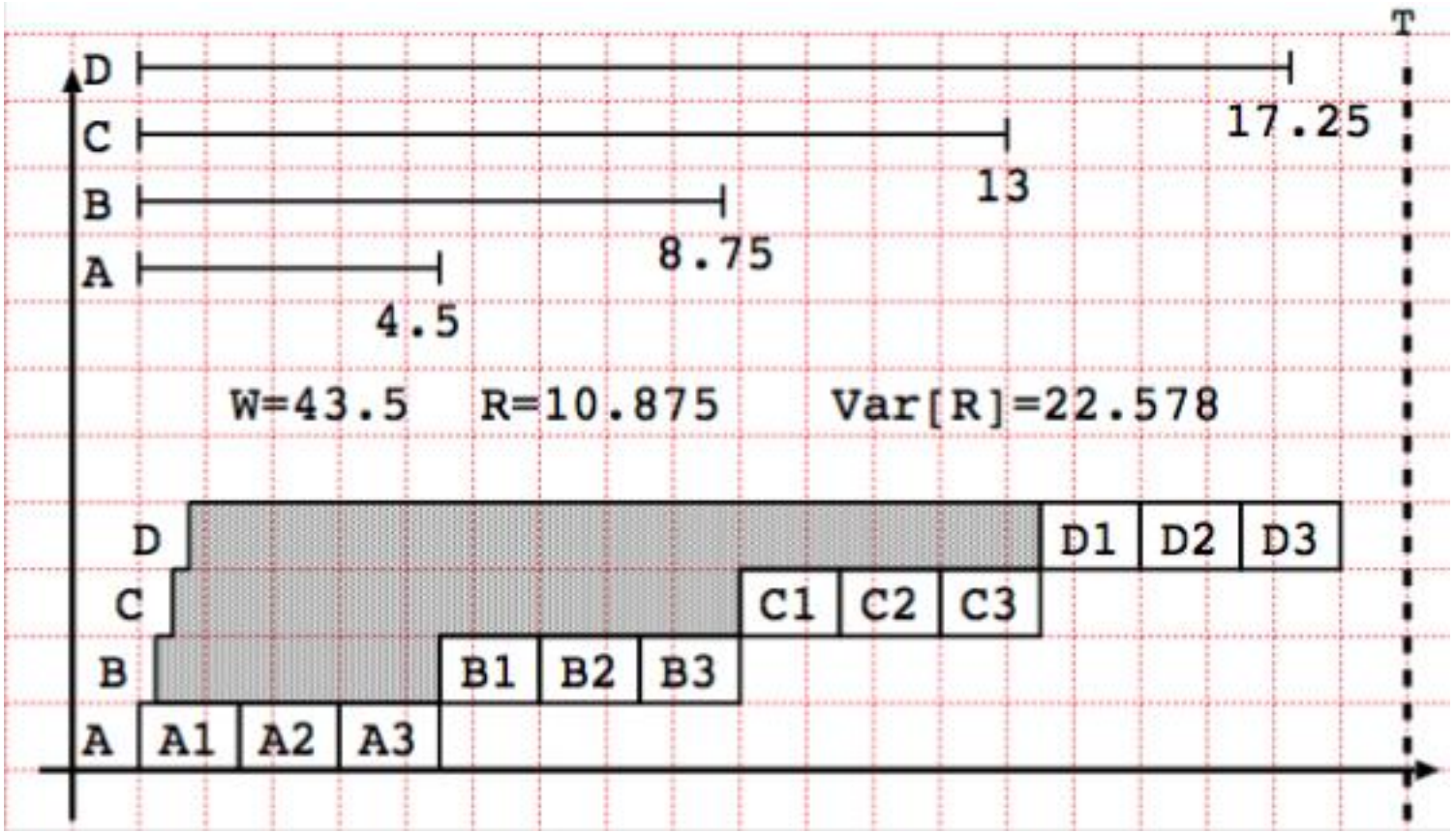
Non-preemptive queuing policies

The most common non-preemptive policies are:

FIFO or FCFS	•	first-in first-out
LIFO or LCFS	•	last-in first-out
SIRO	•	service in random order
SJF	•	shortest jobs first
LJF	•	longest jobs first

Non-preemptive queuing policies

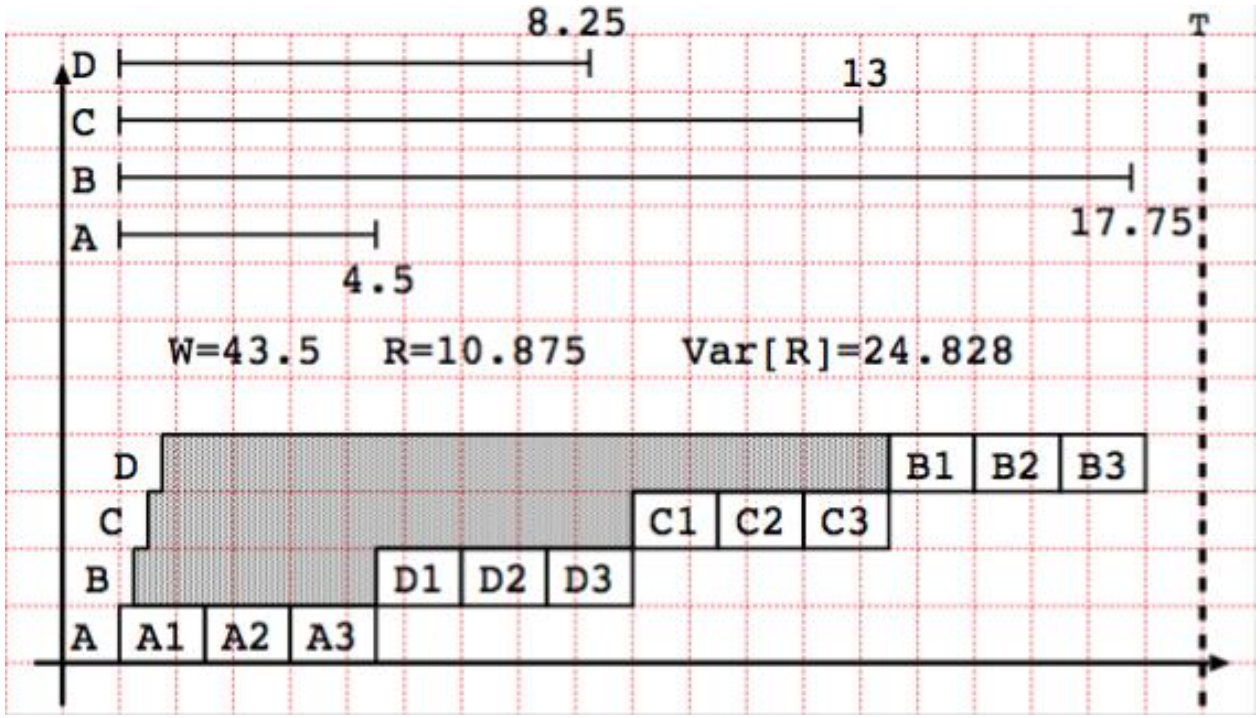
In *First-In First-Out* (also known as *First-Come First-Served*), jobs are served in the order in which they arrive.



Non-preemptive queuing policies

In *(non-preemptive) Last-In First-out* (or *Last-Come First-Served*), jobs are served in the opposite order with respect to their arrival at the station.

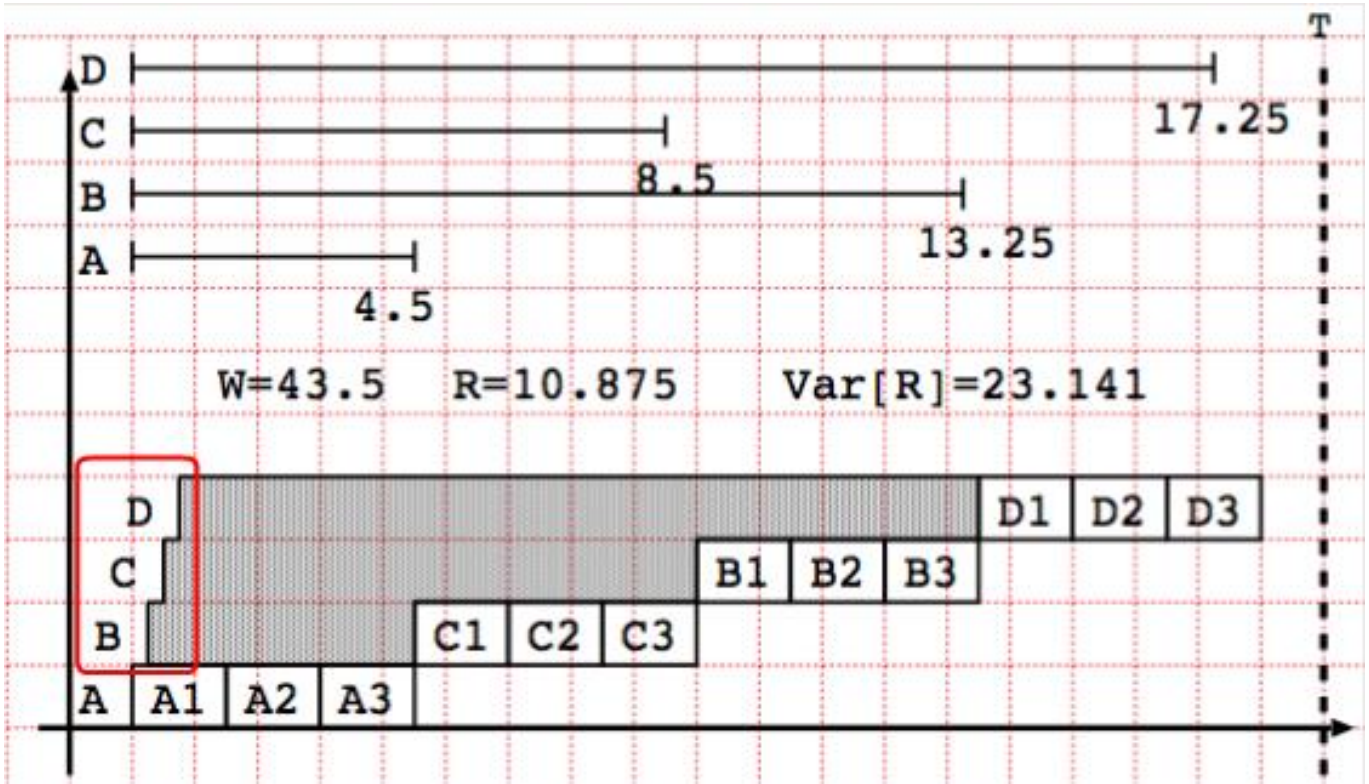
Service is never interrupted: as soon as the job in service exits the system, the last job that arrived meanwhile is selected.



Non-preemptive queuing policies

With the *Service In Random Order* policy, when a job finishes its service, the next one is chosen randomly among the ones waiting in the queue.

All jobs have exactly the same probability of being chosen.



Non-preemptive queuing policies

It can be proven that all the non-preemptive queuing policies that do not use the length of the jobs in service to perform the selection, *have the same average-response time*.

These policies include FCFS, non-preemptive LCFS and SIRO.

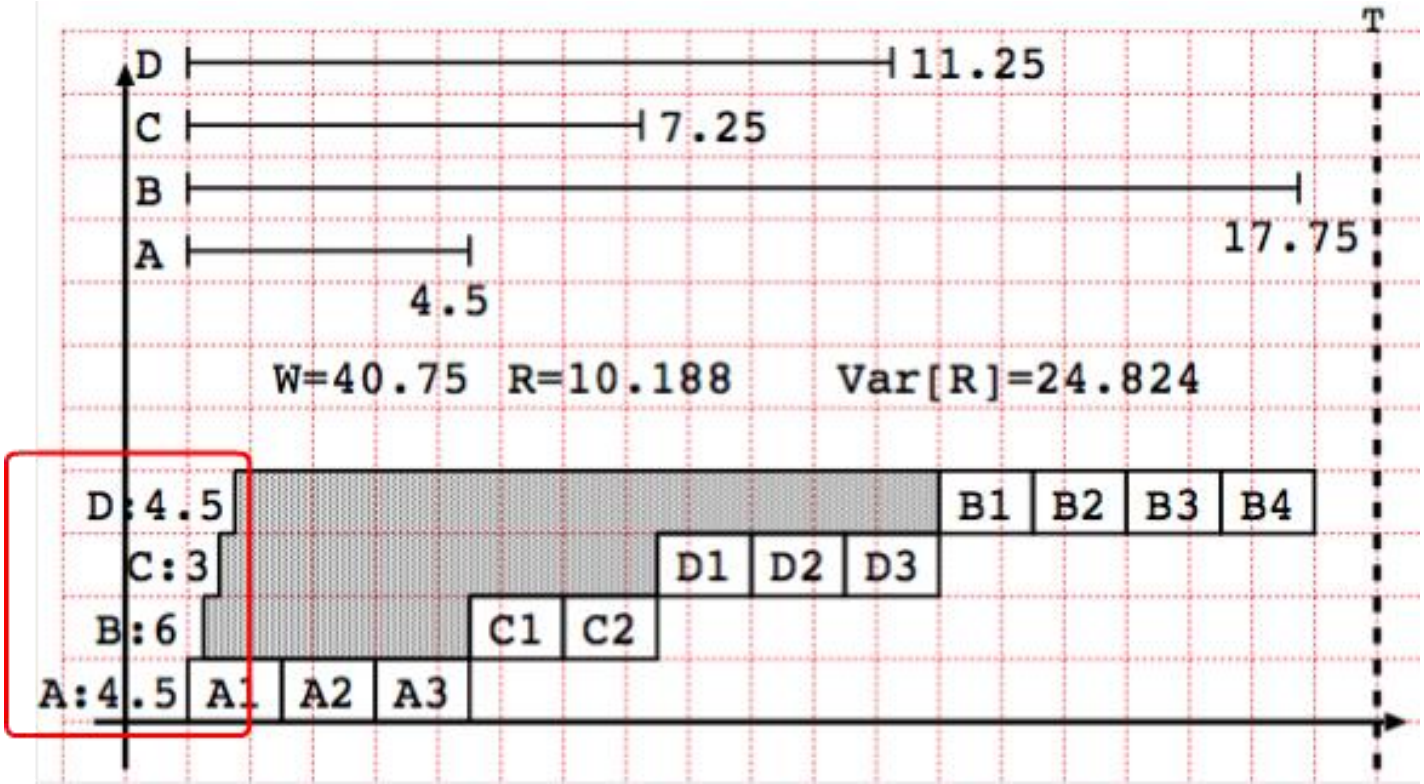
The three policies however have a different variance of the response time, and in particular it can be shown that:

$$Var[FCFS] \leq Var[SIRO] \leq Var[LCFS]$$

Non-preemptive queuing policies

With the *Shortest (Longest) Job First* policy, the duration of the job is known at the time it joins the station.

Jobs are ordered according to their service requirement: when a job leaves the system, the one with the shortest *(longest)* service time is picked from the queue.



Non-preemptive queuing policies

Let us consider a queue with exponential service time (average 0.8) and exponential inter arrival time (average 1).

The following table shows the *average* and the variance of *the service* time computed using the JMT tool.

	Average	Variance
FCFS	3.9884	15.8227
LCFS	4.0072	118.2334
SIRO	3.9935	32.8912
SJF	2.3029	20.3595
LJF	8.1244	309.4463

Preemptive queuing policies

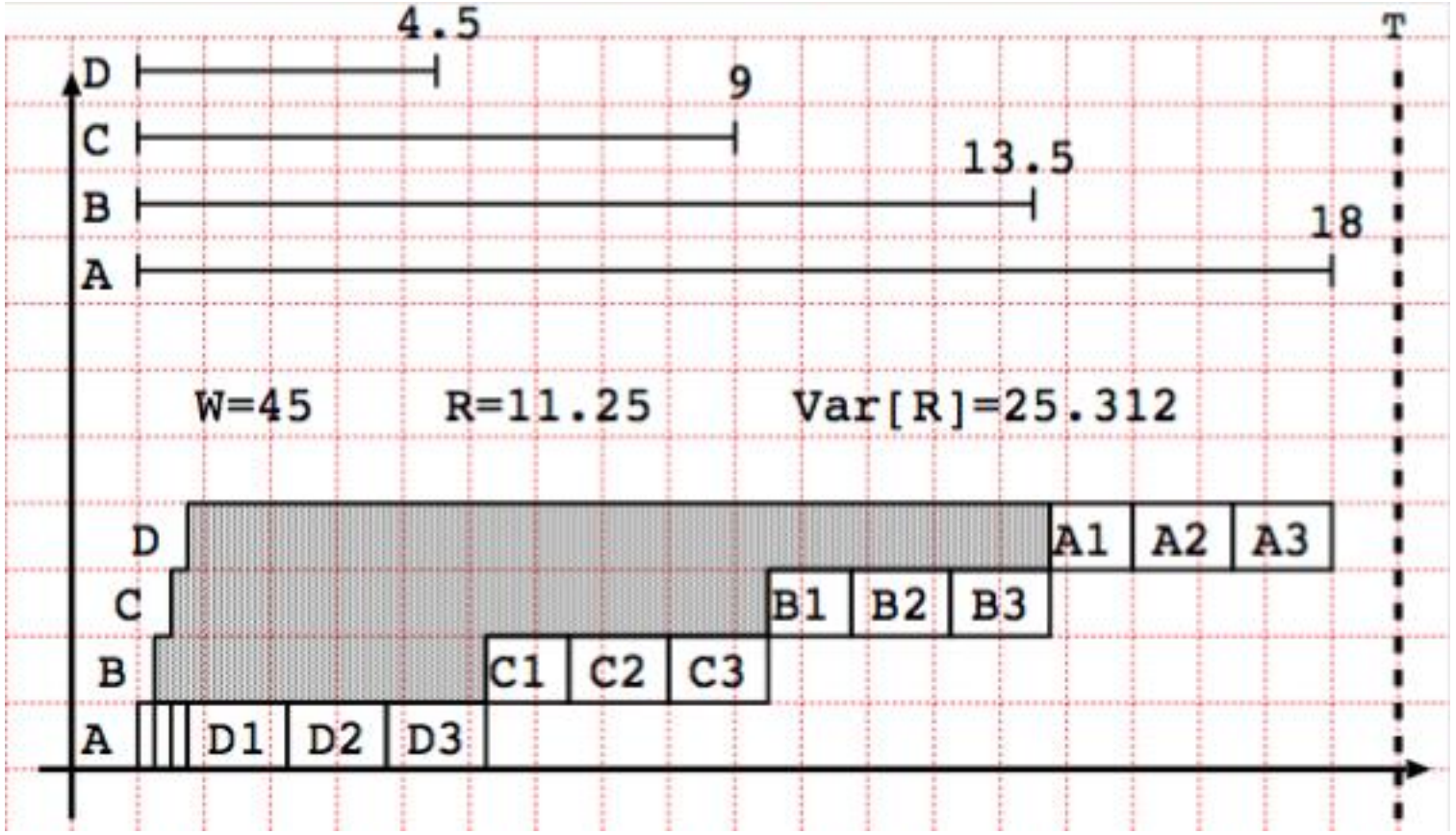
The most common preemptive policies are:

LIFO or LCFS	• last-in first-out *(preemptive)*
RR	• round robin
PS	• processor sharing
SRTF	• shortest remaining time first

Preemptive queuing policies

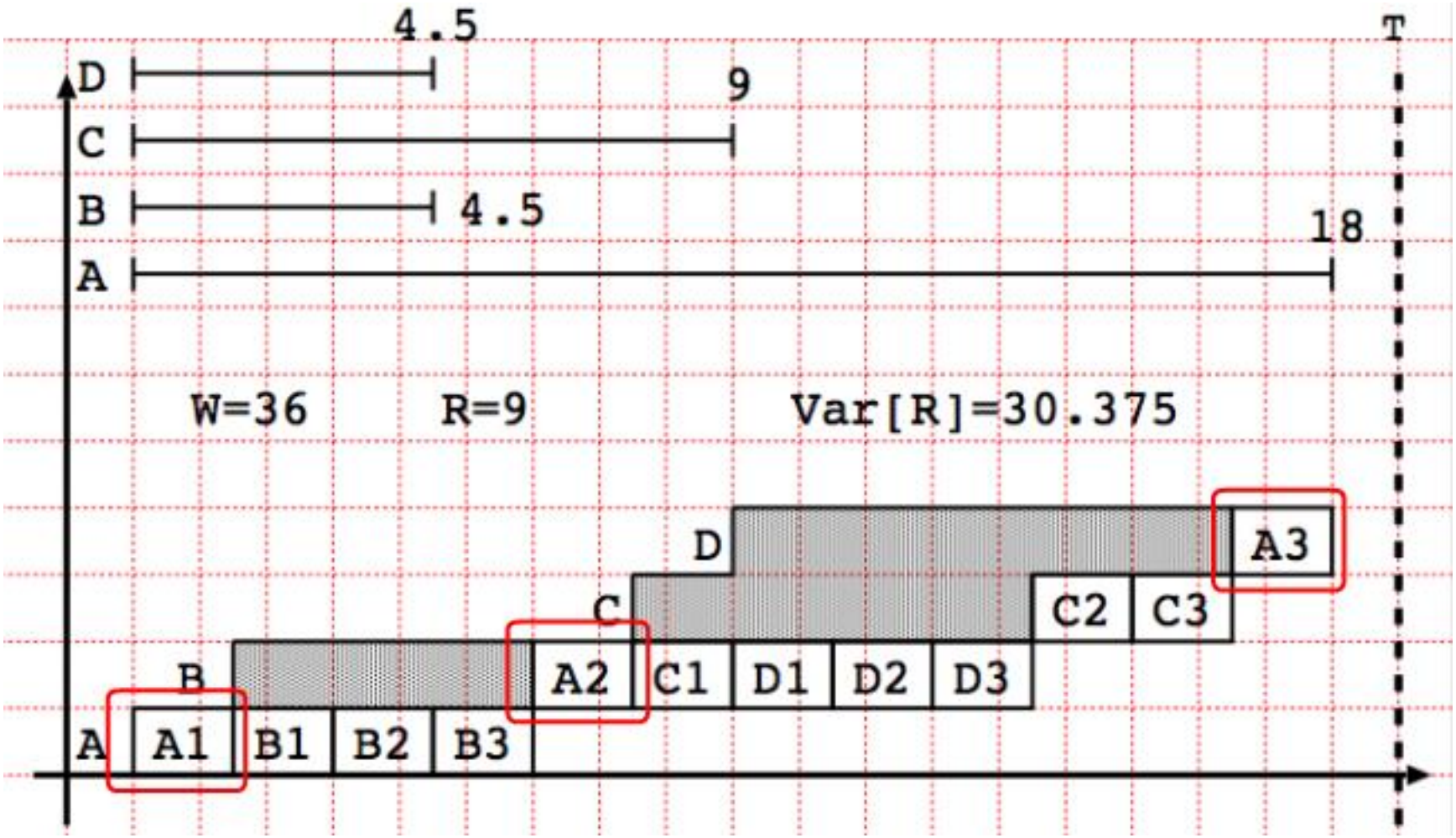
In the *(Preemptive) Last-Come First-Served*, as soon as a new job arrives, it stops the one currently in service and replaces it.

The next job from the queue is chosen in Last-Come First-Served order.



Preemptive queuing policies

Note that a job can be interrupted several times by the following ones.

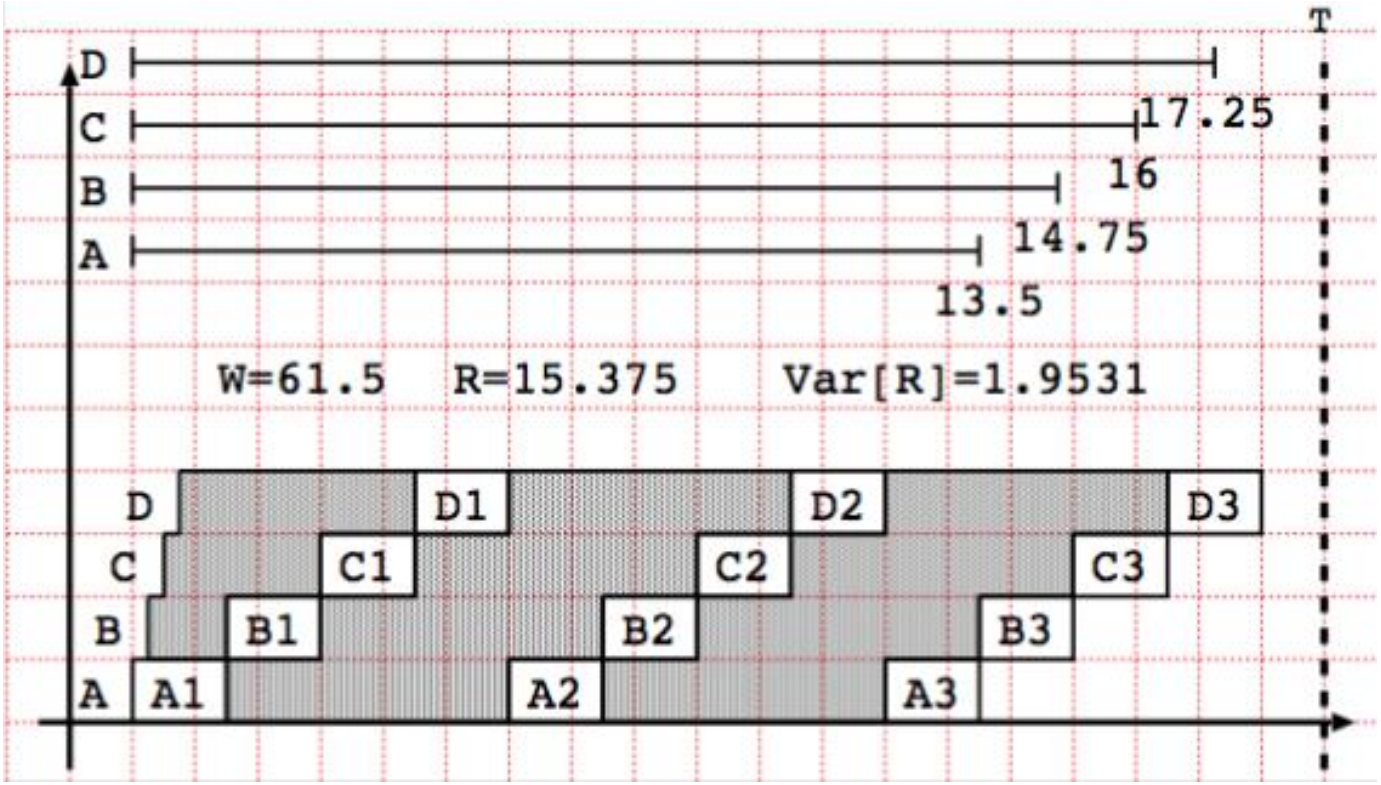


Preemptive queuing policies

In *Round Robin*, each job is assigned a maximum service time called *quantum*.

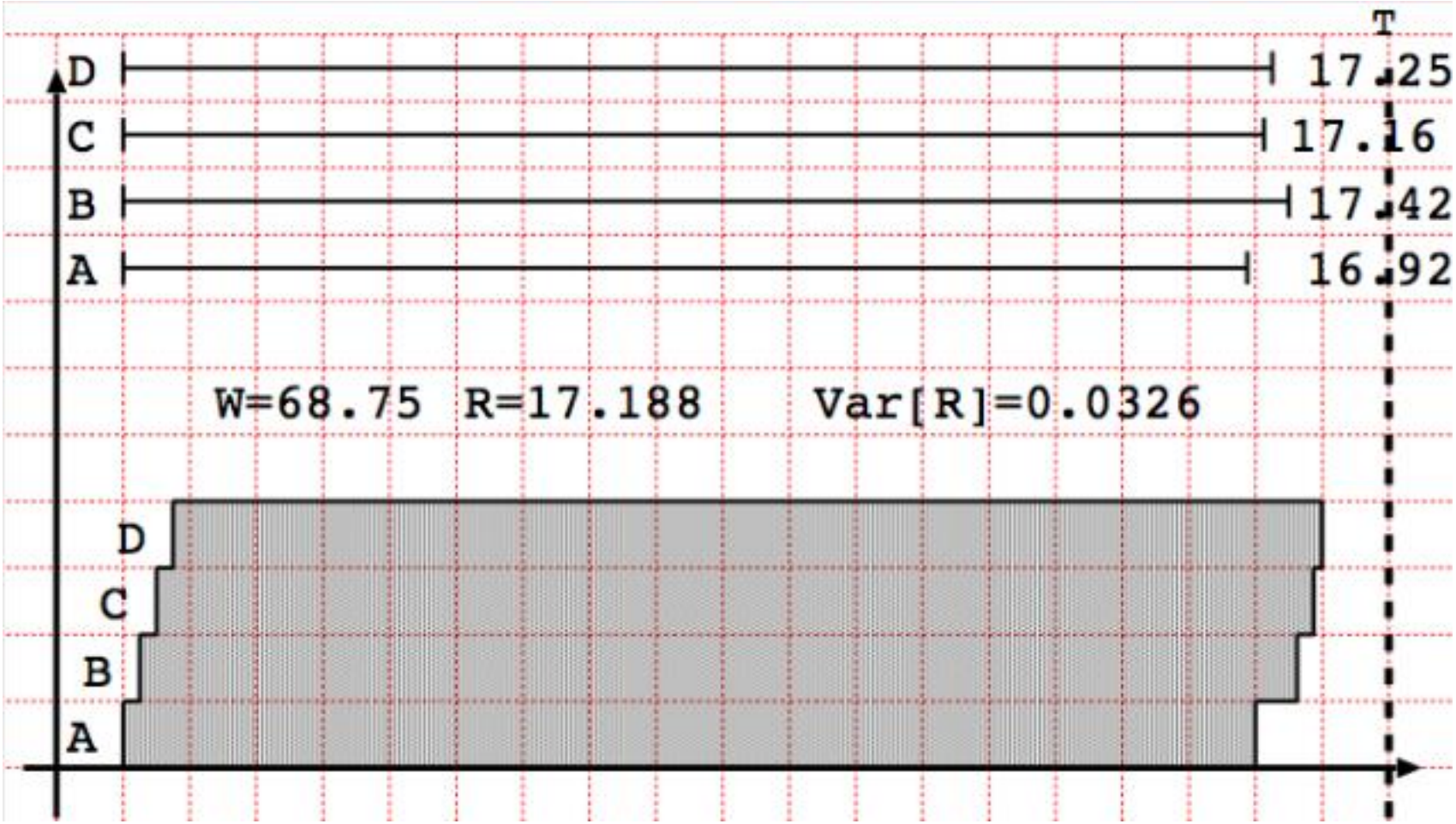
If the job does not finish at the end of its quantum, it is interrupted and replaced by another one in the queue.

The jobs are usually selected from the queue in FIFO order.



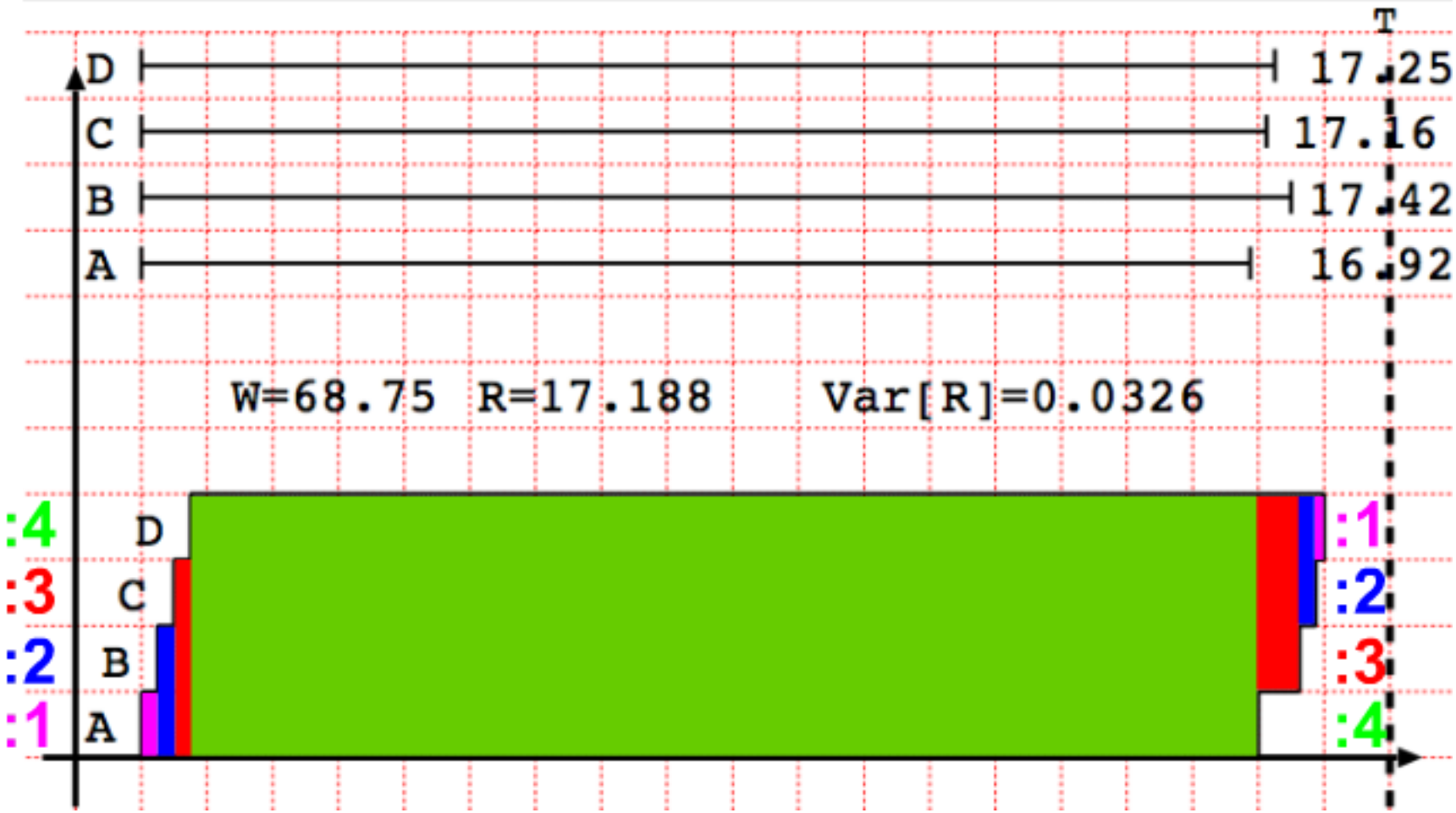
Preemptive queuing policies

Processor Sharing is the equivalent of the Round Robin when the length of the quantum tends to zero.



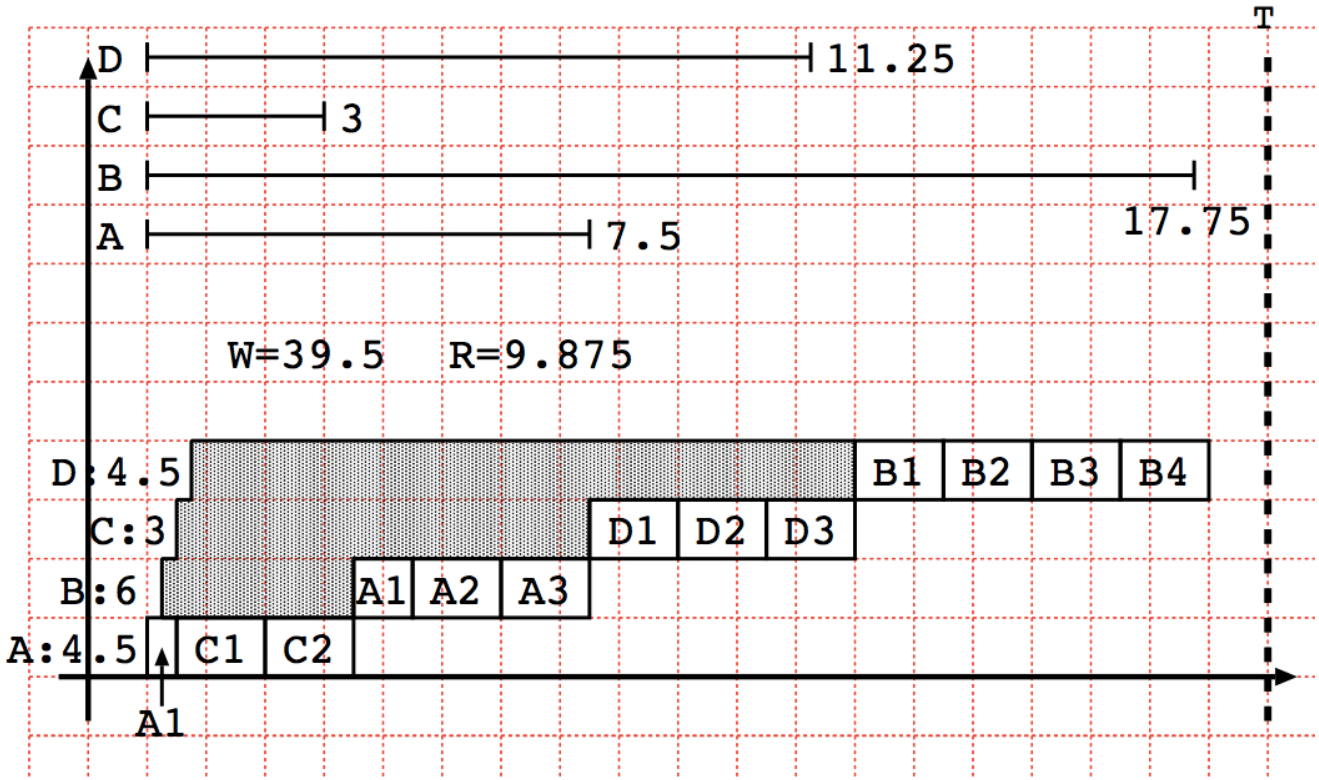
Preemptive queuing policies

In this case, all the jobs are run concurrently at a slower speed, that depends on the total population of the system.



Preemptive queuing policies

Shortest Remaining Time First is the preemptive equivalent of the SJF policy. When a new job arrives, if it is shorter than the one in service, it interrupts the ongoing one, and it starts immediately.



Preemption types

When a service is interrupted, there are different ways in which the service can be resumed.

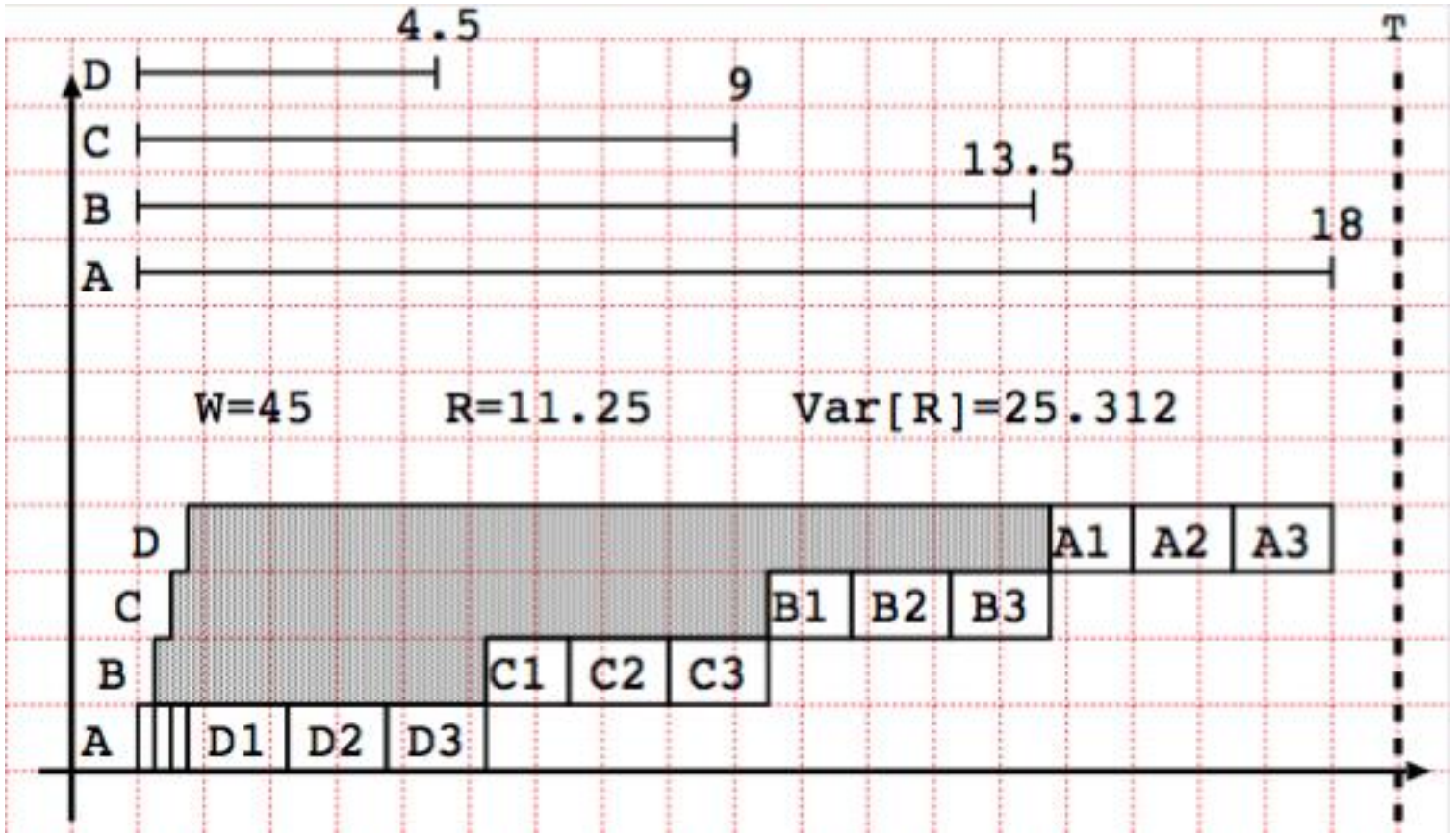
We will present the two main preemption types, and show them for the of LCFS discipline:

- Preemptive Resume - PRS
- Preemptive Repeat (or Restart) - PRD

Note that the same approaches can be used for any job that can be interrupted.

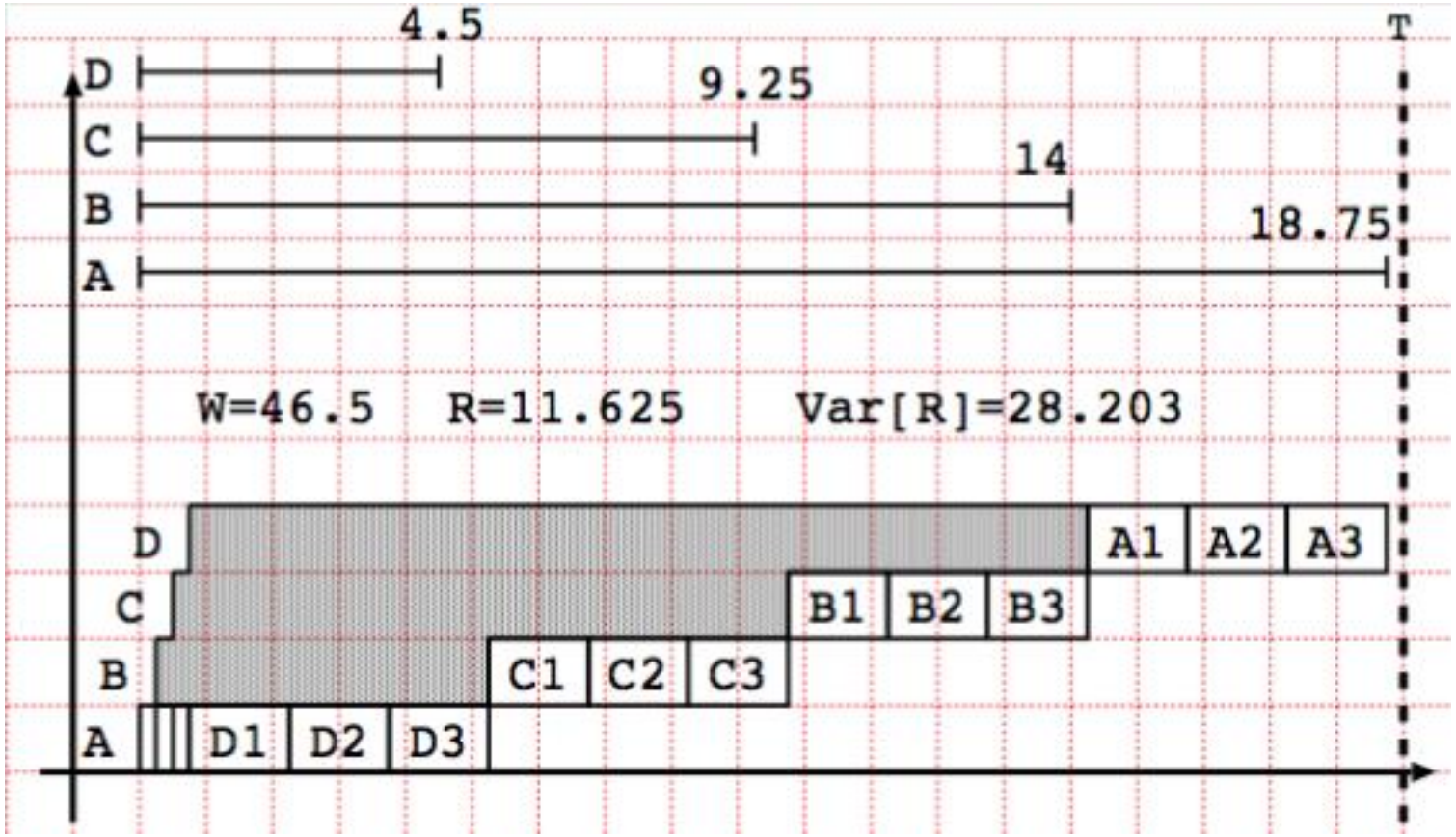
Preemption types

In the *Preemptive Resume* policies, the service continues from the point it stopped, according to the remaining service time at the time of the interruption.



Preemption types

In the *Preemptive Repeat* the job is restarted from the beginning every time it returns in service.



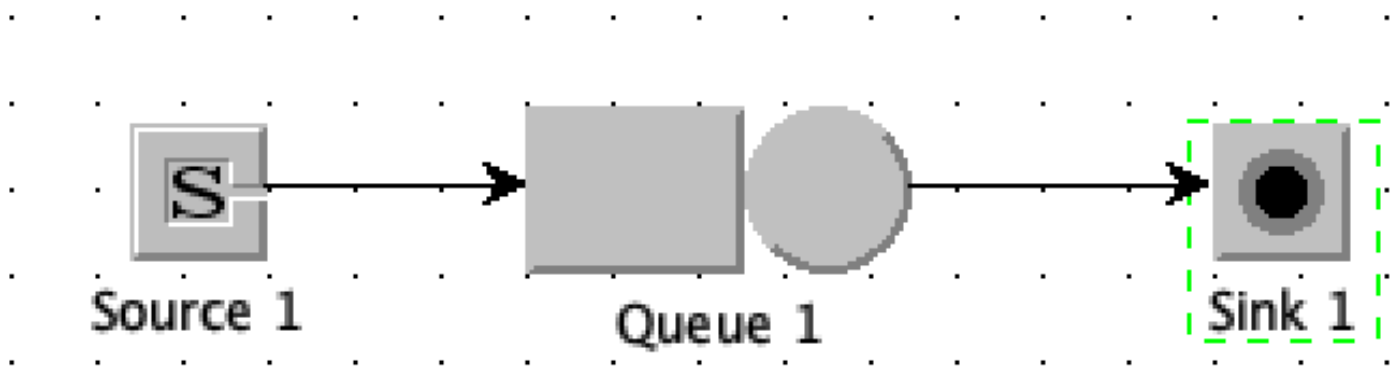
Separable models

Let us consider a simple system composed by a single queue, under different arrival and service processes, and considering different queuing policies.

We consider for both service and arrival process:

- Exponential
- Erlang, with c.v. = 0.5
- Hyper-Exponential, with c.v. = 2

As queuing policies we consider LCFS, PS and FCFS



Separable models

The results obtained using JMT and discrete event simulation for the *Average Response Time (R)* are the following:

Arrival (Average l=0.8 j/s)	Service (Average D = 1s)			
	LCFS			
		Exp	Erl	Hyper
	Exp	4.9845	3.5012	10.9925
	Erlang	3.2733	1.8351	9.2016
	Hyper-exp	10.7959	9.1543	17.2105
	PS			
		Exp	Erl	Hyper
	Exp	4.9813	5.031	4.9152
	Erlang	3.369	2.366	4.1592
	Hyper-exp	10.8972	13.8811	7.8524
	FCFS			
		Exp	Erl	Hyper
	Exp	5.0473	3.5016	11.0676
	Erlang	3.2566	1.8468	9.1904
	Hyper-exp	10.8286	9.0071	17.2209

Average Response Time (R)

Separable models

With the considered arrival rate and demand, analytical results for the M/M/1 model would be:

$$R = \frac{D}{1 - \lambda \cdot D} = \frac{1}{1 - 0.8 \cdot 1} = 5s$$

Note that, beside the M/M/1 case (orange background), such results are also obtained for the PS queuing policy with exponential inter-arrival time, and for M/M/1/LCFS (yellow background).

Arrival (Average l=0.8 j/s)	Service (Average D = 1s)			
	LCFS			
		Exp	Erl	Hyper
	Exp	4.9845	3.5012	10.9925
	Erlang	3.2733	1.8351	9.2016
	Hyper-exp	10.7959	9.1543	17.2105
	PS			
		Exp	Erl	Hyper
	Exp	4.9813	5.031	4.9152
	Erlang	3.369	2.366	4.1592
	Hyper-exp	10.8972	13.8811	7.8524
	FCFS			
		Exp	Erl	Hyper
	Exp	5.0473	3.5016	11.0676
	Erlang	3.2566	1.8468	9.1904
	Hyper-exp	10.8286	9.0071	17.2209

Separable models

Moreover, note that both FCFS and LCFS have basically the same Average Response Time.

Arrival (Average l=0.8 j/s)	Service (Average D = 1s)			
	LCFS			
		Exp	Erl	Hyper
	Exp	4.9845	3.5012	10.9925
	Erlang	3.2733	1.8351	9.2016
	Hyper-exp	10.7959	9.1543	17.2105
	PS			
		Exp	Erl	Hyper
	Exp	4.9813	5.031	4.9152
	Erlang	3.369	2.366	4.1592
	Hyper-exp	10.8972	13.8811	7.8524
	FCFS			
		Exp	Erl	Hyper
	Exp	5.0473	3.5016	11.0676
	Erlang	3.2566	1.8468	9.1904
	Hyper-exp	10.8286	9.0071	17.2209

Average Response Time (R)

Separable models

It has been discovered that the analytical results such the one presented for Open and Closed networks, are valid for a large variety of systems.

In particular, the techniques e can be applied if a system is a *separable queuing network*.

A queueing network is *separable if* it fulfills five main properties.

In the following, we will present the properties directly from the book **Quantitative System Performance** by E. D. Lazoswka.

Assumptions

The first assumption is:

service center flow balance — Service center flow balance is the extension of the flow balance assumption (see Chapter 3) to each individual service center: the number of arrivals at each center is equal to the number of completions there.

Losses clearly violate the *service center flow balance* (1st) assumption, and thus such models are not separable.

Assumptions

The second assumption is:

one step behavior — One step behavior asserts that no two jobs in the system "change state" (i.e., finish processing at some device or arrive to the system) at exactly the same time. Real systems almost certainly display one step behavior.

Assumptions

The third assumption is:

routing homogeneity — To this point we have characterized the behavior of customers in the model simply by their service demands. A more detailed characterization would include the routing patterns of the jobs, that is, the patterns of centers visited. Given this more detailed view, routing homogeneity is satisfied when the proportion of time that a job just completing service at center j proceeds directly to center k is independent of the current queue lengths at any of the centers, for all j and k. (A surprising aspect of separable models is that the routing patterns of jobs are irrelevant to the performance measures of the model. Thus, we will continue to ignore them.)

Assumptions

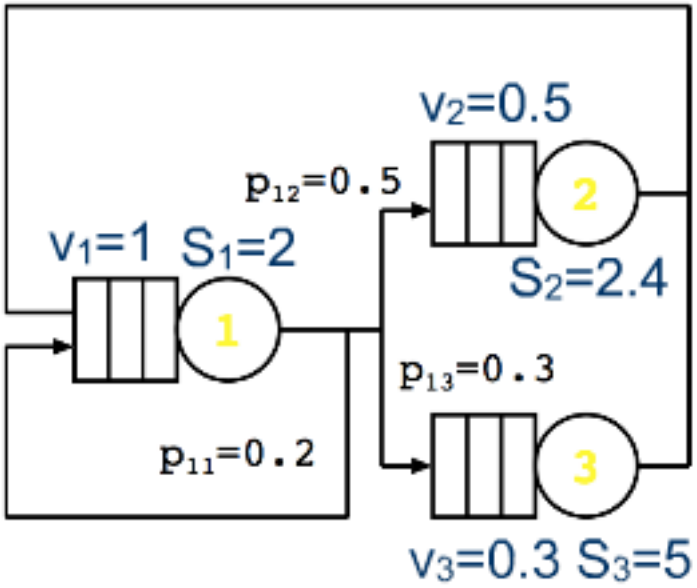
Of the routing policies we have seen, the probabilistic routing fully satisfies the previous assumption.

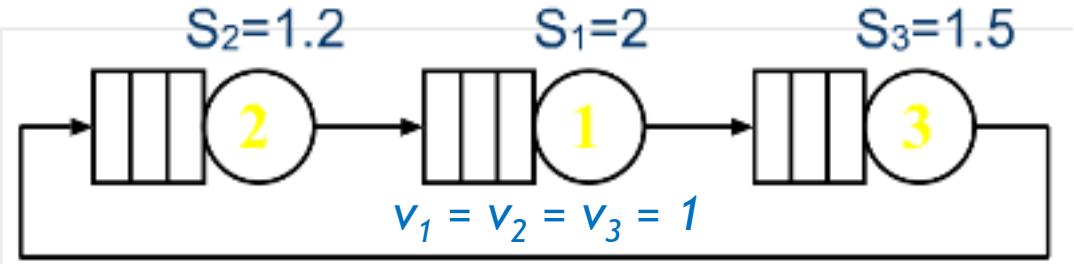
Other routing policies, such as Join the Shortest Queue (JSQ) clearly violate this constraint since the decision of the next station depends on the whole state of the system.

The previous assumption has a very big implication: as long as nodes have the same demands, two queuing networks have the same performance, independently of their topology.

Assumptions

For example, these two networks have exactly the **same performances** in term of *system throughput*, *system response time*, *average number of jobs*, *utilizations* and *residence times*.



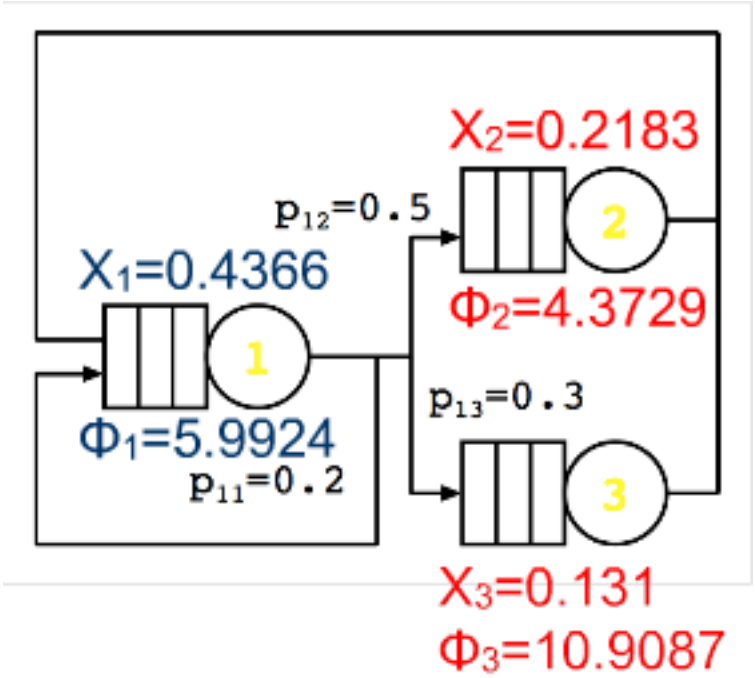


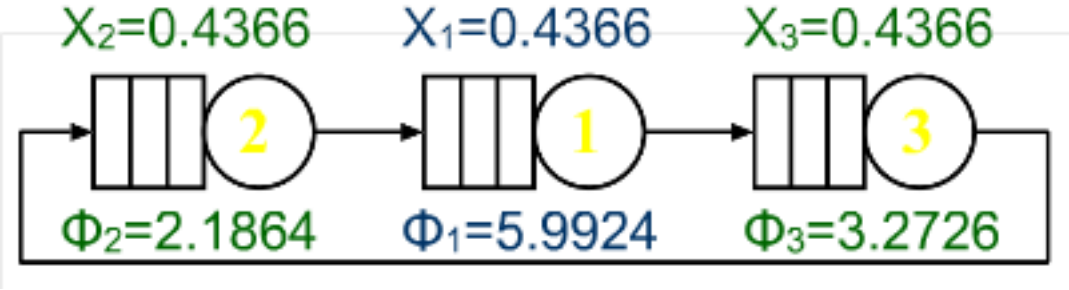
X=0.4366
R=11.4515

U1=0.8732	U2=0.5239	U3=0.6549
N1=2.6165	N2=0.9546	N3=1.4289
R1=5.9925	R2=2.1864	R3=3.2726

Assumptions

However, the two systems have different visits. This means that the two models have **different** *stations throughput* and *resource response times* since these performance indices strongly depend on the visits to the corresponding stations.





Assumptions

In particular, such performance indices are defined as follows:

$$X_k = v_k \cdot X$$

$$\Phi_k = \frac{N_k}{X_k} = \frac{R_k}{v_k}$$

To analyze a queuing network, we need to know its topology just to determine the visits and the demands of the different stations.

We can then perform the analysis using only demands.

Assumptions

The fourth assumption is:

device homogeneity — The rate of completions of jobs from a service center may vary with the number of jobs at that center, but otherwise may not depend on the number or placement of customers within the network.

This is a very tricky and important assumption that considers the combination of service time distributions, type of servers and queuing policies.

Assumptions

The fifth assumption is:

homogeneous external arrivals — The times at which arrivals from outside the network occur may not depend on the number or placement of customers within the network.

Correlated and burst arrivals in open models generally violate this assumption, making them no longer separable.

Arrival and Service time distribution

In general, the distribution alone is not enough to determine whether the properties of separable networks are satisfied or not.

In particular, they might violate or not the *device homogeneity* or the *homogeneous external arrival properties* (4^{th} and 5^{th} respectively).

Non-preemptive queuing policies satisfy the *device homogeneity property* (4^{th}) only for exponential service time distributions.

Finite capacity

Finite capacity almost always do **not** fulfill the *device homogeneity property* (4th assumption).

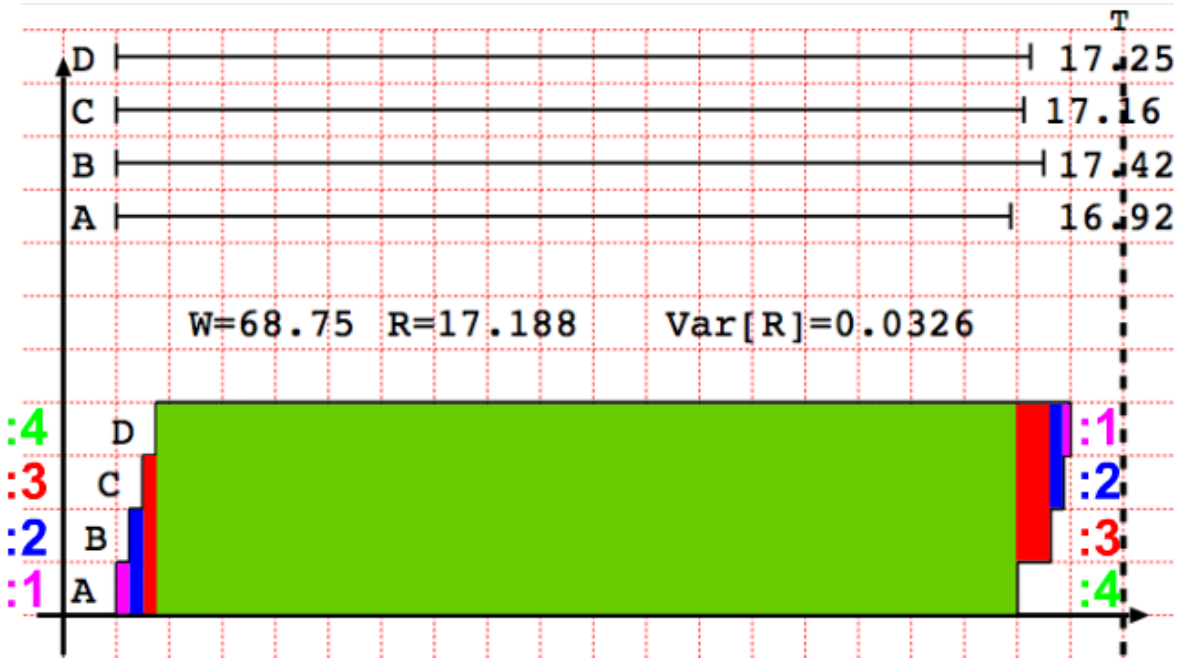
However, for special types of blocking, special network topologies and special service policies, finite capacity models have analytical solution.

Preemptive queuing policies

Preemptive service policies may satisfy the *device homogeneity property* (4^{th}) more often than the non-preemptive ones.

In particular queues with the *Processor Sharing* policy are separable regardless of their service time distribution*.

* Open models have the additional requirement of Poisson arrivals.



Assumptions

If we also add a sixth assumption (that reduces the scope of the fourth) solutions can be computed with simpler algorithms:

service time homogeneity — The rate of completions of jobs from a service center, while it is busy, must be independent of the number of customers at that center, in addition to being independent of the number or placement of customers within the network.

The MVA technique for closed models, and the direct solution for open models, require all 6 assumptions to hold.

Multiple servers and queue length dependencies

Multiple server clearly violates the *service time homogeneity* (6th assumption), so even if separable they require special techniques to be solved.

Infinite server stations (delay centers) instead never create problems in separable models since they do not involve any queuing.

Assumptions: discussion

Even if the previous assumptions seem very strict, they are valid for a large set of models of computer systems.

Moreover, for systems that do not fulfill the previous properties, the considered techniques can provide meaningful approximations.

Network performance indices

On network models, extra performance indices can be defined:

- System Throughput
- Total System Population
- System Response Time

For what concerns utilization, there is not an unique definition of a system-wise measure. Three of the most popular definitions are:

- The fraction of time in which there is at least one job in the system
- The average utilization of all the stations
- The utilization of the bottleneck station

All definitions are have strength and weakness, so we will simply exclude such performance index, and do not consider any in this course.

Network performance indices

If an open model is stable and no blocking or loss occur, the *total arrival rate* corresponds to the *System Throughput*: the average rate at which jobs exit from the system.

$$X = \lambda_0$$

In closed models, the throughput is a monotonic increasing function of the number of jobs in the system *N*, and it tends to an asymptotic value.

Stability in Open models

Open models can be stable only if their utilization is less than one for all their stations:

$$U_k < 1 \quad \forall k$$

$$\lambda < \frac{1}{D_k} \quad \forall k$$

$$\lambda < \frac{1}{\max_k D_k}$$

In closed models, this is the value to which the system throughput tends as the population size N tends to infinity.

Network performance indices

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

$$N = \sum_{i=1}^{K} N_i$$

Of course, this quantity is a constant in closed models.

In open models instead it depends on the workload λ.

Network performance indices

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

$$R = \sum_{i=1}^{K} R_i$$

Note the discrepancy: the system ***response time*** is the sum of ***residence times*** and not of the *response times* at the stations. This is because the response time at the station is conditioned to the fact that the job will pass exactly once through the node, condition which might not always be true.

Conversely, a *system residence time* is identical to the *system response time*, since each different job is assumed to visit the system exactly **once**.

Network performance indices

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

$$N = \sum_{i=1}^{K} N_i = \sum_{i=1}^{K} X \cdot R_i = X \cdot \sum_{i=1}^{K} R_i = X \cdot R$$

Performances of open models

All the main performance indices of separable models can be computed using simple algorithms.

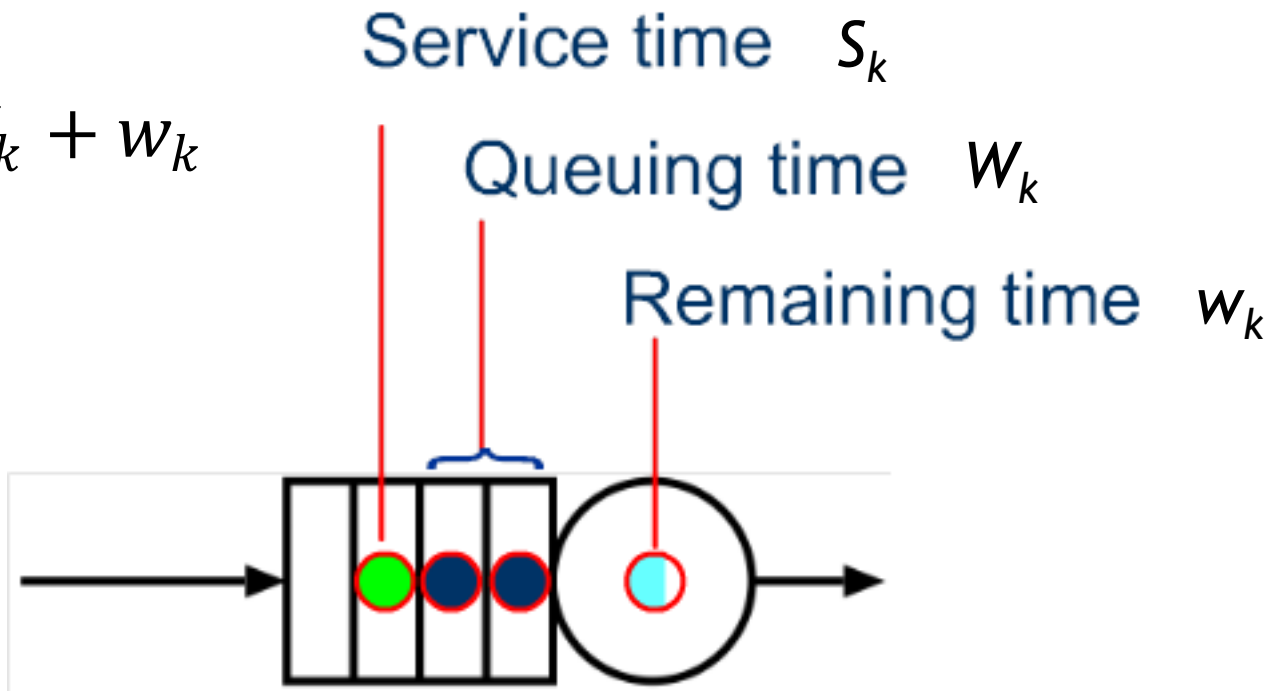
The computations can be carried on manually, or specific tools can be used.

Performances can be computed starting from *the average number of jobs found in the queue by a new arrival to a station*.

Jobs at the arrival

As we already did for the M/G/1, let us focus on the response time Φ_k of a station k (time spent at node k during one visit). It is the sum of 3 components: the service time of the arriving job S_k, the queuing time W_k, and the remaining service time w_k of the job found at the arrival.

$$\Phi_k = S_k + W_k + w_k$$



Jobs at the arrival

For delay stations (i.e. the terminal station in time-sharing systems), both the remaining service time w_k and the queuing time W_k are zero, since each job is served immediately and in parallel with the others.

Response time Φ_k is then identical to the average service time S_k.

$$W_k + w_k = 0$$
$$\Phi_k = S_k$$

Jobs at the arrival

In all the other cases, we assume that the remaining service time w_k plus the waiting time W_k is identical to the average service time of one job S_k multiplied with A_k, the average number of jobs found in the system at the arrival.

$$W_k + w_k = A_k \cdot S_k$$

Response time Φ_k becomes:

$$\Phi_k = S_k + (W_k + w_k) = S_k + A_k \cdot S_k = (1 + A_k) \cdot S_k$$

Jobs at the arrival

The residence time can then be expressed for non-delay stations as:

$$\Phi_k = (1 + A_k) \cdot S_k$$

$$\mathrm{R}_k = v_k \cdot \Phi_k = (1 + A_k) \cdot v_k \cdot S_k = (1 + A_k) \cdot D_k$$

For delay stations instead we simply have:

$$\mathrm{R}_k = D_k$$

Open models

The way in which the jobs found at the arrival can be computed depends on whether the considered system is open or closed.

In open models, the average number of jobs found in a queue at the arrival of a new costumer is identical to the average queue length of the same station.

$$A_k(\lambda) = N_k(\lambda)$$

$$\mathrm{R}_k(\lambda) = (1 + A_k) \cdot D_k = \big(1 + N_k(\lambda)\big) \cdot D_k$$

Open models

Using Little's law, we can express the queue length as function of the residence time and the arrival rate.

$$\mathrm{R}_k(\lambda) = \left(1 + N_k(\lambda)\right) \cdot D_k$$

$$N_k(\lambda) = \mathrm{X}(\lambda) \cdot \mathrm{R}_k(\lambda) = \lambda \cdot \mathrm{R}_k(\lambda)$$

$$\mathrm{R}_k(\lambda) = \left(1 + \lambda \cdot \mathrm{R}_k(\lambda)\right) \cdot D_k$$

Open models

We can then rework the equation and apply the utilization law to find an expression for the residence time as function of the arrival rate (or of the utilization).

$$\mathrm{R}_k(\lambda) = D_k + \lambda \cdot \mathrm{R}_k(\lambda) \cdot D_k$$

$$(1 - \lambda \cdot D_k) \cdot \mathrm{R}_k(\lambda) = D_k$$

$$\mathrm{R}_k(\lambda) = \frac{D_k}{1 - \lambda \cdot D_k} = \frac{D_k}{1 - \mathrm{U}_k(\lambda)}$$

Open models

Applying Little's law, we can compute the average number of jobs in a station.

$$N_k(\lambda) = \lambda \cdot \mathrm{R}_k(\lambda) = \frac{\lambda \cdot D_k}{1 - \mathrm{U}_k(\lambda)} = \frac{\mathrm{U}_k(\lambda)}{1 - \mathrm{U}_k(\lambda)}$$

Please note that such relation can be computed for each node k, using the corresponding demand D_k

Open models

As expected, the previous results agree with the ones obtained analyzing the M/M/1 queue, with traffic intensity $\rho = \lambda \cdot D$.

Open Model	M/M/1
$U = \lambda \cdot D$	$U = \rho$
$N = \frac{U}{1-U}$	$N = \frac{\rho}{1-\rho}$
$R = \frac{D}{1-U}$	$R = \frac{D}{1-\rho}$

Open models

To summarize:

(from the book of Lazowska)

$$\textit{processing capacity}: \quad \lambda_{sat} = 1 / D_{max}$$

$$\textit{throughput}: \quad X(\lambda) = \lambda$$

$$\textit{utilization}: \quad U_k(\lambda) = \lambda D_k$$

$$\textit{residence time}: \quad R_k(\lambda) = \begin{cases} D_k & \text{(delay centers)} \\ \dfrac{D_k}{1 - U_k(\lambda)} & \text{(queueing centers)} \end{cases}$$

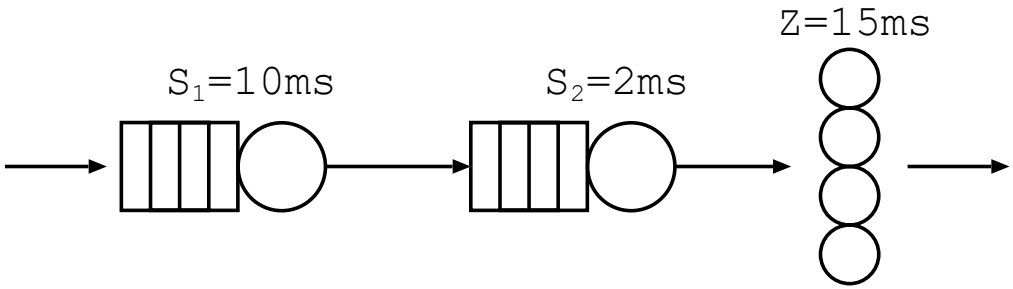
$$\textit{queue length}: \quad Q_k(\lambda) = \lambda R_k(\lambda) = \begin{cases} U_k(\lambda) & \text{(delay centers)} \\ \dfrac{U_k(\lambda)}{1 - U_k(\lambda)} & \text{(queueing centers)} \end{cases}$$

$$\textit{system response time}: \quad R(\lambda) = \sum_{k=1}^{K} R_k(\lambda)$$

$$\textit{average number in system}: \quad Q(\lambda) = \lambda R(\lambda) = \sum_{k=1}^{K} Q_k(\lambda)$$

Analysis of Motivating Example

We can model the considered system with the following model.



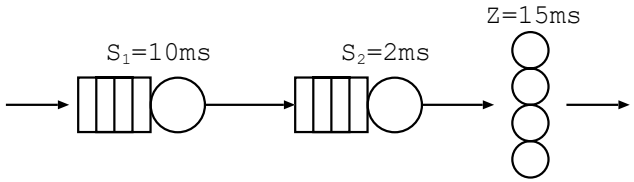
The system response time, as function of the arrival rate λ, can be computed as:

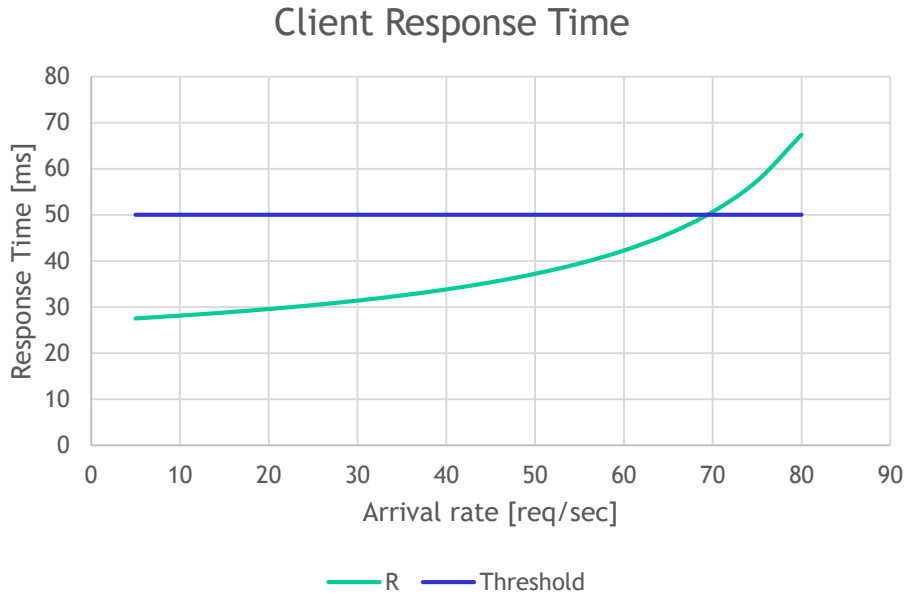
$$R(\lambda) = \frac{0.01}{1 - 0.01\lambda} + \frac{0.002}{1 - 0.002\lambda} + 0.015$$

Analysis of Motivating Example

We can then study the response time to be less than the given threshold:

$$R(\lambda) = \frac{0.01}{1-0.01\lambda} + \frac{0.002}{1-0.002\lambda} + 0.015 < 0.05 \text{ ms.}$$





The maximum arrival rate to respect the given threshold is around λ=70 job/s