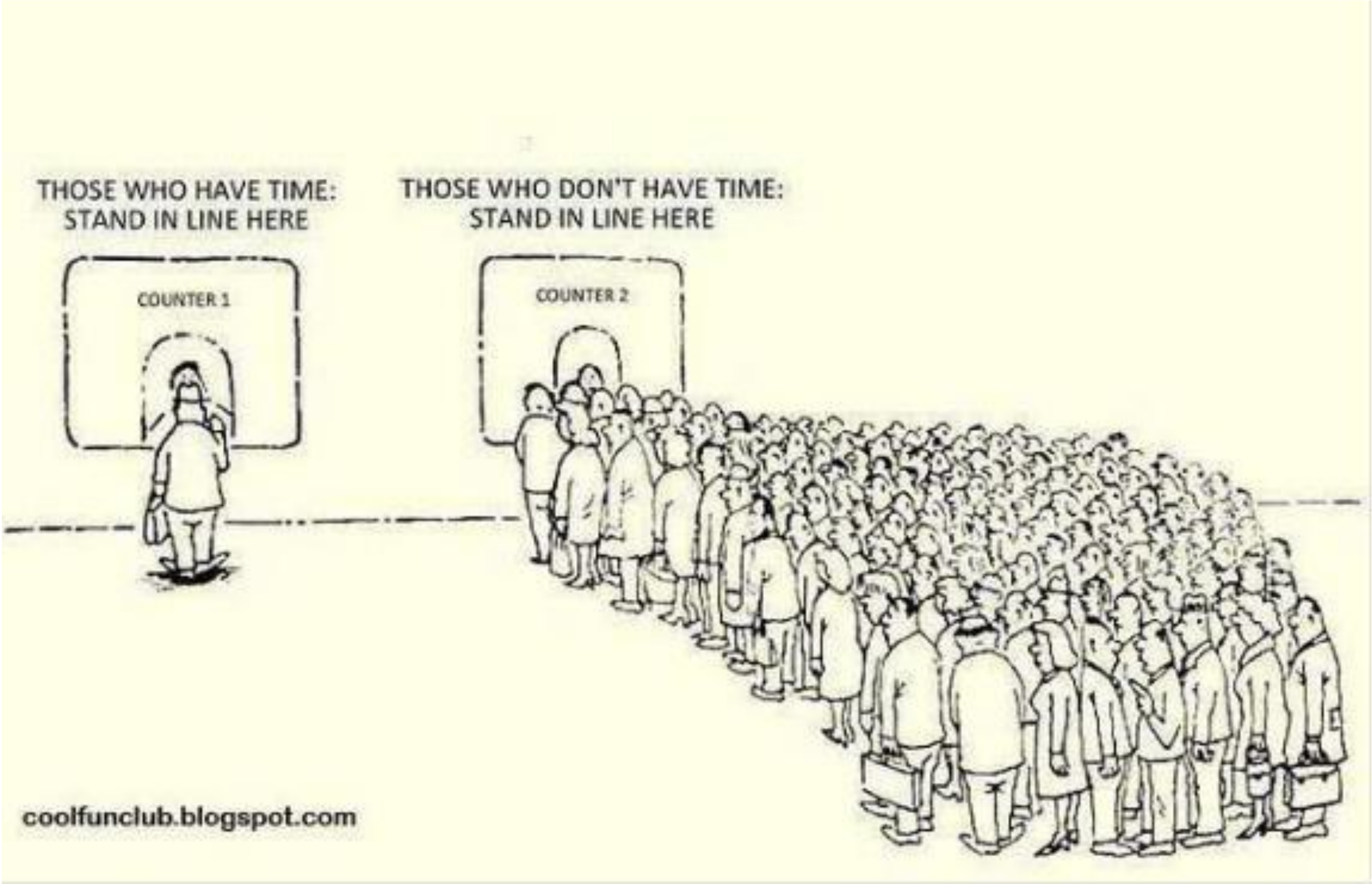
POLITECNICO DI MILANO

Performance Evaluation
and Applications

Modeling with Queueing Networks and JMT

Queuing networks

Many modeling languages exists: we will start focusing on "queuing networks"



Queuing Networks

A *Queuing Network* is a description of a system, composed by several *Service Centers* that executes *Jobs*.

Queuing Networks: Service Centers

In a Service Center, at a given time instant, there could be *several Jobs competing* for that service at the same time.

The number of Jobs at a give time in a Service Center, is called its *Current Population*.

Current Population = 7

Queuing Networks: Service Centers

Some of the *Jobs* in a service center, will be actually being *Served*.

Some other might be *Waiting to be served*.

Moreover, the *Current Population* might influence the *Speed* at which *Jobs* are being *Served*.

All these features are essential in describing how a *Service Center* of *Queueing Network* works… But we will return on them in a future lesson.

Queuing Networks: terminology and conventions

Please note, that the terminology shown here is the convention we decide to follow in this course.

Unfortunately, in many other contexts, even focusing on Queueing Networks, many of the terms used here have a different meaning.

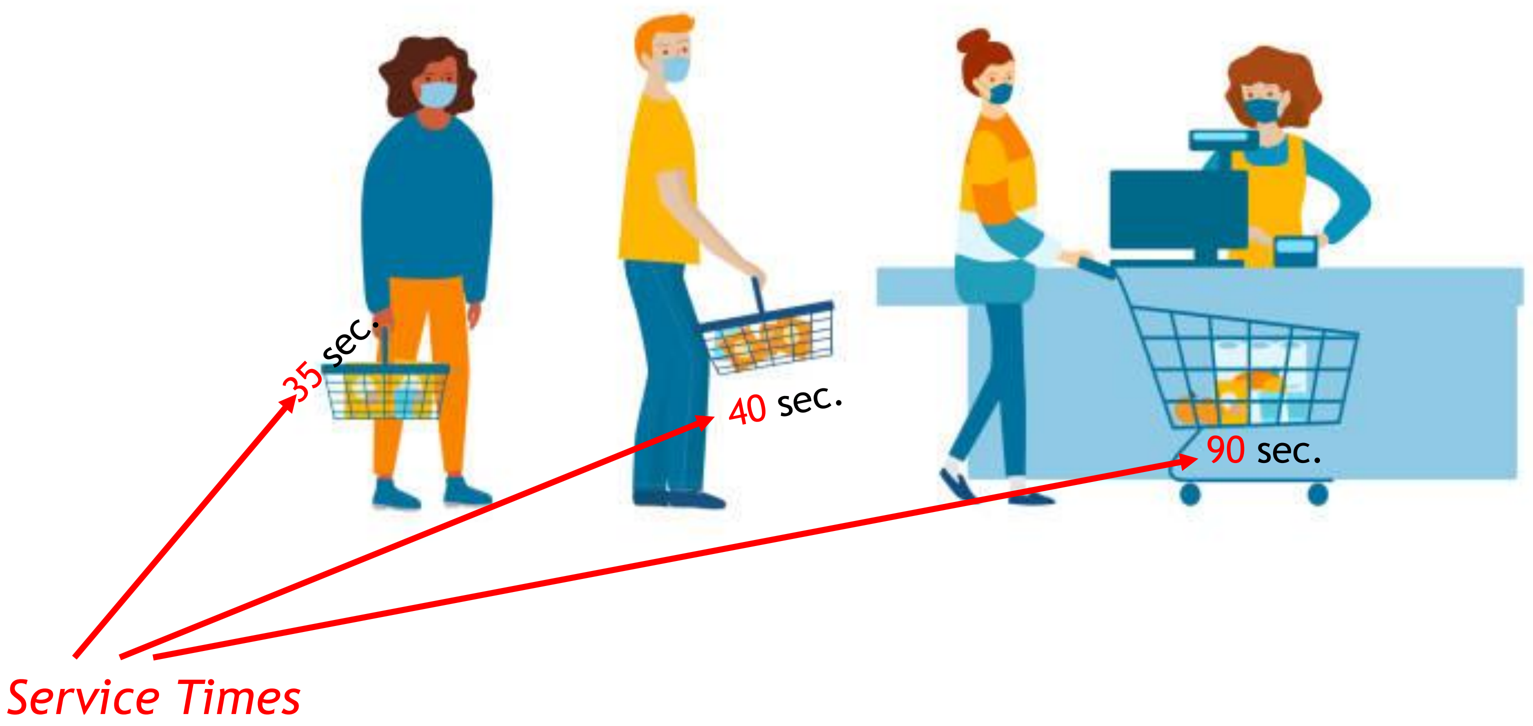
Queuing Networks: Queue and Delay

Initially we will consider two types of *Service Centers*:

- *Queueing Stations*
- *Delay Stations*

Queuing Networks: Queueing Stations

Jobs arrive at a *Queueing Station* to receive a *Service* characterized by a given *Duration* that corresponds to its *Service Time*.



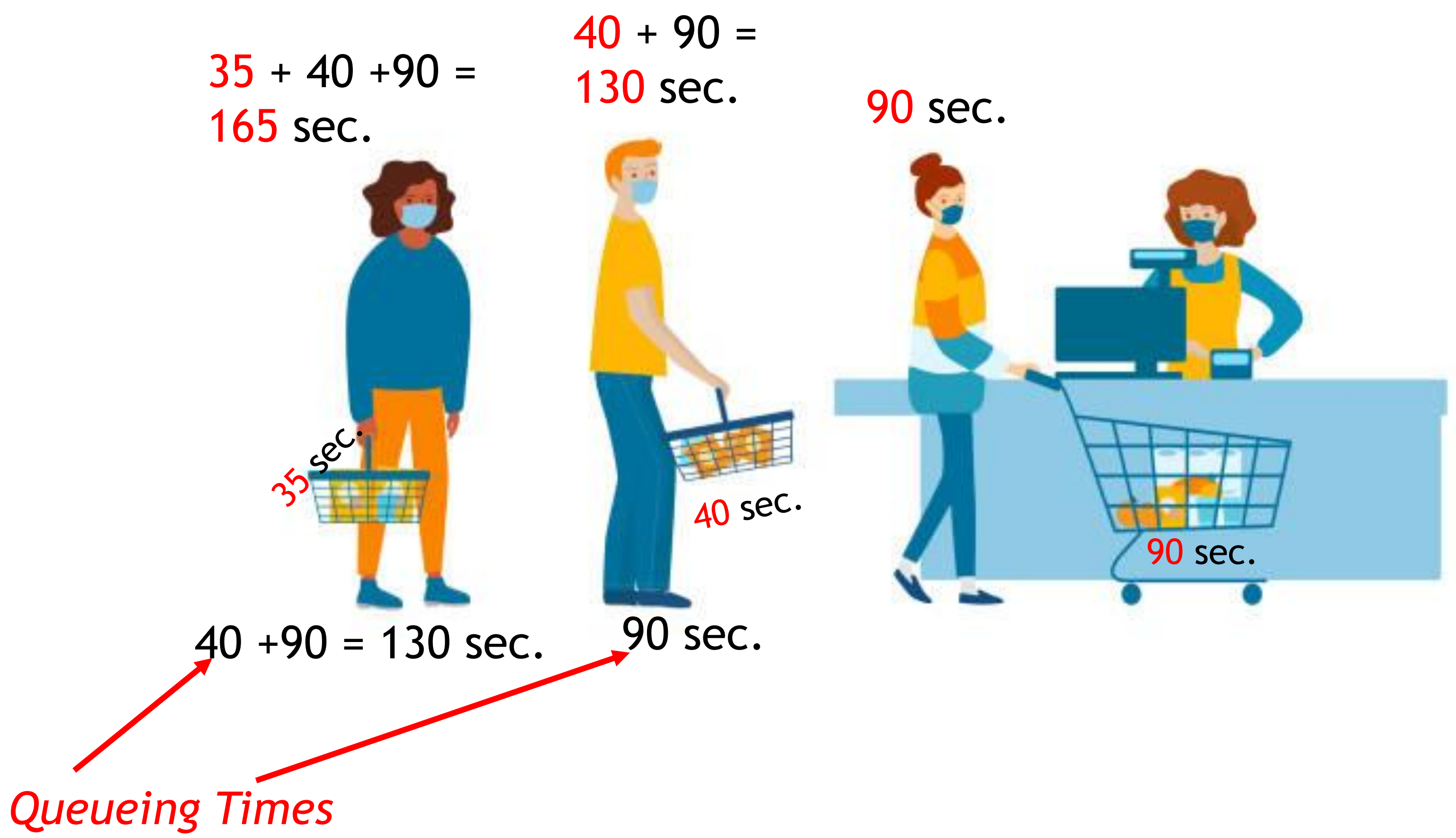
Queuing Networks: Queueing Stations

If the *Job* will be the only one in service for its entire *Duration*, then the time spent at its station, will correspond to its *Service Time*.

90 sec.

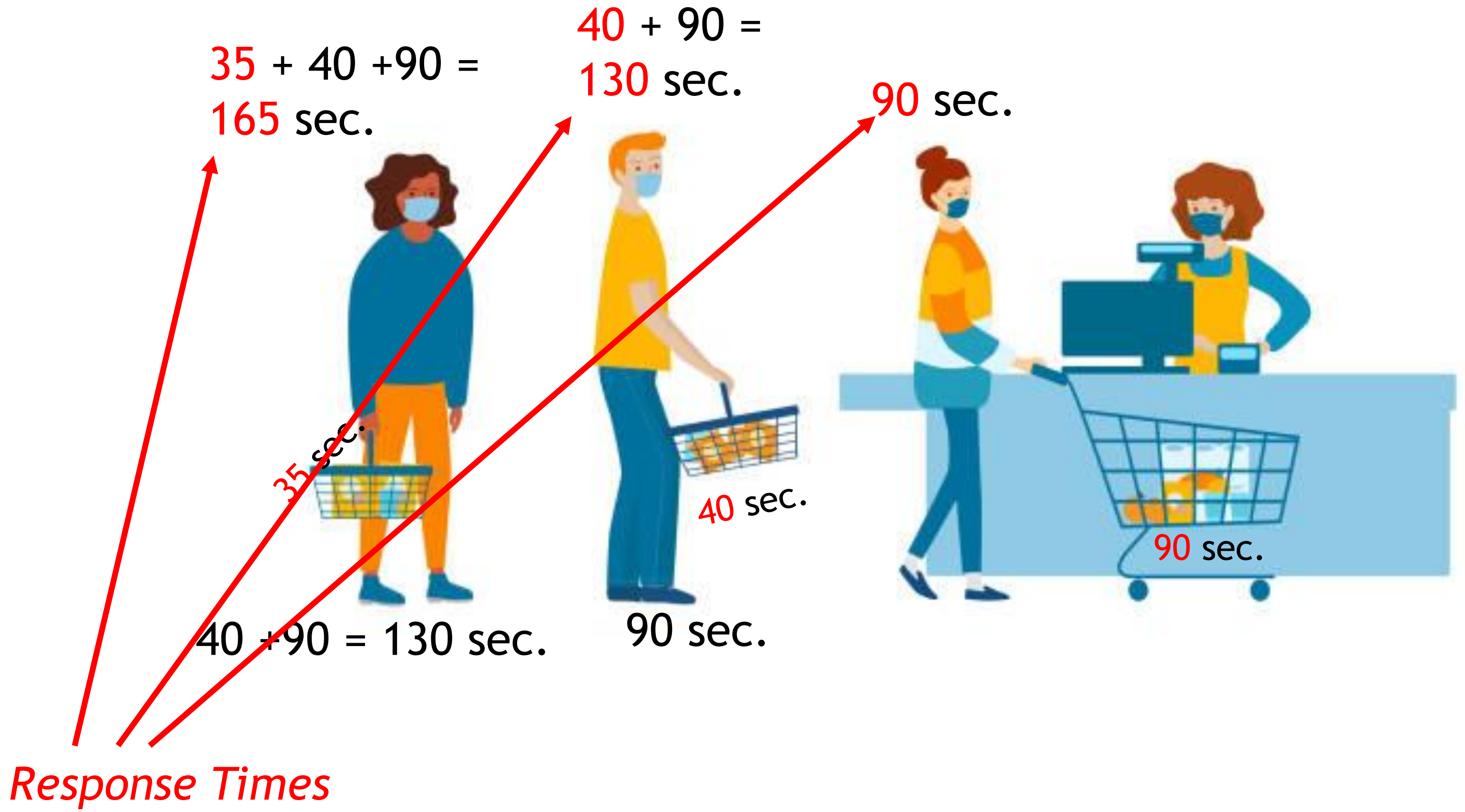
Queuing Networks: Queueing Stations

If other *Jobs* will be at same time in the considered station, then the *Service* might be delayed or interrupted. This causes some waiting time, called *Queueing Time*.



Queuing Networks: Queueing Stations

The total time spent in a *Queueing Station*, which is the sum of the *Service Time* and the *Queueing Time*, corresponds to the *Response Time*.



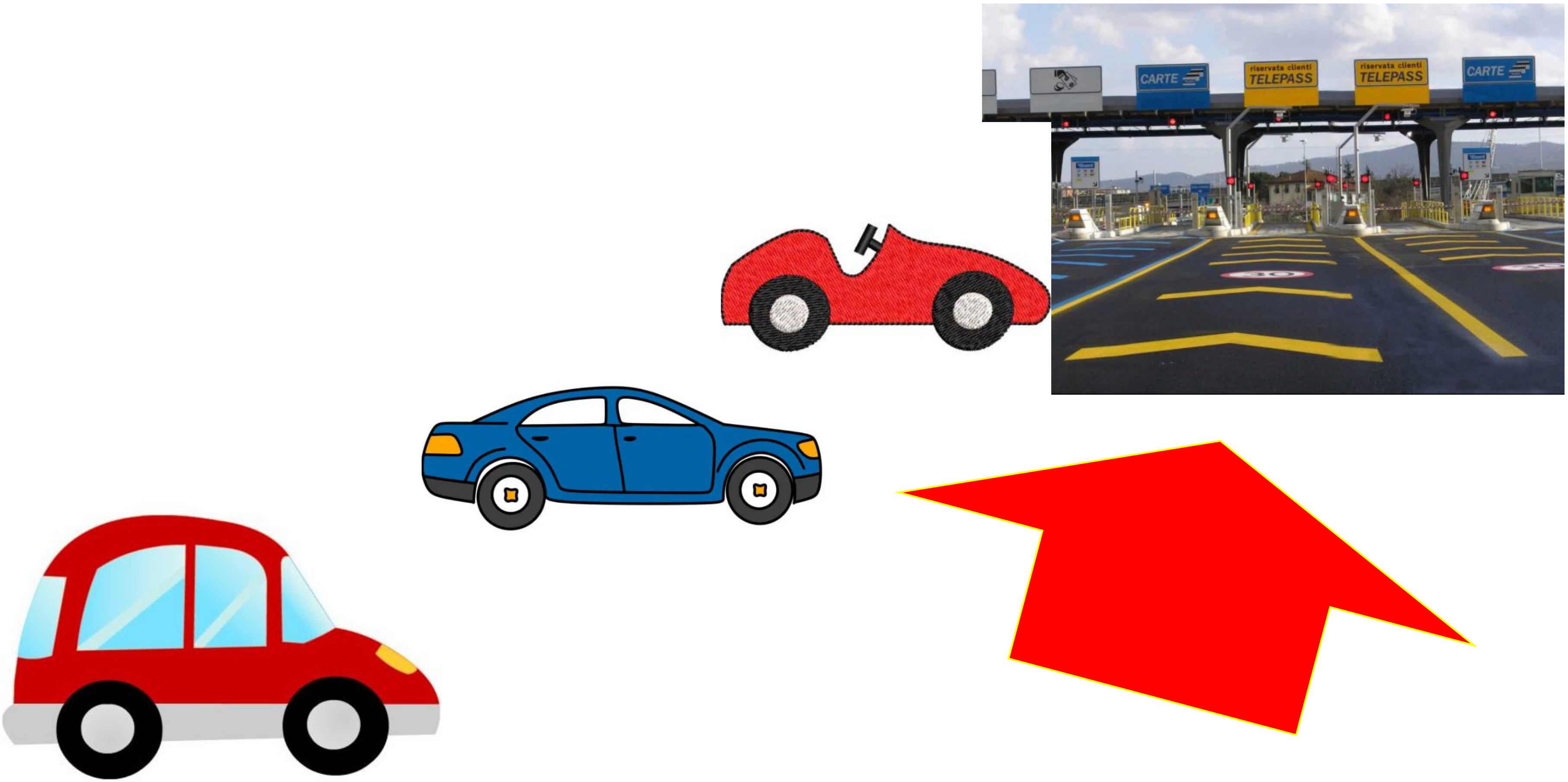
Queuing Networks: Delay Stations

Delay Stations represent actions that are performed by each *Job Individually*, and not influenced by the others.

No Queue can happen at a *Delay Station*, and the *Response Time* always correspond to its *Service Time*.

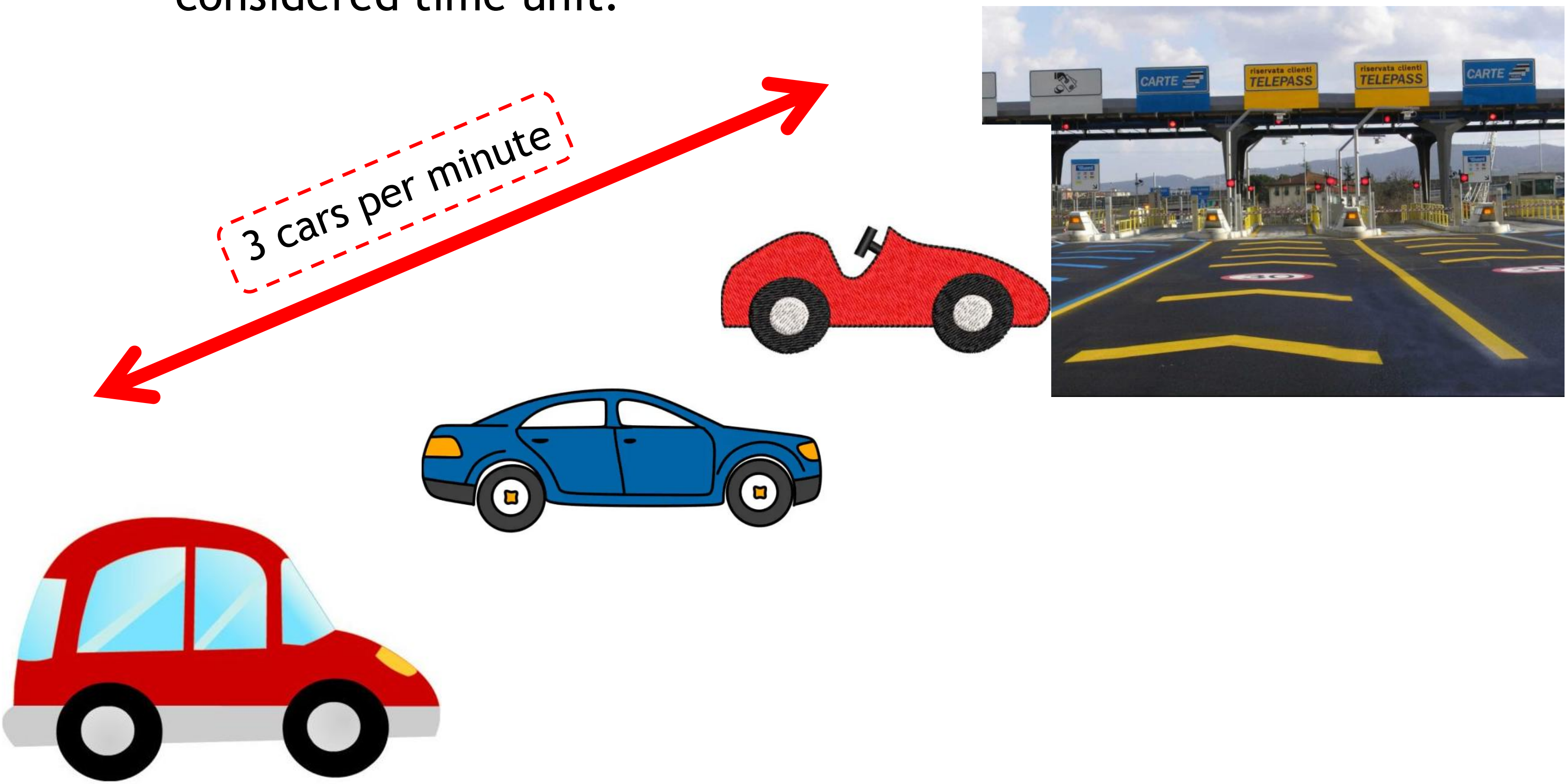
Queuing Networks: Arrivals

Arrivals model the entrance of new jobs in the *System*.



Queuing Networks: Arrivals

Arrivals can be described in several ways, as we will consider in the following lessons. The simplest definition uses the *Arrival Rates*, which counts the number of new *Jobs* entering the system in a considered time unit.



Queuing Networks: Stability

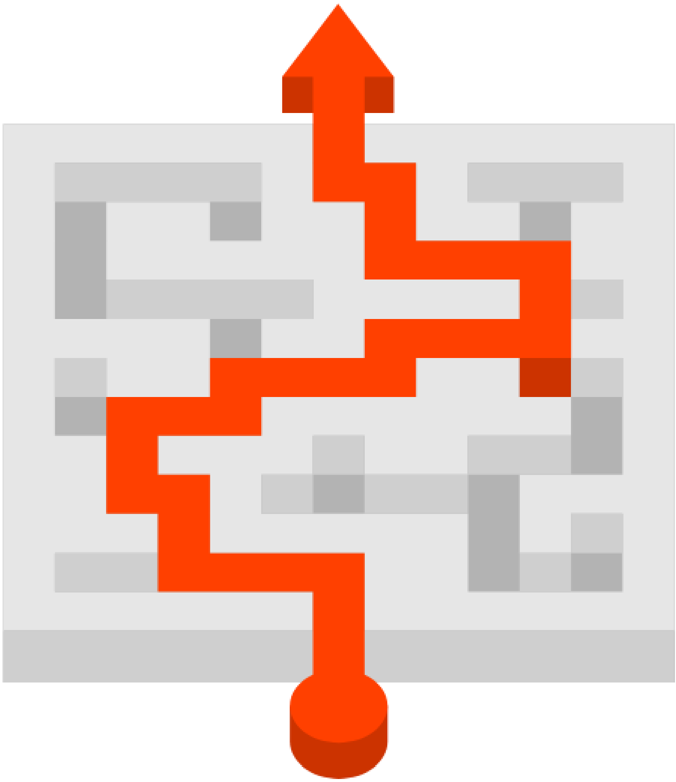
If the *Arrival Rate* is too high compared to the *Service Time* required by each Job in a *Queueing Station*, the system becomes *Unstable*.

Queuing Networks: Stability

If the system is *Unstable*, then the *Queueing Time* increases and it tends to the *Infinity*. We will return on *Stability* in later in the course.

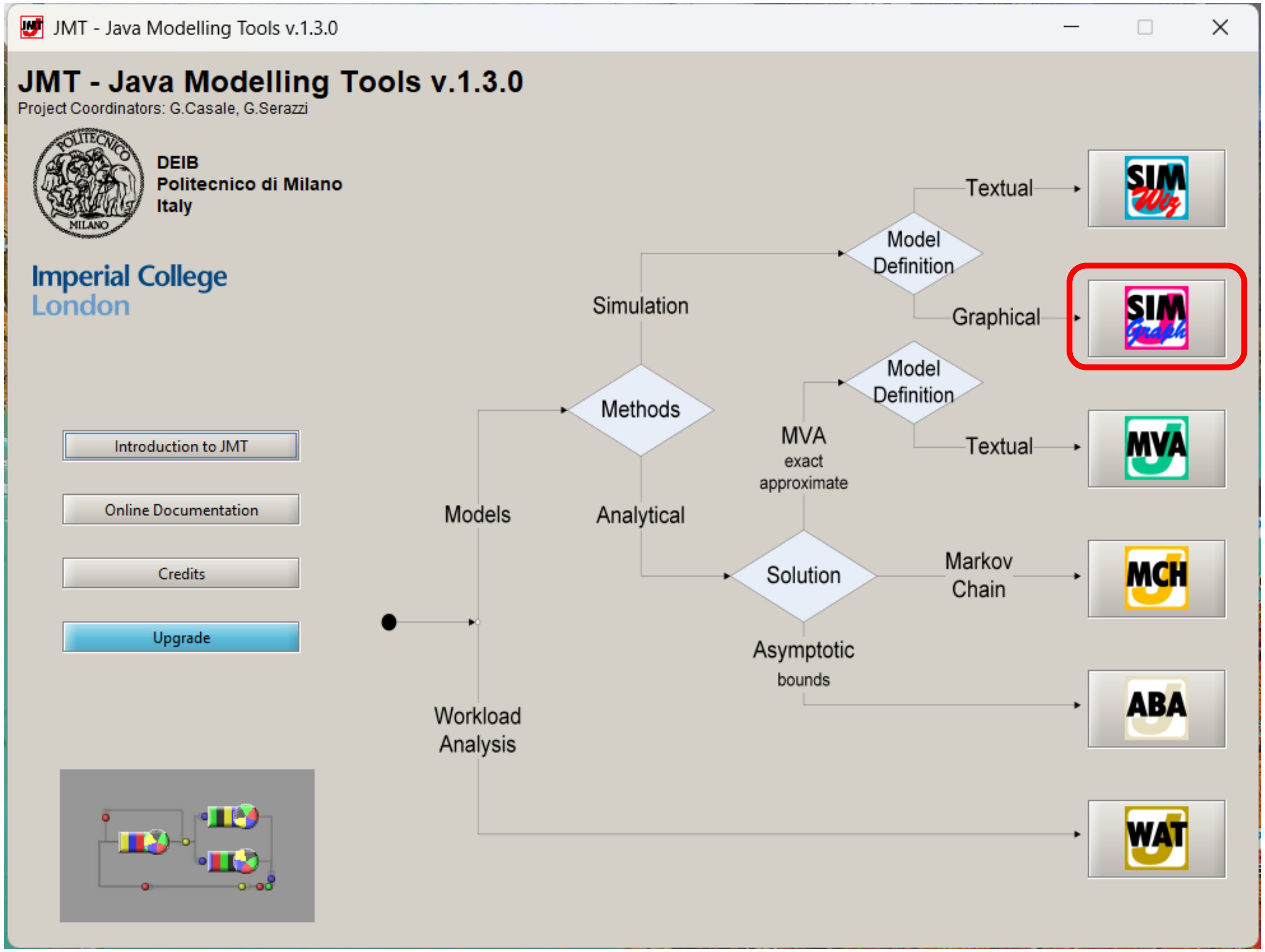
Queuing Networks: Departures

When a *Job* completes its service, it leaves the *System*. This event is called *Departure*.



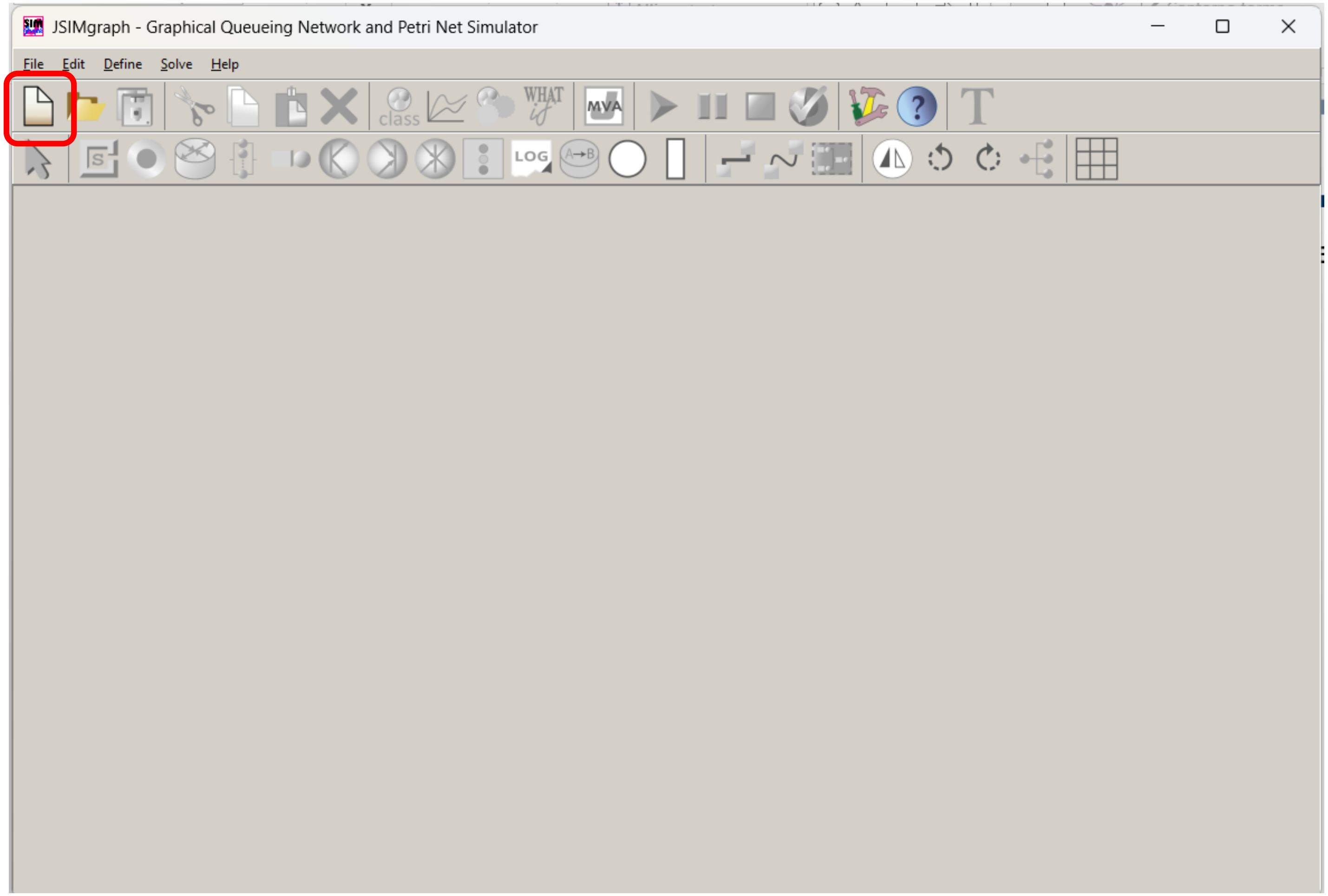
Queuing Networks: JMT

JMT allows to consider Queueing Networks in several components.
We will first focus on the *JSimGraph Tool*.



Queuing Networks: JMT

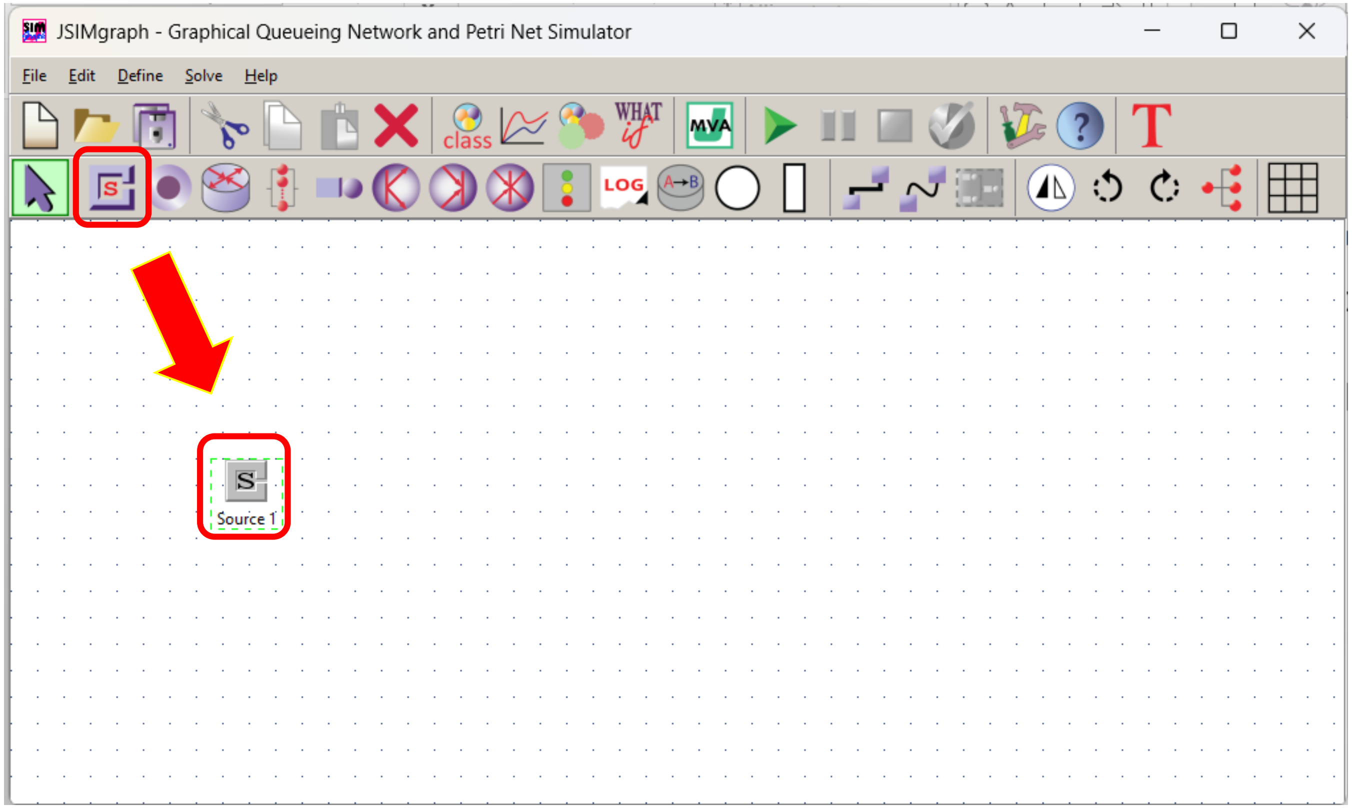
Once opened JSimGraph, a New Model should be created pressing the corresponding button.



JMT: Arrivals

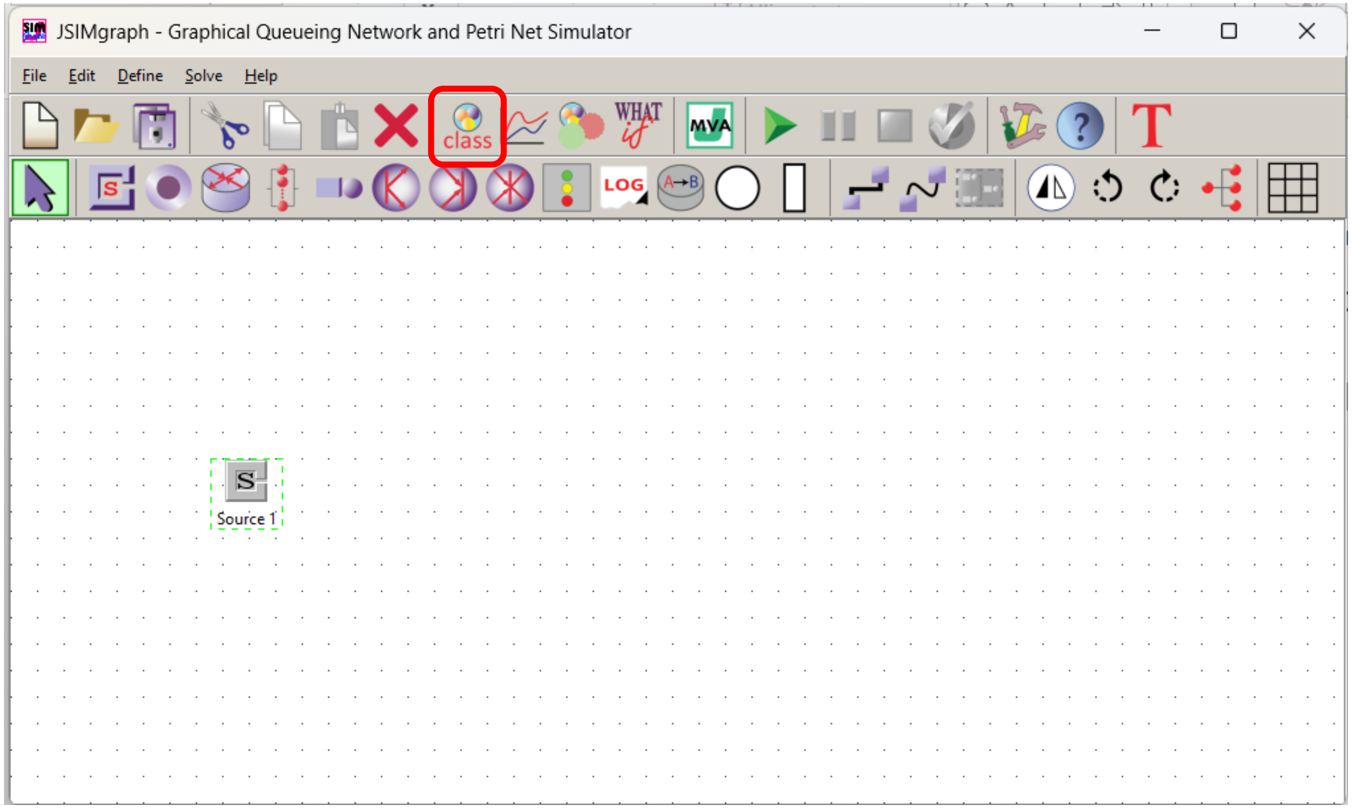
Arrivals in JMT are defined by placing a *Source* node in the central panel, using the corresponding icon.

After pressing the Source button, the element is placed by clicking the mouse in the desired position.



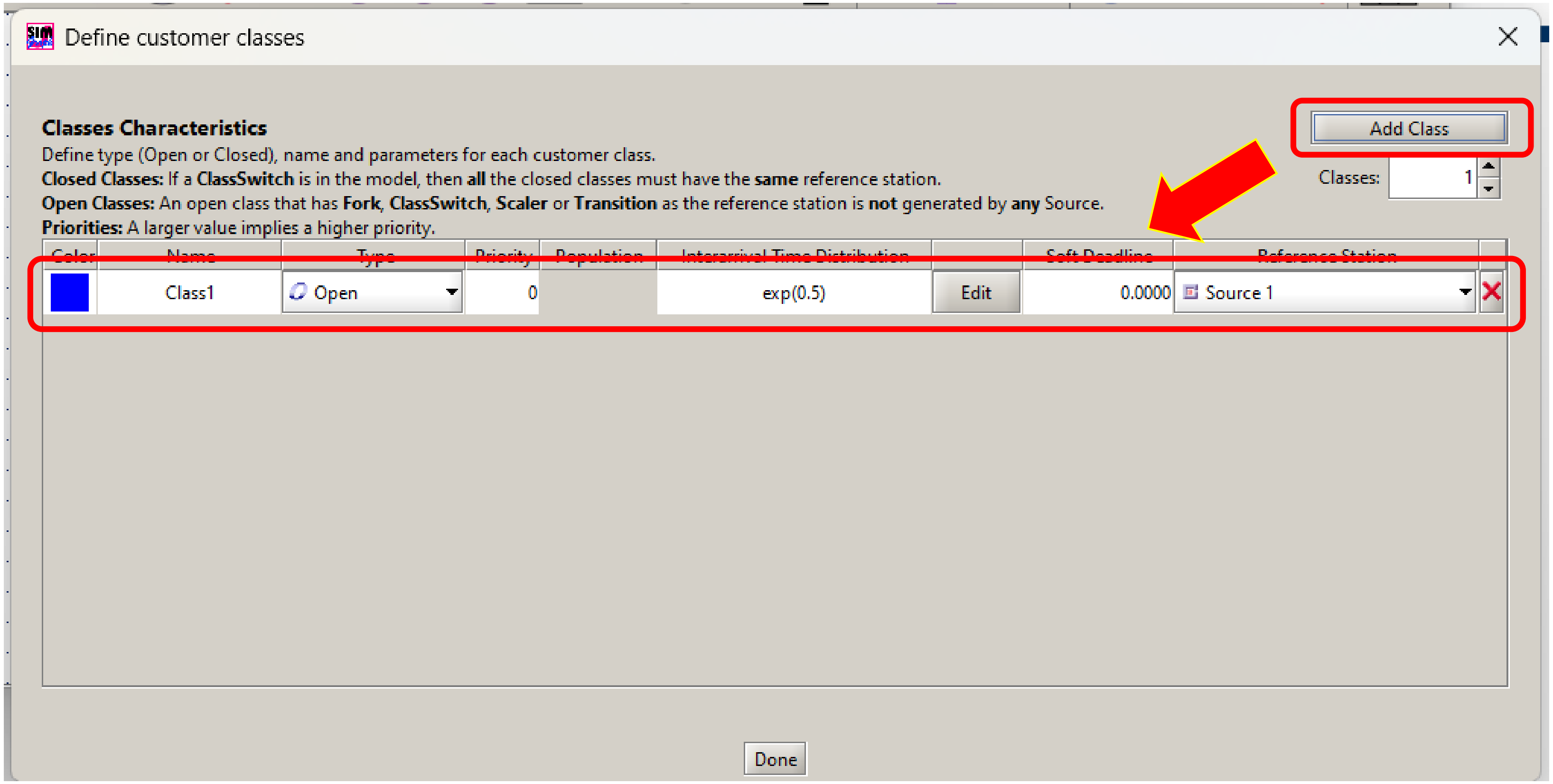
JMT: Arrivals

To specify the *Arrival Rate*, a new *Job Class* must be created first. Start pressing the *Class Button* to open the class *Panel*.



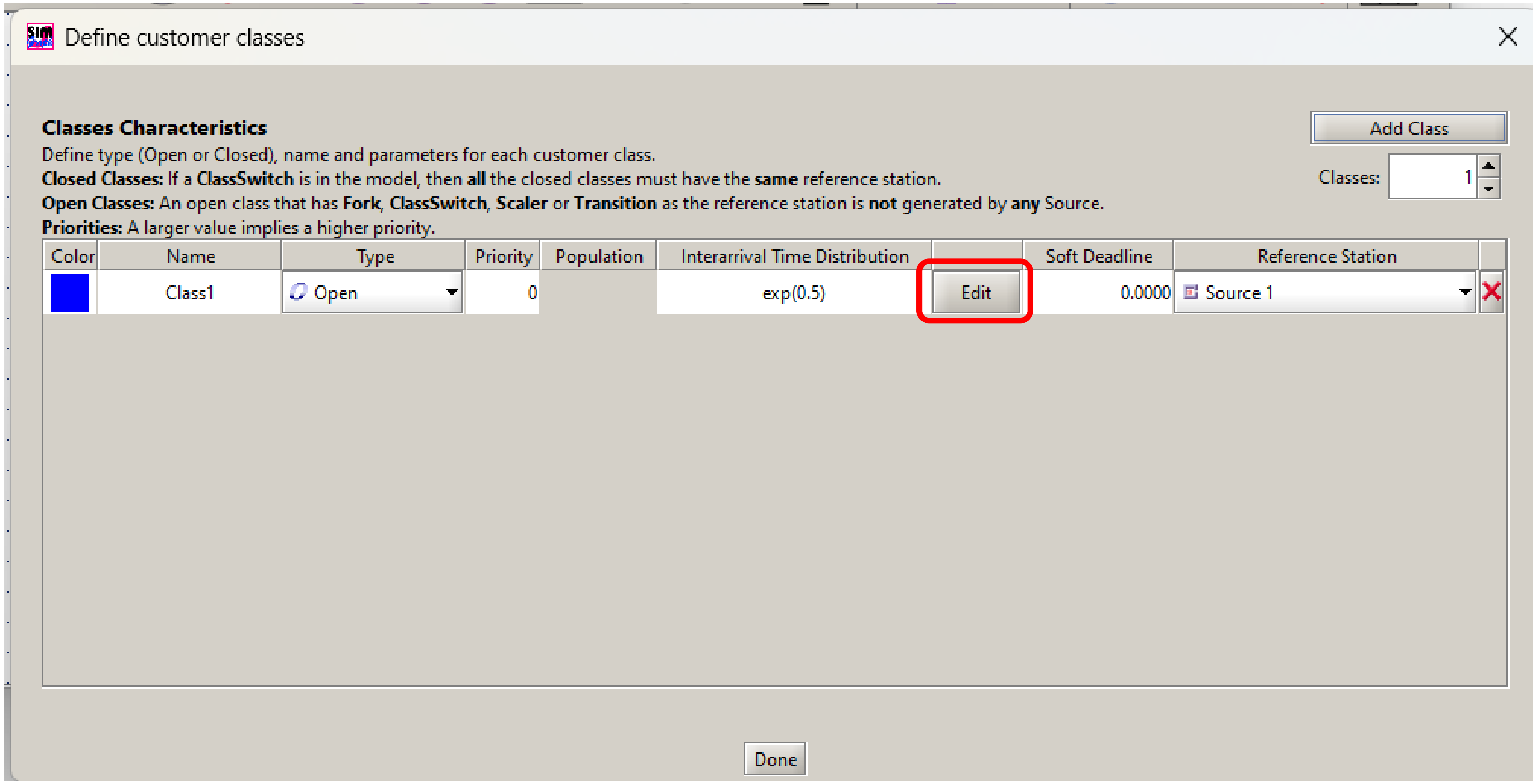
JMT: Arrivals

A *New Class* is created by pressing the corresponding button.



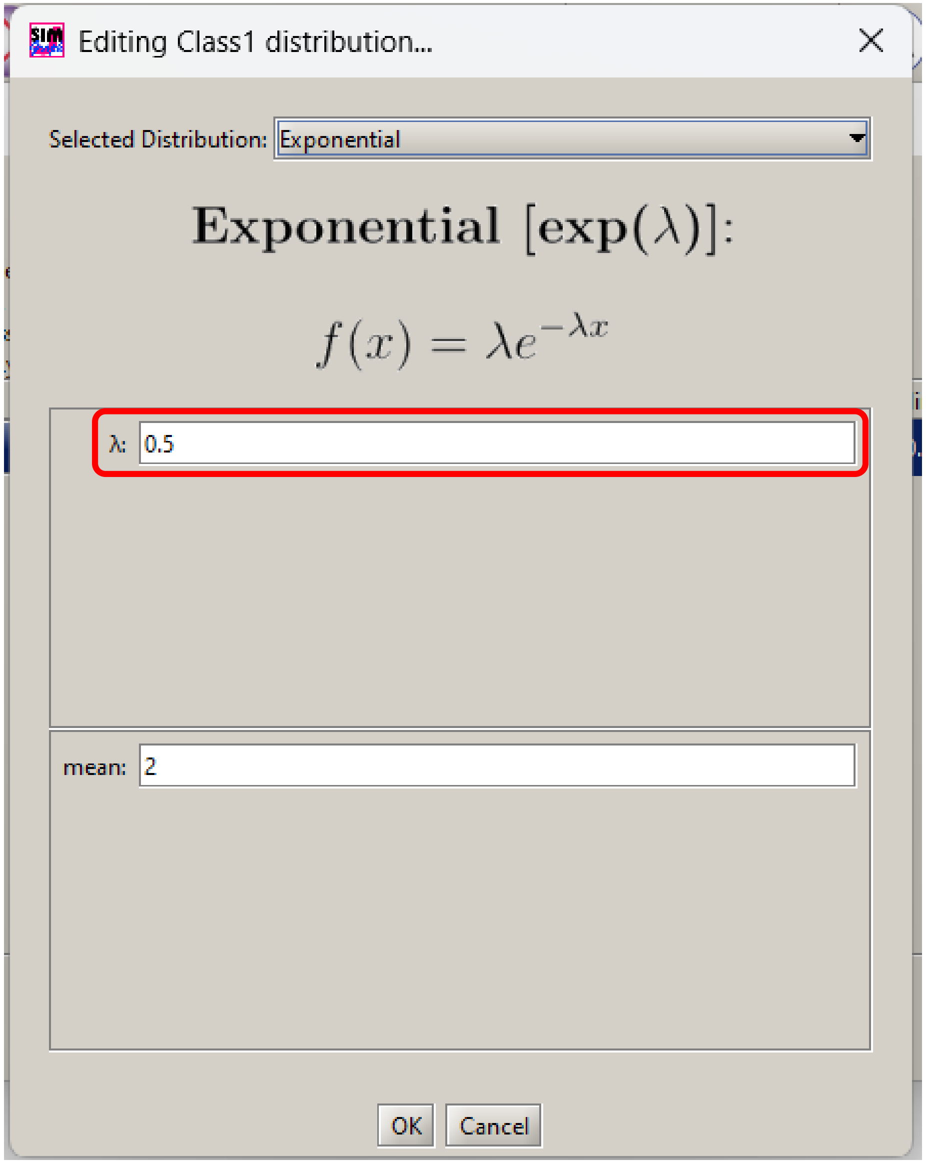
JMT: Arrival Rate

The Arrival Rate is specified pressing the “Edit” Button in the line corresponding to the newly created class.



JMT: Arrival Rate

- The actual rate is entered in the λ field.
- The panel allows to define very complex arrival patterns. We will return on this in the following lessons.



Queueing Networks parameters and units

JMT uses by convention the *[seconds]* as the main unit to measure time.

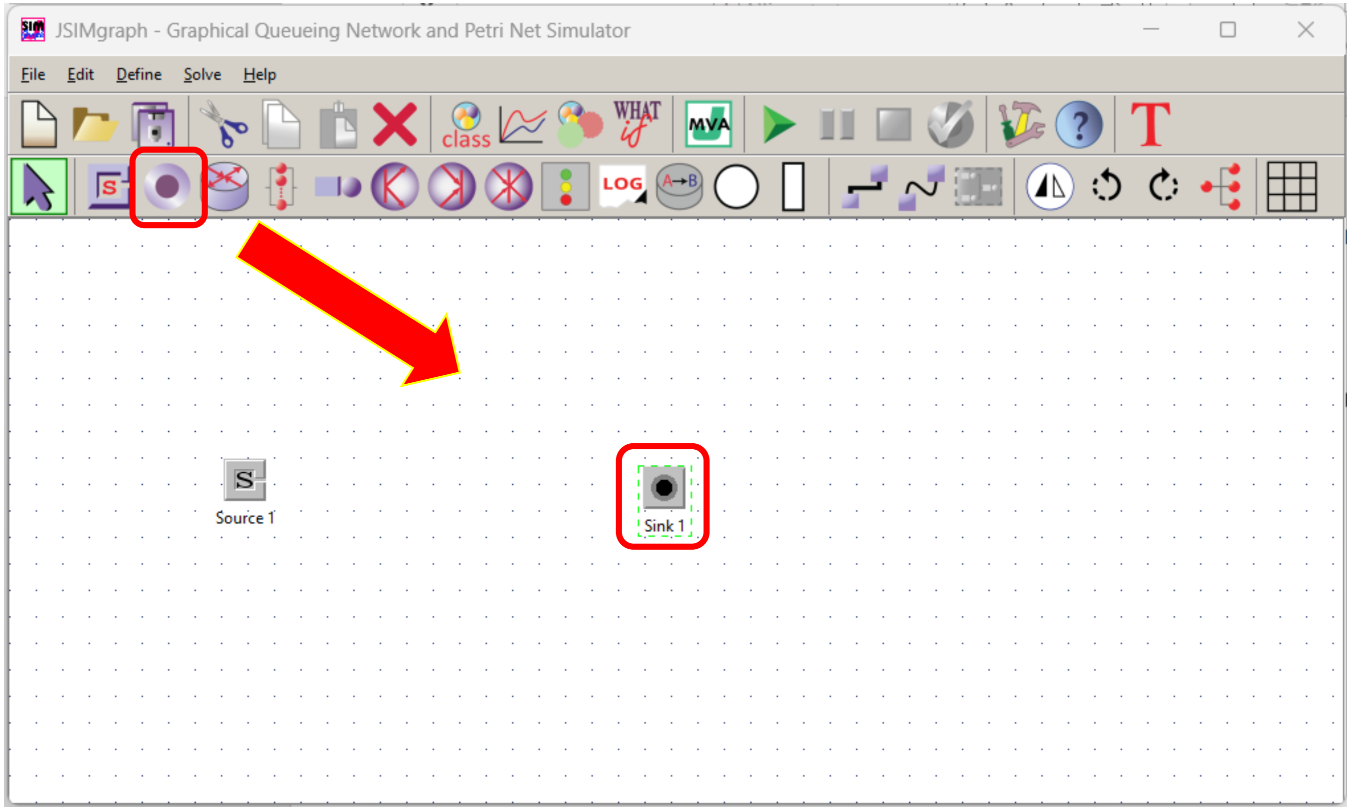
This however is just a convention: the user might simply ignore this definition and consider temporal specification given in *[minutes]*, *[hours]*, *[milliseconds]* and so on.

The only important requirement, is that all the temporal specification, in all the parameters considered by the tool, match the same units.

JMT: Departures

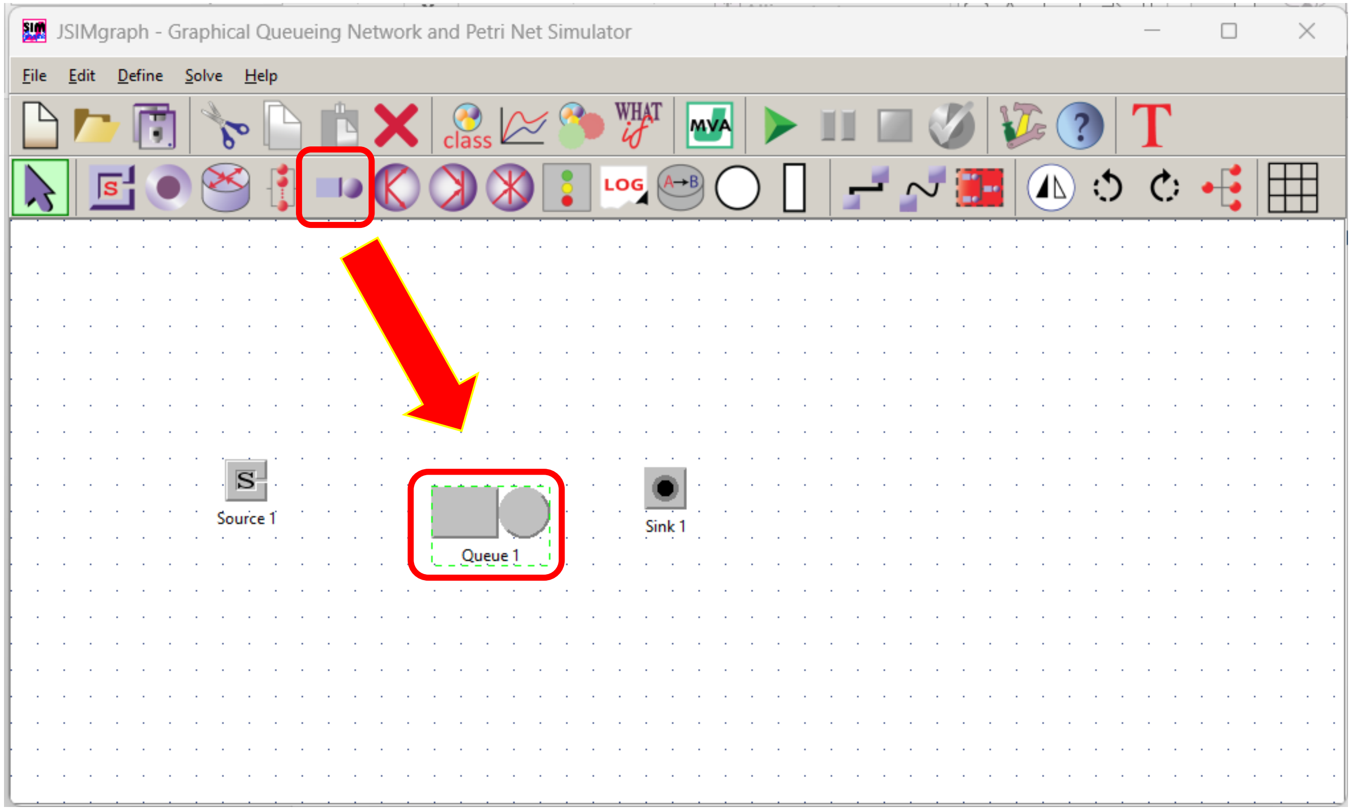
Departures in JMT are defined by placing a *Sink* node in the central panel, using the corresponding icon.

Again, the element is placed by clicking the mouse in the desired position.



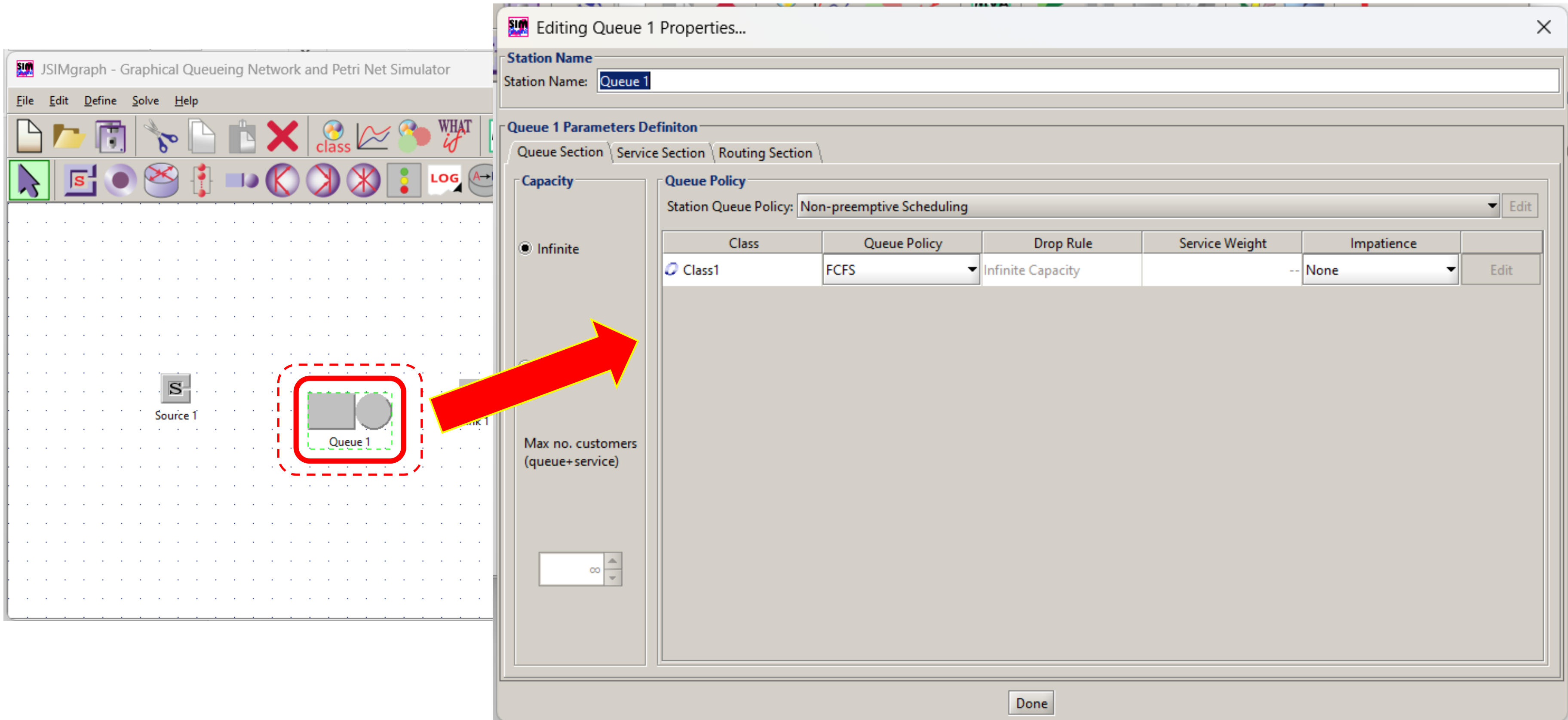
JMT: Queueing Stations

Queueing Stations are defined by placing *Queue* nodes.



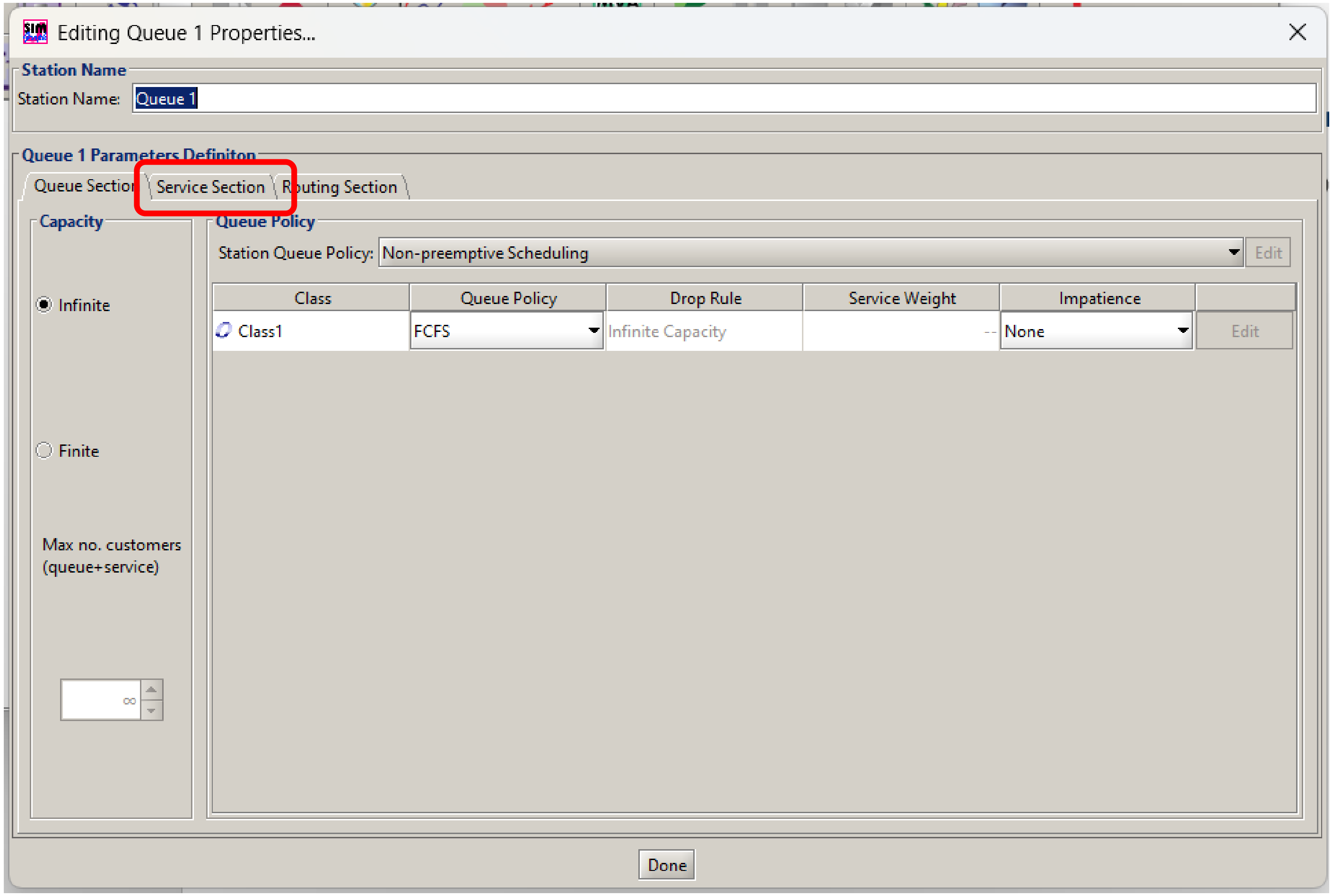
JMT: Queueing Stations

Service Times are defined by first opening the parameter panel, by double-clicking on the queue icon in the central panel.



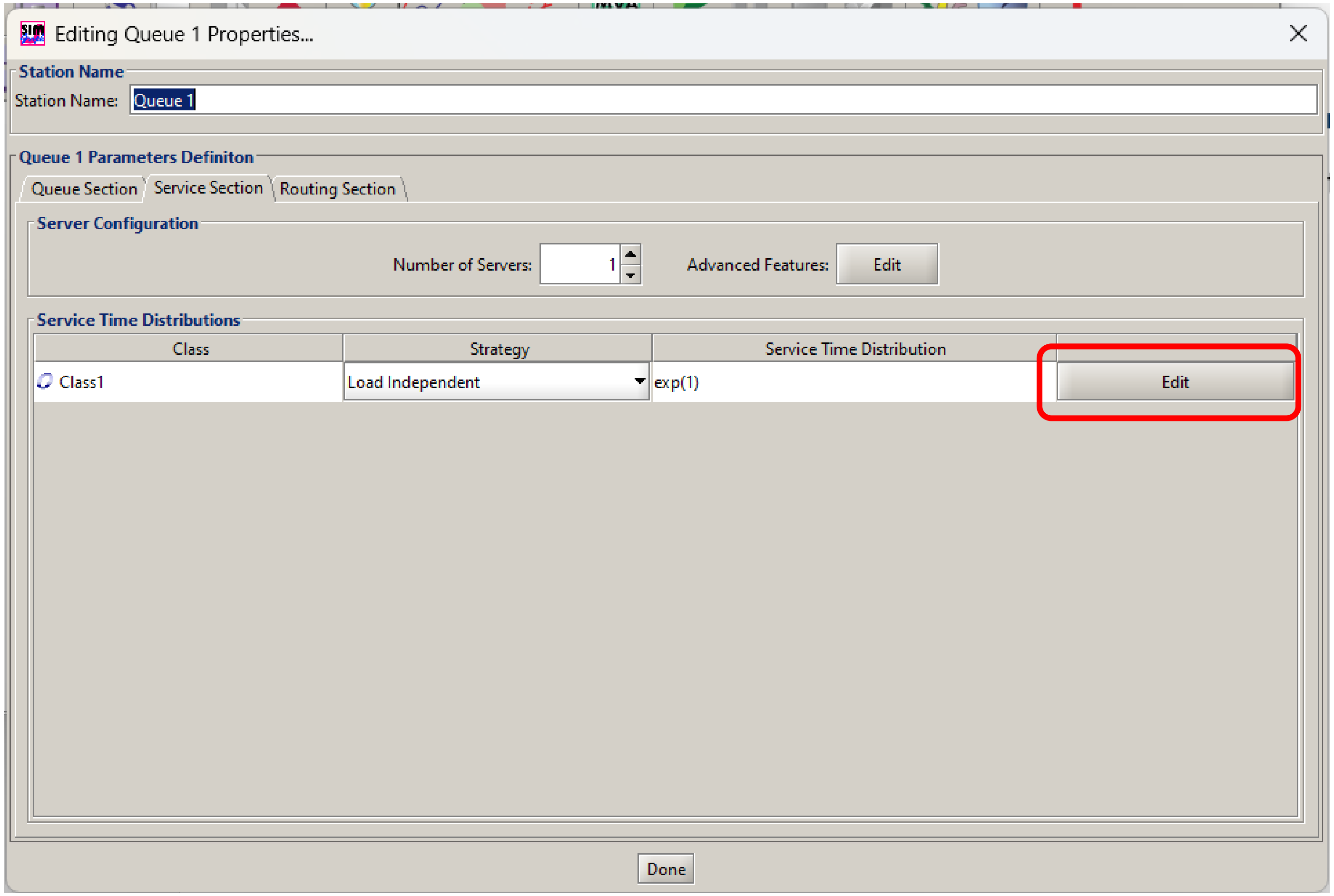
JMT: Queueing Stations

Then, the *Service Sectiom Tab* should be selected.



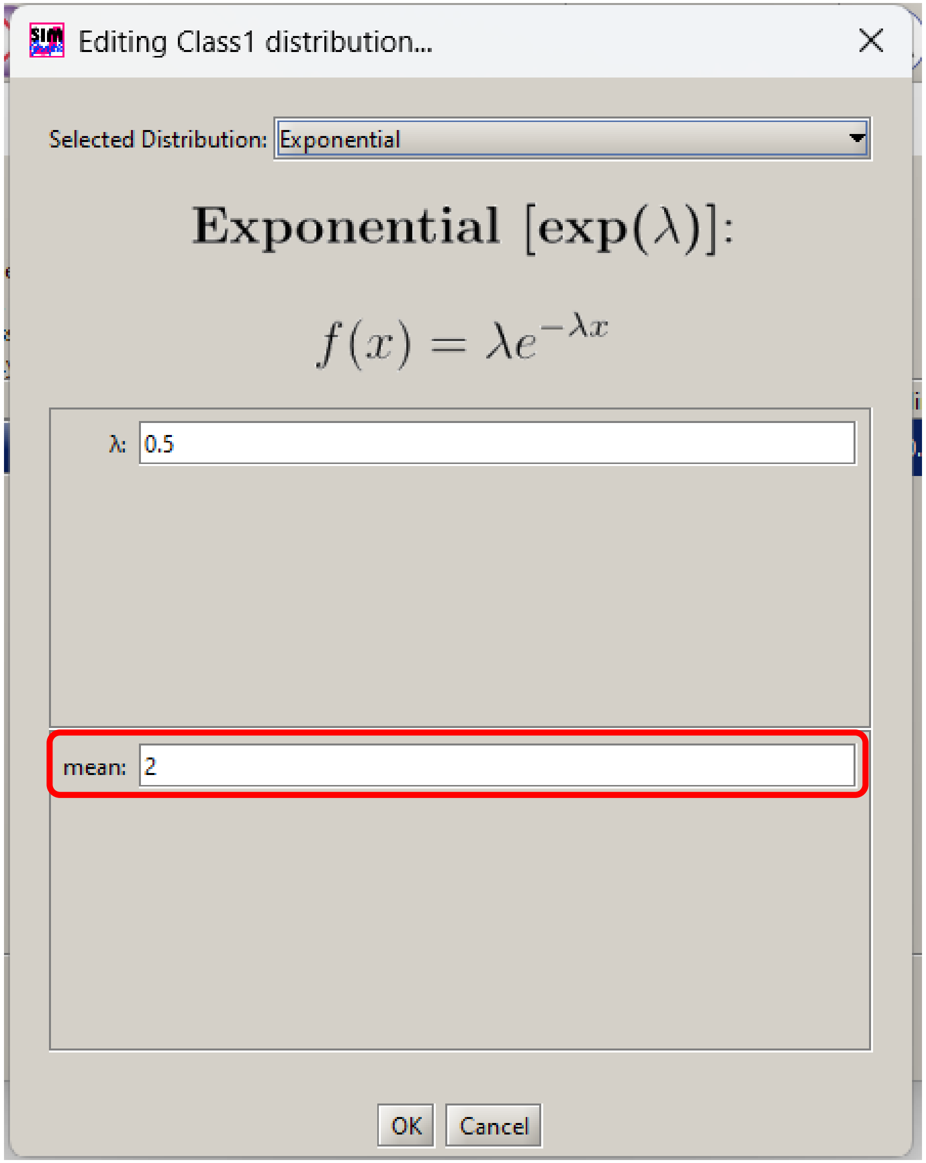
JMT: Queueing Stations

Service Time is then finally reached by clicking the *Edit* button, in the table in the center of the window.



JMT: Queueing Stations

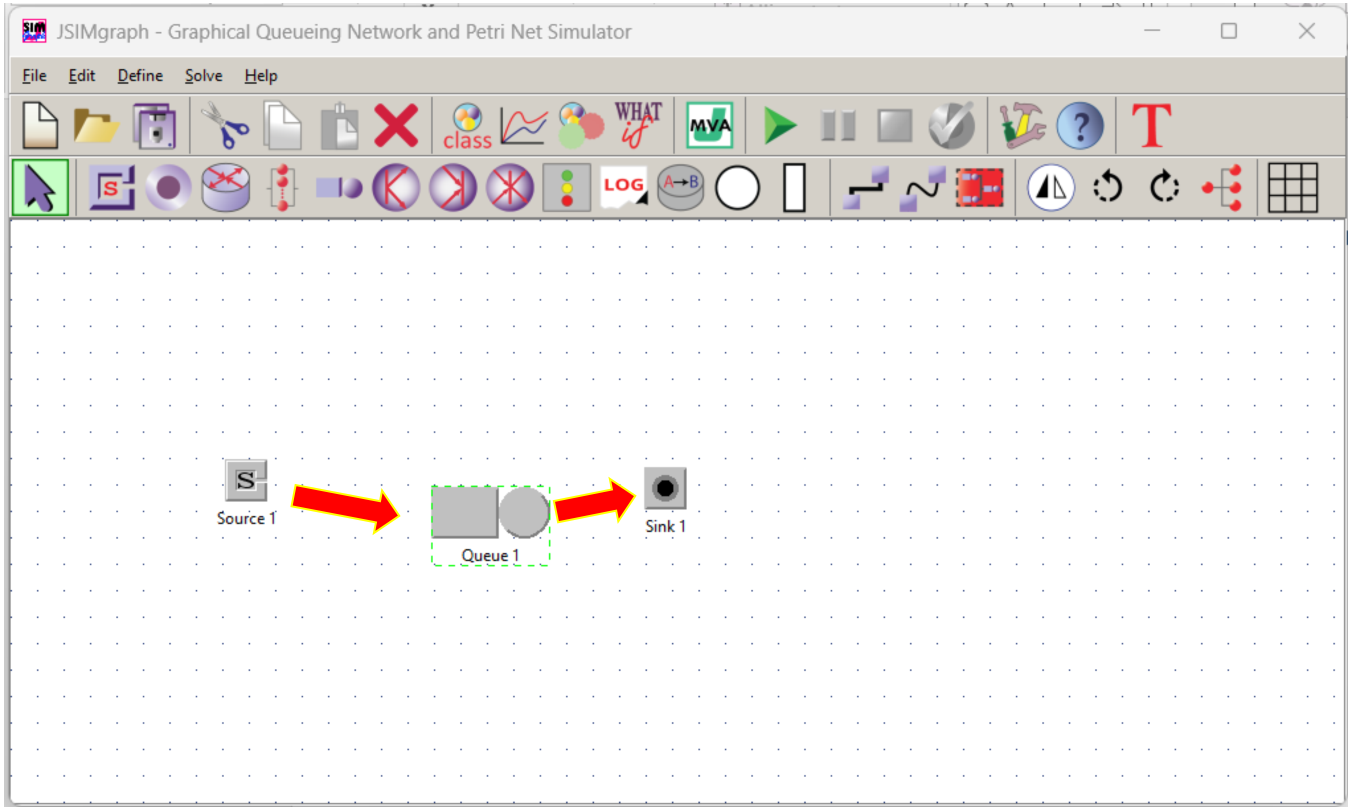
- The *Service Duration* can be specified in the *mean* section of the proposed window.
- Again, the tool allows to define very complex service processes: we will return on them in the following.



JMT: Connections

Once a Source, a Queueing Station and a Sink have been included in the model, they must be connect using two arcs:

- One directed from the Source to the Queue
- The other from the Queue to the Sink

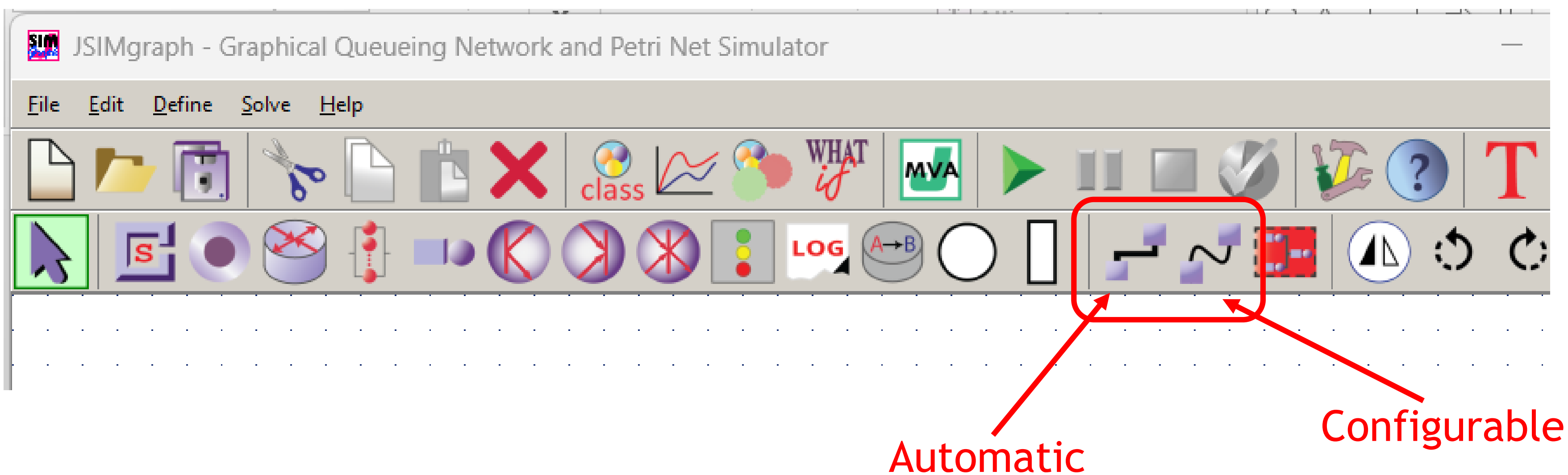


JMT: Connections

Two types of arcs are available:

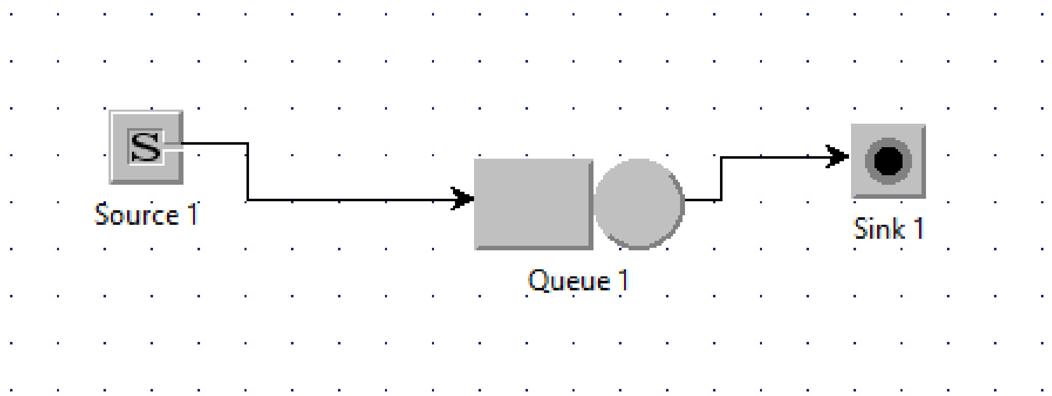
- Automatic Arcs
- Configurable Arcs

They are equivalent from a semantic point of view, and their difference is purely graphical.



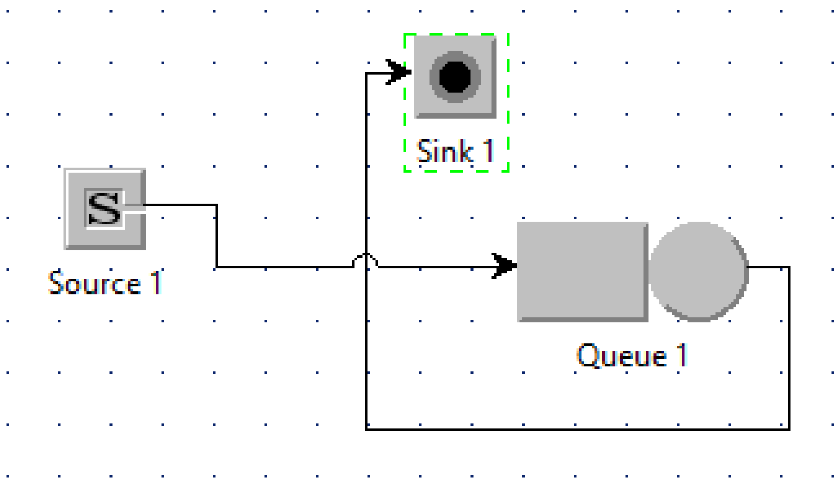
JMT: Connections

Automatic arcs, are composed of lines always parallel to the borders of the screen, and are entered by clicking first on the starting node, then releasing the mouse press on the destination.



JMT: Connections

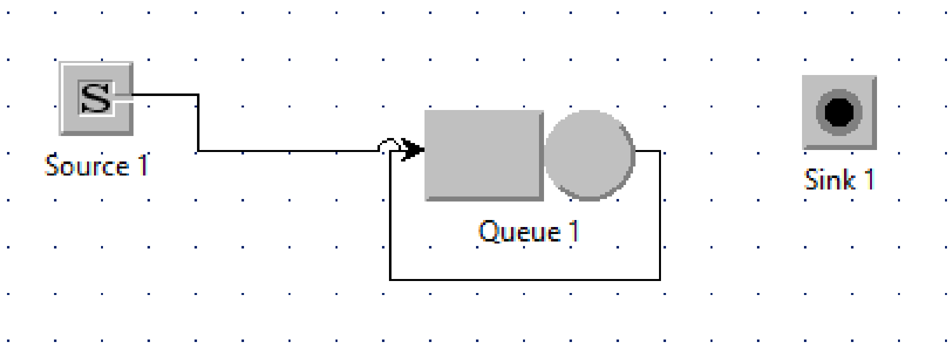
The tool automatically choose what it considers the best graphical path for connecting the two nodes.



JMT: Connections

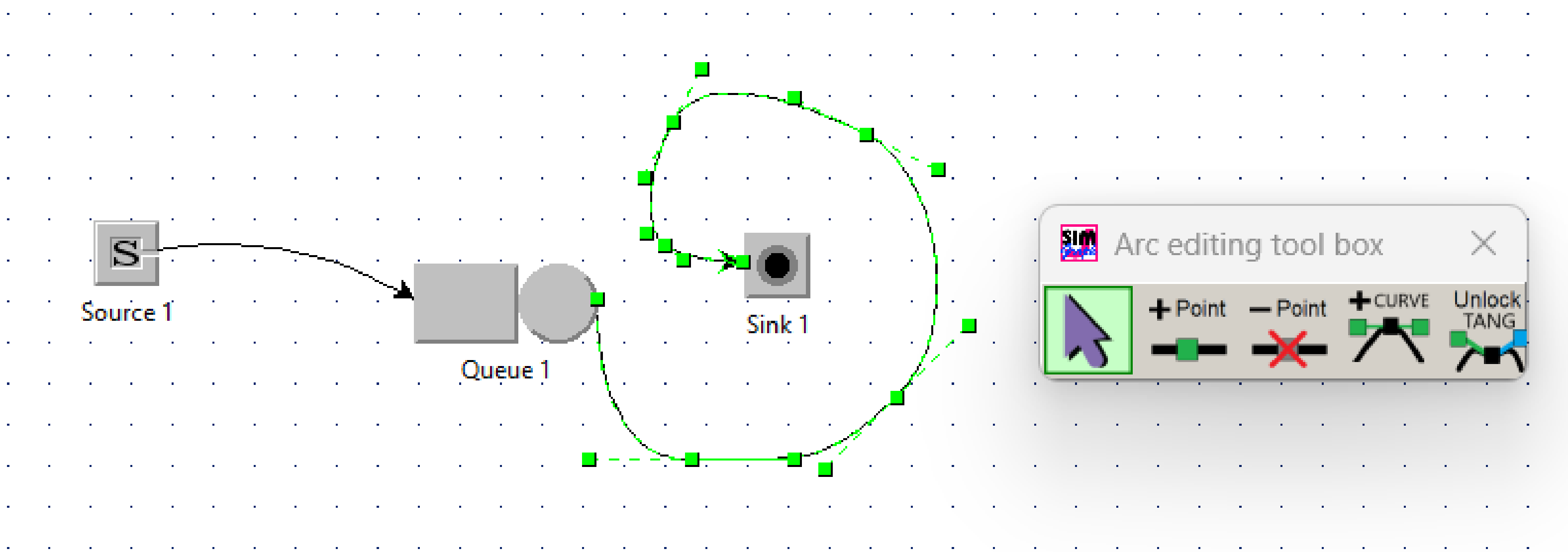
Keep in mind that models can have a *Loops*: arcs that starts from one node and ends on itself.

They are obtained by clicking and releasing the mouse press on the same node.



JMT: Connections

Configurable arcs allow the user to precisely define the path that the arc follows, including the possibility of adding curved segments.

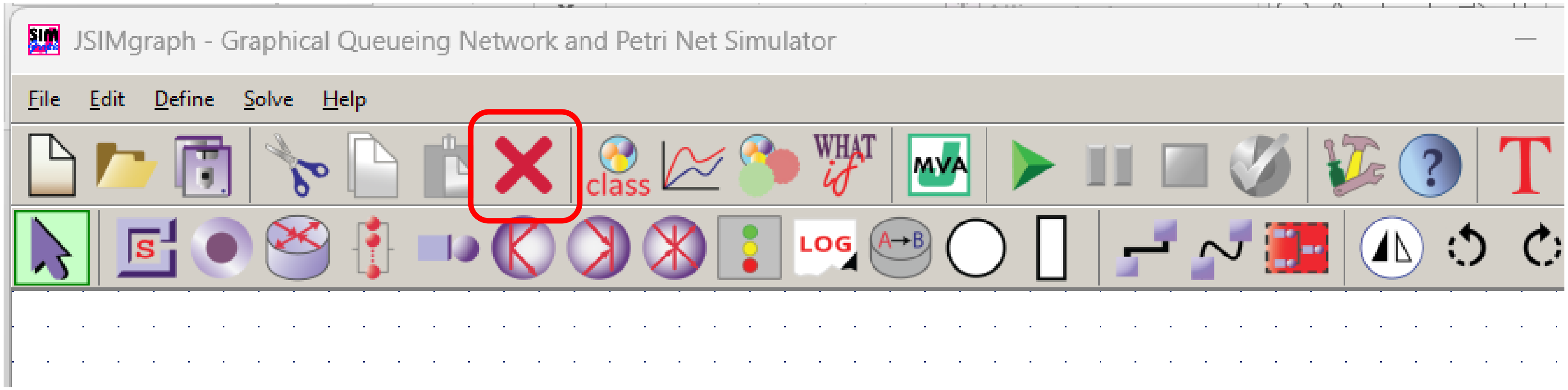


If you are interested in knowing more on how to configure these arcs, please refer to the JMT manual.

JMT: Undo and Deleting

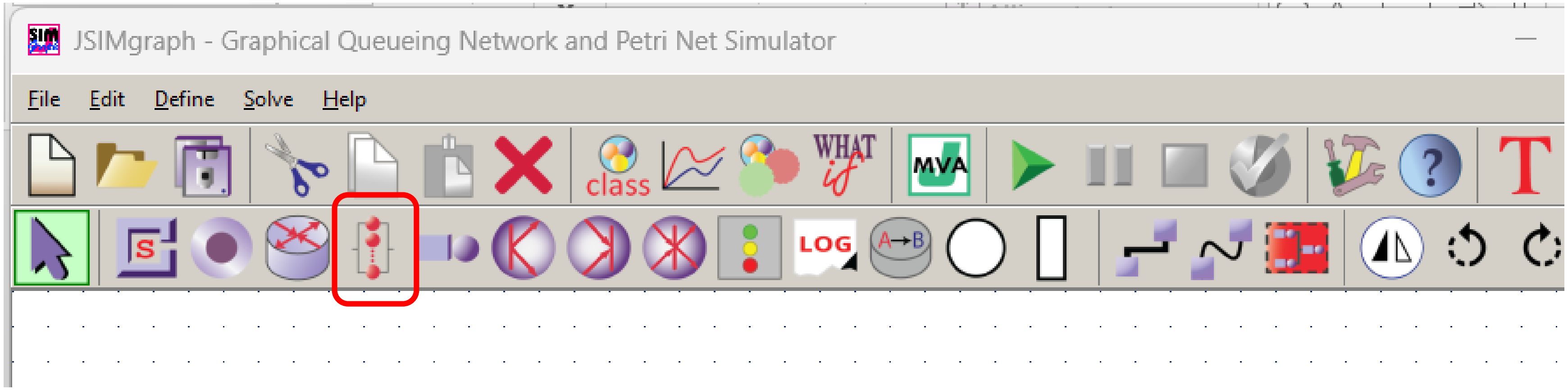
Elements (both nodes and arcs) can be deleted by first selecting them, and then pressing the red cross shaped button.

Be aware that JMT does not have an Undo feature, so you must be extra careful when deleting elements.



JMT: Delay Stations

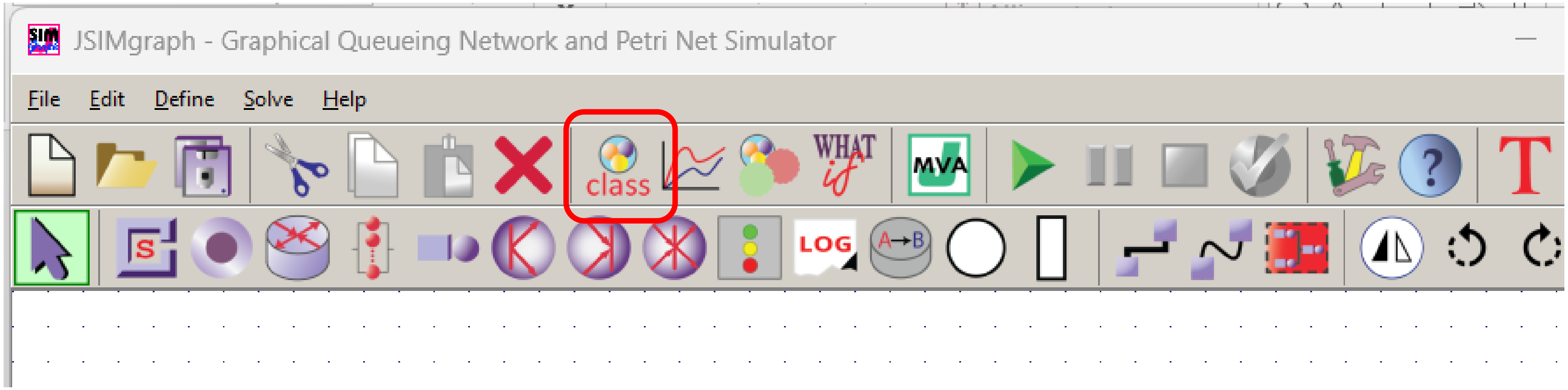
Delay Stations can be entered with the corresponding Icon, and can be configured in a similar way.



JMT: Performance Indices

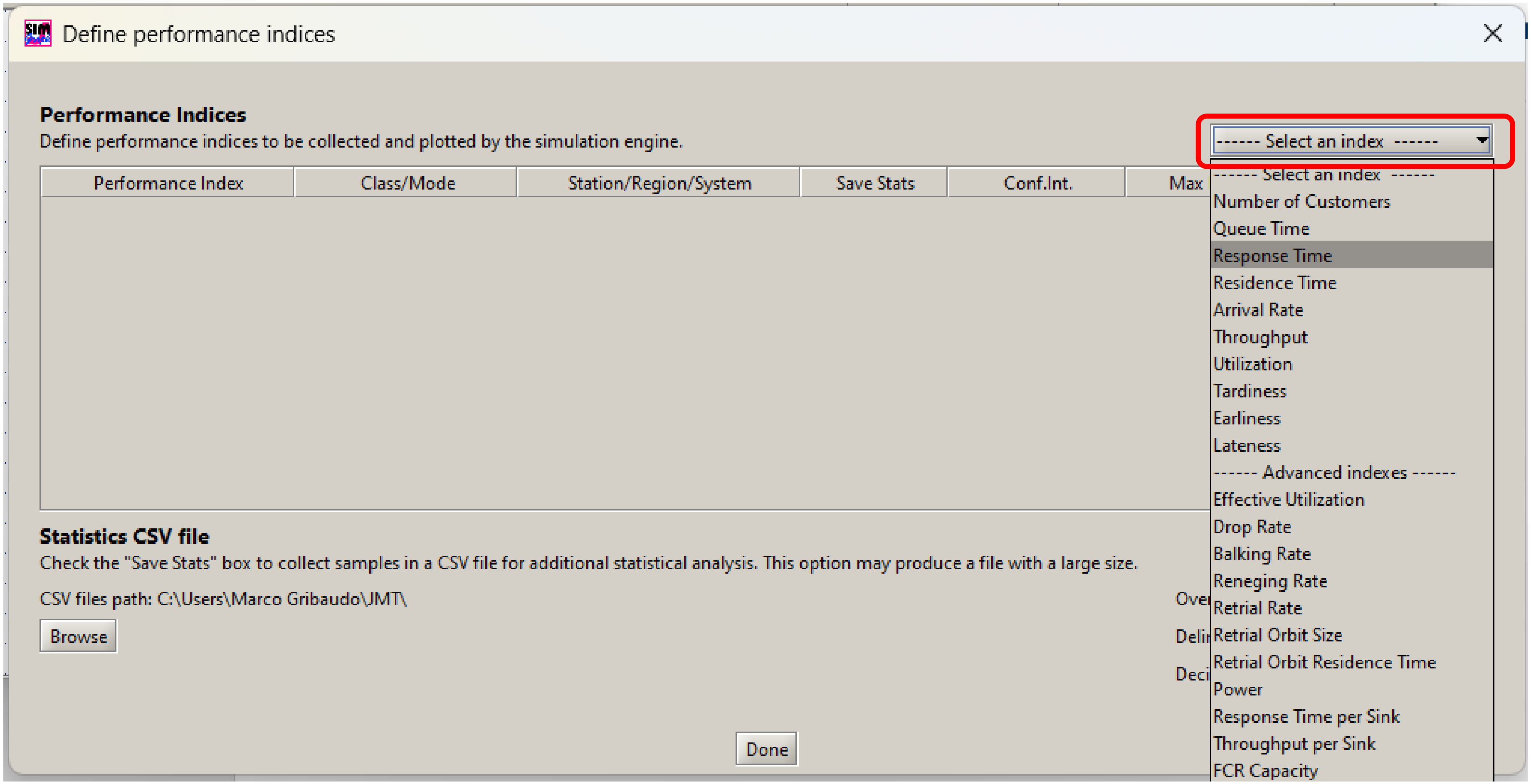
Once a model has been drawn, the *Performance Indices* in which we are interested must be selected.

This is done by pressing the corresponding button.



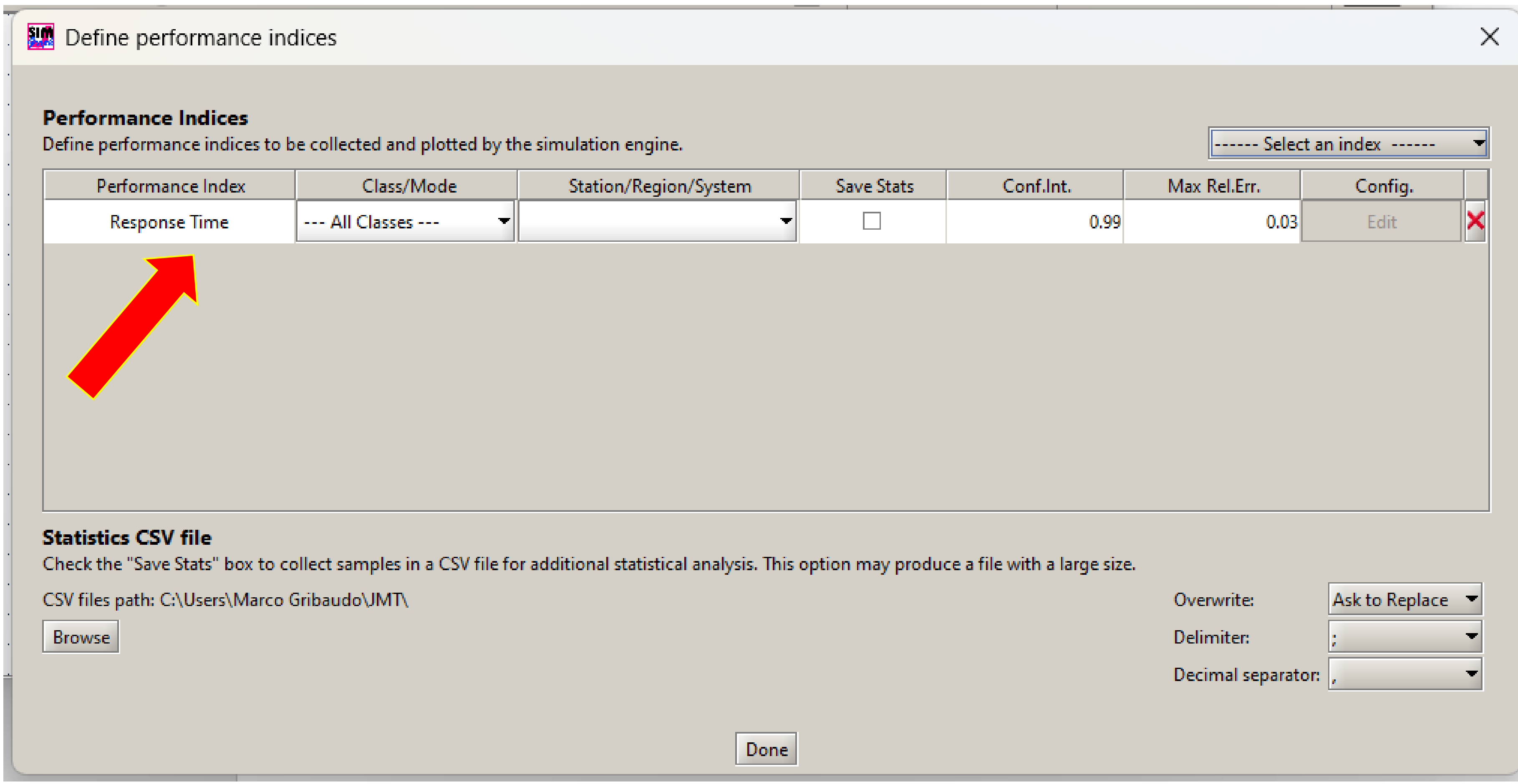
JMT: Performance Indices

The type of performance measure must be selected from the drop-down menu at the top right.



JMT: Performance Indices

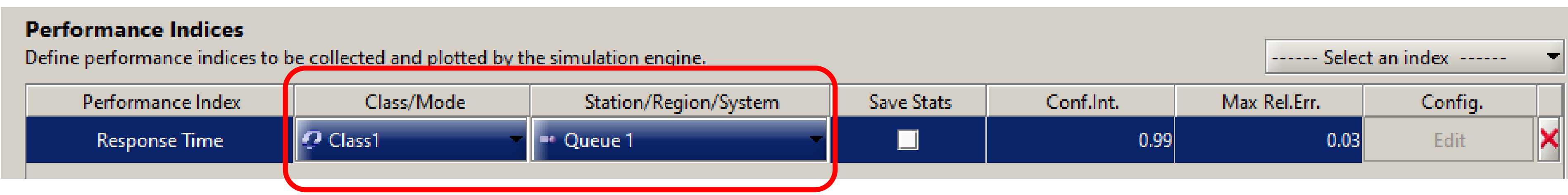
This adds a corresponding line in the table in the center of the window.



JMT: Performance Indices

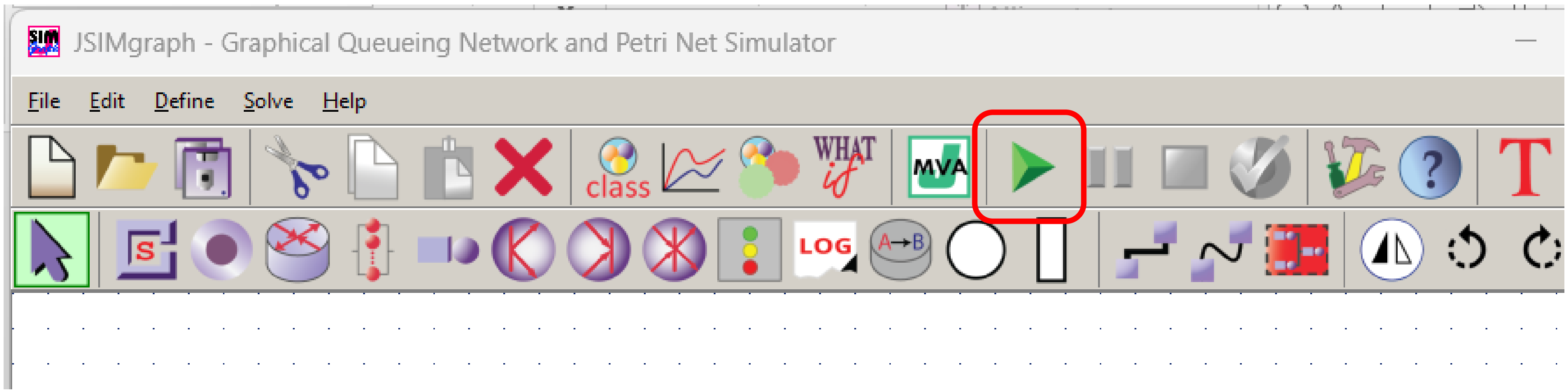
The performance metrics should be configured, by choosing to which element of the model they refer.

We will return on the meaning of the available metrics, and how they can be configured, in the following lessons.



JMT: Solution

The model can be solved by pressing the “Play” button.



JMT: Solution

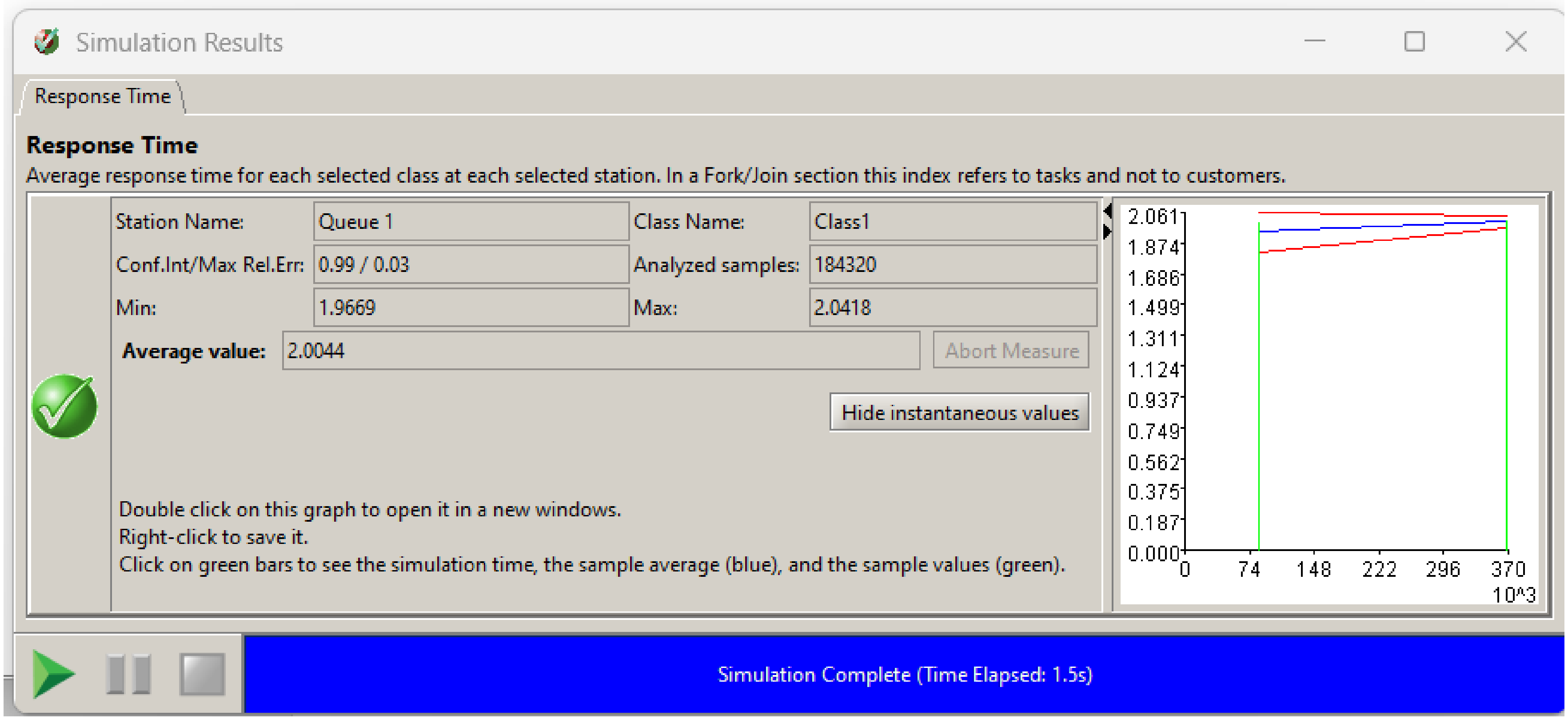
The technique used by *JSimGraph* to compute the solution is called *Discrete Event Simulation*. We will return on its characteristics in the following lessons. However, in the following, we will very frequently call the process of computing the performance indices of a model “*Simulation*”.

Since it performs several executions and averages them to compute the performance indices, it might requires some time to converge.

The computation of the metrics might even fail: we will see how to deal with these cases in the future.

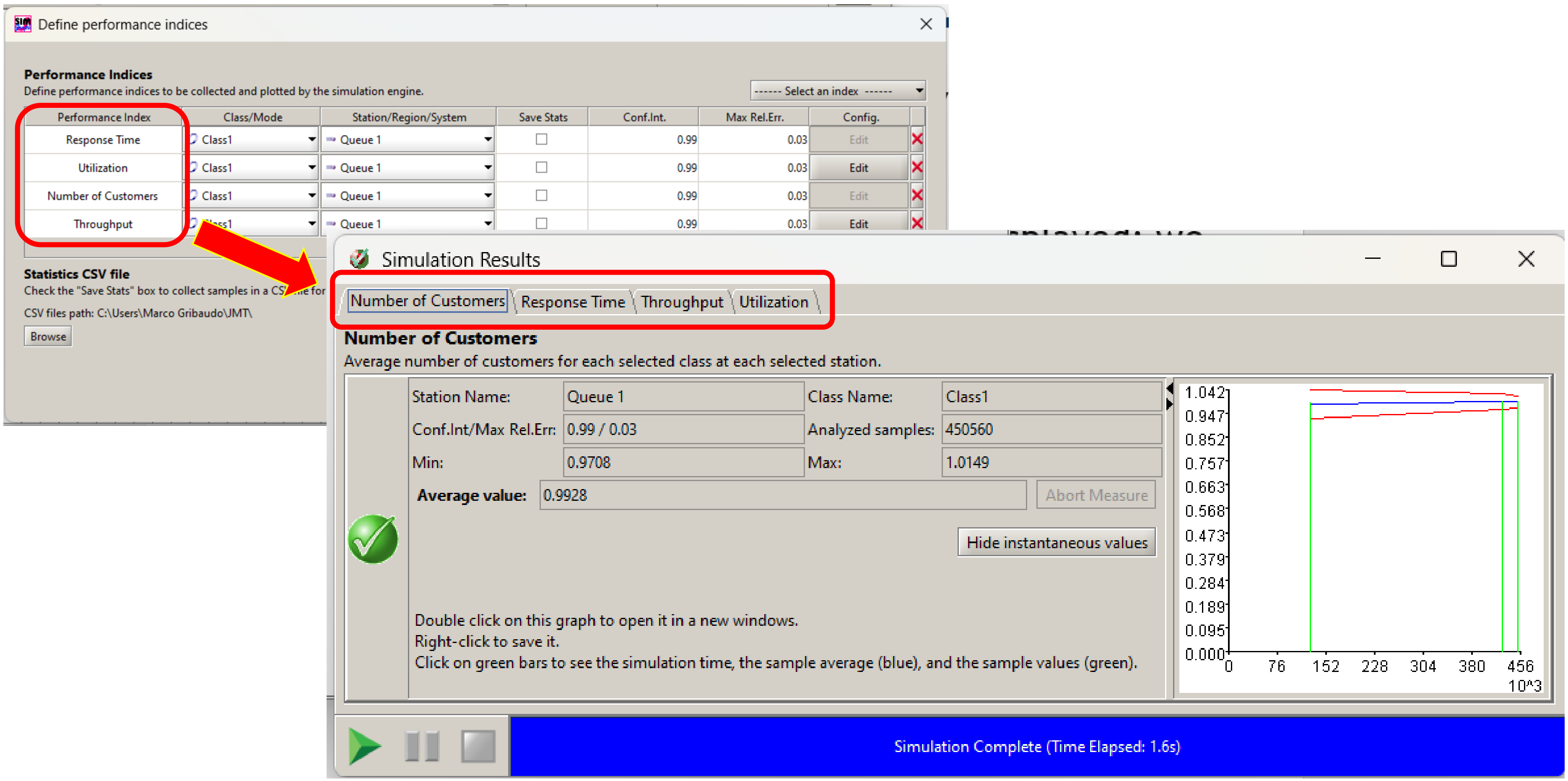
JMT: Solution

If everything went well, the values of the performance metrics will be shown with a green icon indicating that their computation was successful.



JMT: Solution

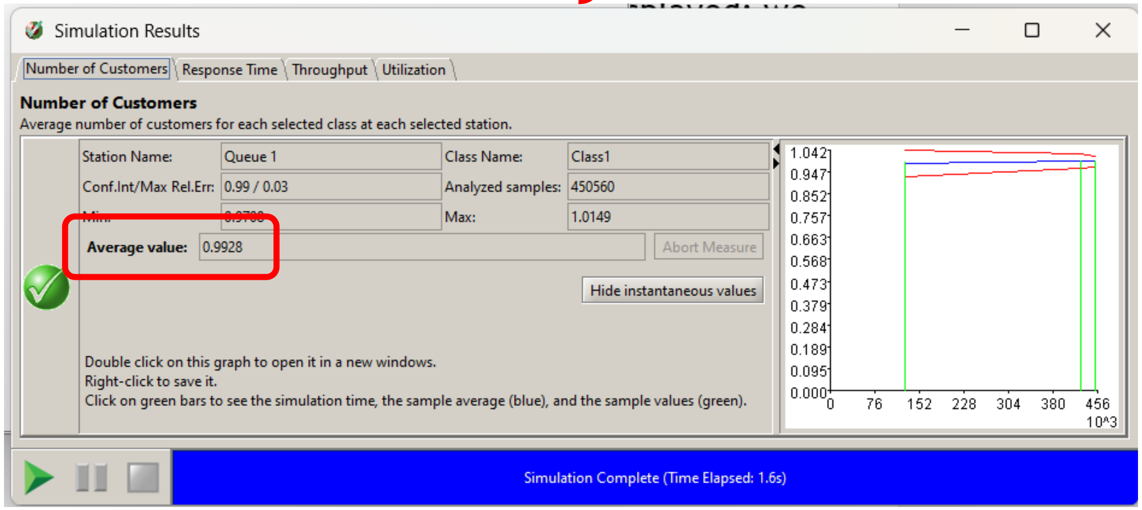
Metrics will be grouped in different tabs, according to their definition. Moreover several other features will be displayed: we will return on them later.



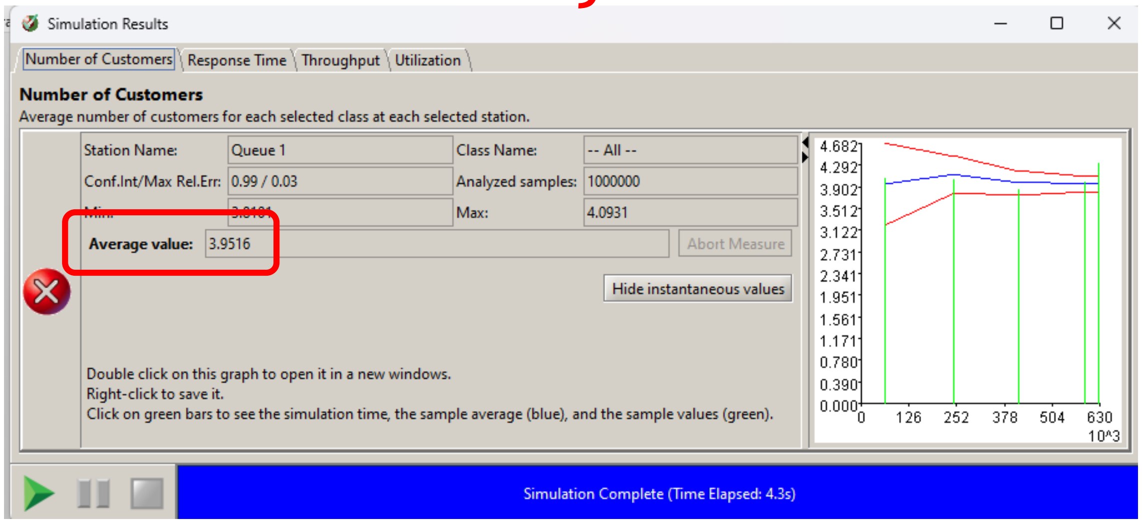
JMT: Instability

If we increase for example the arrival rate, all the performance indices will increase accordingly.

λ = 0.5 job/sec



λ = 0.8 job/sec

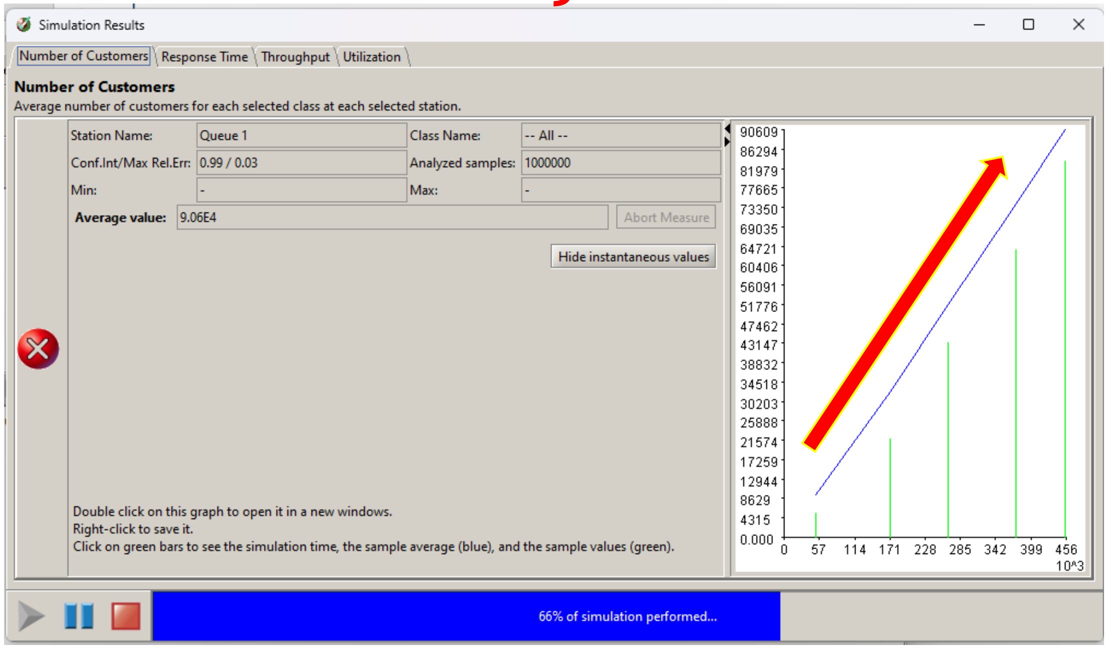


In this case, the desired accuracy could have not been reached. This is represented by the red circle with a cross inside.
We will return on this topi later.

JMT: Instability

If the arrival rate increases up to the point in which the system becomes unstable, performance indices like the *Response Time* and *Number of Jobs* will continuously grow, and the solution will not converge.

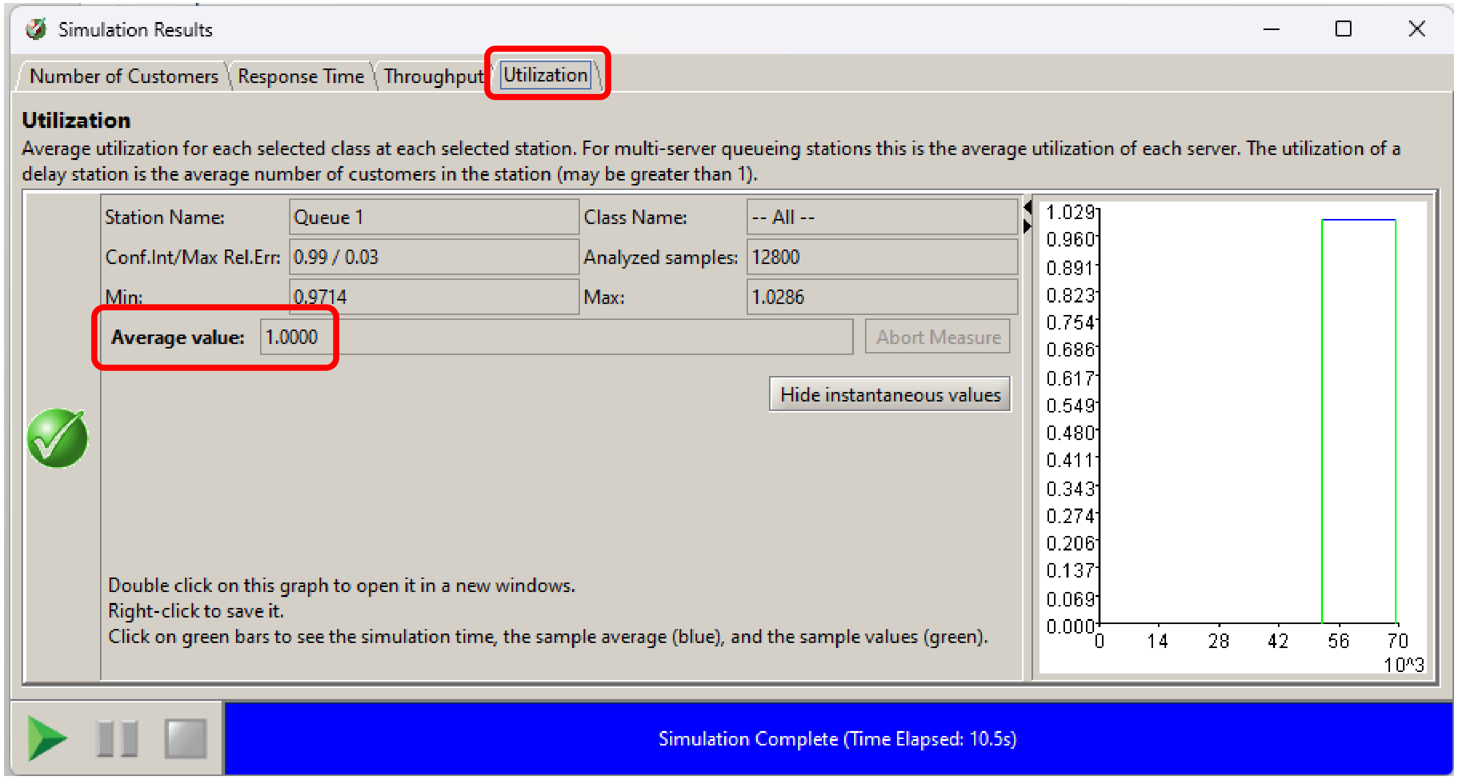
λ = *1.2 job/sec*



In the worst case, the JAVA virtual machine might consume all the available memory and create an unrecoverable error that will force the user to force-quit the application, loosing all the un-saved changes!

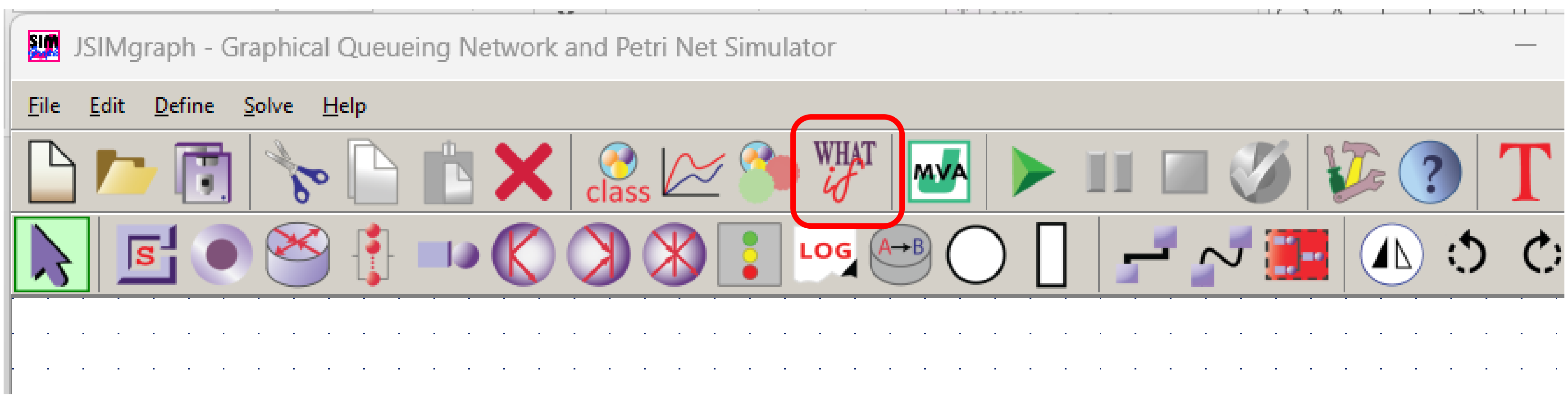
JMT: Instability

Please note that other performance indices instead, such as the Utilization and the Throughput, will reach a maximum value, and their computation will converge even if the model is unstable.



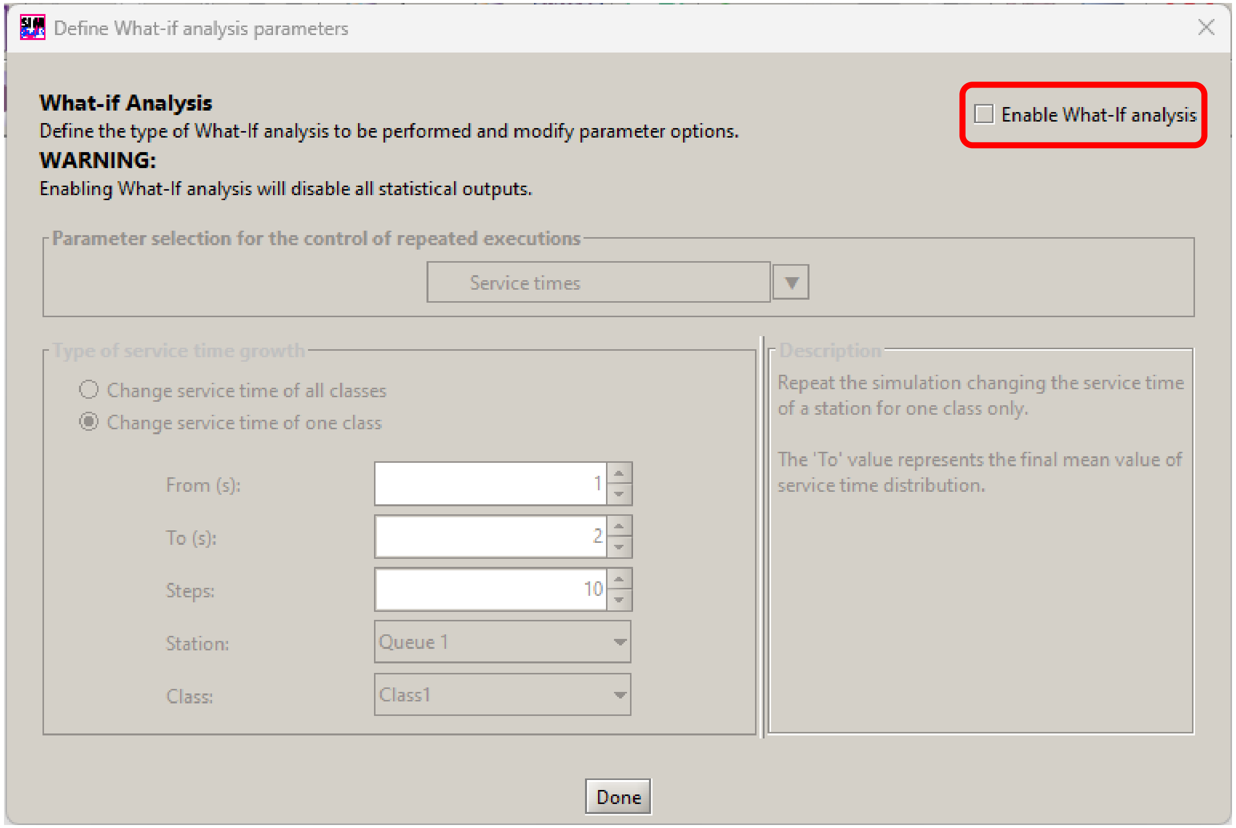
JMT: What-if analysis

When we want to study the evolution of our model as function of one of its parameters, such for example the Arrival Rate, we can exploit the What-if analysis feature of JMT, which can be activated from the corresponding window.



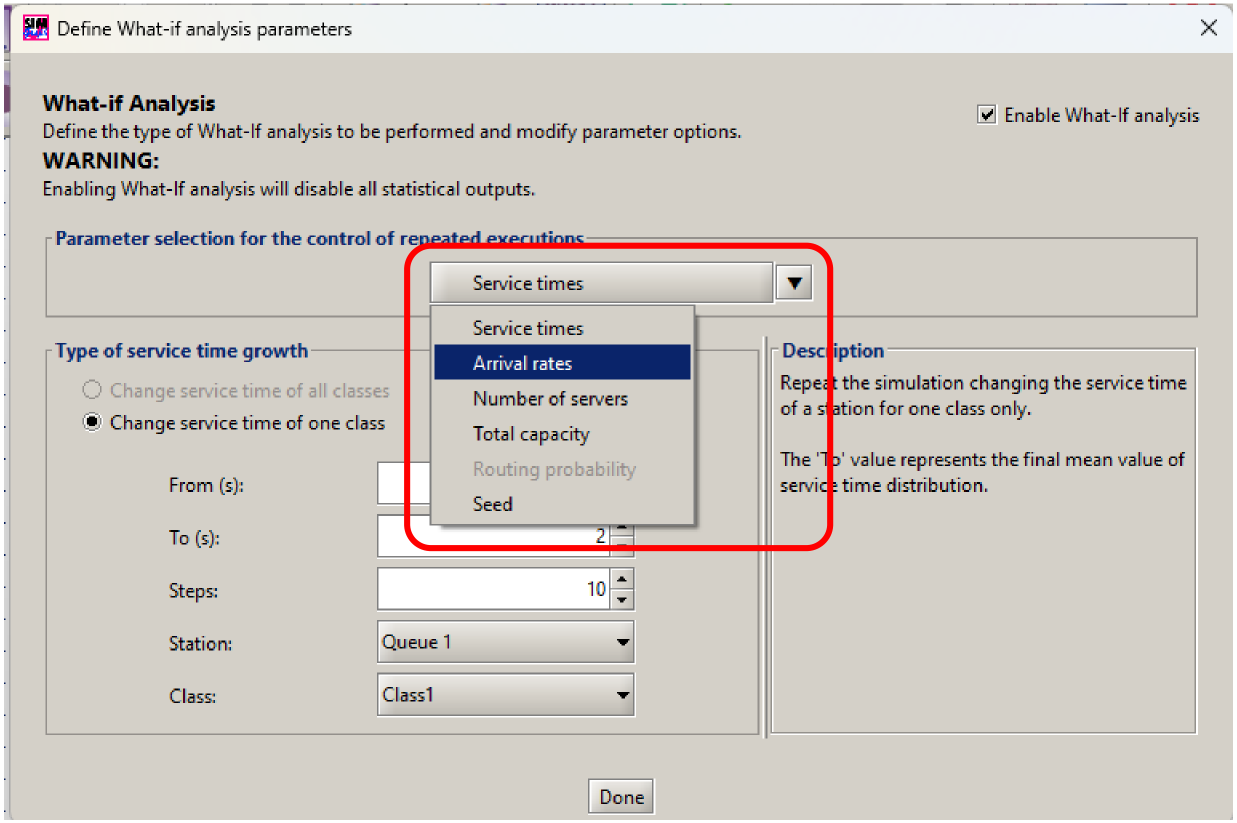
JMT: What-if analysis

The analysis must be activated by checking the corresponding option.



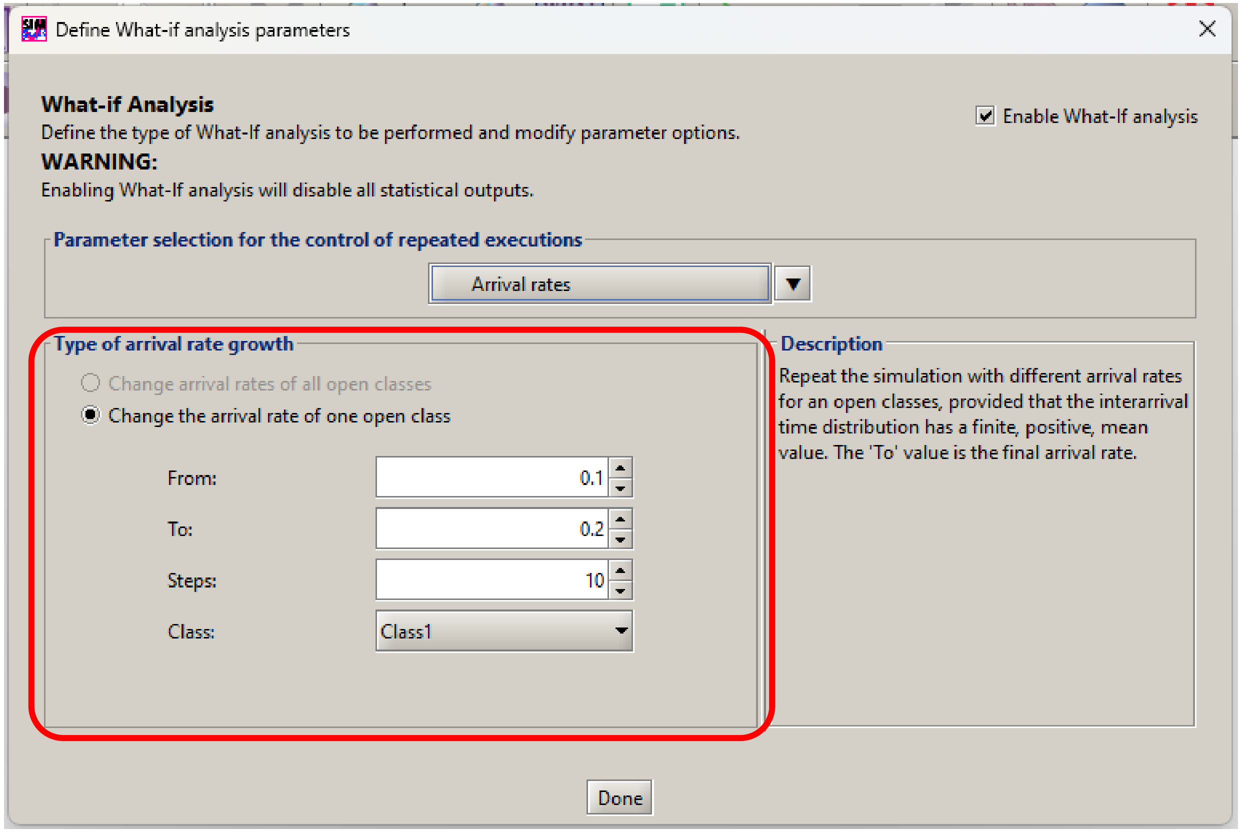
JMT: What-if analysis

The type of parameter that is varied is then selected from the drop-down menu.



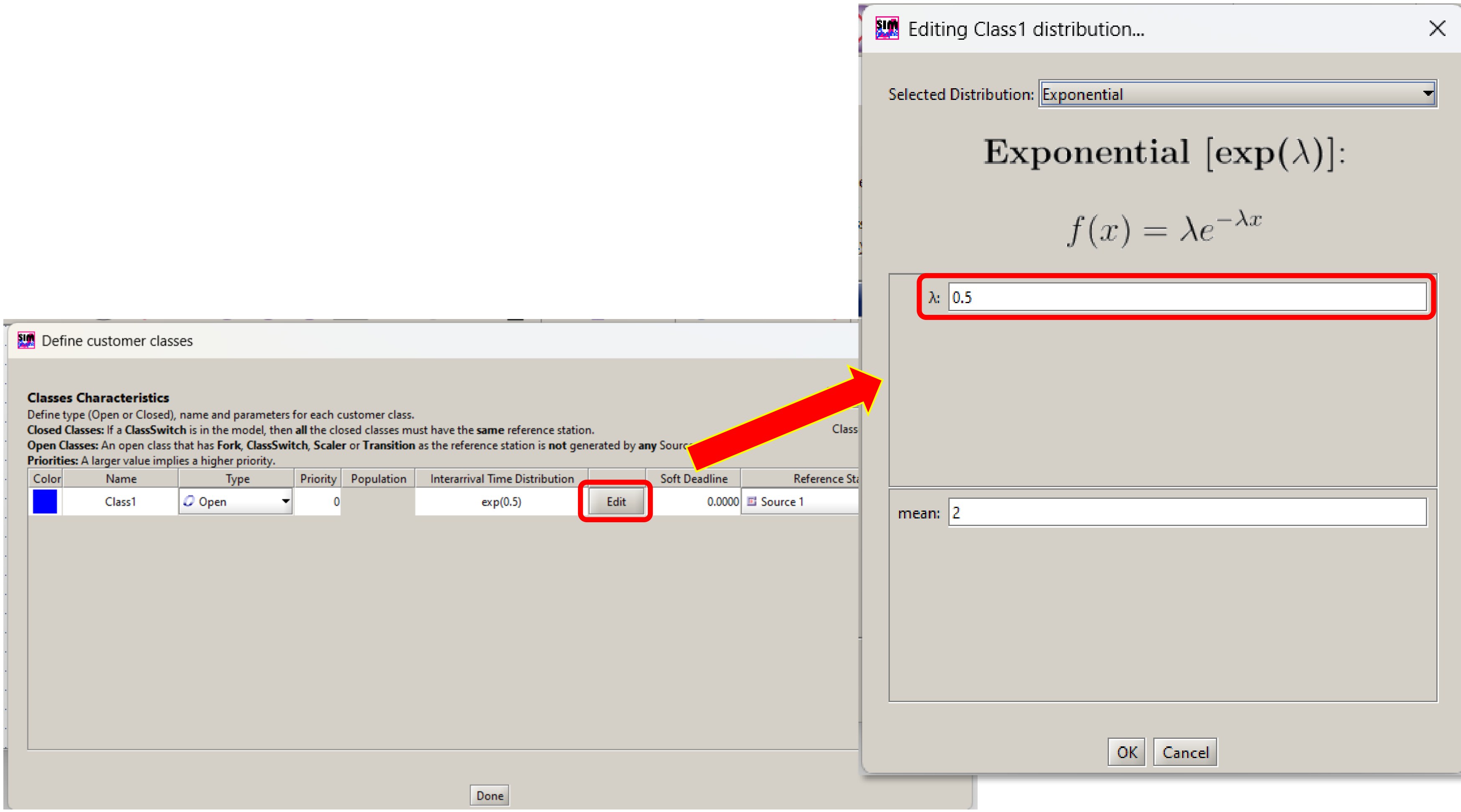
JMT: What-if analysis

It is further configured from the fields that appear below, allowing, for example, to select the station or the class to which it refers.



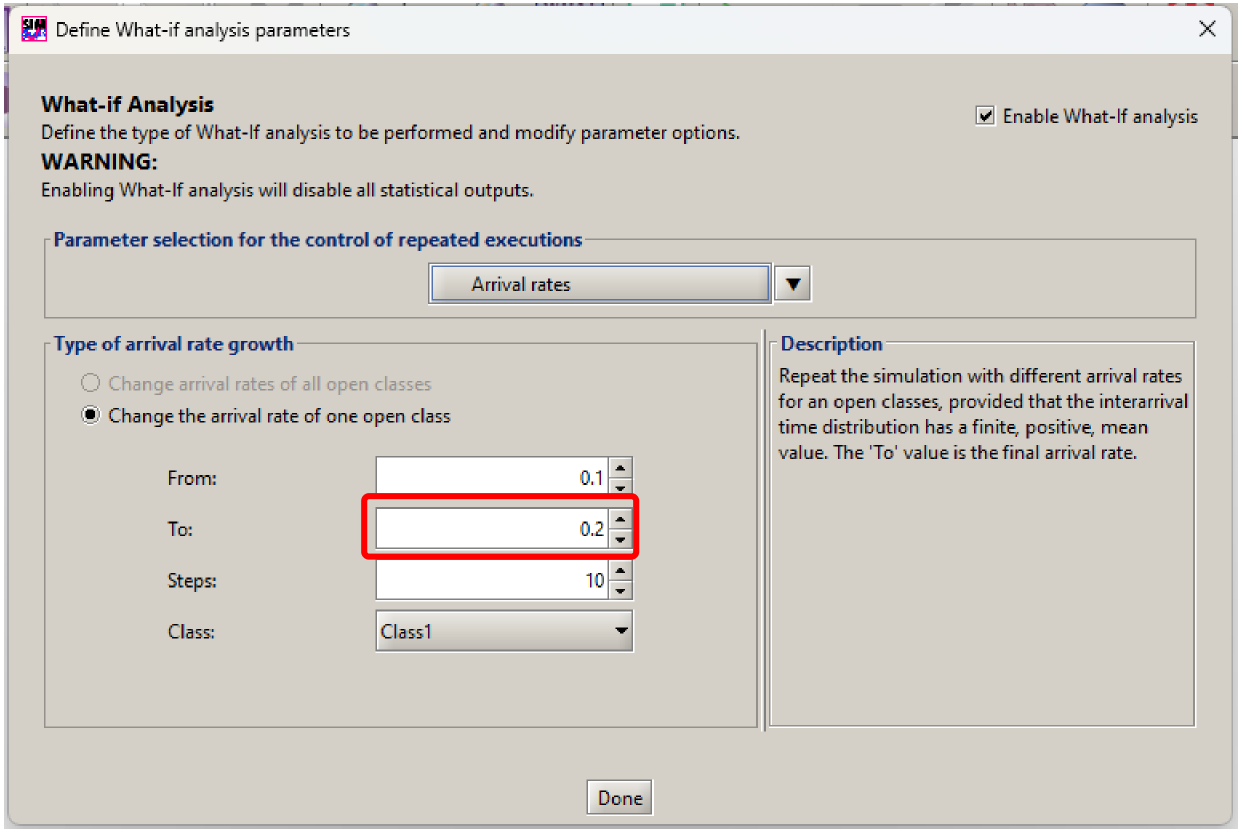
JMT: What-if analysis

The starting value of the analysis, is the one entered when creating the model.



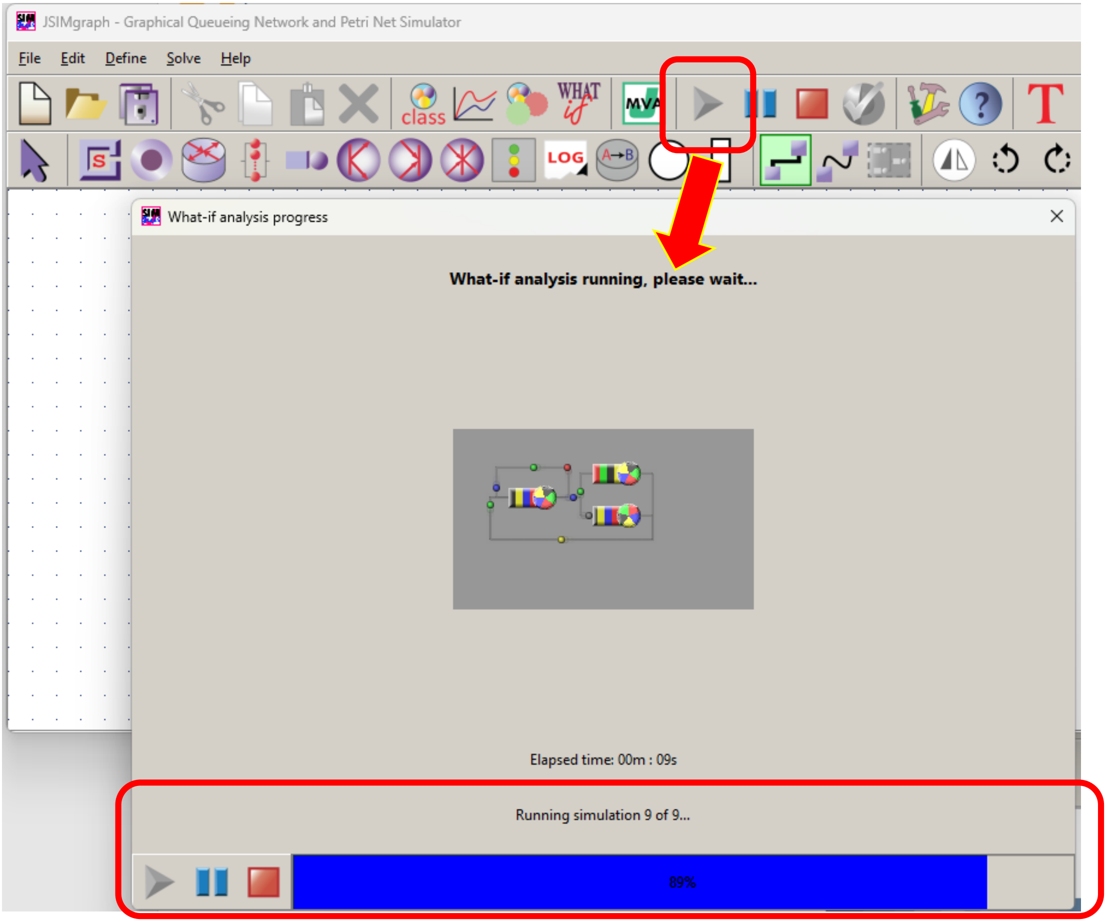
JMT: What-if analysis

The final value is instead entered in the What-if panel. The same window, also allows to specify the steps the solution should be computed.



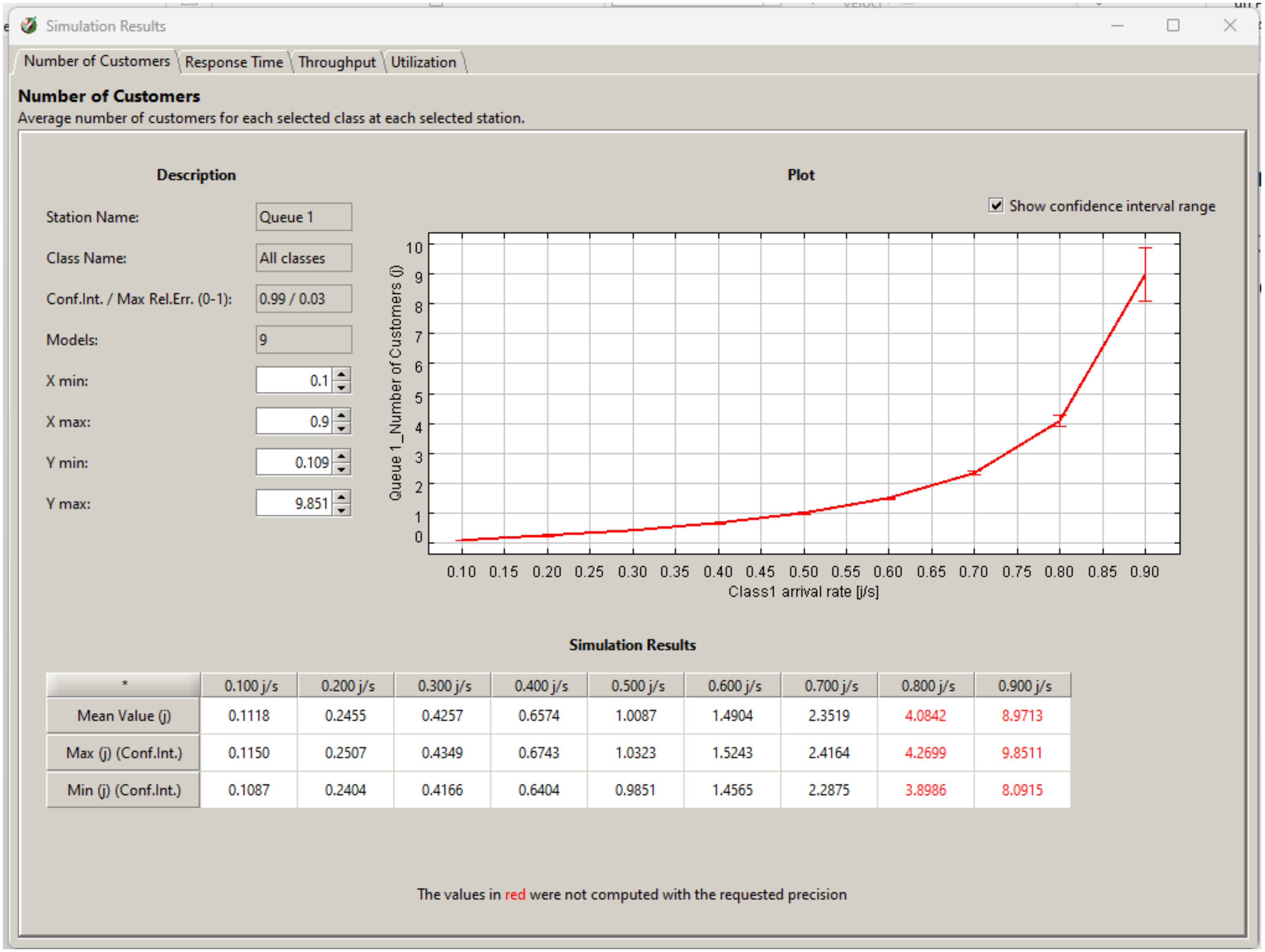
JMT: What-if analysis

When the solution is executed, the output window is replaced by a progress bar that shows how many configurations have been computed.



JMT: What-if analysis

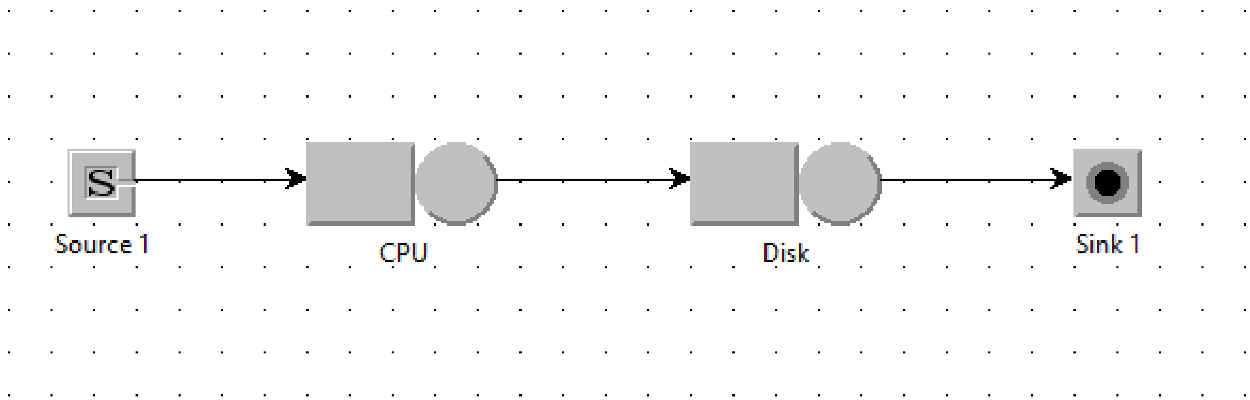
Results are then presented as tables and plots.



*	0.100 j/s	0.200 j/s	0.300 j/s	0.400 j/s	0.500 j/s	0.600 j/s	0.700 j/s	0.800 j/s	0.900 j/s
Mean Value (j)	0.1118	0.2455	0.4257	0.6574	1.0087	1.4904	2.3519	4.0842	8.9713
Max (j) (Conf.Int.)	0.1150	0.2507	0.4349	0.6743	1.0323	1.5243	2.4164	4.2699	9.8511
Min (j) (Conf.Int.)	0.1087	0.2404	0.4166	0.6404	0.9851	1.4565	2.2875	3.8986	8.0915

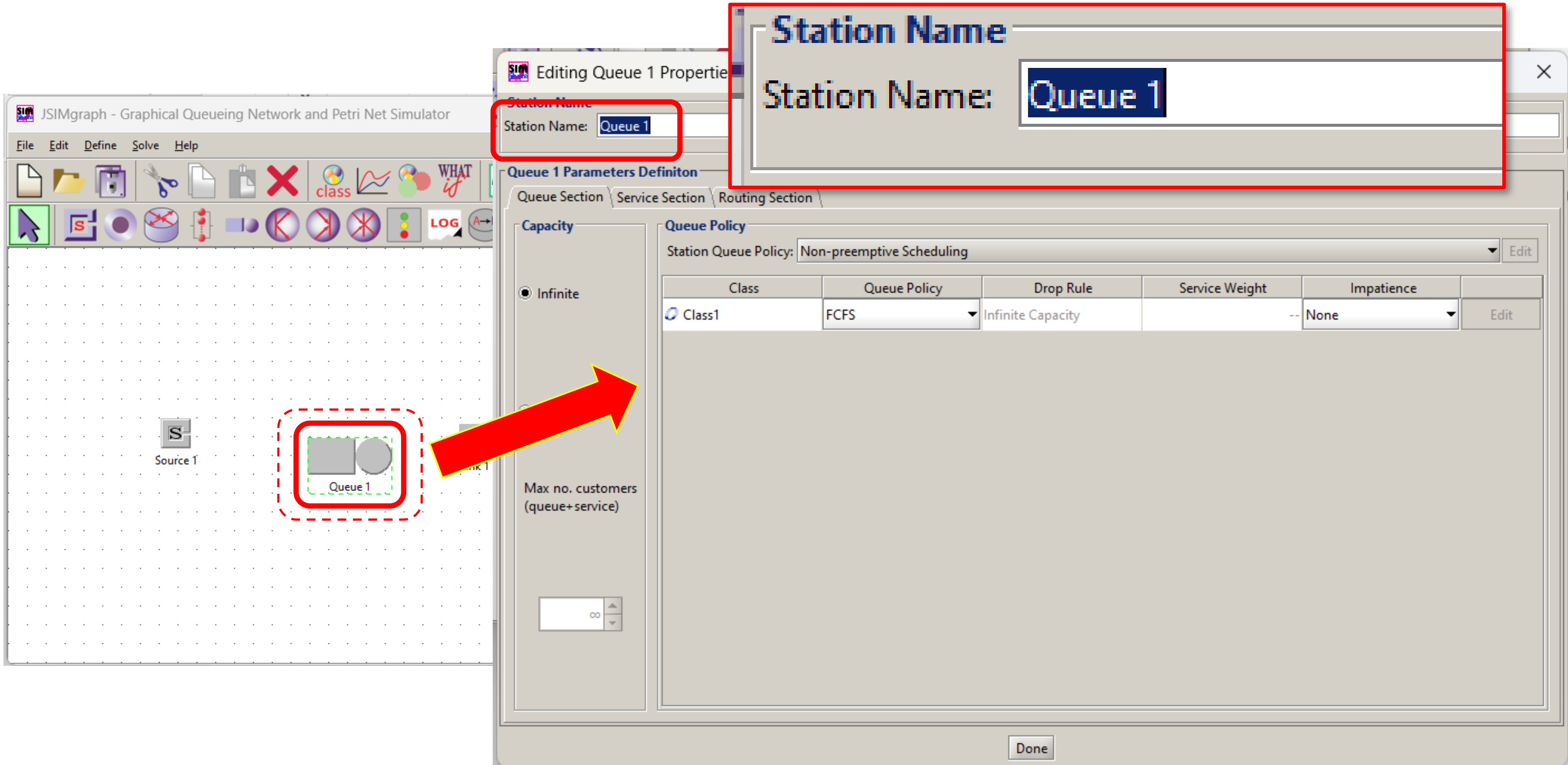
More queues

A model can be composed by several queues.



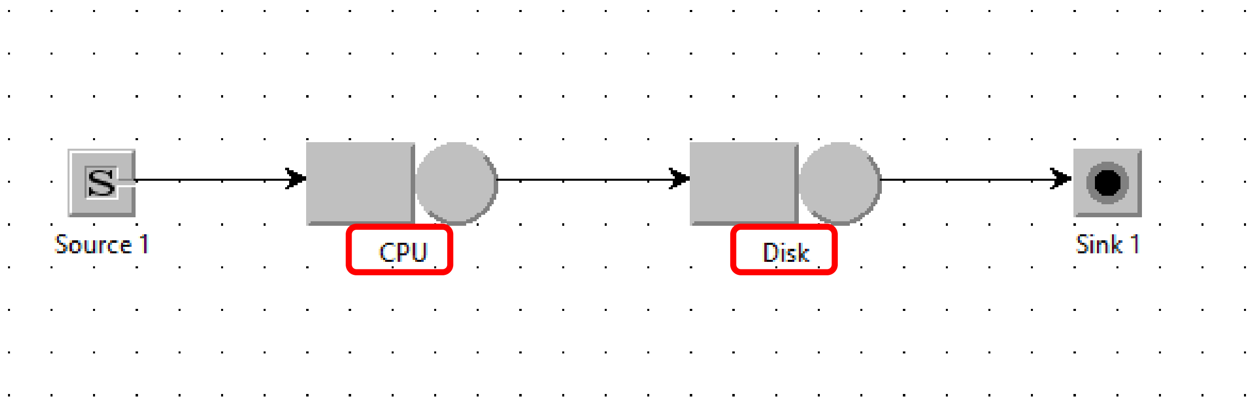
More queues

When a node is edited by performing a double click on it, a *Name* can be assigned to it.



More queues

This name can be used to identify the node in the model: it become of paramount importance when considering more than one queue.

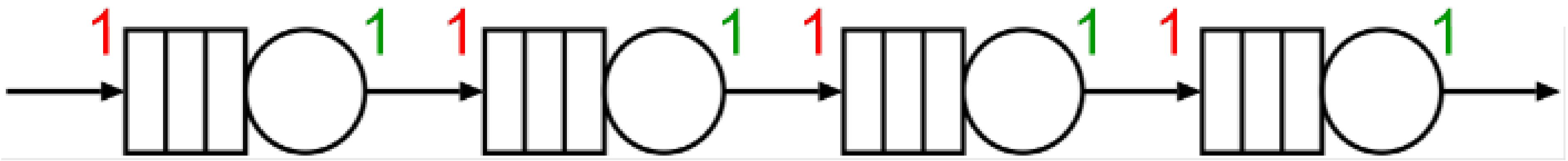


Tandem Networks

The simplest case of systems with more stations are the Tandem Networks….

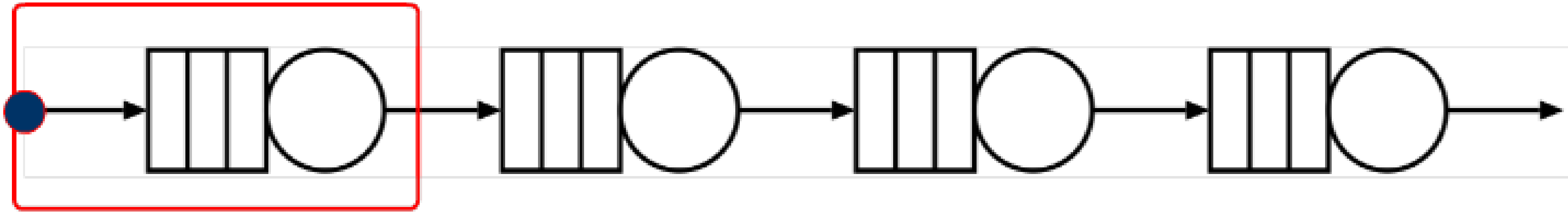
Tandem networks

A *Tandem Open Queuing Network* is an open system where every queue has exactly one input and one output connection.



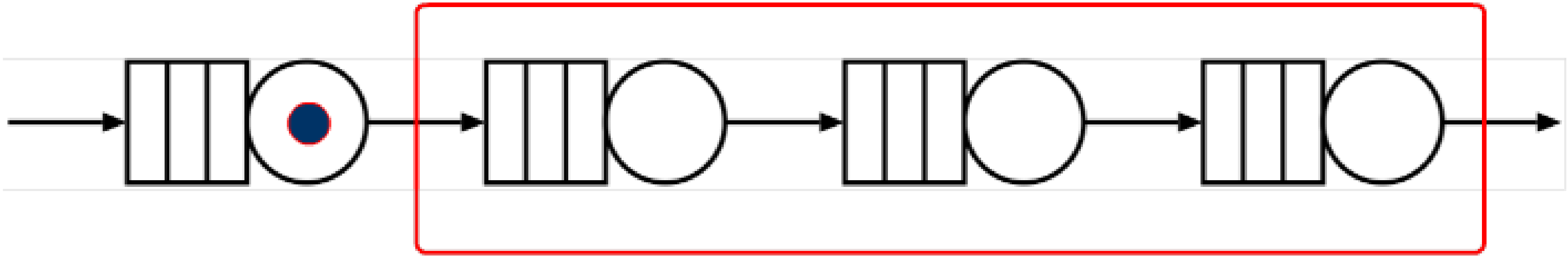
Tandem networks

Jobs enter the first node (queue) of the line...



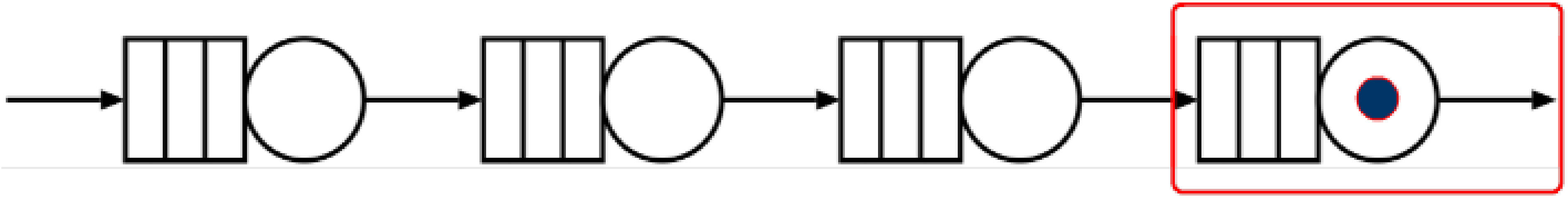
Tandem networks

... and proceeds stage after stage up to the last station.



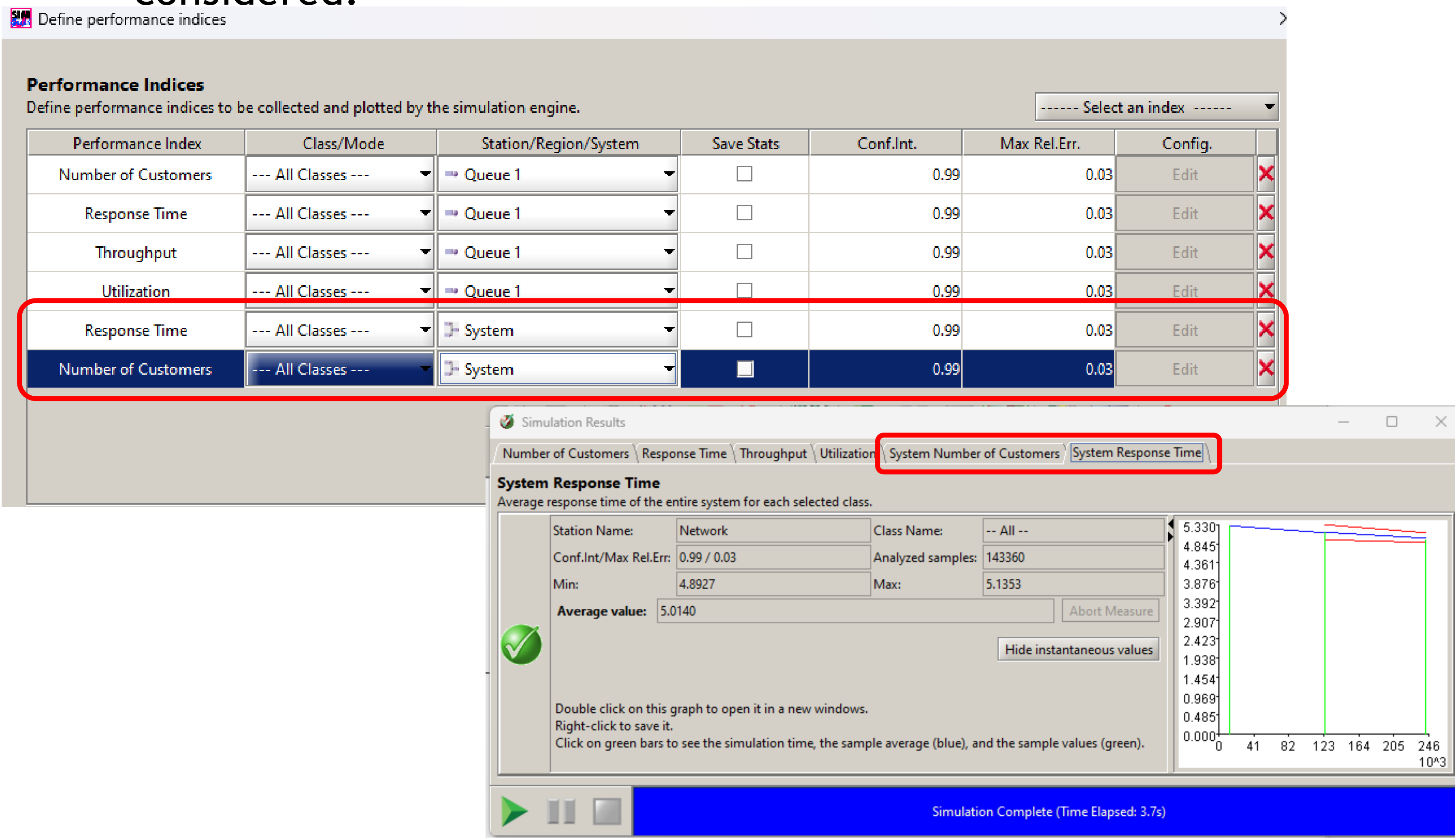
Tandem networks

After jobs have completed the final stage, they leave the system.



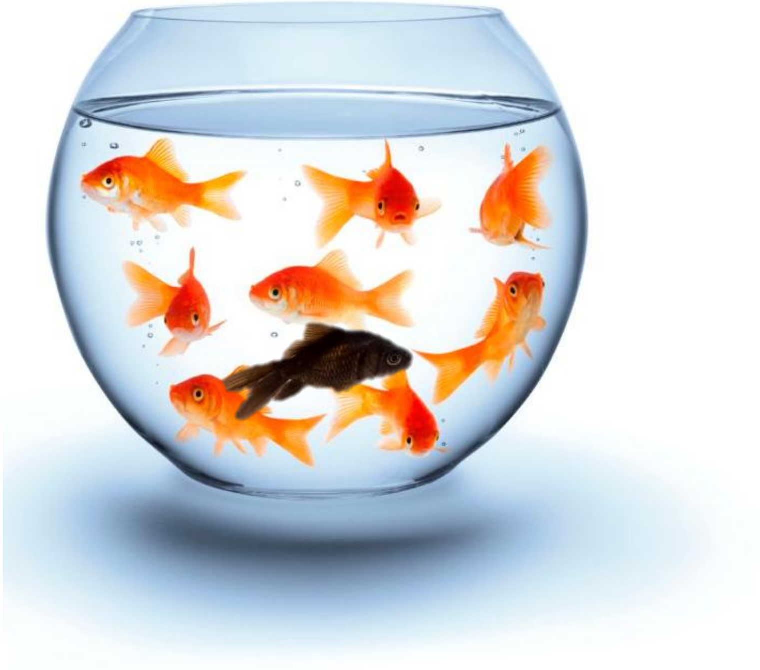
Network Performance Indices

When considering networks, some new performance metrics can be considered.



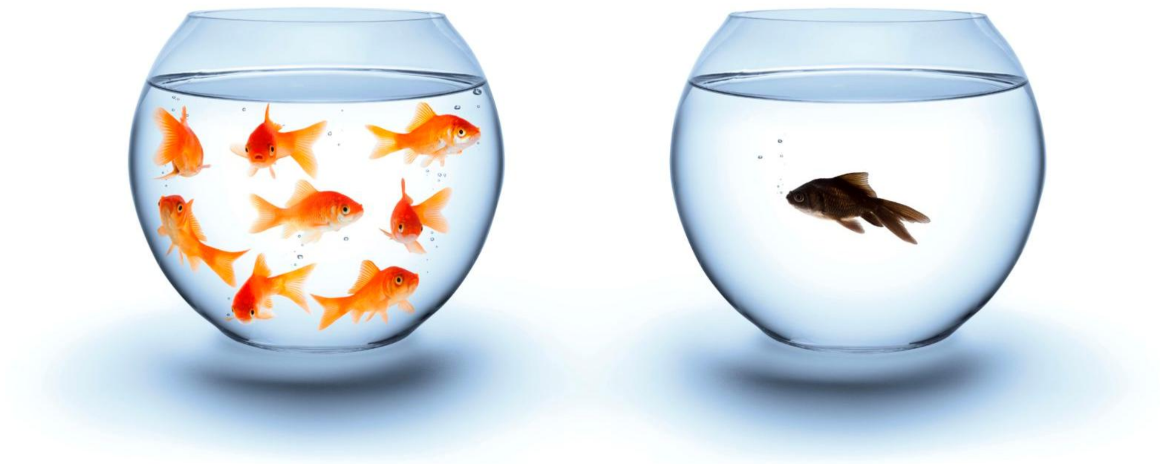
Closed systems: motivation

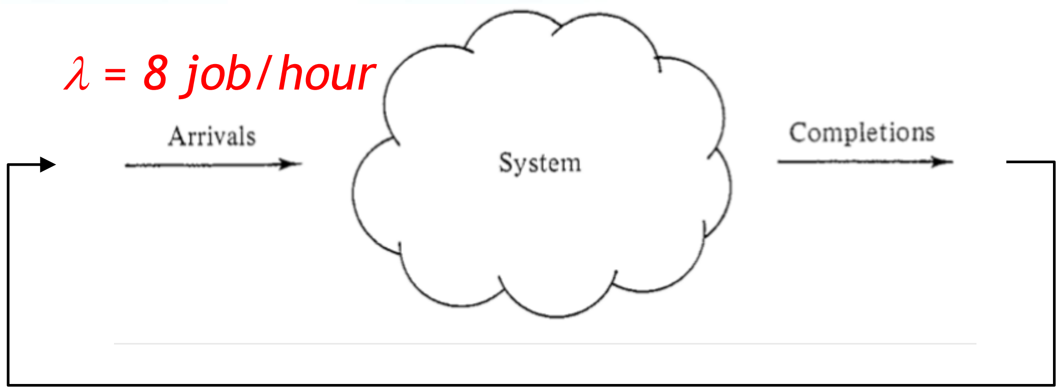
If the jobs that enter the system come from a fixed population…



Closed systems: motivation

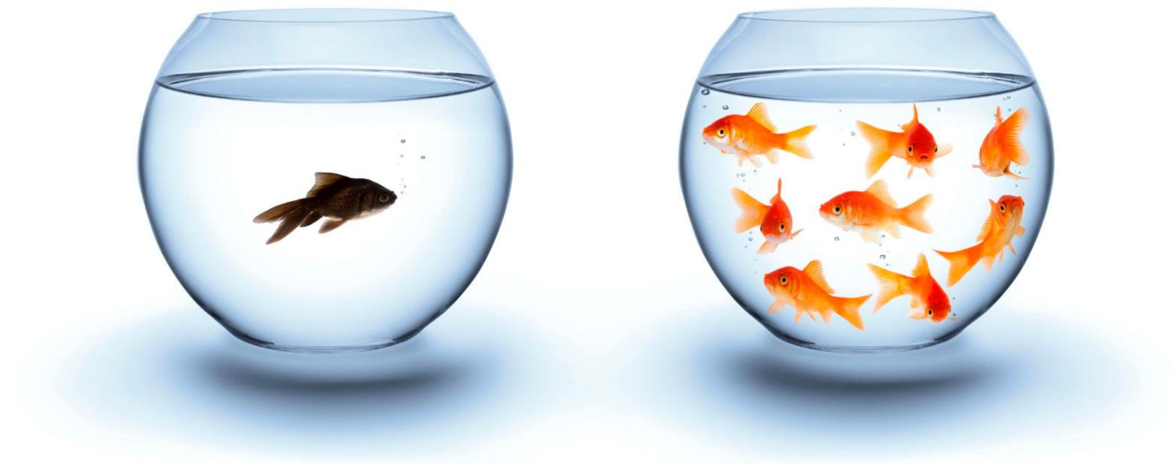
When most of the population is outside, jobs arrives at a high rate.

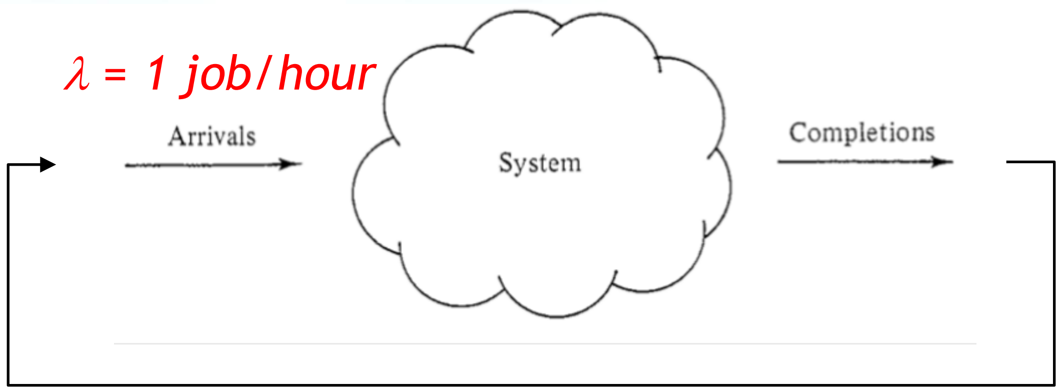




Closed systems: motivation

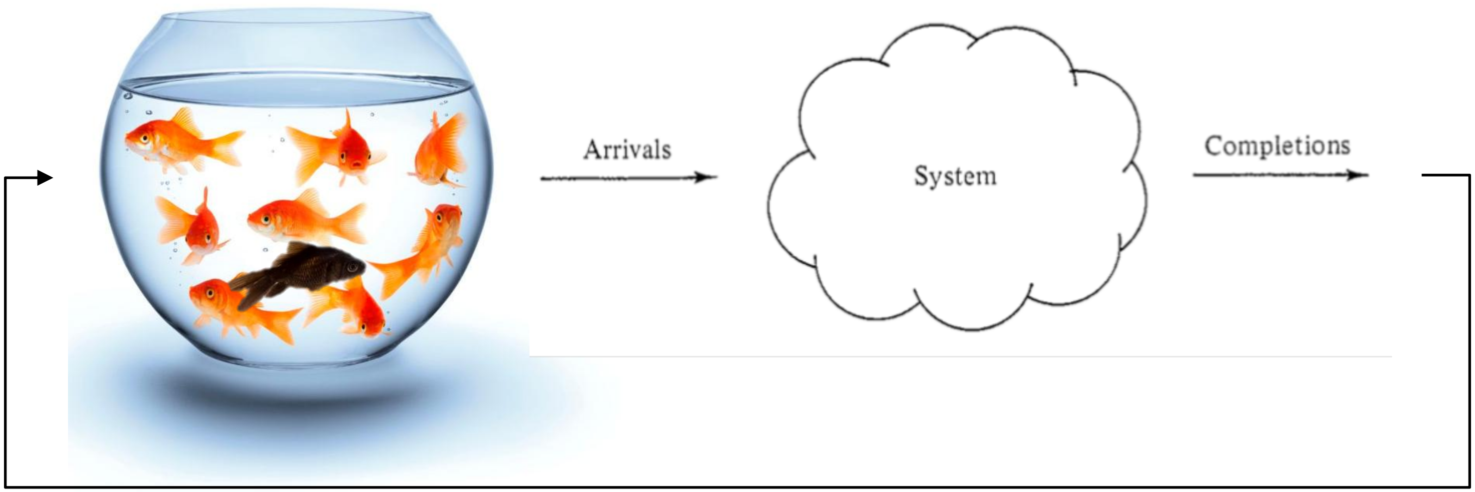
Conversely, when most of the population is inside, jobs arrives at a very low rate.





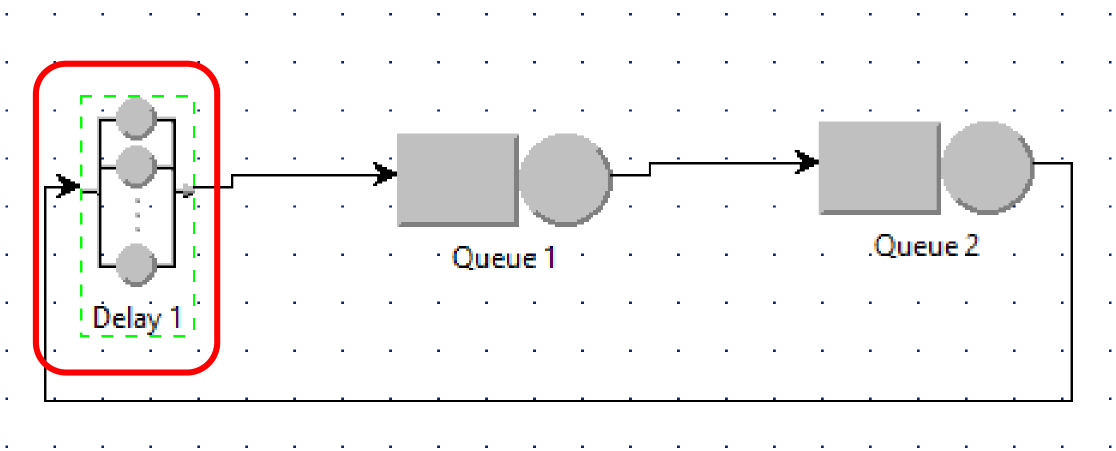
Closed systems: stability

For this reason, closed systems are *Always Stable* and needs specific techniques to be studied.



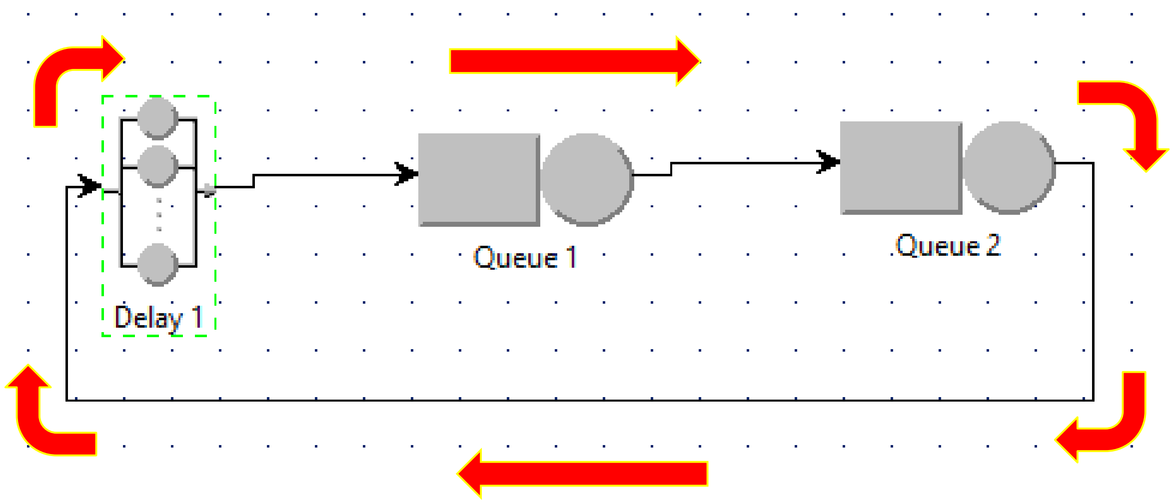
Closed systems: think time and delay station

The time spent “outside” the system is generally modelled by a delay station…



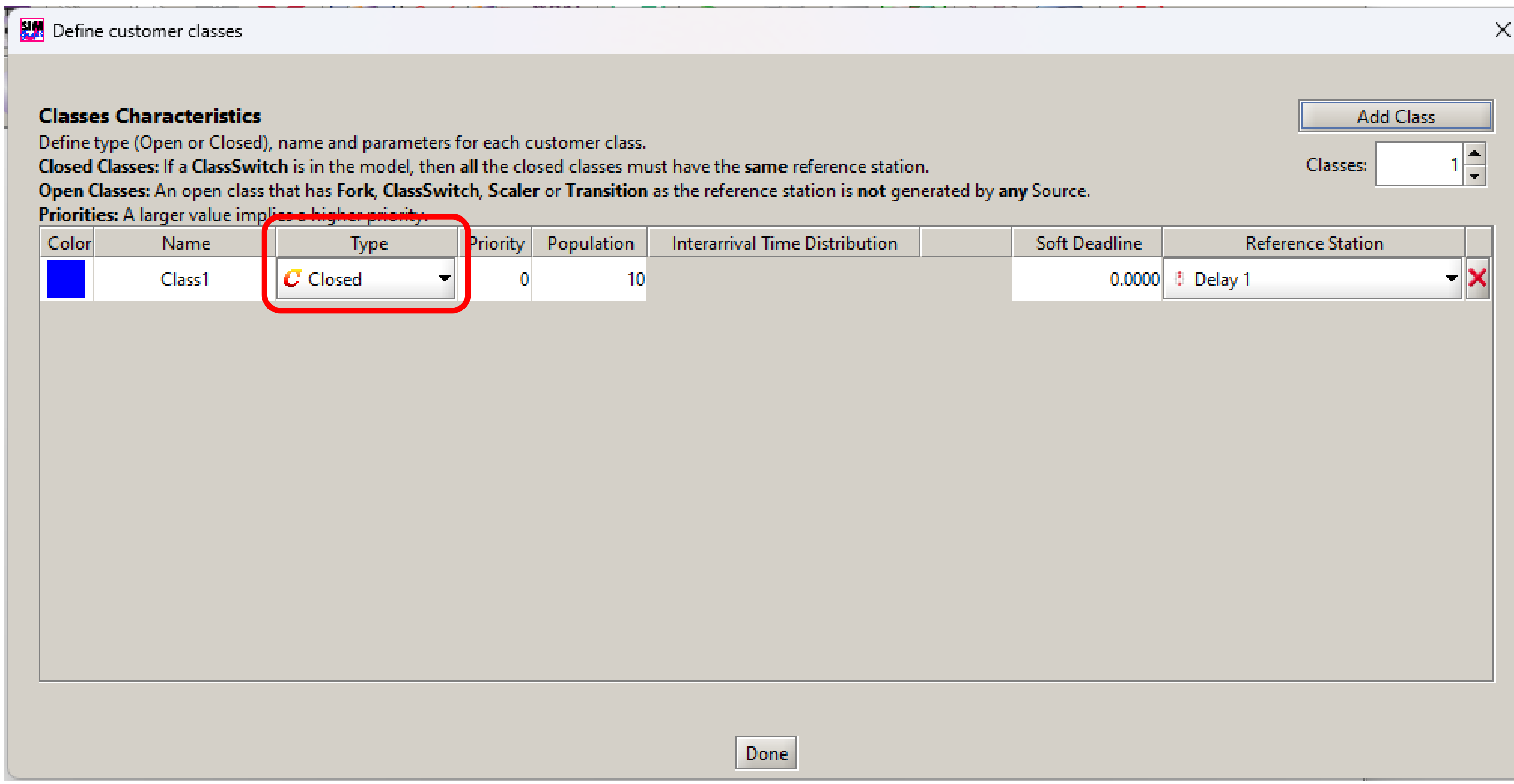
Closed systems: closed classes

Then, a closed model is created by defining a loop between the stations, passing through the terminal (delay) station.



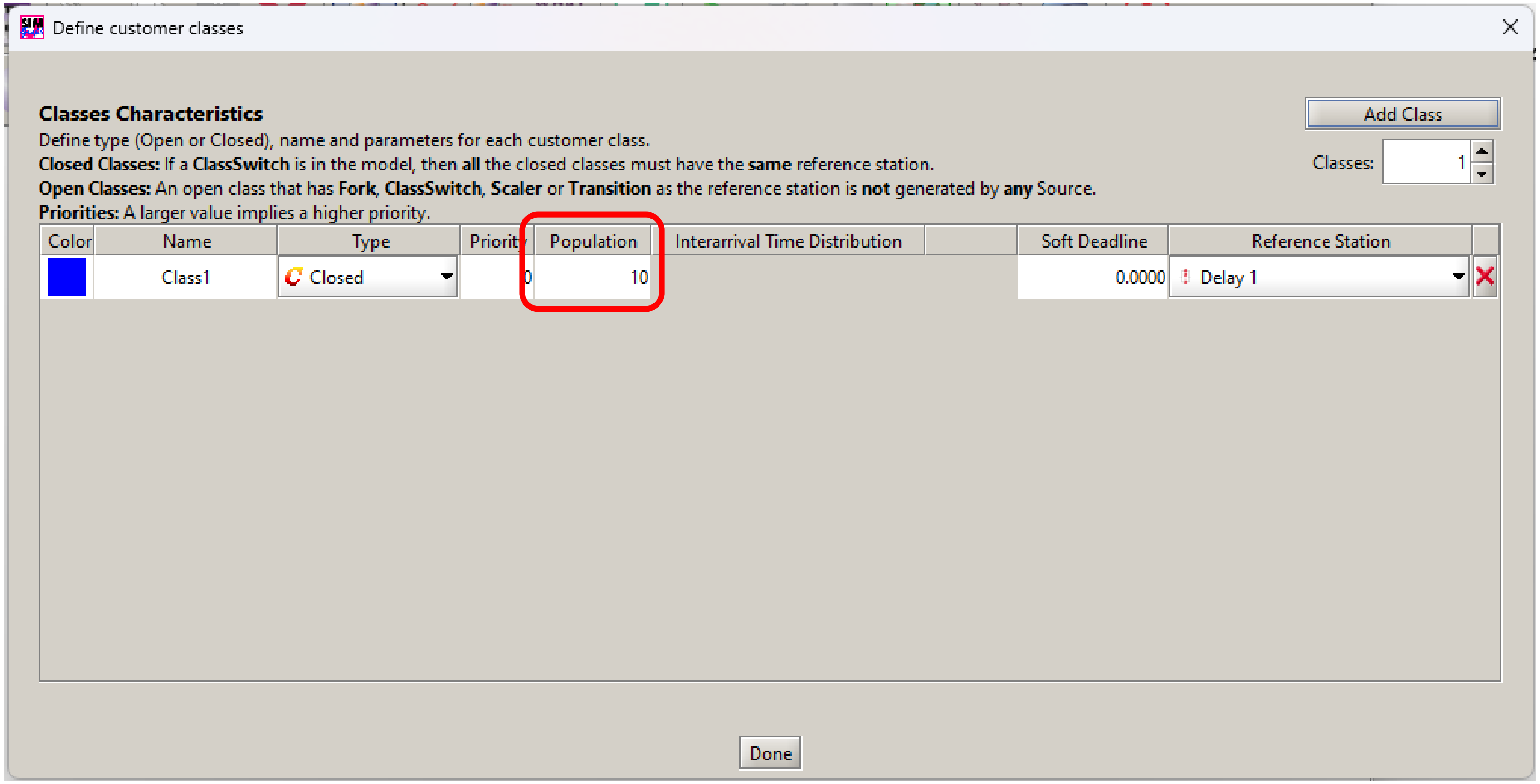
Closed systems: closed classes

The corresponding class is defined as closed in the class definition panel.



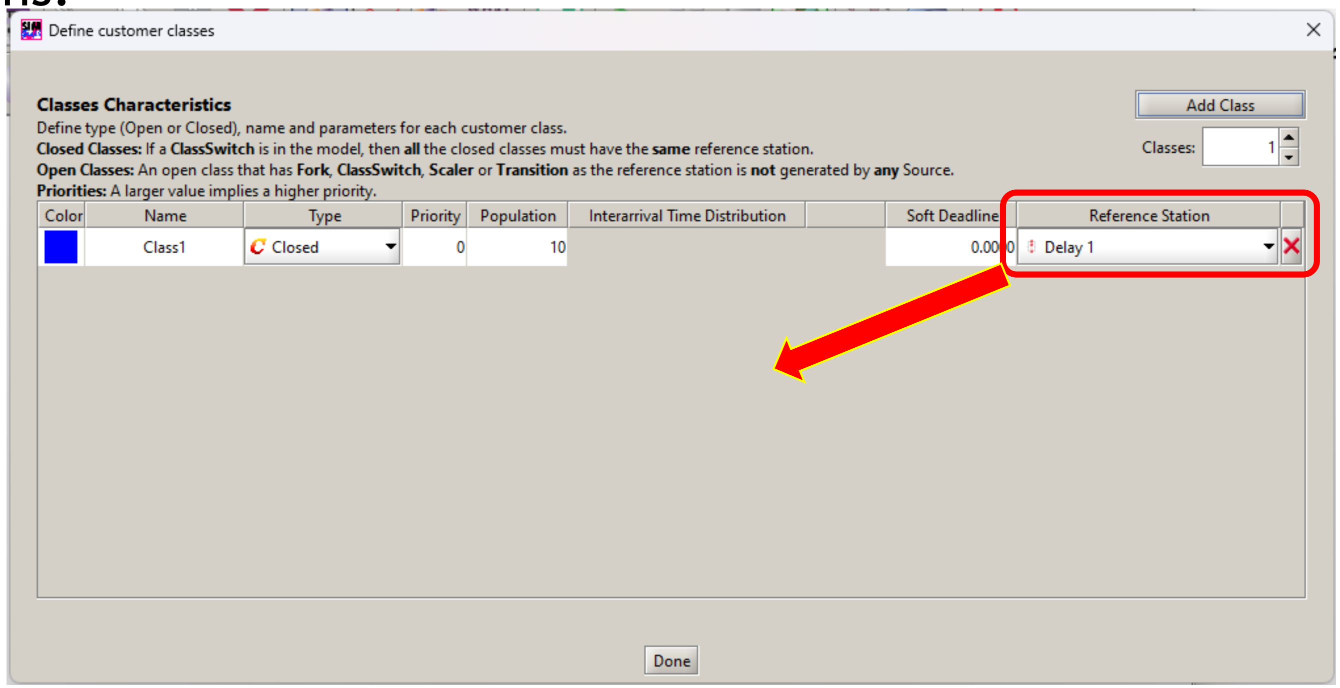
Closed systems: closed classes

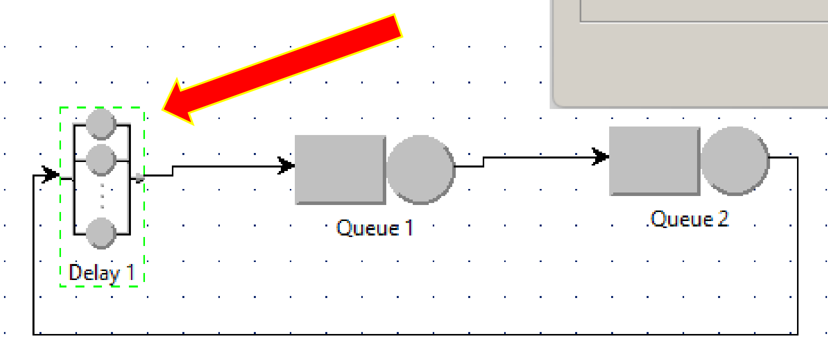
The fixed population is then specified in the corresponding field.



Closed systems: closed classes

The terminal station should be specified in the “reference station” field of the class definition. We will return on this concept in the following lessons.



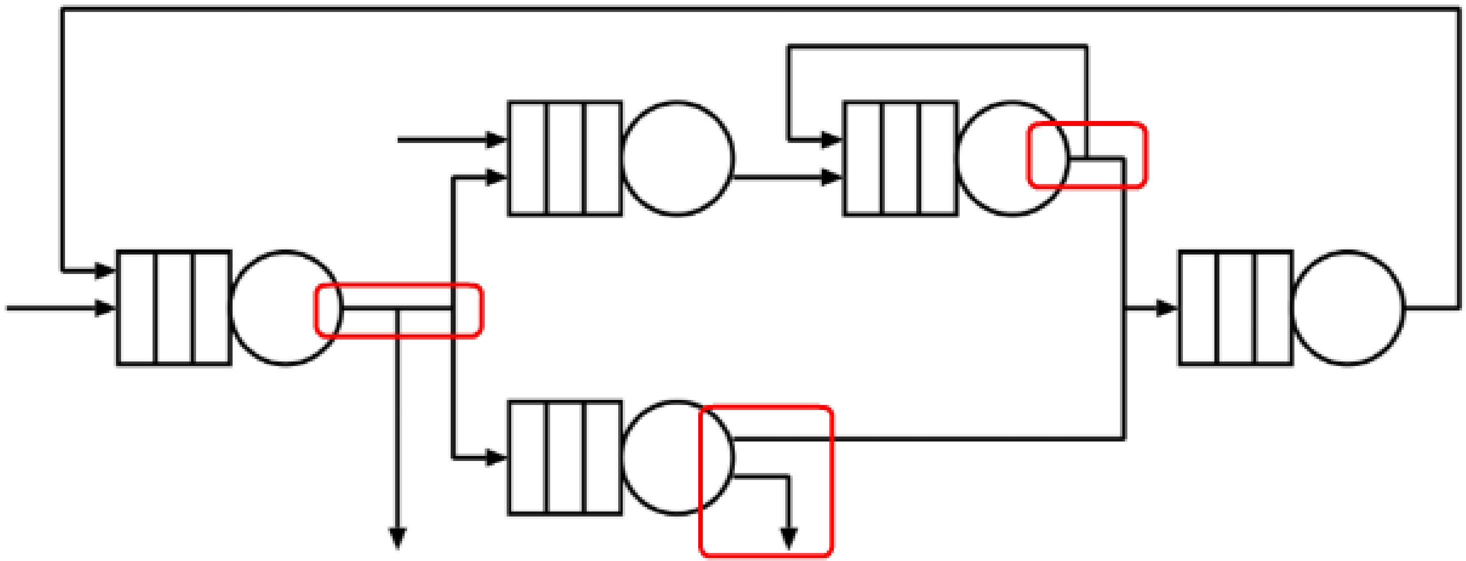


Routing

A node can be connected to more than one destination nodes.

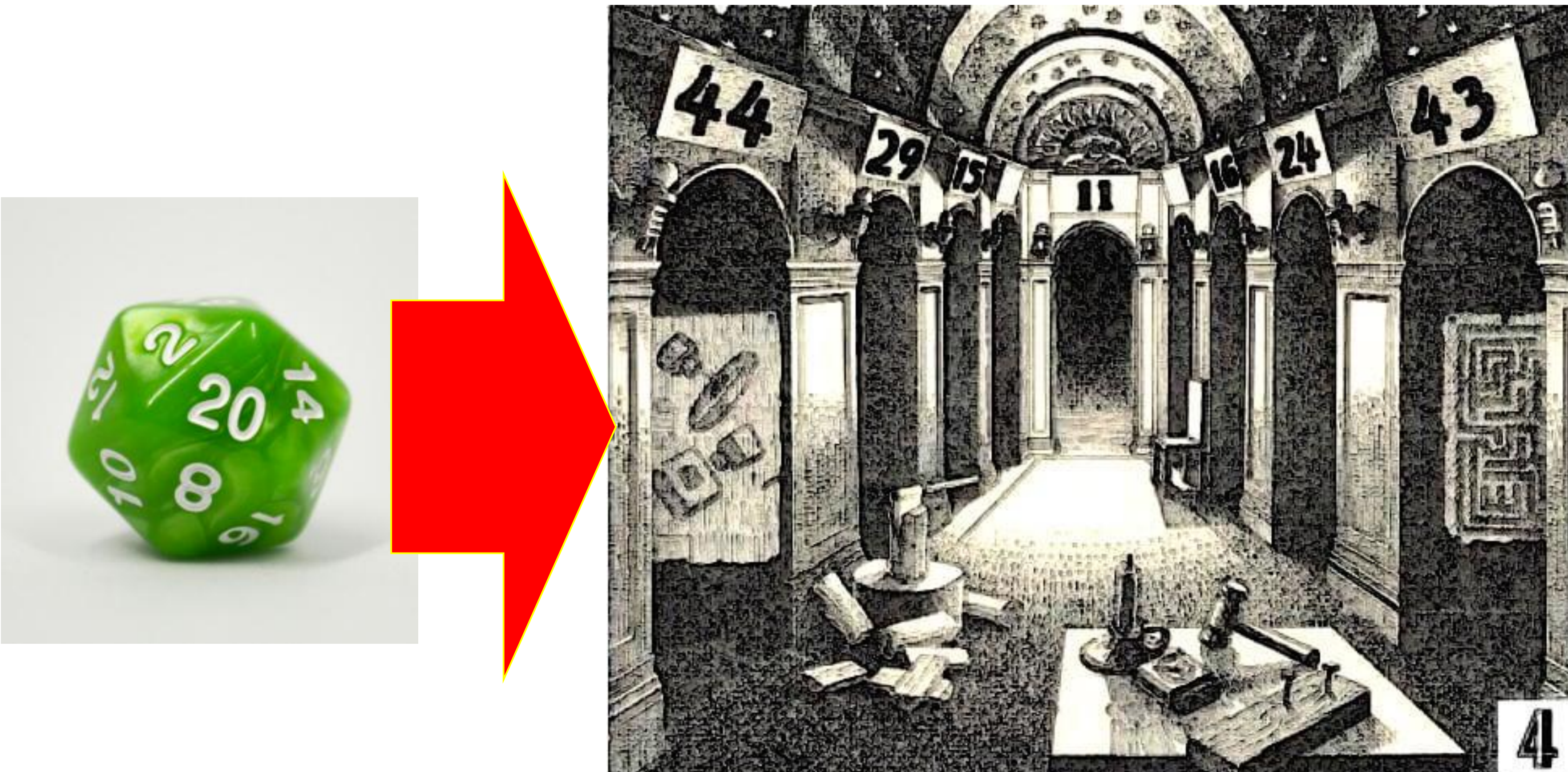
In this case, the modeler must define which service center a job finishing at one node will join.

This definition is called *Routing*.



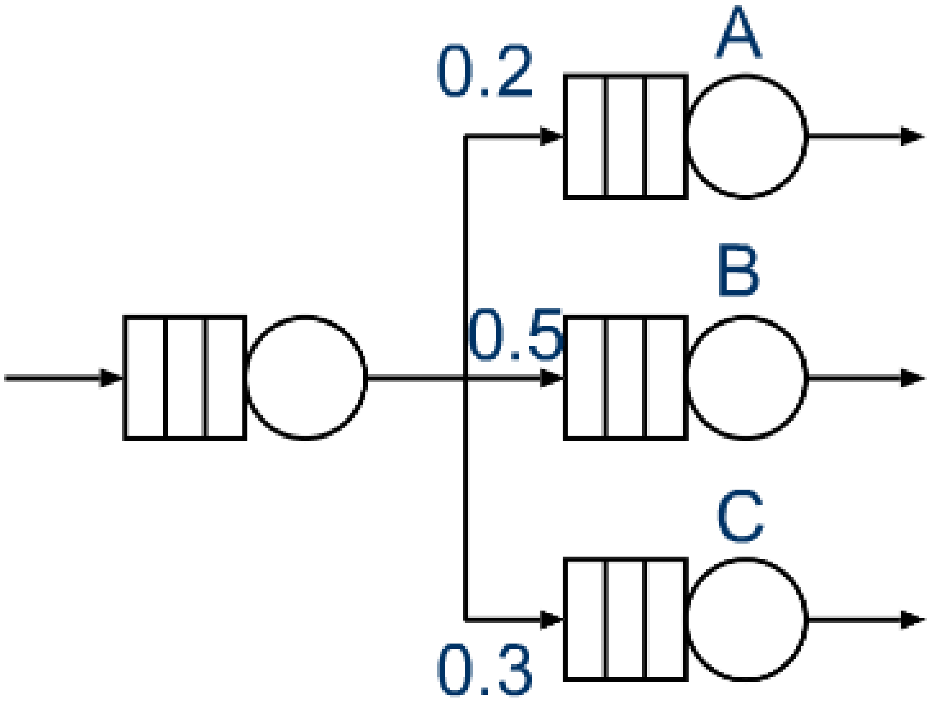
Probabilistic Routing

Several routing policies exists: we will present them in a future lesson. For the moment we focus on the *Probabilistic Routing* policy.



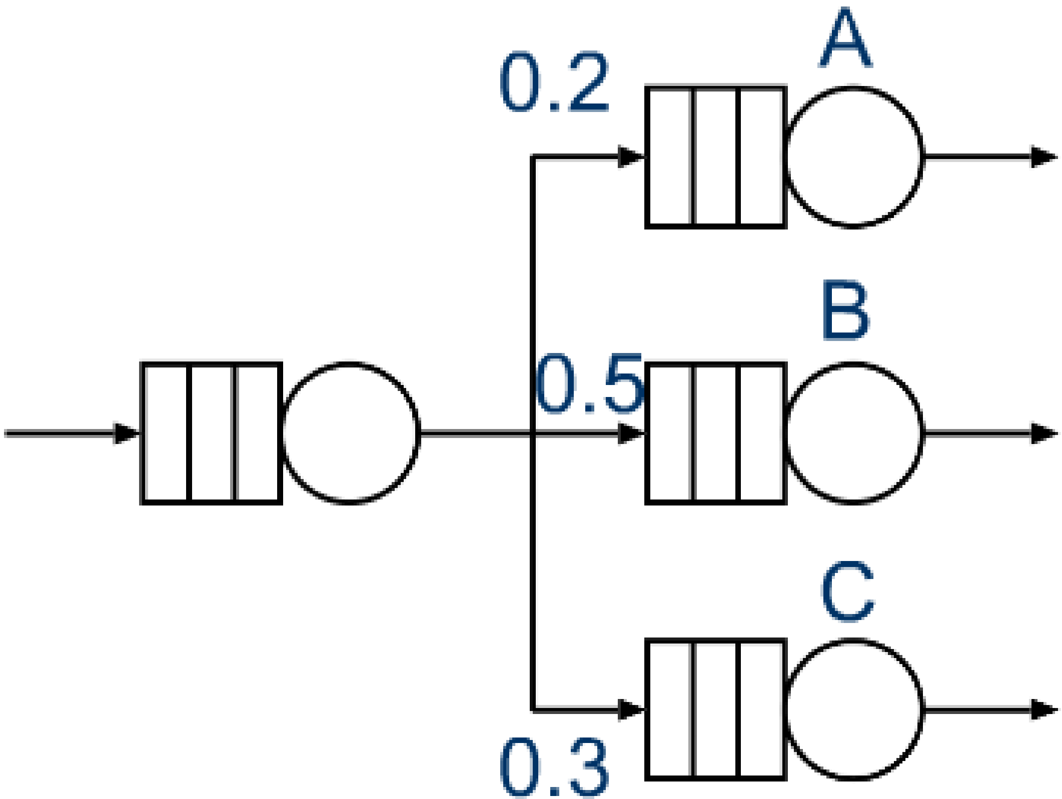
Probabilistic routing

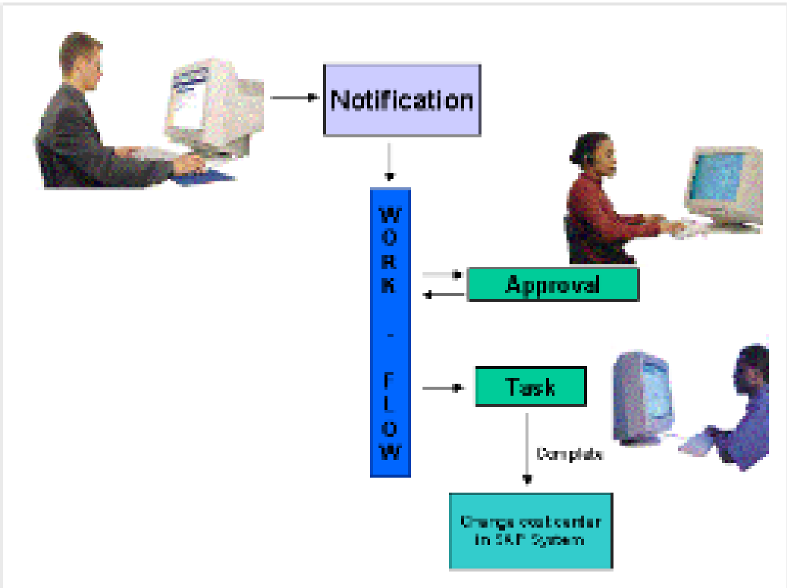
In the *probabilistic routing*, each path has assigned a probability of being chosen by the job that left the considered station.



Probabilistic routing

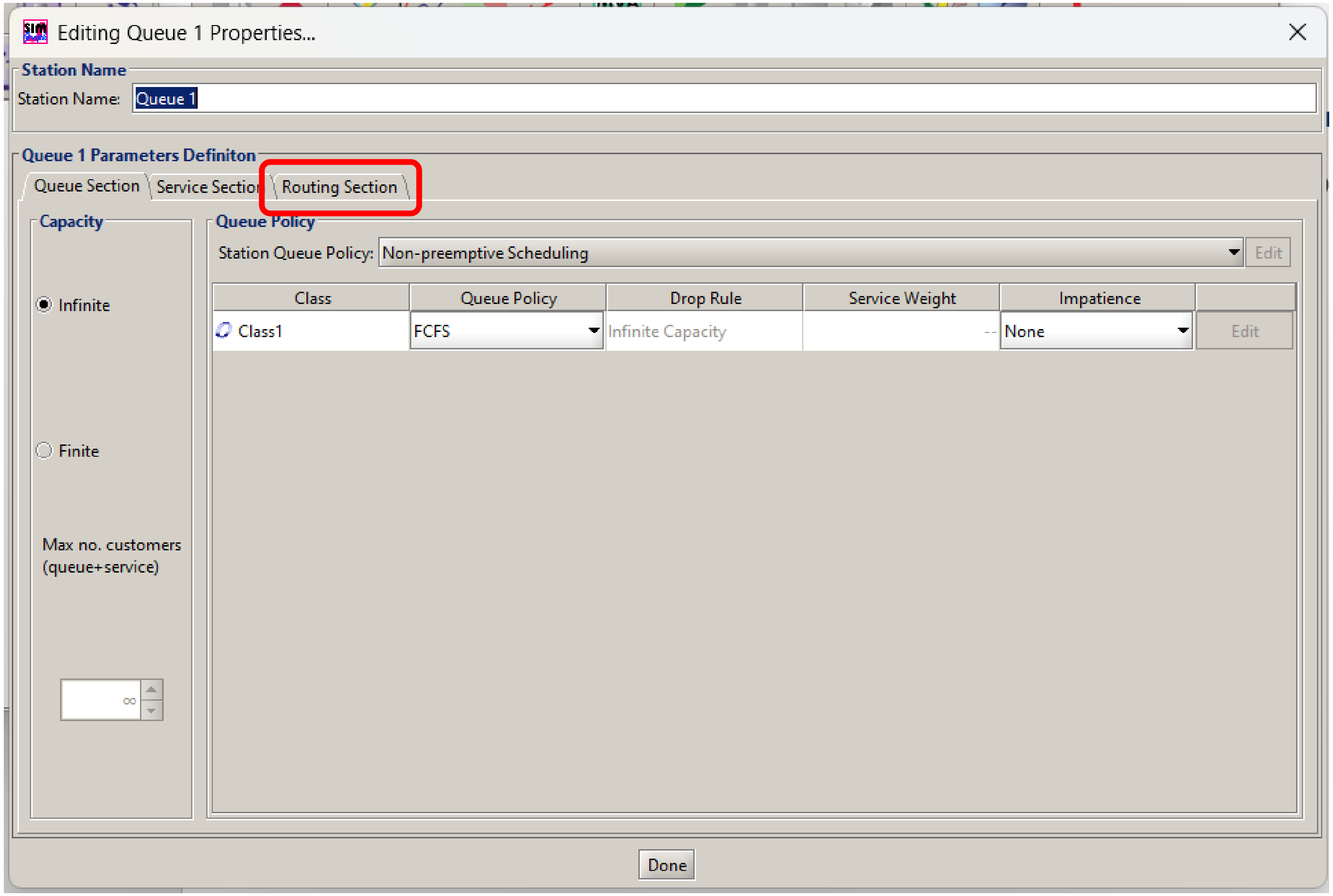
By appropriately assigning values to the probabilities associated to each possible next node, the modeler can match the flux of jobs in the considered systems.





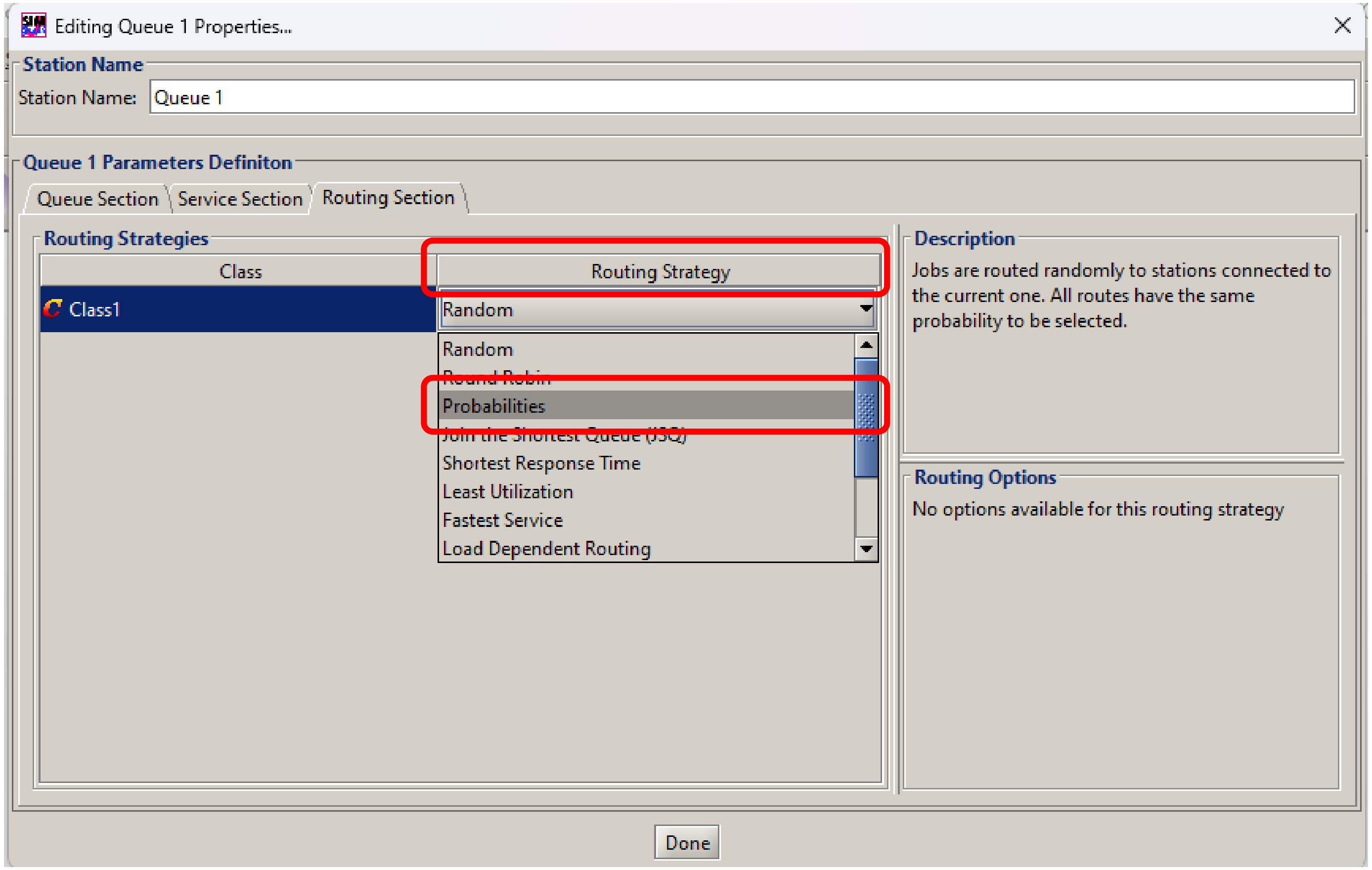
Probabilistic Routing in JMT

Sources and *Service Stations* configuration panels have a tab where routing can be defined.



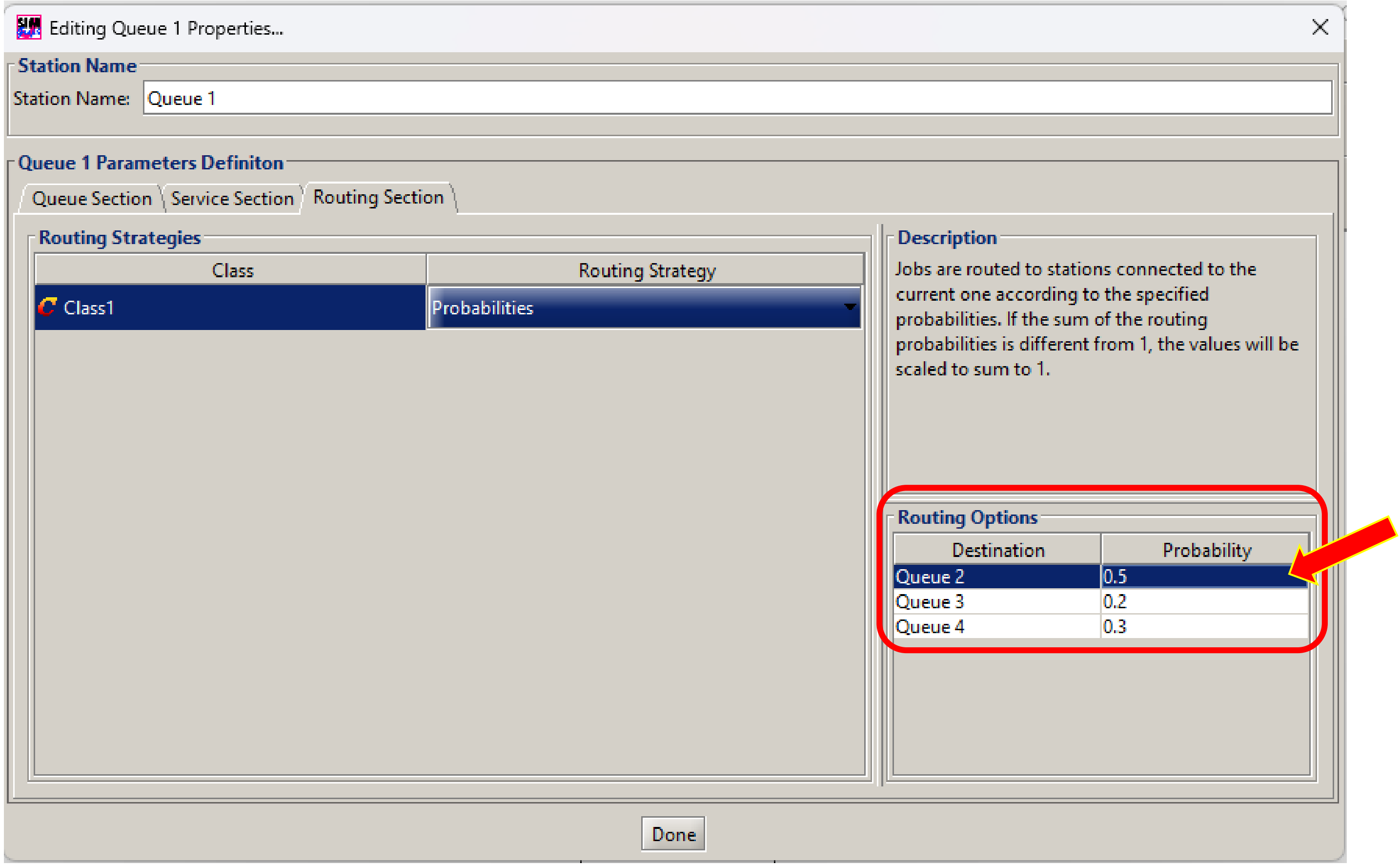
Probabilistic Routing in JMT

Probabilistic Routing can be selected in the drop-down menu in the table in the middle of the window.



Probabilistic Routing in JMT

For each possible destination, the corresponding routing probability can be specified in the table at the bottom-right of the window.



Modelling a web service

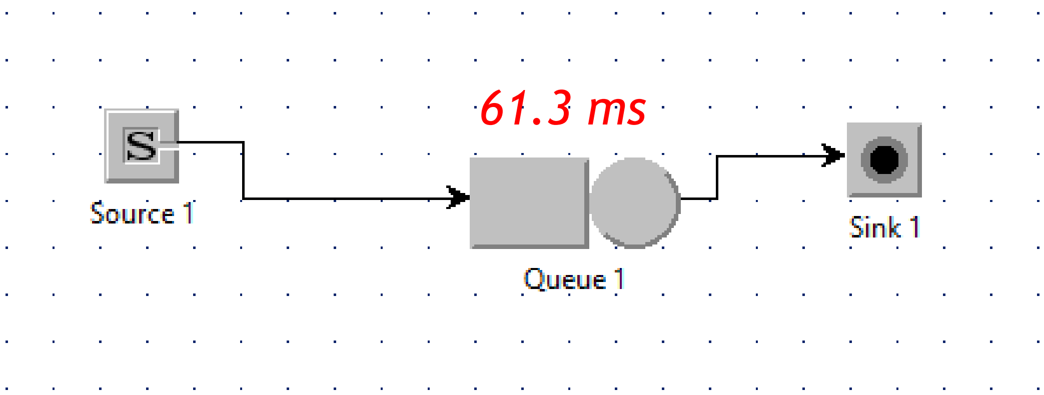
Let us consider a web service.

We study it in isolation: we send a request, and we measure how long it takes to be completed (on the average).

61.3 ms

Modelling a web service

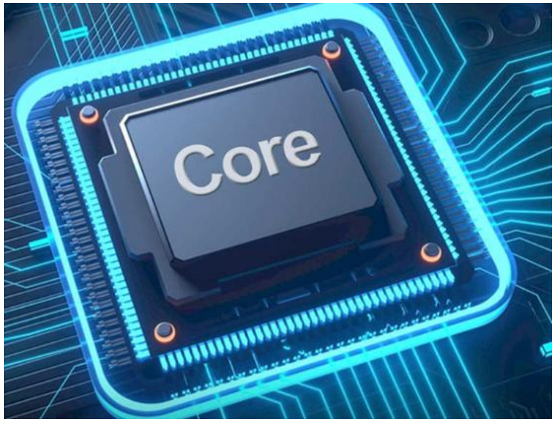
We can model the web service as a single station, whose service time corresponds to the quantity we have just measured.



Modelling a web service

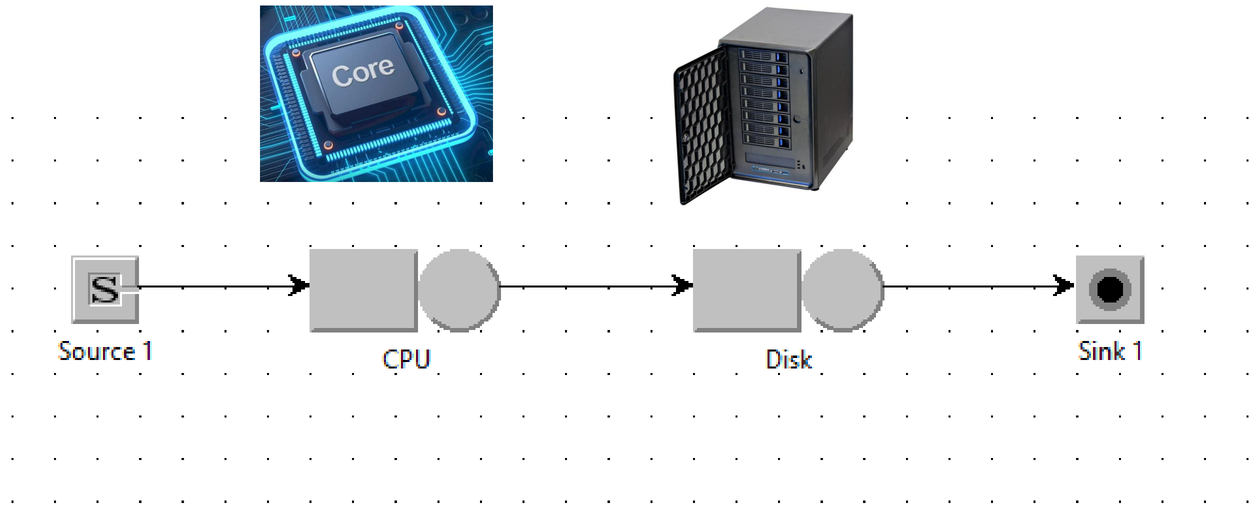
By analyzing a little bit more the real system, we find out that it is actually composed by two entities:

- A CPU
- A storage component



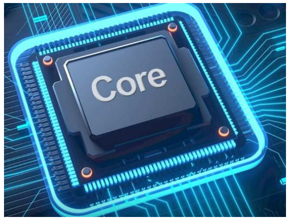
Modelling a web service

While the a Job is waiting for receiving data from the storage, another can run in parallel on the CPU, thus providing a higher throughput.



Modelling a web service

We measure again our system, this time probing the time spent in CPU and the one spent in the disk separately.

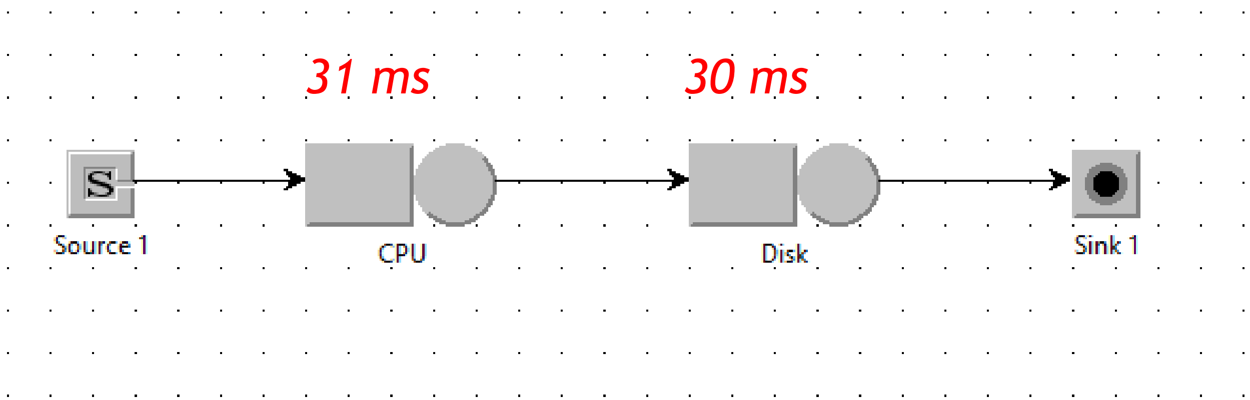


31 ms

30 ms

Modelling a web service

We then compose a tandem mode, with two Queueing Stations, representing respectively the CPU and the Storage.

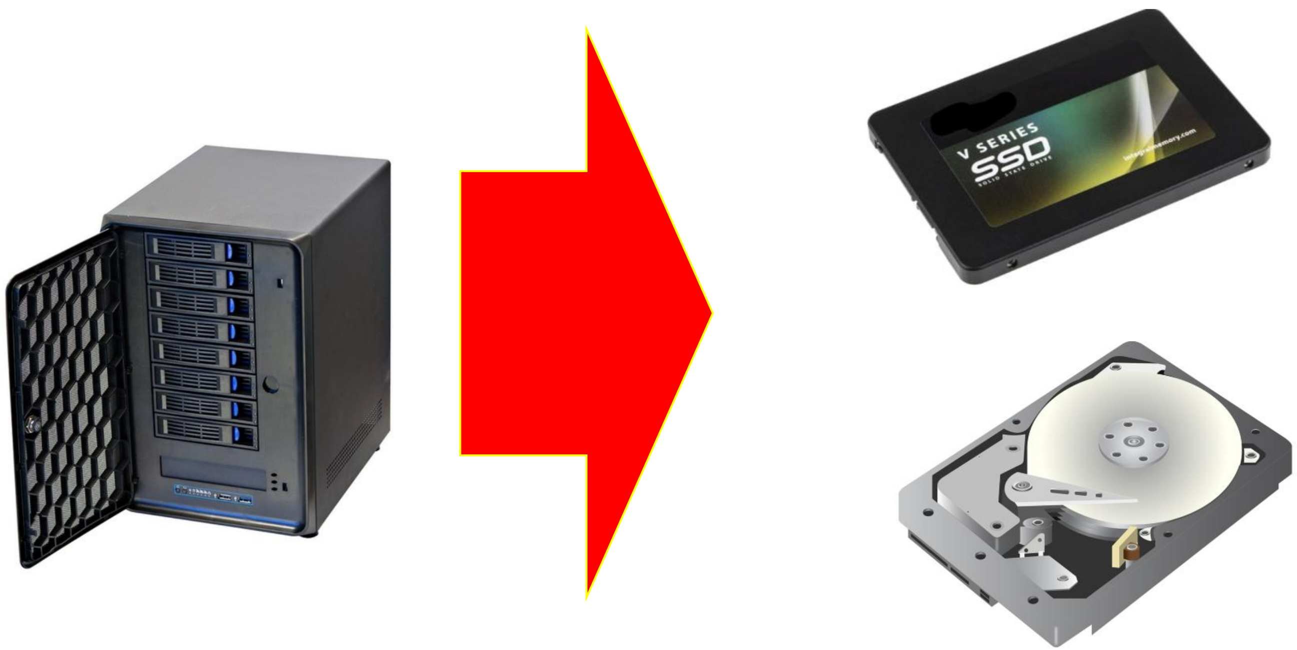


Modelling a web service

By looking even deeper in the system, we realize that the storage is indeed composed of two devices:

- An HDD for the large files
- An SSD for the smaller files

The former is very slow, and used rarely. The latter is much faster, and used more frequently.



Modelling a web service

We measure again our system, focusing on the two disks.
In particular we measure:

-> the fraction of request directed to one disk
    This tells us the routing probabilities

-> the response time of that disk, when used
    This tells us the service time of the two disks

87%

16 ms

13%

124 ms

Modelling a web service

We create a three-stations model that exploits the information we collected…