

Performance Evaluation and Applications



POLITECNICO DI MILANO



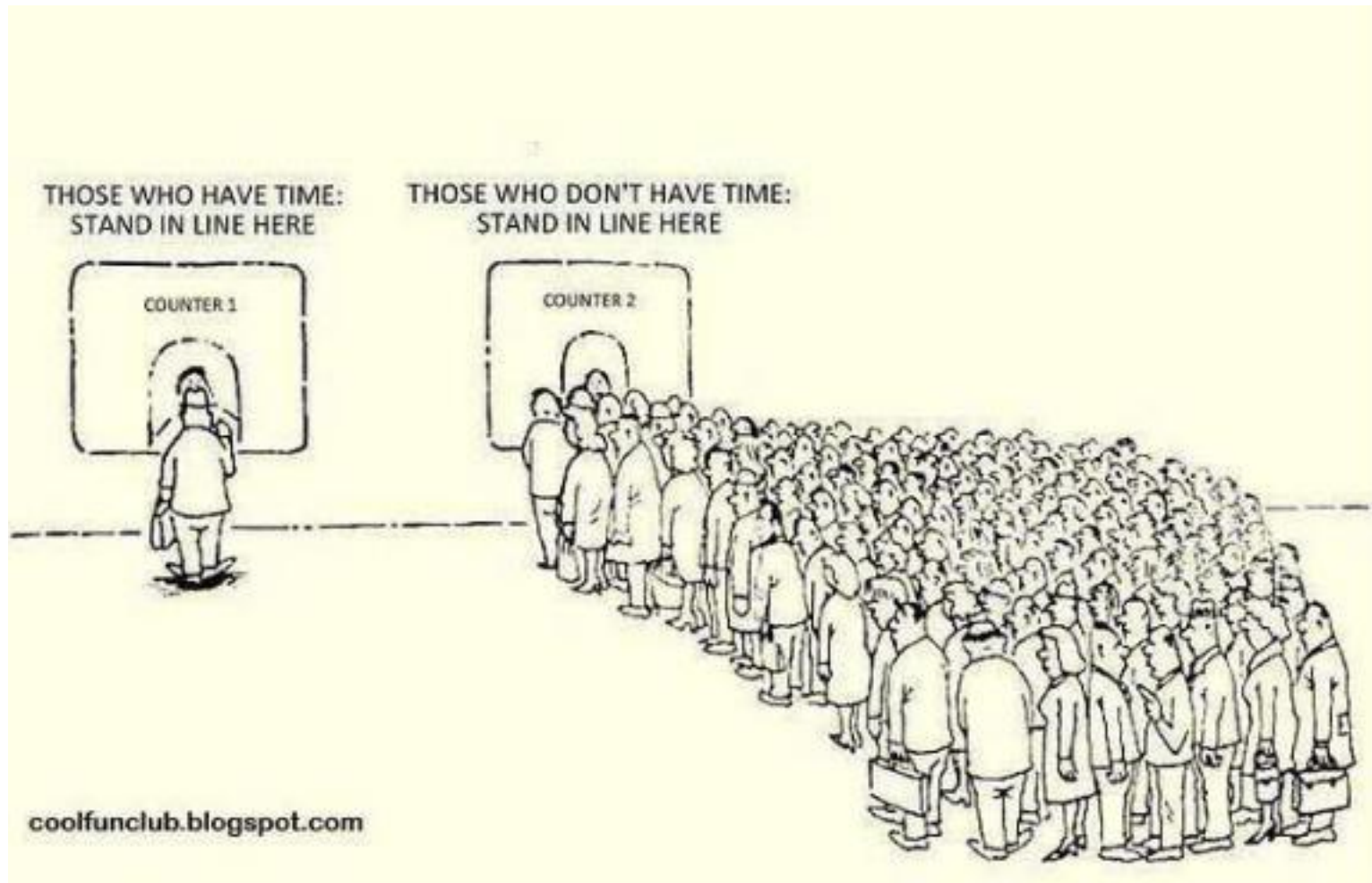
Modeling with Queueing Networks and JMT

POLITECNICO DI MILANO



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

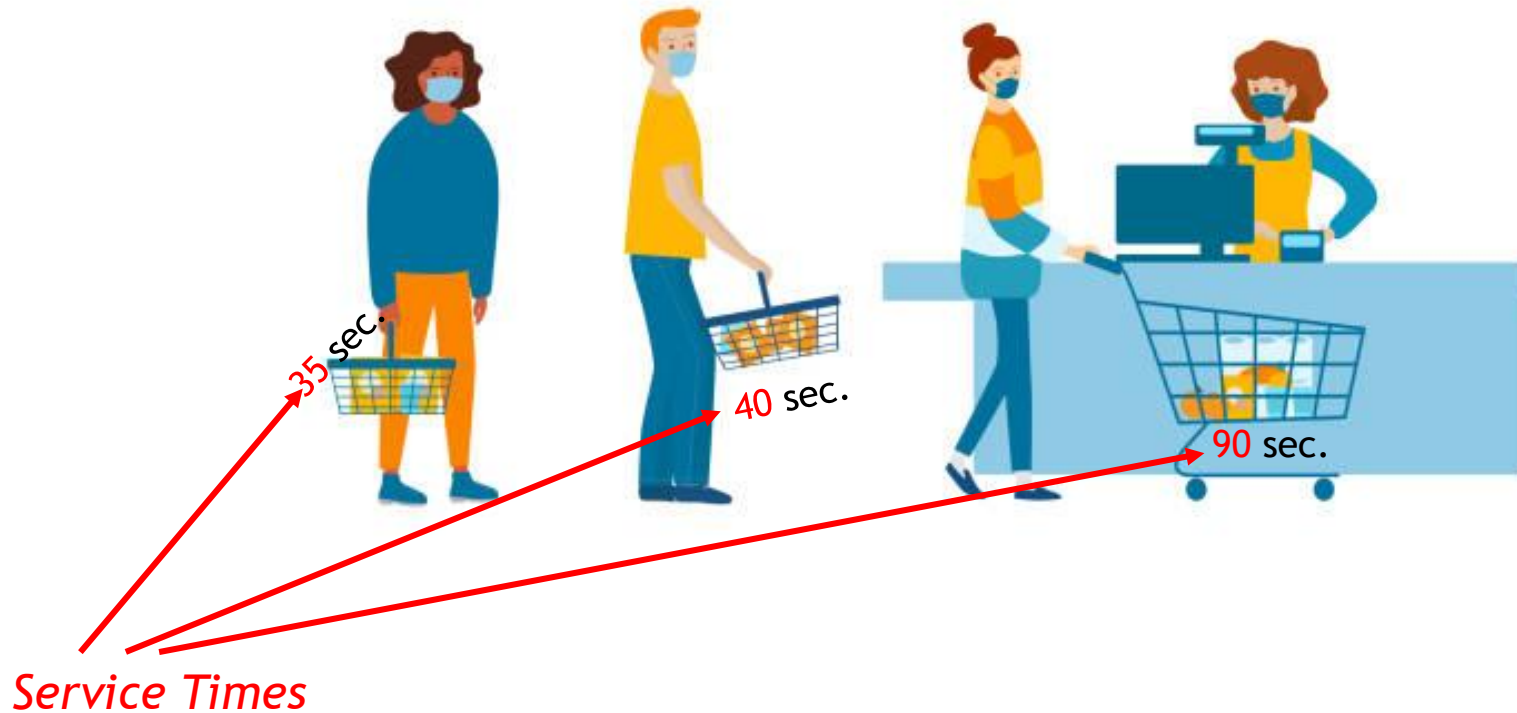
Initially we will consider two types of *Service Centers*:

- *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*.

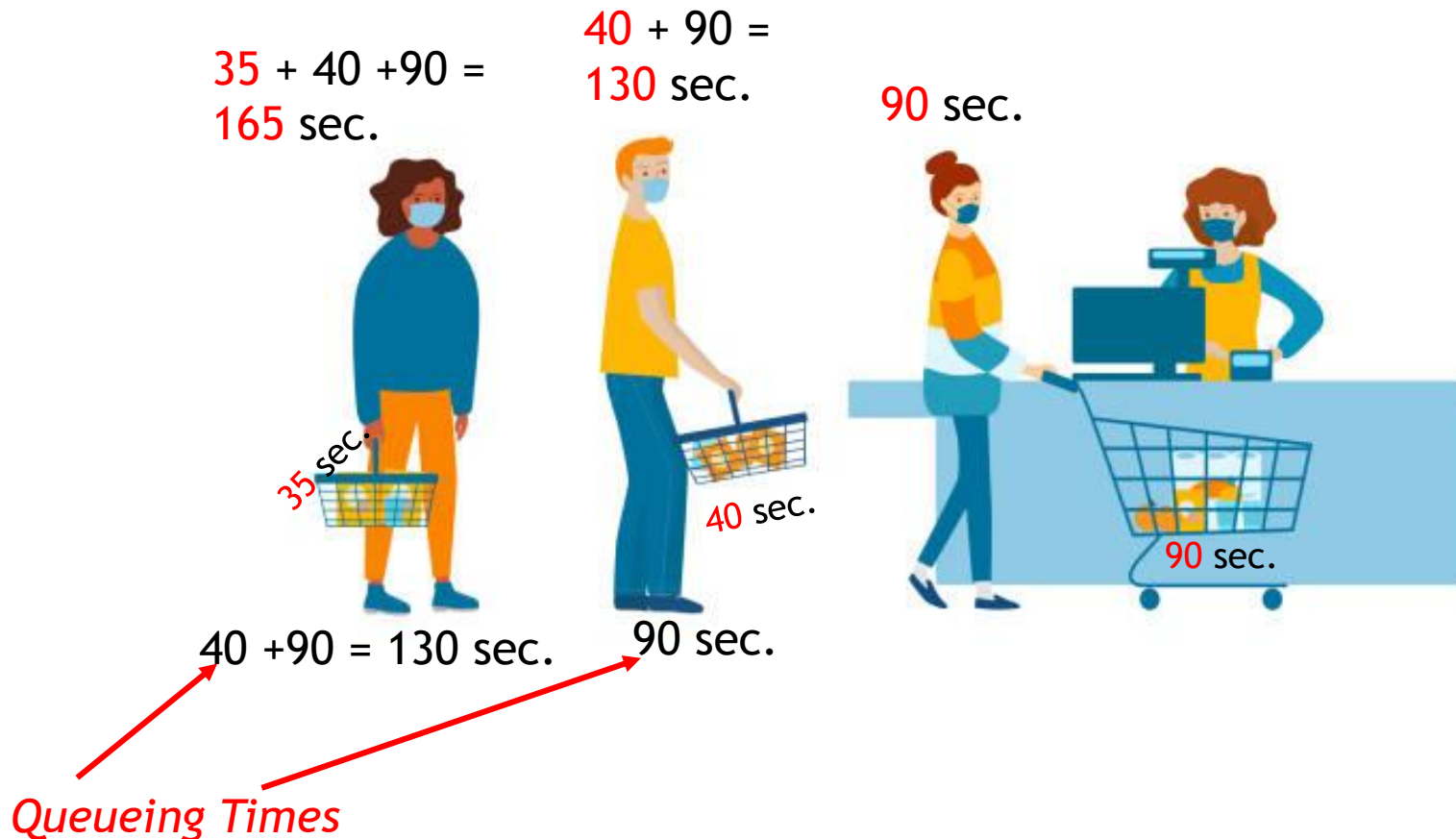
90 sec.





Queuing Networks: Queueing Stations

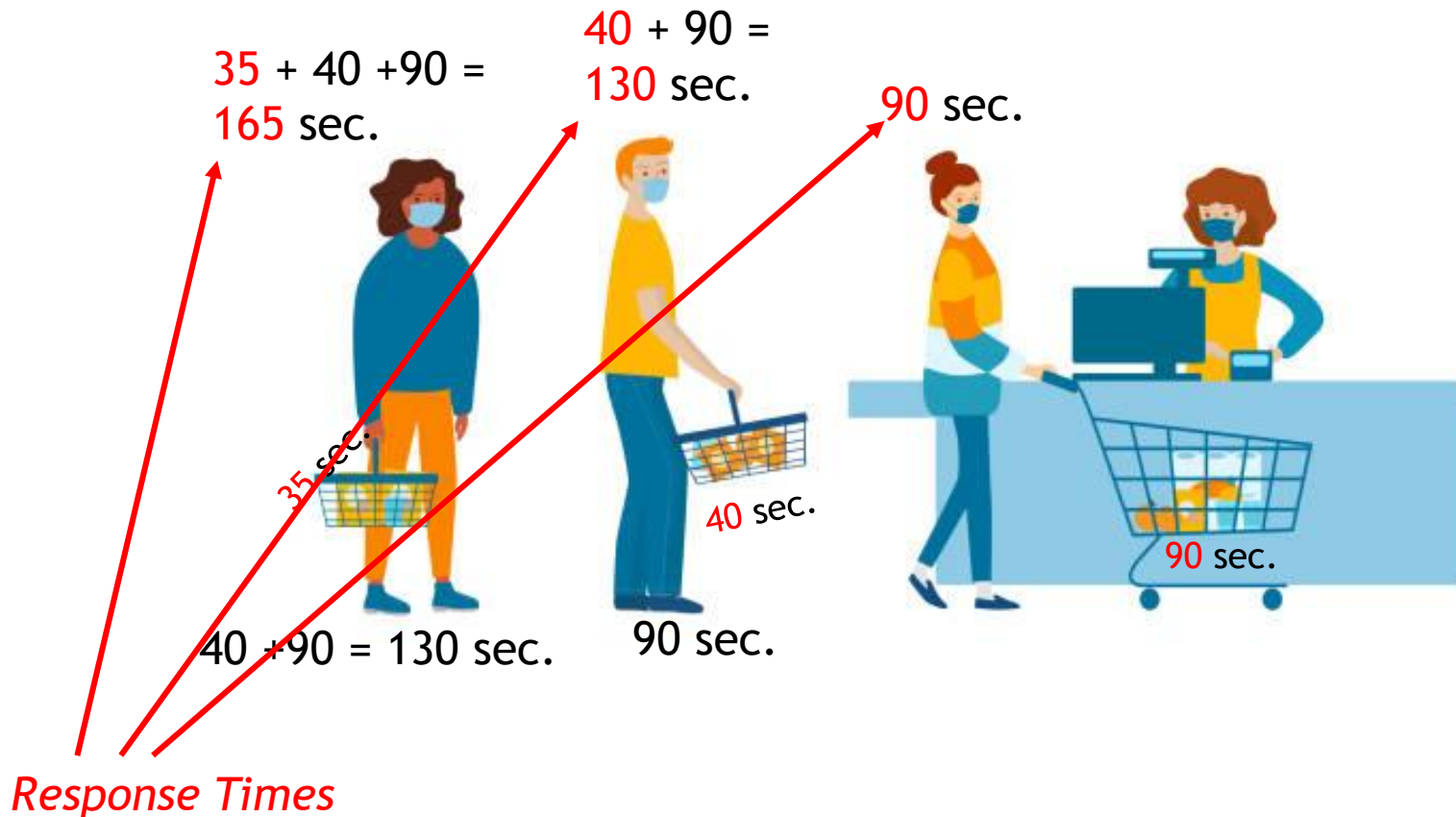
If other *Jobs* will be at same time in the considered station, then the *Service* might be delayed or interrupted. This causes some waiting time, called *Queueing Time*.





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.

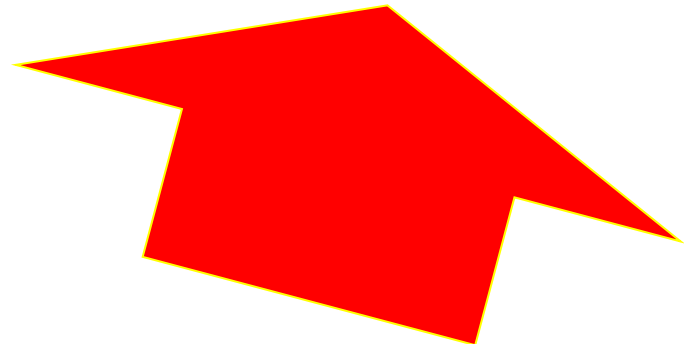
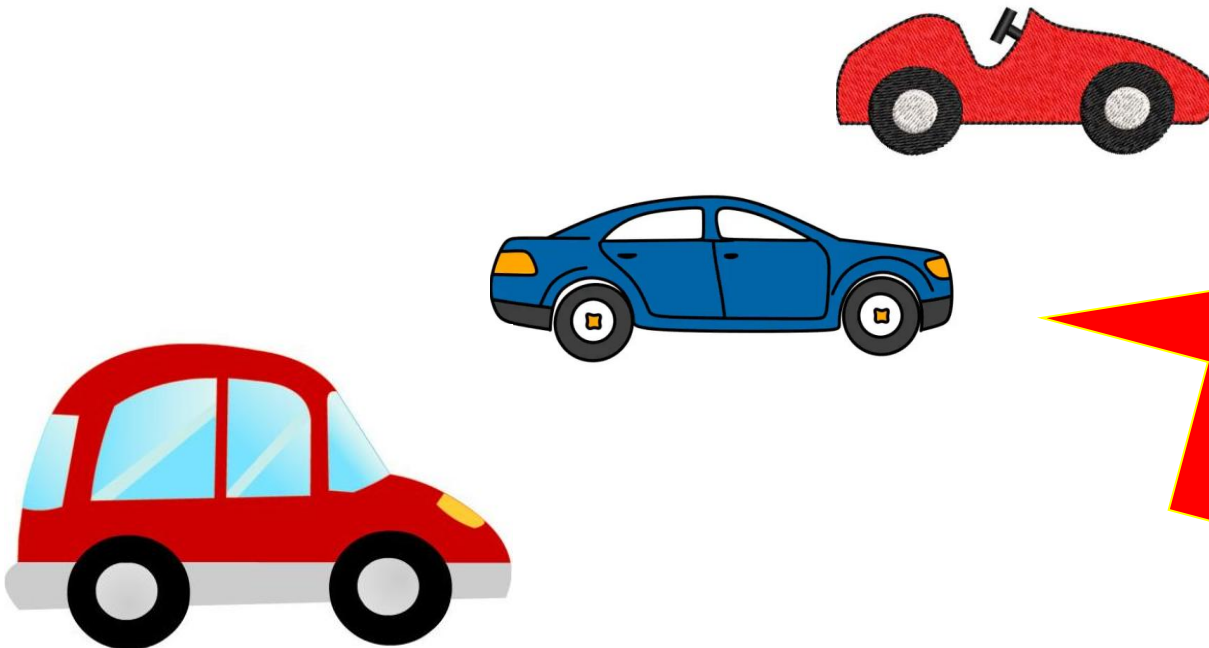
No Queue can happen at a *Delay Station*, and the *Response Time* always correspond to its *Service Time*.





Queuing Networks: Arrivals

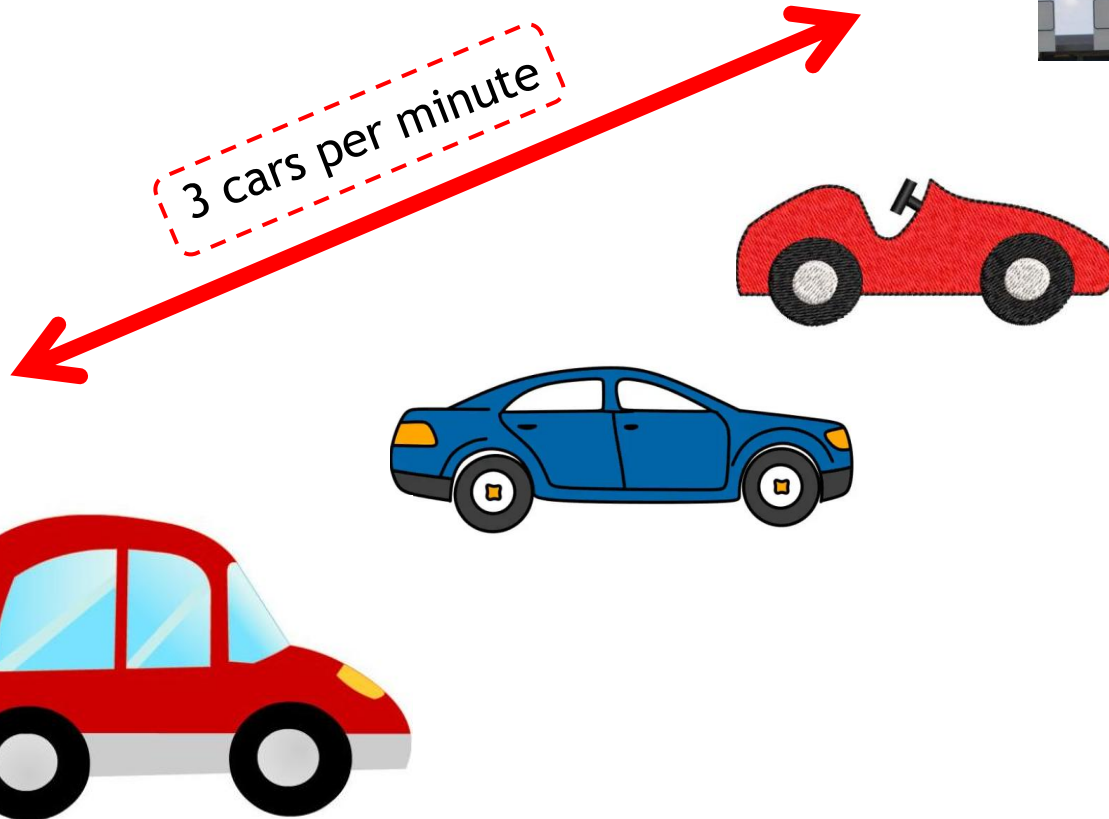
Arrivals model the entrance of new jobs in the *System*.





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

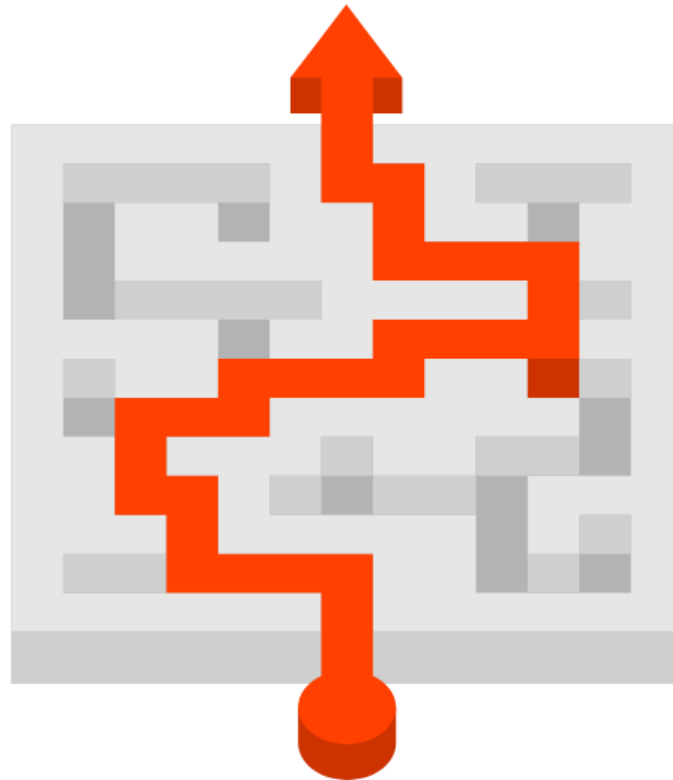
If the system is *Unstable*, then the *Queueing Time* increases and it tends to the *Infinity*. We will return on *Stability* in later in the course.





Queuing Networks: Departures

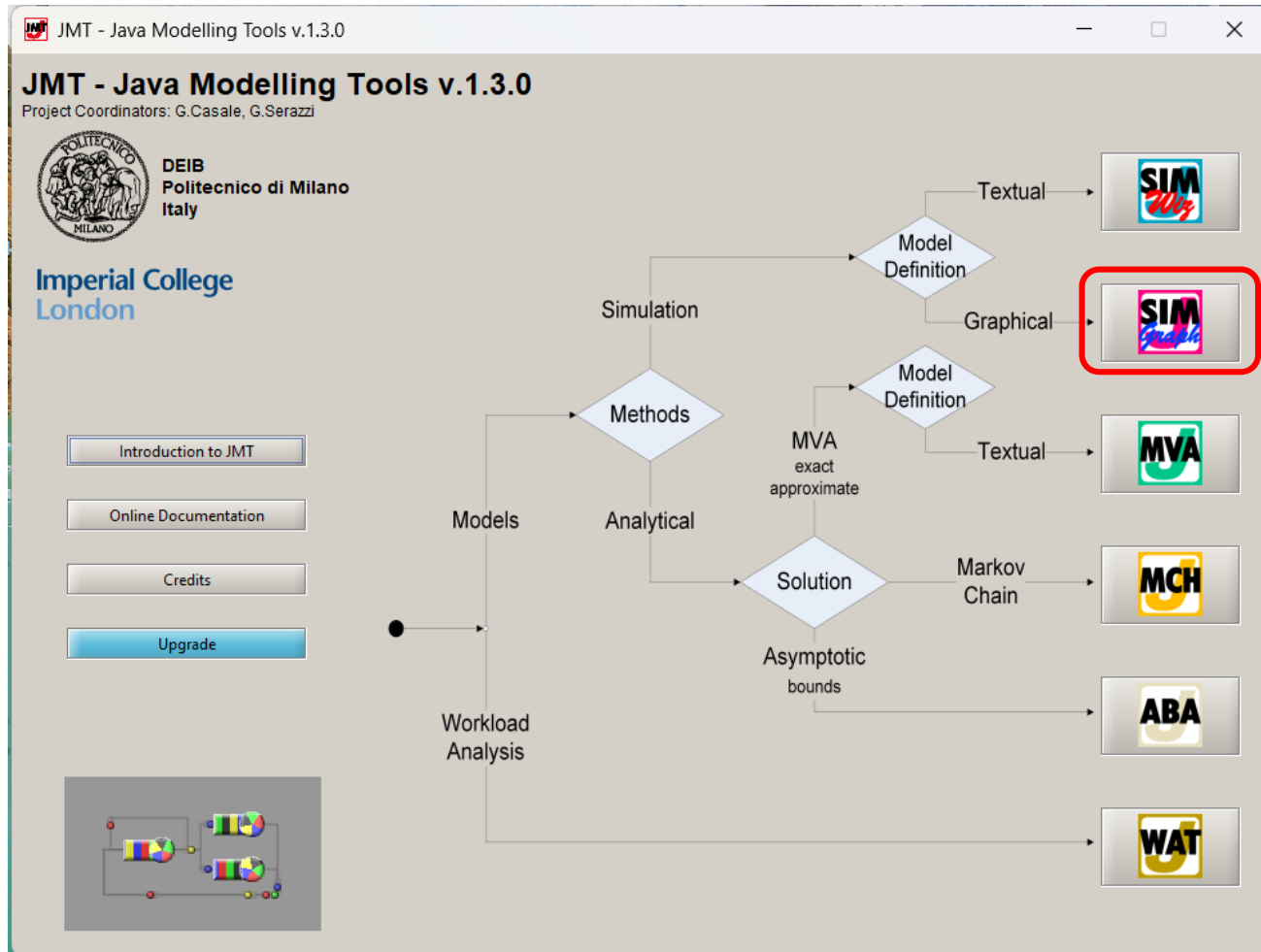
When a *Job* completes its service, it leaves the *System*. This event is called *Departure*.





Queuing Networks: JMT

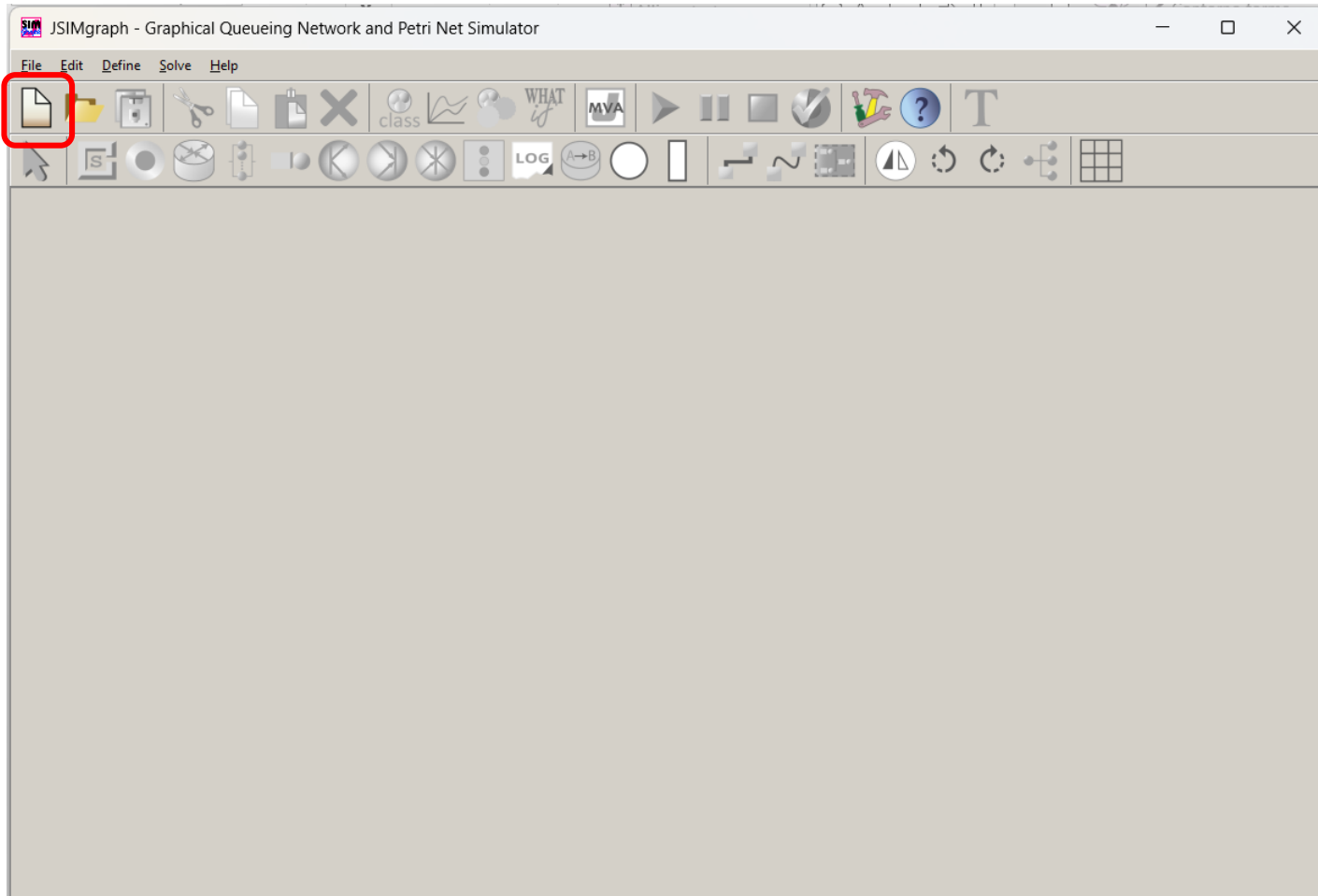
JMT allows to consider Queueing Networks in several components. We will first focus on the *JSimGraph Tool*.





Queuing Networks: JMT

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

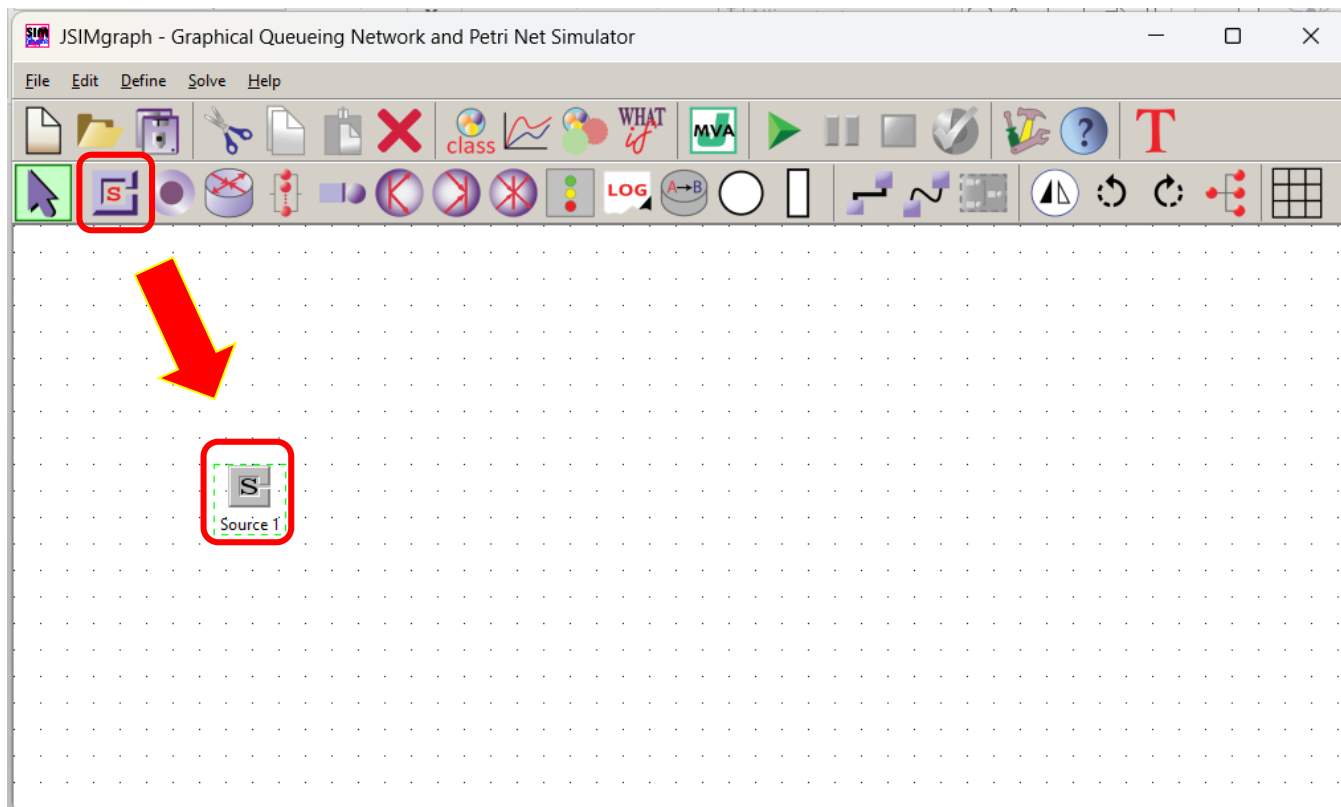




JMT: Arrivals

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

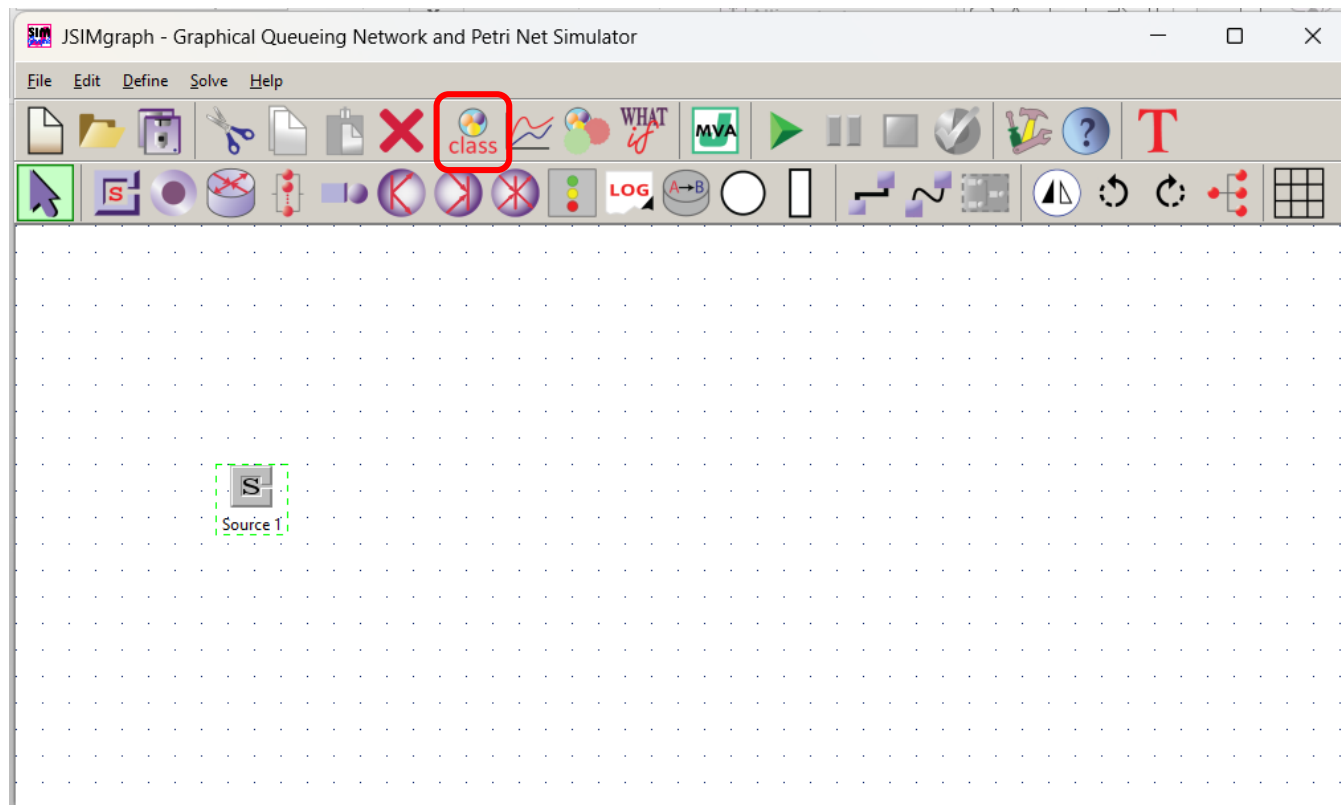
After pressing the Source button, the element is placed by clicking the mouse in the desired position.





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.

Define customer classes

Classes Characteristics
Define type (Open or Closed), name and parameters for each customer class.
Closed Classes: If a **ClassSwitch** is in the model, then **all** the closed classes must have the **same** reference station.
Open Classes: An open class that has **Fork**, **ClassSwitch**, **Scaler** or **Transition** as the reference station is **not** generated by **any** Source.
Priorities: A larger value implies a higher priority.

Add Class

Classes: 1

Color	Name	Type	Priority	Population	Interarrival Time Distribution	Soft Deadline	Reference Station
Blue	Class1	Open	0		exp(0.5)	0.0000	Source 1

Edit

Done



JMT: Arrival Rate

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

Define customer classes

Classes Characteristics
Define type (Open or Closed), name and parameters for each customer class.
Closed Classes: If a **ClassSwitch** is in the model, then **all** the closed classes must have the **same** reference station.
Open Classes: An open class that has **Fork**, **ClassSwitch**, **Scaler** or **Transition** as the reference station is **not** generated by **any** Source.
Priorities: A larger value implies a higher priority.

Classes: 1

Color	Name	Type	Priority	Population	Interarrival Time Distribution		Soft Deadline	Reference Station
Blue	Class1	Open	0		exp(0.5)	Edit	0.0000	Source 1

Done



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.

Editing Class1 distribution...

Selected Distribution: Exponential

Exponential $[\exp(\lambda)]$:

$$f(x) = \lambda e^{-\lambda x}$$

λ : 0.5

mean: 2

OK Cancel



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.

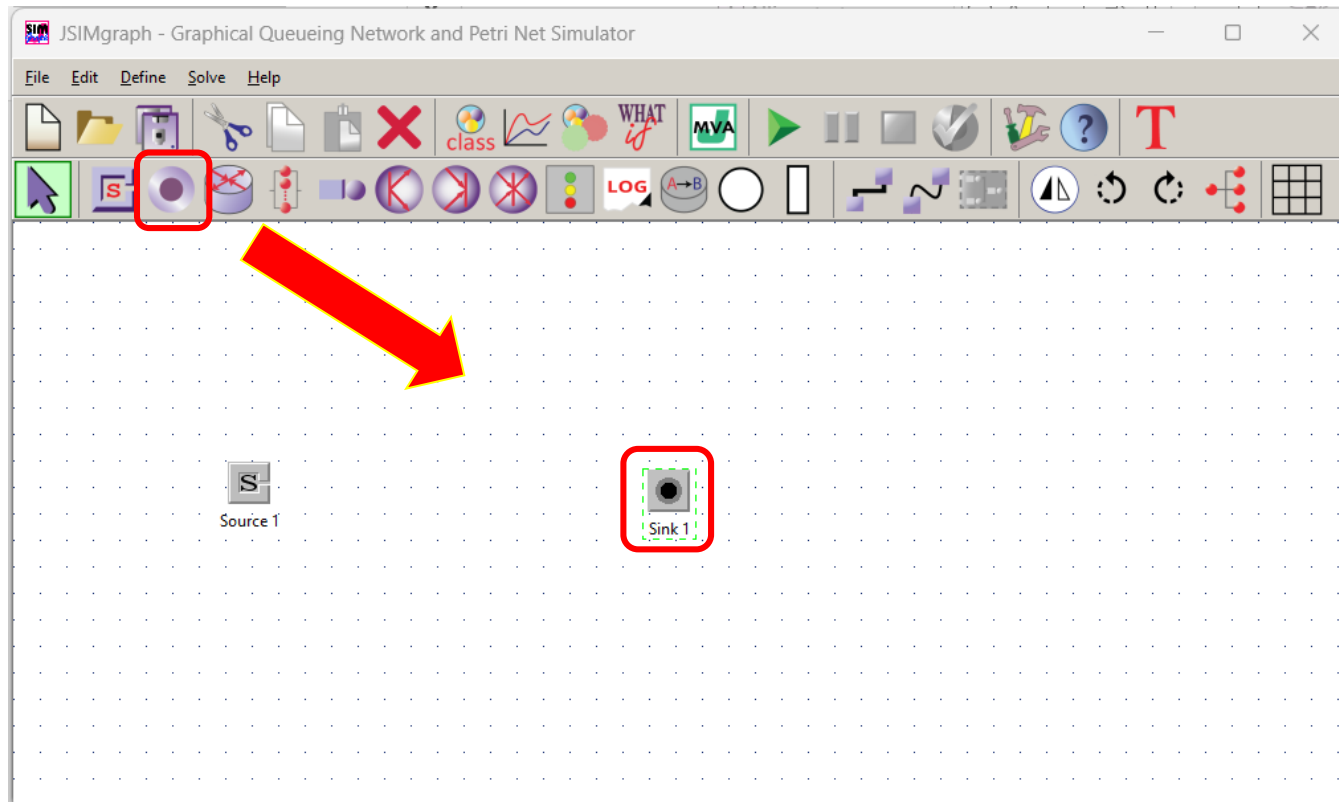
The only important requirement, is that all the temporal specification, in all the parameters considered by the tool, match the same units.



JMT: Departures

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

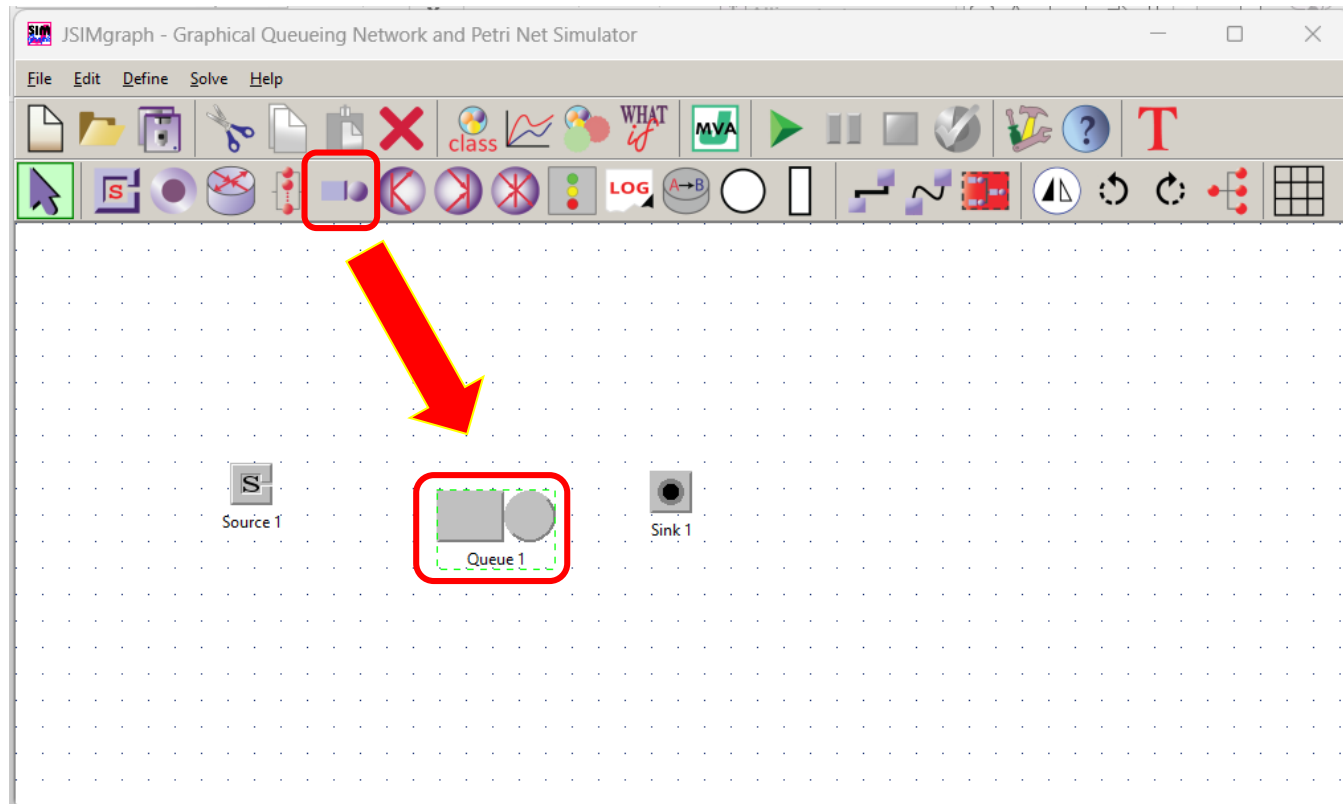
Again, the element is placed by clicking the mouse in the desired position.





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.

JSIMgraph - Graphical Queueing Network and Petri Net Simulator

File Edit Define Solve Help

class WHAT if LOG

Source 1

Queue 1

Editing Queue 1 Properties...

Station Name: Queue 1

Queue 1 Parameters Definition

Queue Section Service Section Routing Section

Capacity

☒ Infinite

Max no. customers (queue+service)

Queue Policy

Station Queue Policy: Non-preemptive Scheduling

Class	Queue Policy	Drop Rule	Service Weight	Impatience	
Class1	FCFS	Infinite Capacity	--	None	Edit

Done



JMT: Queueing Stations

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

Editing Queue 1 Properties...

Station Name: Queue 1

Queue 1 Parameters Definition

Queue Section | **Service Section** | Routing Section

Capacity

☒ Infinite

☐ Finite

Max no. customers (queue+service)

Queue Policy

Station Queue Policy: Non-preemptive Scheduling [Edit]

Class	Queue Policy	Drop Rule	Service Weight	Impatience	
Class1	FCFS	Infinite Capacity	--	None	[Edit]

Done



JMT: Queueing Stations

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

Editing Queue 1 Properties...

Station Name: Queue 1

Queue 1 Parameters Definition

Queue Section Service Section Routing Section

Server Configuration

Number of Servers: 1 Advanced Features: Edit

Service Time Distributions

Class	Strategy	Service Time Distribution	
Class1	Load Independent	exp(1)	Edit

Done



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.

Editing Class1 distribution...

Selected Distribution: Exponential

Exponential [exp(λ)]:

$$f(x) = \lambda e^{-\lambda x}$$

λ : 0.5

mean: 2

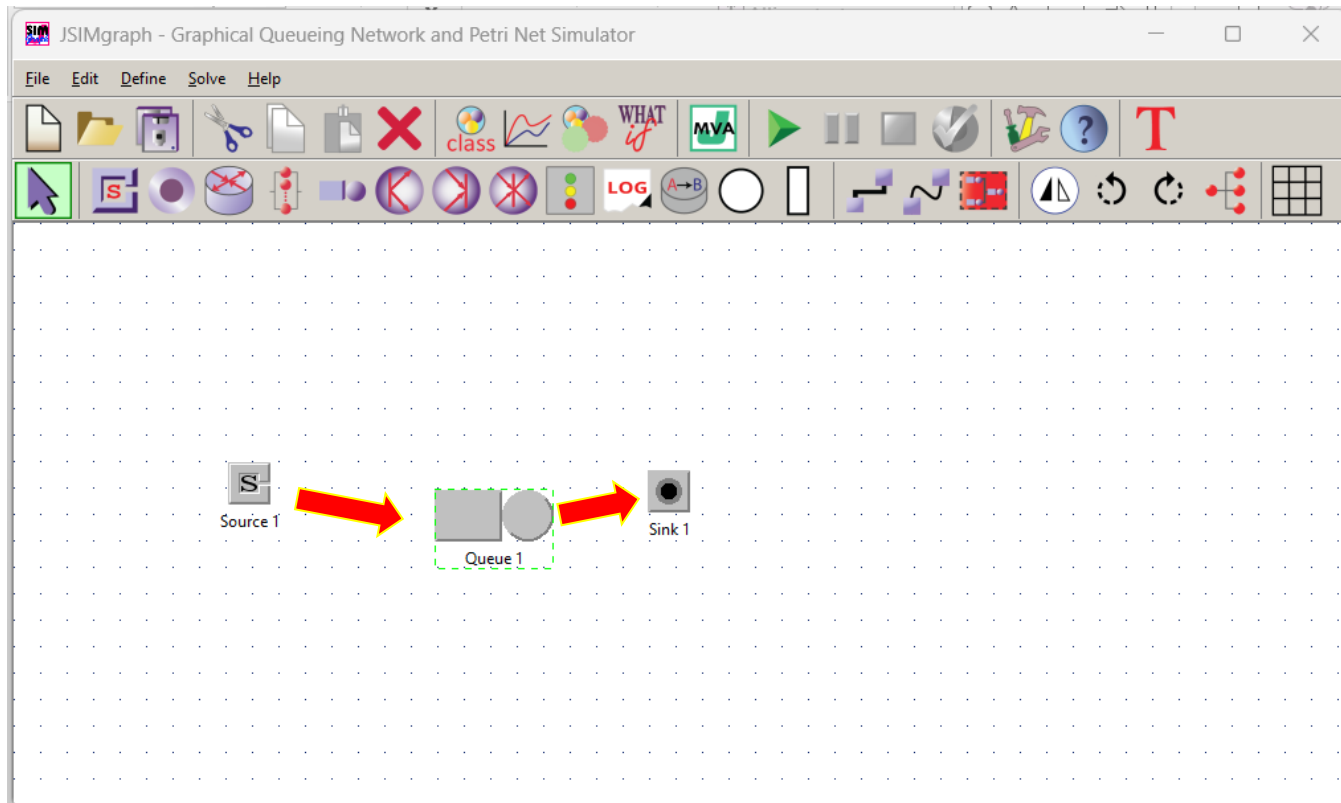
OK Cancel



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

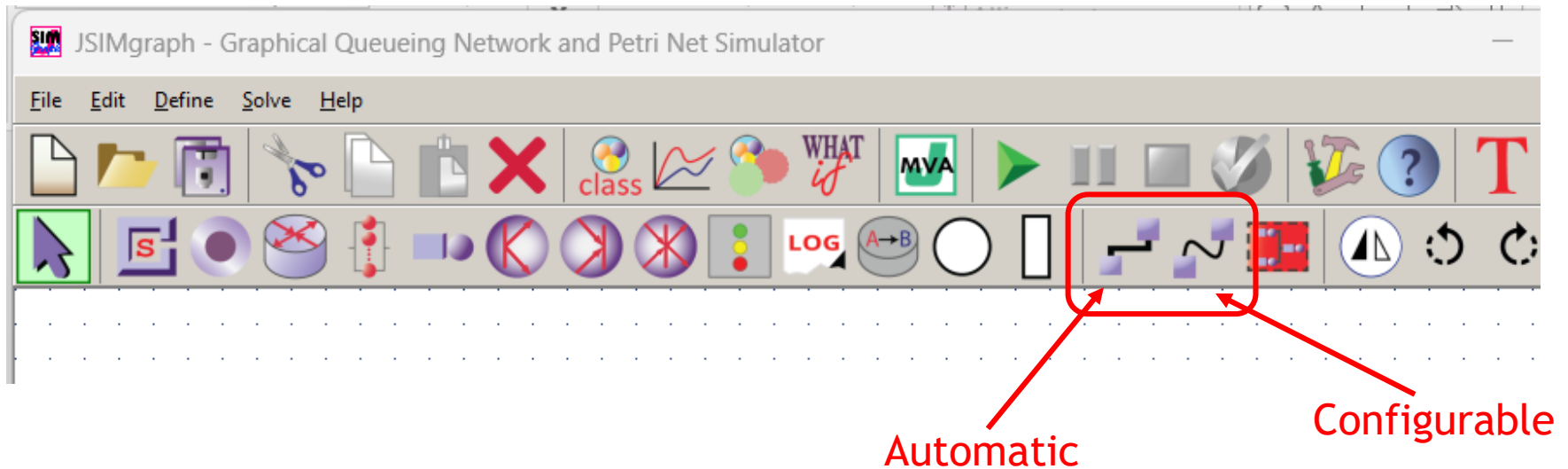


JMT: Connections

Two types of arcs are available:

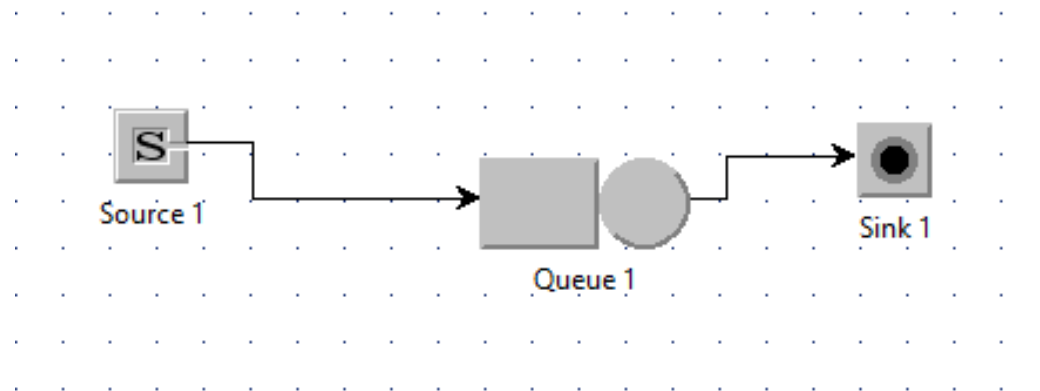
- Automatic Arcs
- Configurable Arcs

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



JMT: Connections

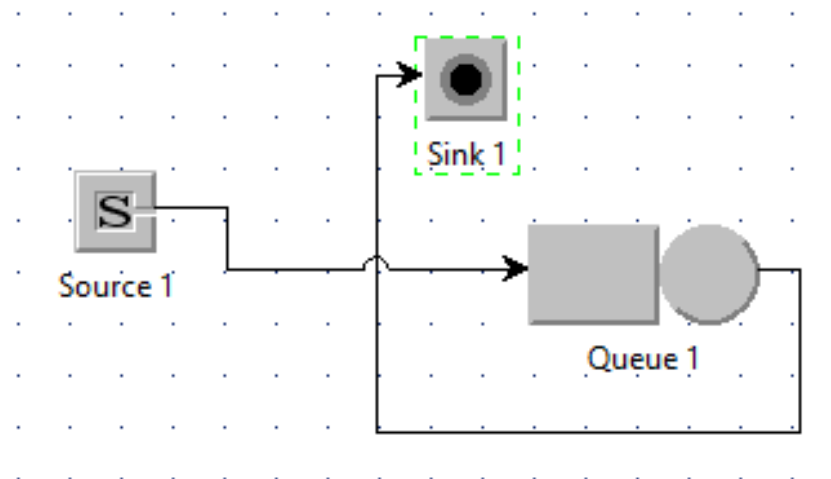
Automatic arcs, are composed of lines always parallel to the borders of the screen, and are entered by clicking first on the starting node, then releasing the mouse press on the destination.





JMT: Connections

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

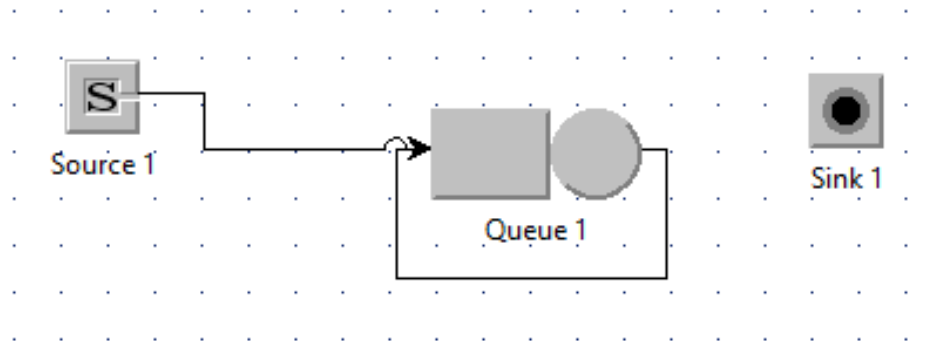




JMT: Connections

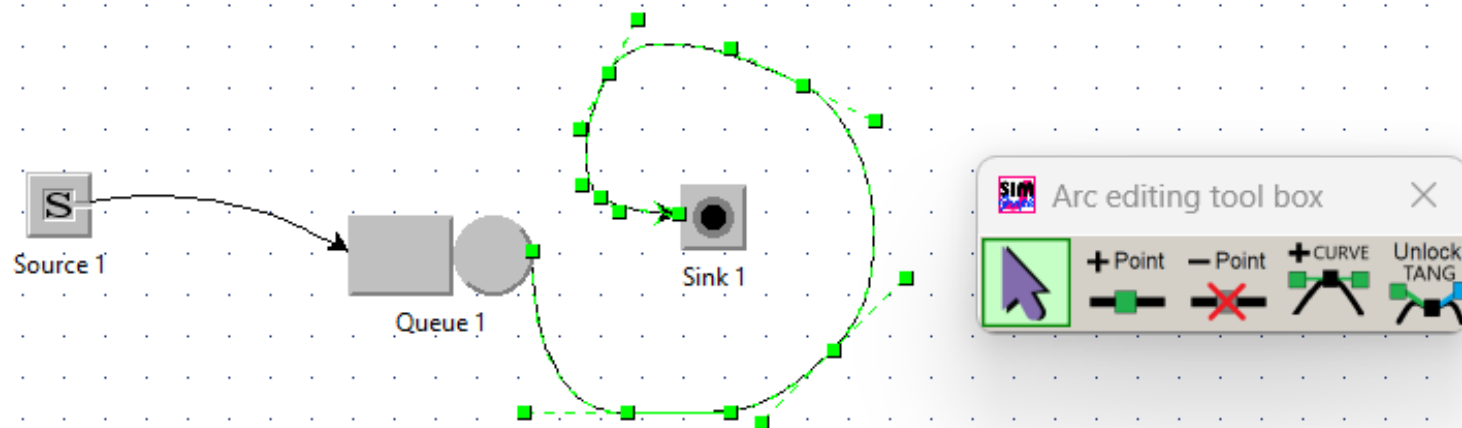
Keep in mind that models can have a *Loops*: arcs that starts from one node and ends on itself.

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





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



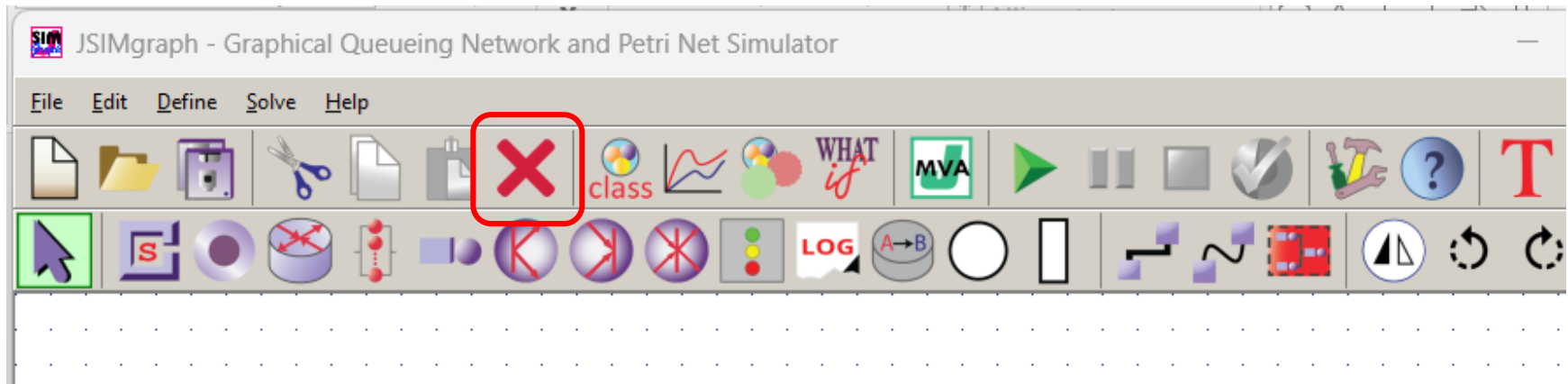
If you are interested in knowing more on how to configure these arcs, please refer to the JMT manual.



JMT: Undo and Deleting

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

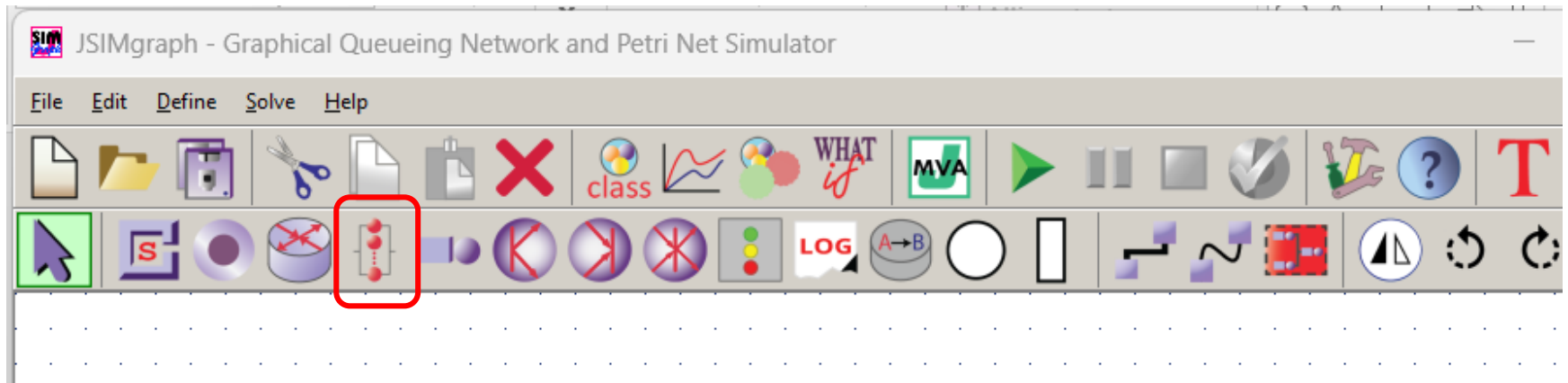
Be aware that JMT does not have an Undo feature, so you must be extra careful when deleting elements.



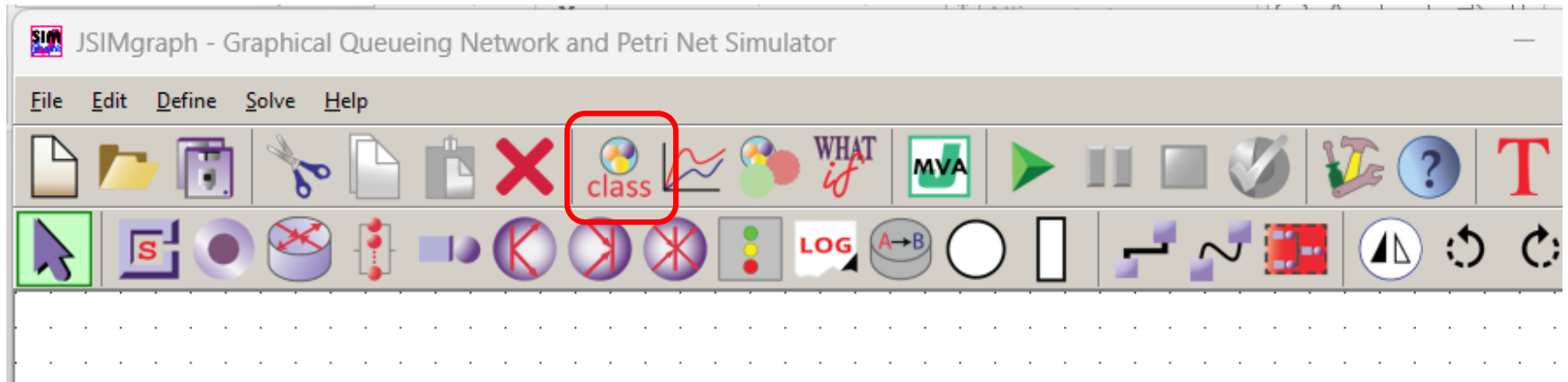


JMT: Delay Stations

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



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.

Define performance indices

Performance Indices
Define performance indices to be collected and plotted by the simulation engine.

Performance Index	Class/Mode	Station/Region/System	Save Stats	Conf.Int.	Max
----- Select an index -----					
Number of Customers					
Queue Time					
Response Time					
Residence Time					
Arrival Rate					
Throughput					
Utilization					
Tardiness					
Earliness					
Lateness					
----- Advanced indexes -----					
Effective Utilization					
Drop Rate					
Balking Rate					
Reneging Rate					
Overload Rate					
Retrial Rate					
Delivered Retrial Orbit Size					
Retrial Orbit Residence Time					
Decision Power					
Response Time per Sink					
Throughput per Sink					
FCR Capacity					

Statistics CSV file
Check the "Save Stats" box to collect samples in a CSV file for additional statistical analysis. This option may produce a file with a large size.
CSV files path: C:\Users\Marco Gribaudo\JMT\



JMT: Performance Indices

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

Define performance indices

Performance Indices
Define performance indices to be collected and plotted by the simulation engine.

----- Select an index -----

Performance Index	Class/Mode	Station/Region/System	Save Stats	Conf.Int.	Max Rel.Err.	Config.
Response Time	--- All Classes ---		<input type="checkbox"/>	0.99	0.03	Edit

Statistics CSV file
Check the "Save Stats" box to collect samples in a CSV file for additional statistical analysis. This option may produce a file with a large size.

CSV files path: C:\Users\Marco Gribaudo\JMT\

Overwrite:
Delimiter:
Decimal separator:



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.

Performance Indices

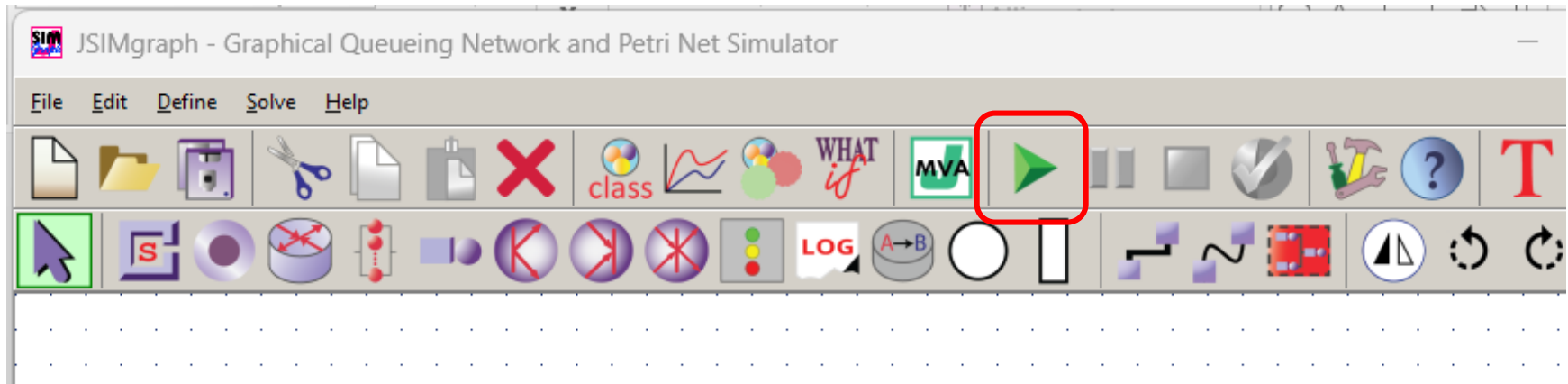
Define performance indices to be collected and plotted by the simulation engine.

----- Select an index ----- ▼

Performance Index	Class/Mode	Station/Region/System	Save Stats	Conf.Int.	Max Rel.Err.	Config.	
Response Time	Class1	Queue 1	<input type="checkbox"/>	0.99	0.03	Edit	<input type="button" value="X"/>



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





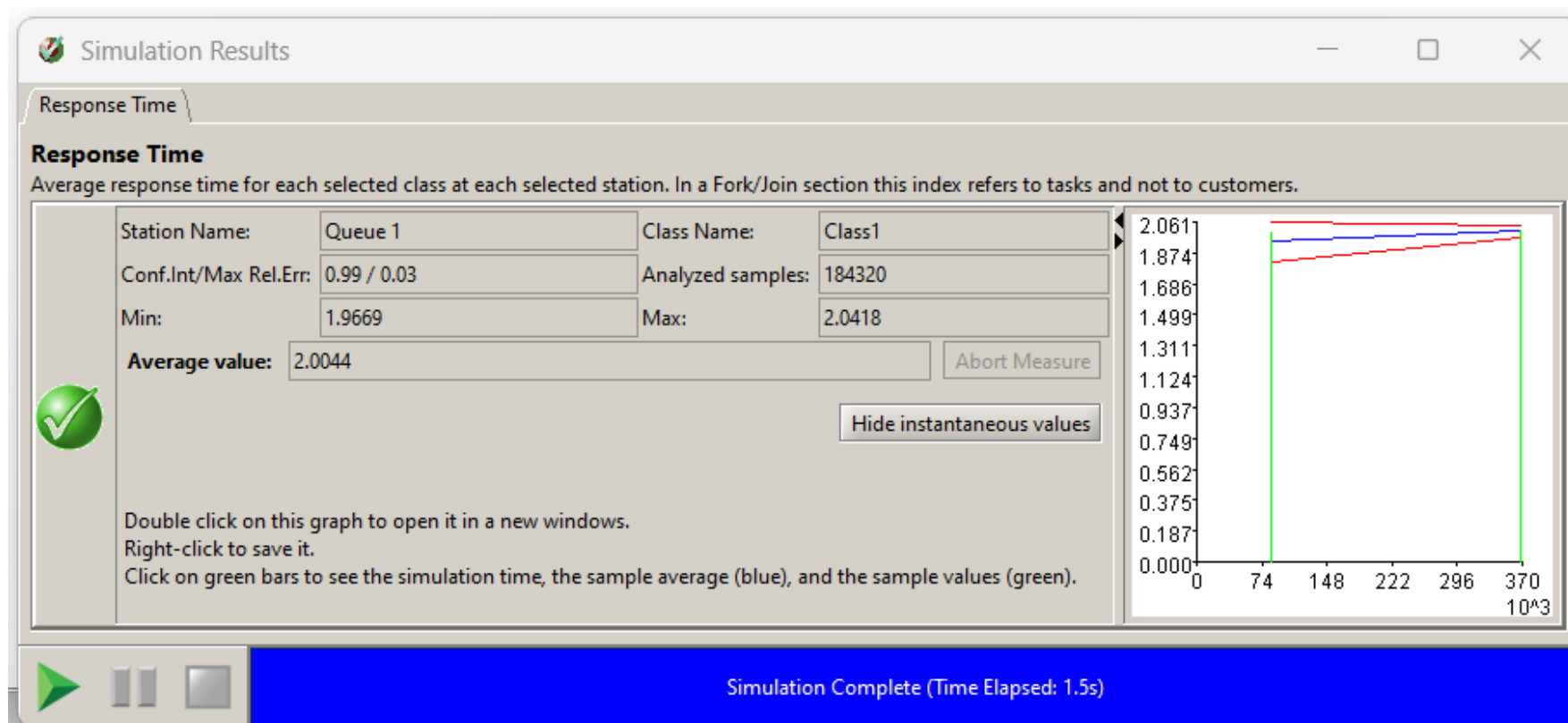
The technique used by *JSimGraph* to compute the solution is called *Discrete Event Simulation*. We will return on its characteristics in the following lessons. However, in the following, we will very frequently call the process of computing the performance indices of a model “*Simulation*”.

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.



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





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

The screenshot displays two windows from the JMT software. The 'Define performance indices' window is in the background, showing a table of performance metrics. The 'Number of Customers' metric is highlighted with a red box, and a red arrow points from it to the 'Simulation Results' window. The 'Simulation Results' window is in the foreground, showing the 'Number of Customers' tab selected. It displays a graph of the average number of customers over time, with a blue line representing the sample average and green bars representing the sample values. The simulation is complete, with a time elapsed of 1.6s.

Define performance indices

Performance Indices
Define performance indices to be collected and plotted by the simulation engine.

Performance Index	Class/Mode	Station/Region/System	Save Stats	Conf.Int.	Max Rel.Err.	Config.
Response Time	Class1	Queue 1	<input type="checkbox"/>	0.99	0.03	Edit
Utilization	Class1	Queue 1	<input type="checkbox"/>	0.99	0.03	Edit
Number of Customers	Class1	Queue 1	<input type="checkbox"/>	0.99	0.03	Edit
Throughput	Class1	Queue 1	<input type="checkbox"/>	0.99	0.03	Edit

Simulation Results

Number of Customers | Response Time | Throughput | Utilization

Number of Customers
Average number of customers for each selected class at each selected station.

Station Name:	Queue 1	Class Name:	Class1
Conf.Int/Max Rel.Err:	0.99 / 0.03	Analyzed samples:	450560
Min:	0.9708	Max:	1.0149
Average value:	0.9928	Abort Measure	

Hide instantaneous values

Double click on this graph to open it in a new windows.
Right-click to save it.
Click on green bars to see the simulation time, the sample average (blue), and the sample values (green).

Simulation Complete (Time Elapsed: 1.6s)

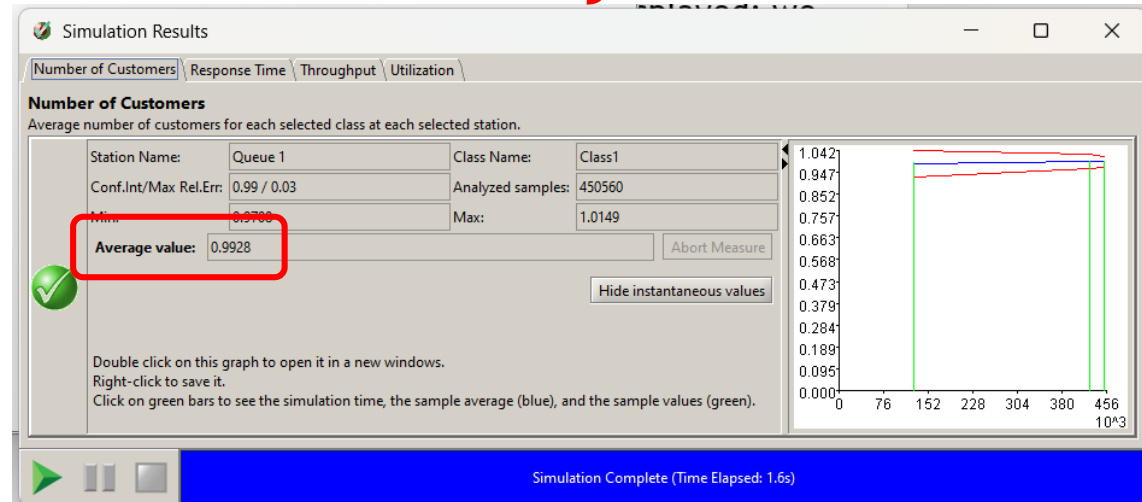


JMT: Instability

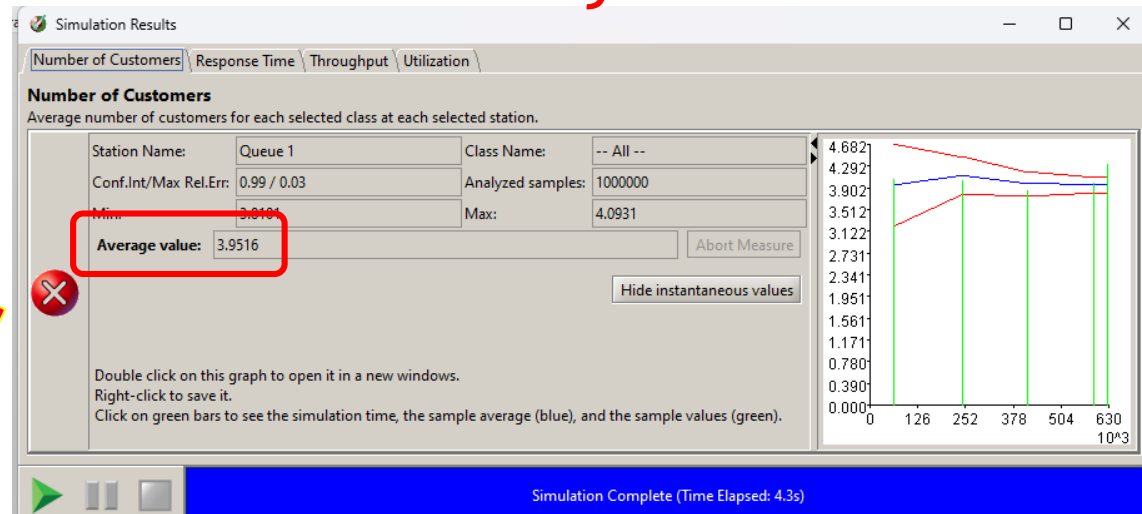
48

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

$$\lambda = 0.5 \text{ job/sec}$$



$$\lambda = 0.8 \text{ job/sec}$$

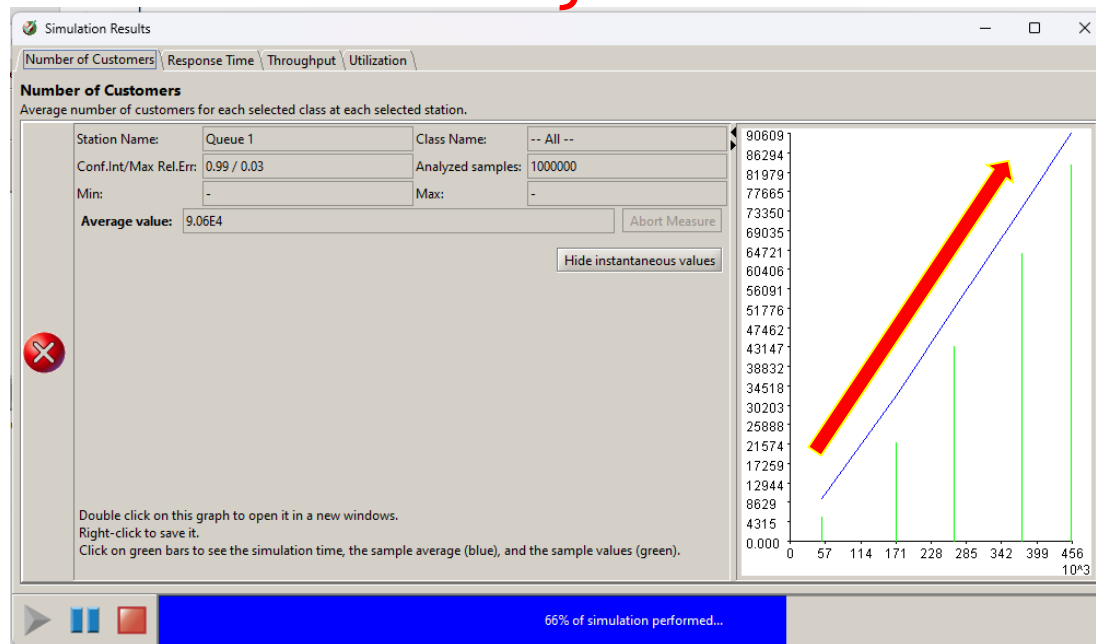


In this case, the desired accuracy could have not been reached. This is represented by the red circle with a cross inside. We will return on this topic later.



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.

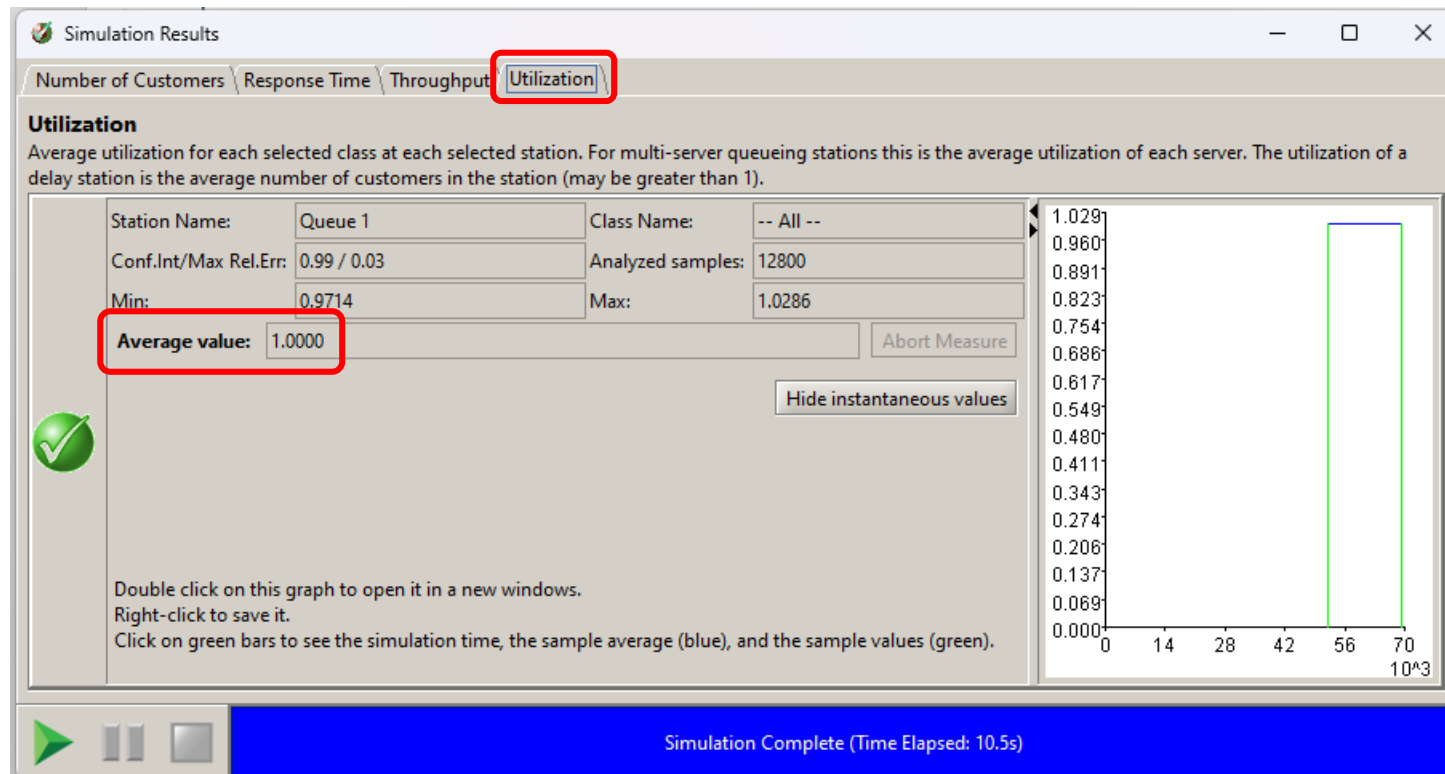
$$\lambda = 1.2 \text{ job/sec}$$



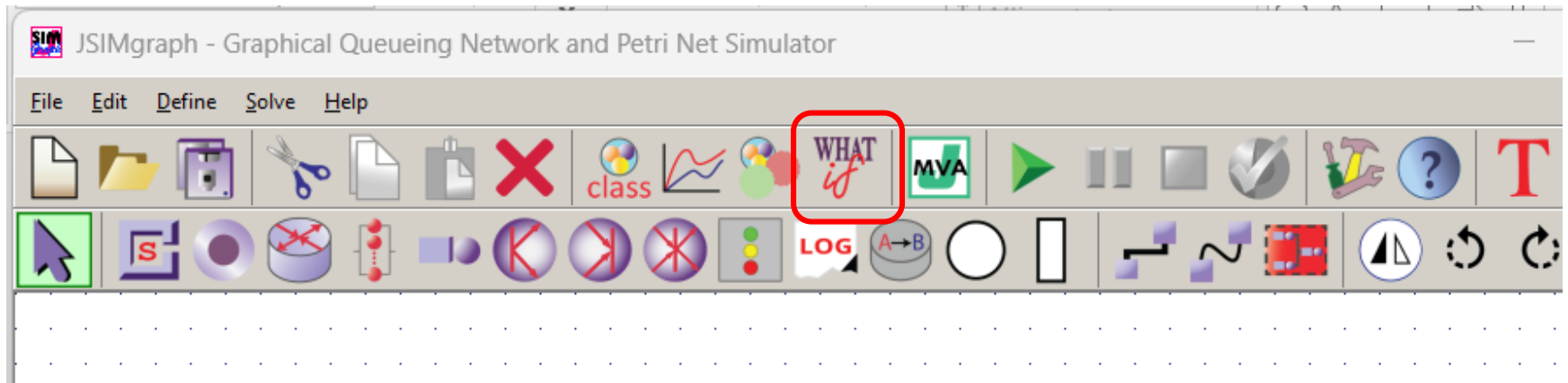
In the worst case, the JAVA virtual machine might consume all the available memory and create an unrecoverable error that will force the user to force-quit the application, losing all the un-saved changes!



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.



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.

The screenshot shows a dialog box titled "Define What-if analysis parameters" with a close button (X) in the top right corner. The dialog is divided into several sections:

- What-if Analysis**: A section with a description "Define the type of What-If analysis to be performed and modify parameter options." and a **WARNING:** "Enabling What-If analysis will disable all statistical outputs." In the top right of this section, there is a checkbox labeled "Enable What-If analysis" which is currently unchecked and is highlighted with a red rectangular box.
- Parameter selection for the control of repeated executions**: A section containing a dropdown menu currently set to "Service times".
- Type of service time growth**: A section with two radio buttons: "Change service time of all classes" (unselected) and "Change service time of one class" (selected). Below these are several input fields:
 - From (s):** A text box with the value "1" and a spinner control.
 - To (s):** A text box with the value "2" and a spinner control.
 - Steps:** A text box with the value "10" and a spinner control.
 - Station:** A dropdown menu currently set to "Queue 1".
 - Class:** A dropdown menu currently set to "Class1".
- Description**: A text area on the right side of the dialog containing the text: "Repeat the simulation changing the service time of a station for one class only." and "The 'To' value represents the final mean value of service time distribution."
- Done**: A button at the bottom center of the dialog.



JMT: What-if analysis

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

Define What-if analysis parameters

What-if Analysis
Define the type of What-If analysis to be performed and modify parameter options.

☒ Enable What-If analysis

WARNING:
Enabling What-If analysis will disable all statistical outputs.

Parameter selection for the control of repeated executions

Service times

Service times
Arrival rates
Number of servers
Total capacity
Routing probability
Seed

Type of service time growth

☐ Change service time of all classes
☒ Change service time of one class

From (s):

To (s):

Steps:

Station:

Class:

Description
Repeat the simulation changing the service time of a station for one class only.
The 'To' value represents the final mean value of service time distribution.

Done



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.

Define What-if analysis parameters

What-if Analysis
Define the type of What-If analysis to be performed and modify parameter options.

☒ Enable What-If analysis

WARNING:
Enabling What-If analysis will disable all statistical outputs.

Parameter selection for the control of repeated executions

Arrival rates ▼

Type of arrival rate growth

☐ Change arrival rates of all open classes

☒ Change the arrival rate of one open class

From: 0.1 ▲▼

To: 0.2 ▲▼

Steps: 10 ▲▼

Class: Class1 ▼

Description
Repeat the simulation with different arrival rates for an open classes, provided that the interarrival time distribution has a finite, positive, mean value. The 'To' value is the final arrival rate.

Done



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

The image shows two dialog boxes from a simulation software. The background dialog is 'Define customer classes' with a table of customer classes. The foreground dialog is 'Editing Class1 distribution...' showing the configuration for an exponential distribution.

Define customer classes

Classes Characteristics
Define type (Open or Closed), name and parameters for each customer class.
Closed Classes: If a **ClassSwitch** is in the model, then **all** the closed classes must have the **same** reference station.
Open Classes: An open class that has **Fork**, **ClassSwitch**, **Scaler** or **Transition** as the reference station is **not** generated by **any** Source.
Priorities: A larger value implies a higher priority.

Color	Name	Type	Priority	Population	Interarrival Time Distribution	Soft Deadline	Reference Station
Blue	Class1	Open	0		exp(0.5)	0.0000	Source 1

Editing Class1 distribution...

Selected Distribution: Exponential

Exponential $[\exp(\lambda)]$:

$$f(x) = \lambda e^{-\lambda x}$$

λ : 0.5

mean: 2

OK Cancel



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.

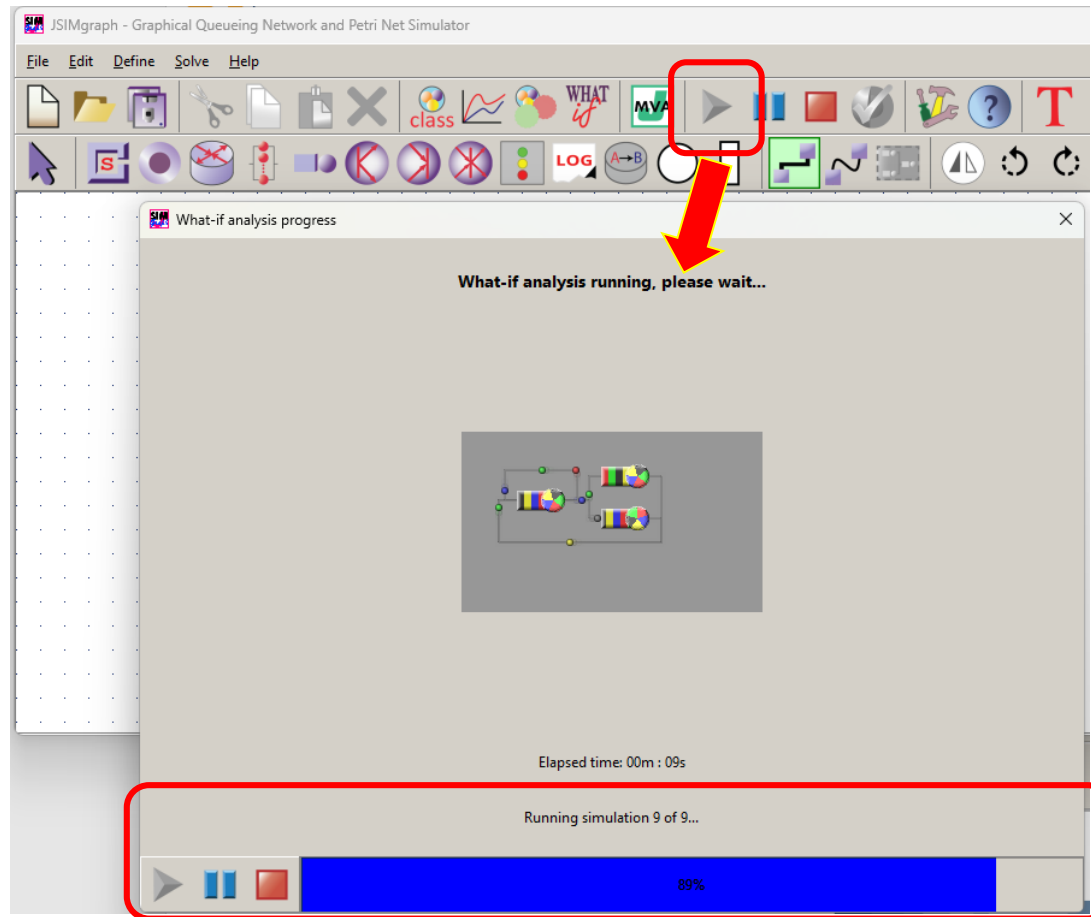
The screenshot shows a dialog box titled "Define What-if analysis parameters". It contains the following elements:

- What-if Analysis** section with a checkbox "Enable What-If analysis" which is checked.
- WARNING:** Enabling What-If analysis will disable all statistical outputs.
- Parameter selection for the control of repeated executions** section with a dropdown menu showing "Arrival rates".
- Type of arrival rate growth** section with two radio buttons:
 - ☐ Change arrival rates of all open classes
 - ☒ Change the arrival rate of one open class
- Fields for the selected option:
 - From:** 0.1
 - To:** 0.2 (highlighted with a red rectangle)
 - Steps:** 10
 - Class:** Class1
- Description** section: Repeat the simulation with different arrival rates for an open classes, provided that the interarrival time distribution has a finite, positive, mean value. The 'To' value is the final arrival rate.
- Done** button at the bottom.



JMT: What-if analysis

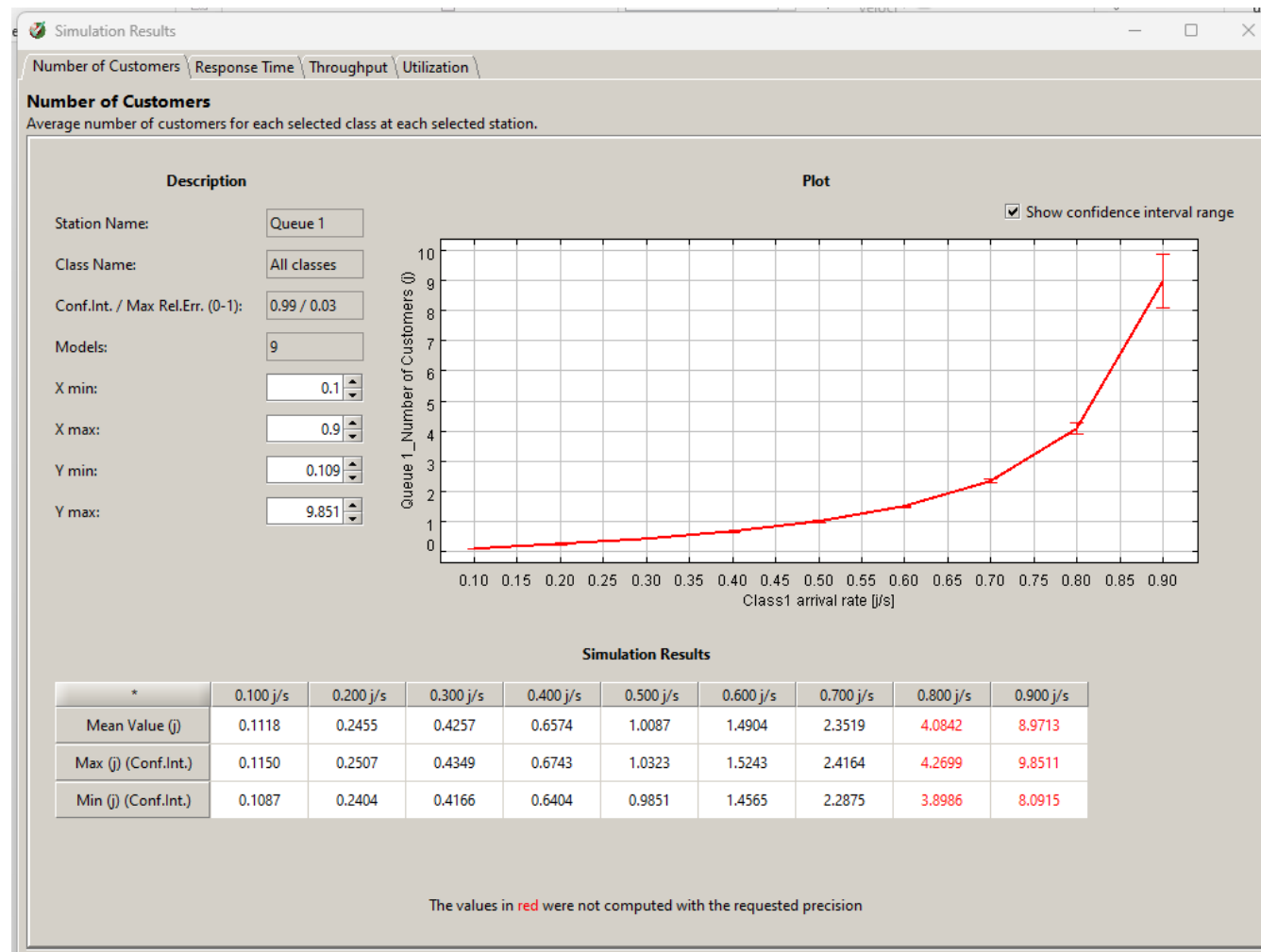
When the solution is executed, the output window is replaced by a progress bar that shows how many configurations have been computed.





JMT: What-if analysis

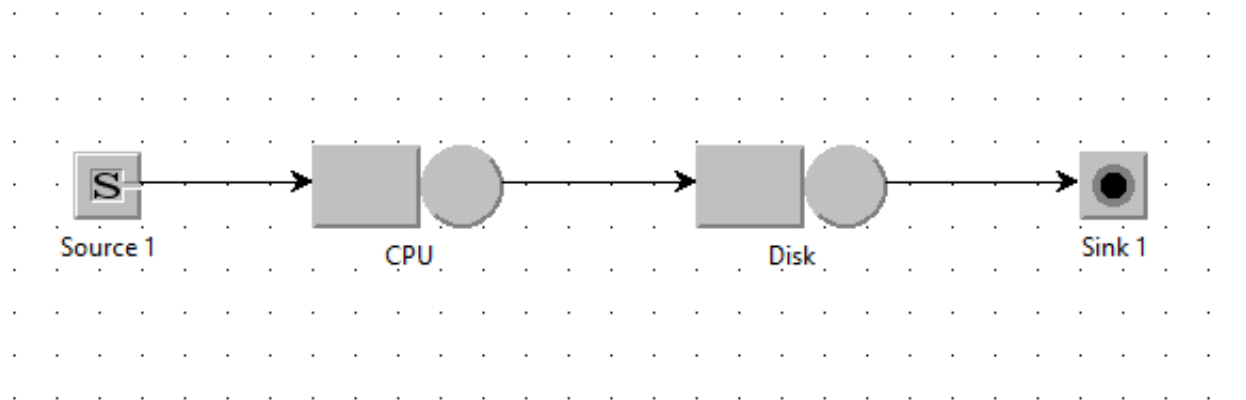
Results are then presented as tables and plots.





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.

JSIMgraph - Graphical Queueing Network and Petri Net Simulator

File Edit Define Solve Help

class WHAT LOG

Source 1

Queue 1

Editing Queue 1 Properties

Station Name: Queue 1

Queue 1 Parameters Definition

Queue Section Service Section Routing Section

Capacity

☒ Infinite

Max no. customers (queue+service)

Queue Policy

Station Queue Policy: Non-preemptive Scheduling

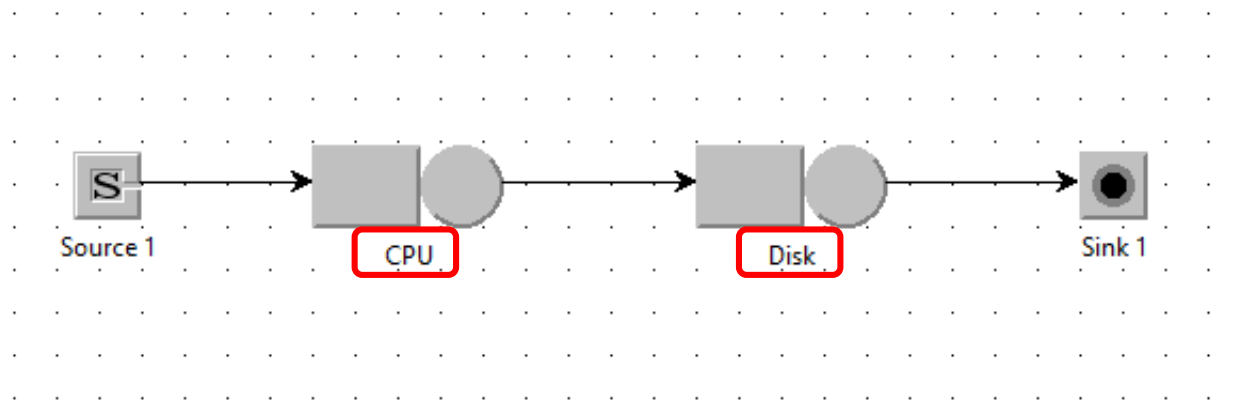
Class	Queue Policy	Drop Rule	Service Weight	Impatience	
Class1	FCFS	Infinite Capacity	--	None	Edit

Done



More queues

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





Tandem Networks

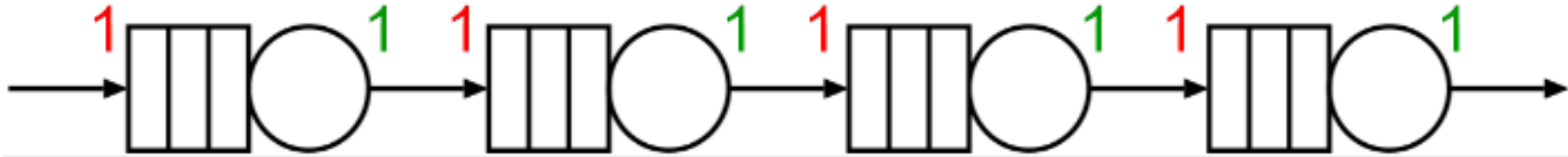
The simplest case of systems with more stations are the Tandem Networks....





Tandem networks

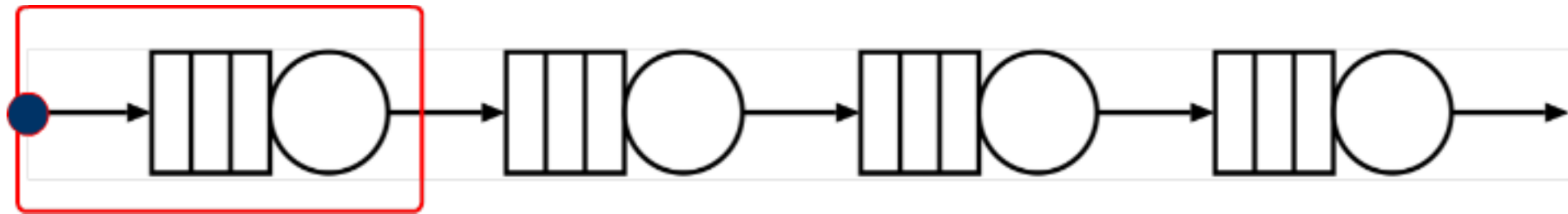
A *Tandem Open Queuing Network* is an open system where every queue has exactly one input and one output connection.





Tandem networks

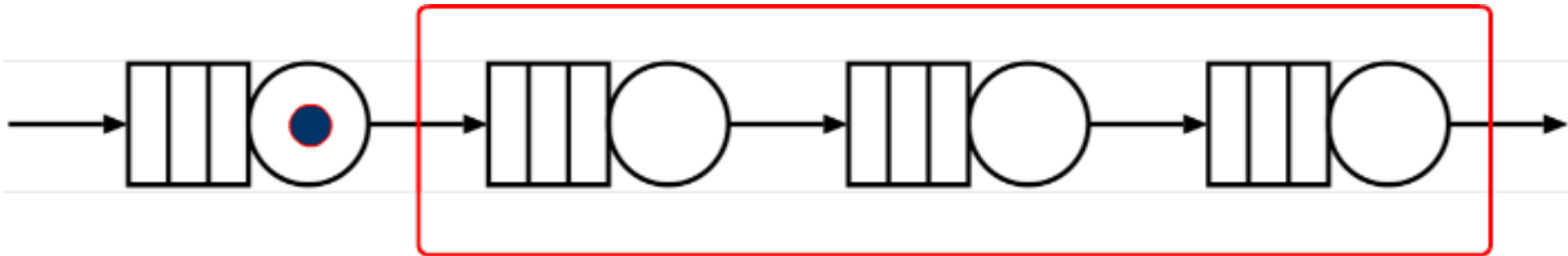
Jobs enter the first node (queue) of the line...





Tandem networks

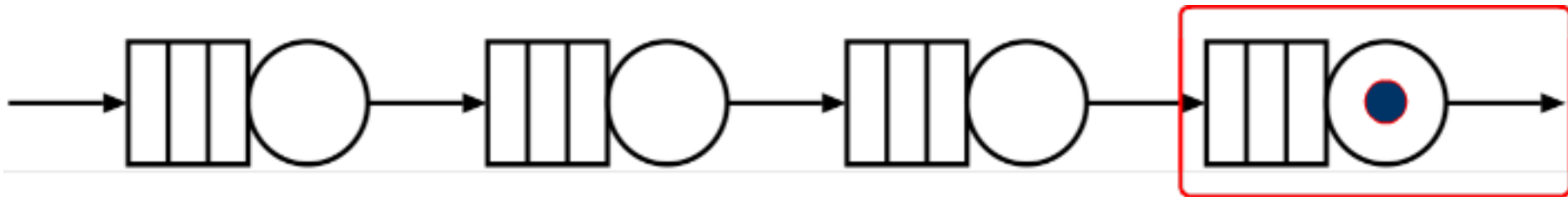
... and proceeds stage after stage up to the last station.





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.

Define performance indices

Performance Indices
Define performance indices to be collected and plotted by the simulation engine.

----- Select an index -----

Performance Index	Class/Mode	Station/Region/System	Save Stats	Conf.Int.	Max Rel.Err.	Config.	
Number of Customers	--- All Classes ---	Queue 1	<input type="checkbox"/>	0.99	0.03	Edit	✗
Response Time	--- All Classes ---	Queue 1	<input type="checkbox"/>	0.99	0.03	Edit	✗
Throughput	--- All Classes ---	Queue 1	<input type="checkbox"/>	0.99	0.03	Edit	✗
Utilization	--- All Classes ---	Queue 1	<input type="checkbox"/>	0.99	0.03	Edit	✗
Response Time	--- All Classes ---	System	<input type="checkbox"/>	0.99	0.03	Edit	✗
Number of Customers	--- All Classes ---	System	<input checked="" type="checkbox"/>	0.99	0.03	Edit	✗

Simulation Results

Number of Customers | Response Time | Throughput | Utilization | **System Number of Customers** | **System Response Time**

System Response Time
Average response time of the entire system for each selected class.

Station Name: Network Class Name: -- All --
Conf.Int./Max Rel.Err: 0.99 / 0.03 Analyzed samples: 143360
Min: 4.8927 Max: 5.1353
Average value: 5.0140

Abort Measure
Hide instantaneous values

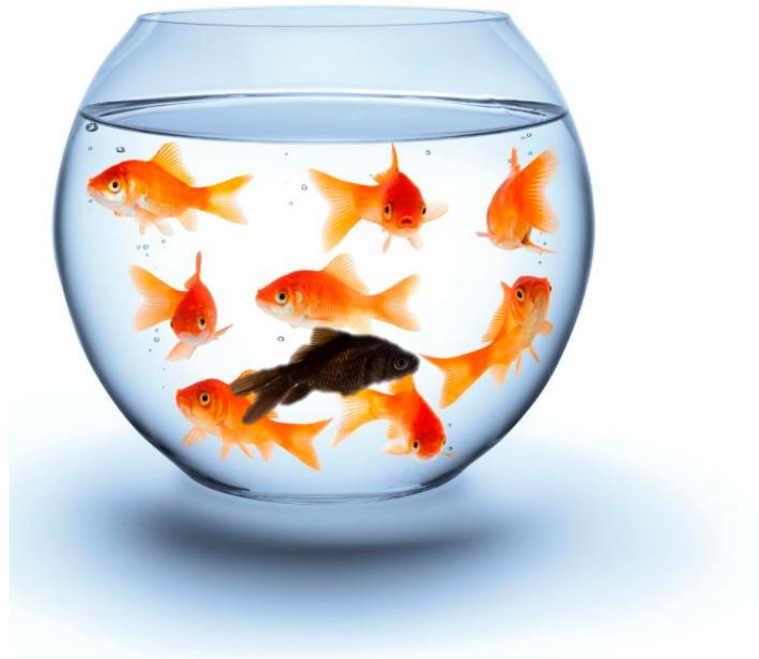
Double click on this graph to open it in a new windows.
Right-click to save it.
Click on green bars to see the simulation time, the sample average (blue), and the sample values (green).

Simulation Complete (Time Elapsed: 3.7s)



Closed systems: motivation

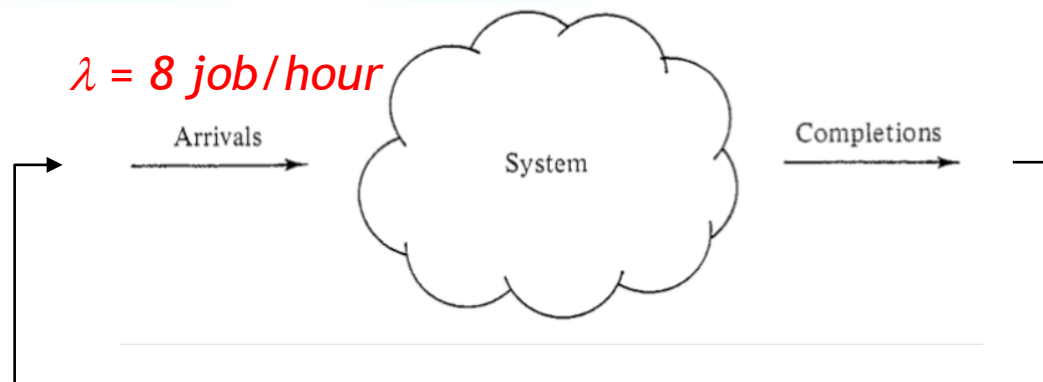
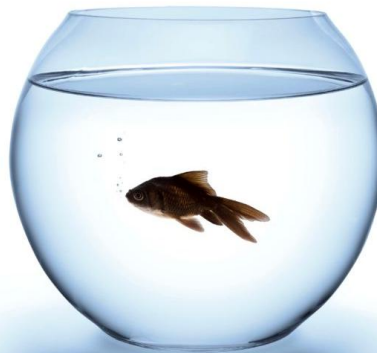
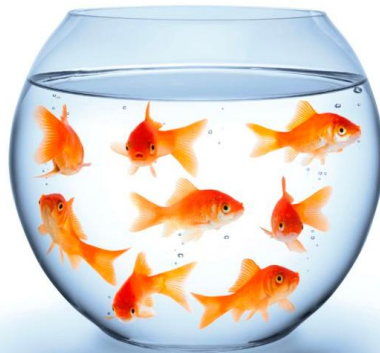
If the jobs that enter the system come from a fixed population...





Closed systems: motivation

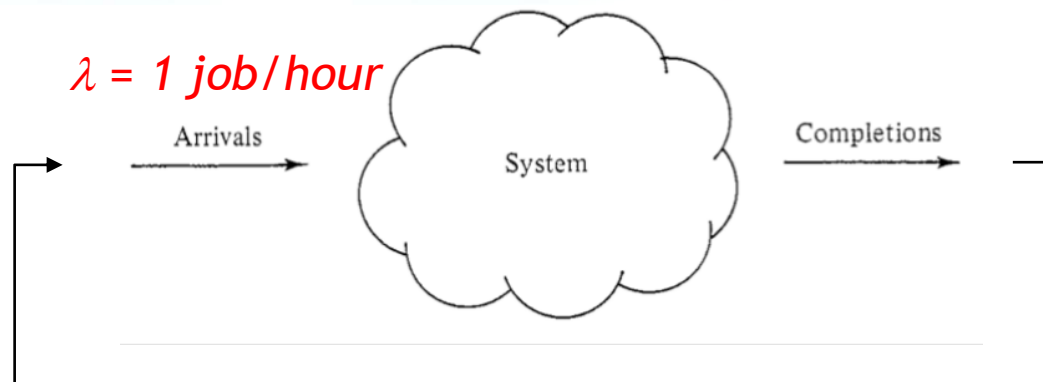
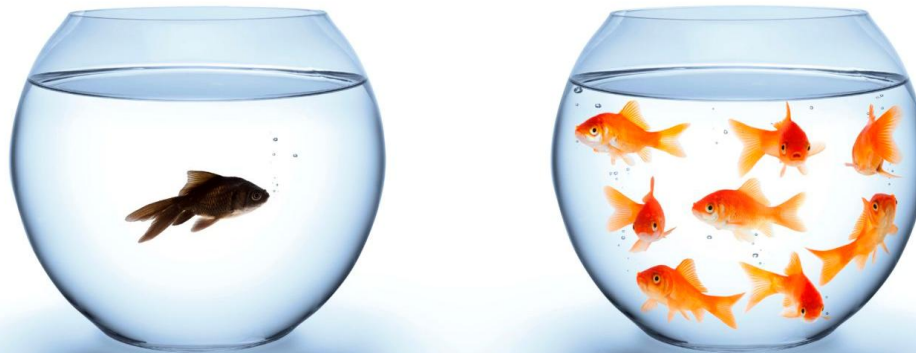
When most of the population is outside, jobs arrives at a high rate.





Closed systems: motivation

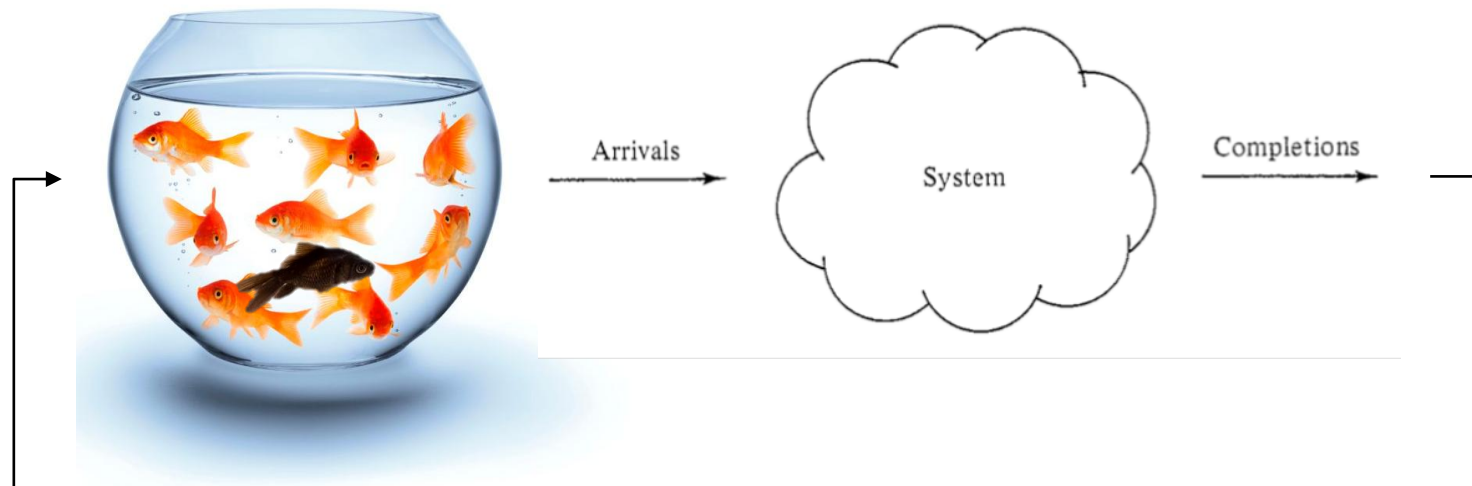
Conversely, when most of the population is inside, jobs arrives at a very low rate.





Closed systems: stability

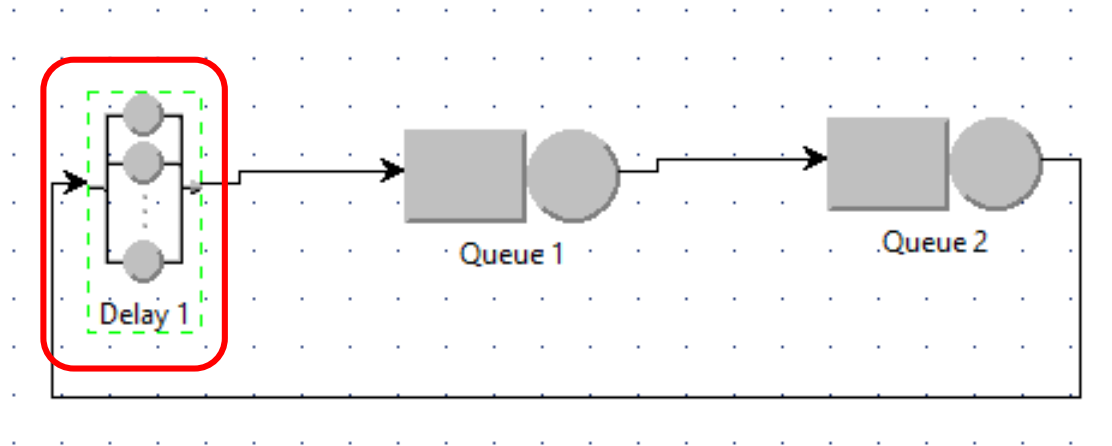
For this reason, closed systems are *Always Stable* and needs specific techniques to be studied.





Closed systems: think time and delay station

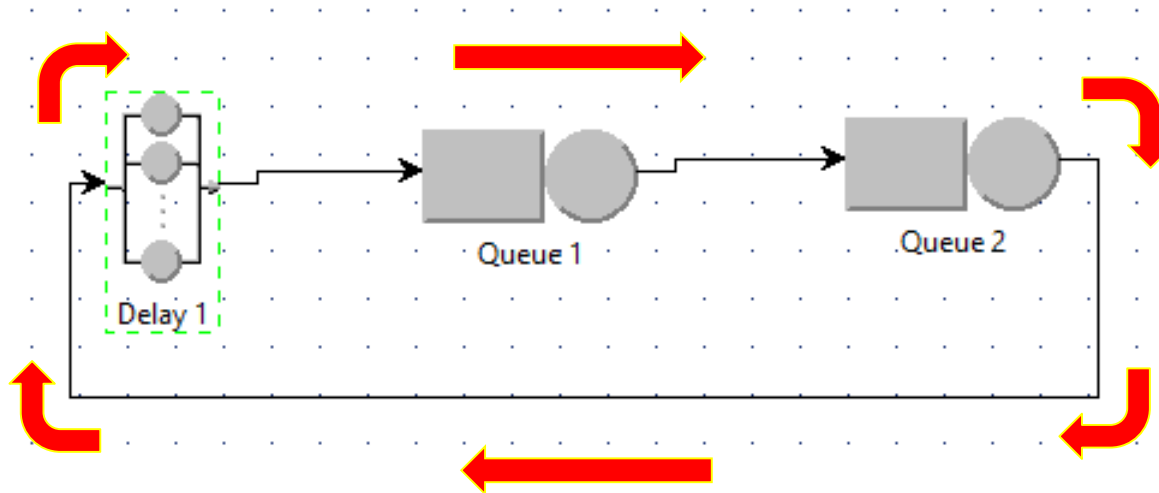
The time spent “outside” the system is generally modelled by a delay station...





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.

Define customer classes

Classes Characteristics
Define type (Open or Closed), name and parameters for each customer class.
Closed Classes: If a **ClassSwitch** is in the model, then **all** the closed classes must have the **same** reference station.
Open Classes: An open class that has **Fork**, **ClassSwitch**, **Scaler** or **Transition** as the reference station is **not** generated by **any** Source.
Priorities: A larger value implies a higher priority.

Classes: 1

Color	Name	Type	Priority	Population	Interarrival Time Distribution	Soft Deadline	Reference Station
Blue	Class1	Closed	0	10		0.0000	Delay 1

Done



Closed systems: closed classes

The fixed population is then specified in the corresponding field.

Define customer classes

Classes Characteristics
Define type (Open or Closed), name and parameters for each customer class.
Closed Classes: If a **ClassSwitch** is in the model, then **all** the closed classes must have the **same** reference station.
Open Classes: An open class that has **Fork**, **ClassSwitch**, **Scaler** or **Transition** as the reference station is **not** generated by **any** Source.
Priorities: A larger value implies a higher priority.

Add Class

Classes: 1

Color	Name	Type	Priority	Population	Interarrival Time Distribution	Soft Deadline	Reference Station
	Class1	Closed	0	10		0.0000	Delay 1

Done



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.

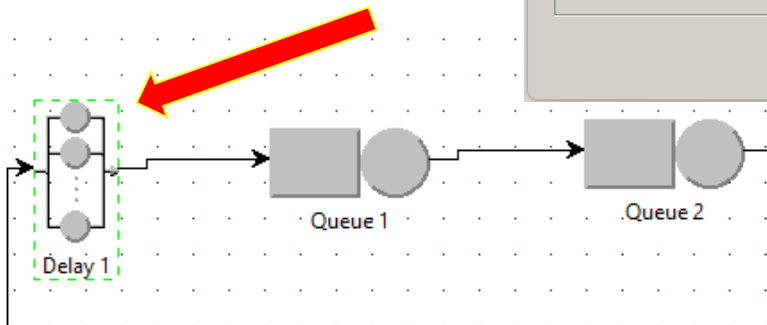
Define customer classes

Classes Characteristics
Define type (Open or Closed), name and parameters for each customer class.
Closed Classes: If a **ClassSwitch** is in the model, then **all** the closed classes must have the **same** reference station.
Open Classes: An open class that has **Fork**, **ClassSwitch**, **Scaler** or **Transition** as the reference station is **not** generated by **any** Source.
Priorities: A larger value implies a higher priority.

Classes: 1

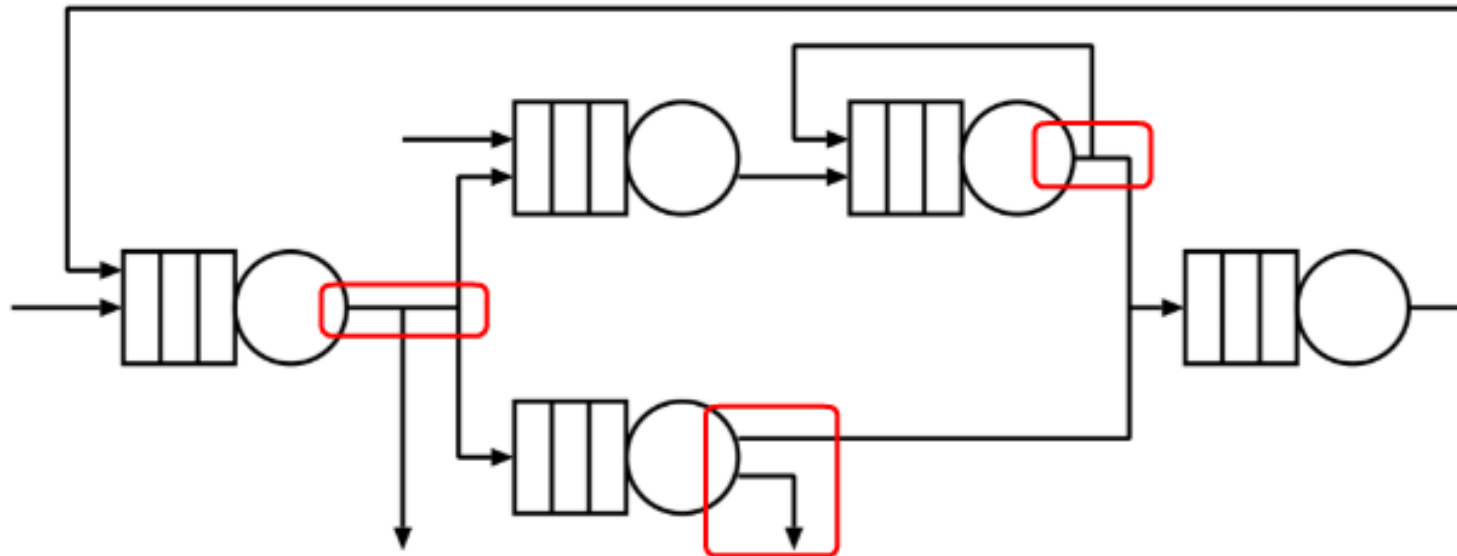
Color	Name	Type	Priority	Population	Interarrival Time Distribution	Soft Deadline	Reference Station
Blue	Class1	Closed	0	10		0.00 0	Delay 1

Done





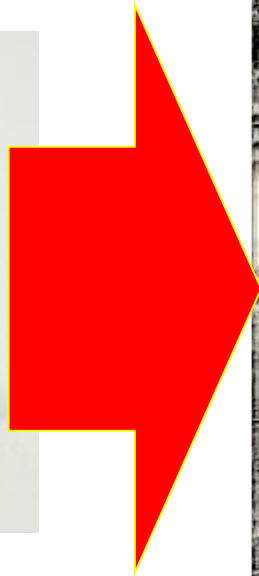
A node can be connected to more than one destination nodes.
In this case, the modeler must define which service center a job finishing at one node will join.
This definition is called *Routing*.





Probabilistic Routing

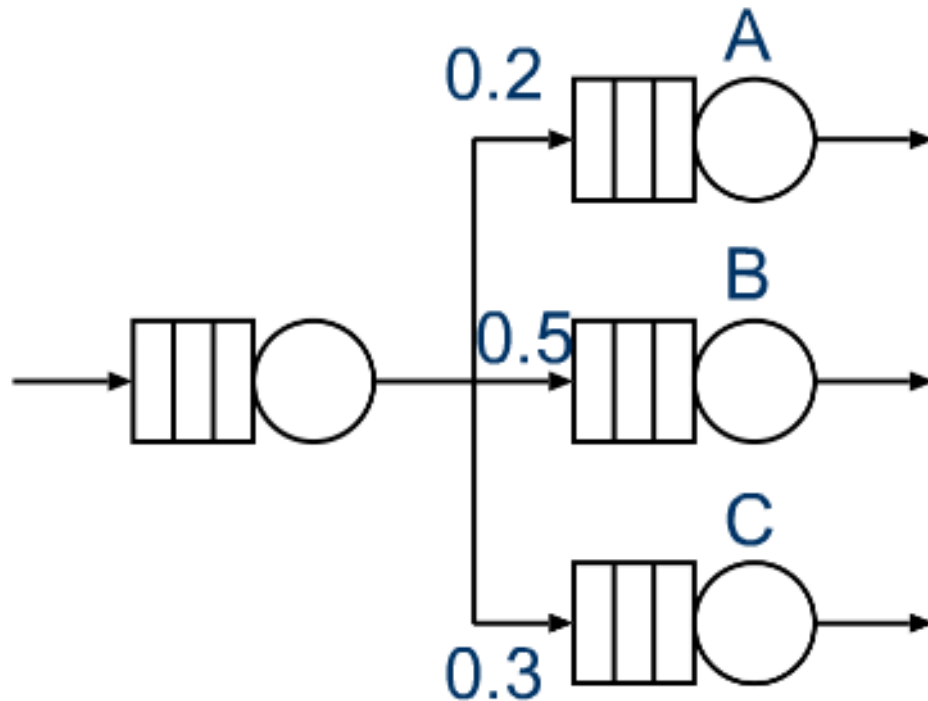
Several routing policies exist: we will present them in a future lesson. For the moment we focus on the *Probabilistic Routing* policy.





Probabilistic routing

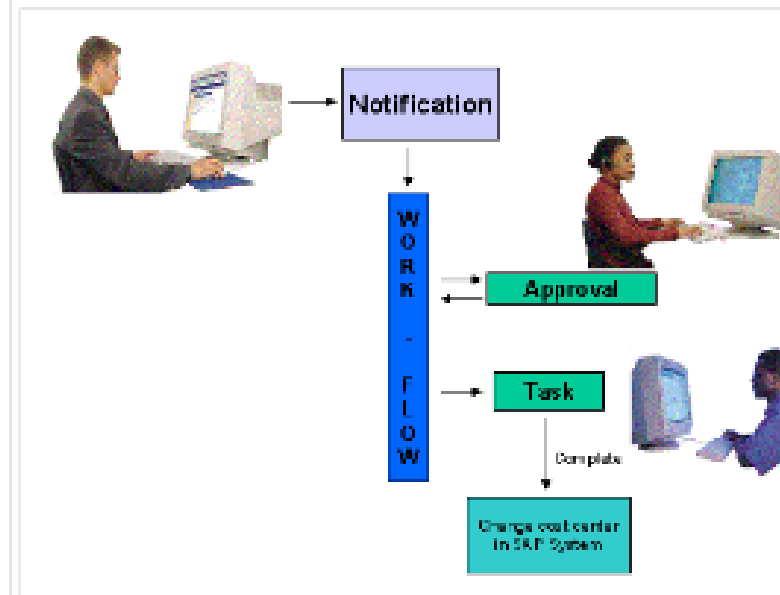
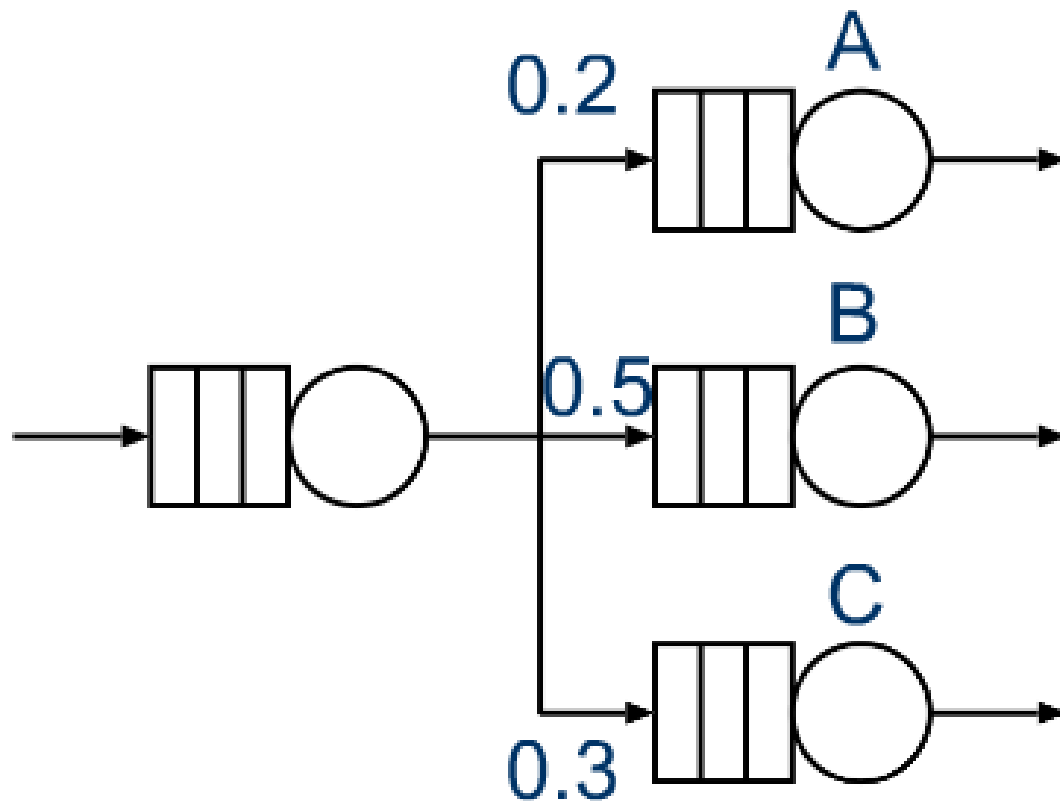
In the *probabilistic routing*, each path has assigned a probability of being chosen by the job that left the considered station.





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.

Editing Queue 1 Properties...

Station Name: Queue 1

Queue 1 Parameters Definition

Queue Section | Service Section | **Routing Section**

Capacity

☒ Infinite

☐ Finite

Max no. customers (queue+service)

Queue Policy

Station Queue Policy: Non-preemptive Scheduling [Edit]

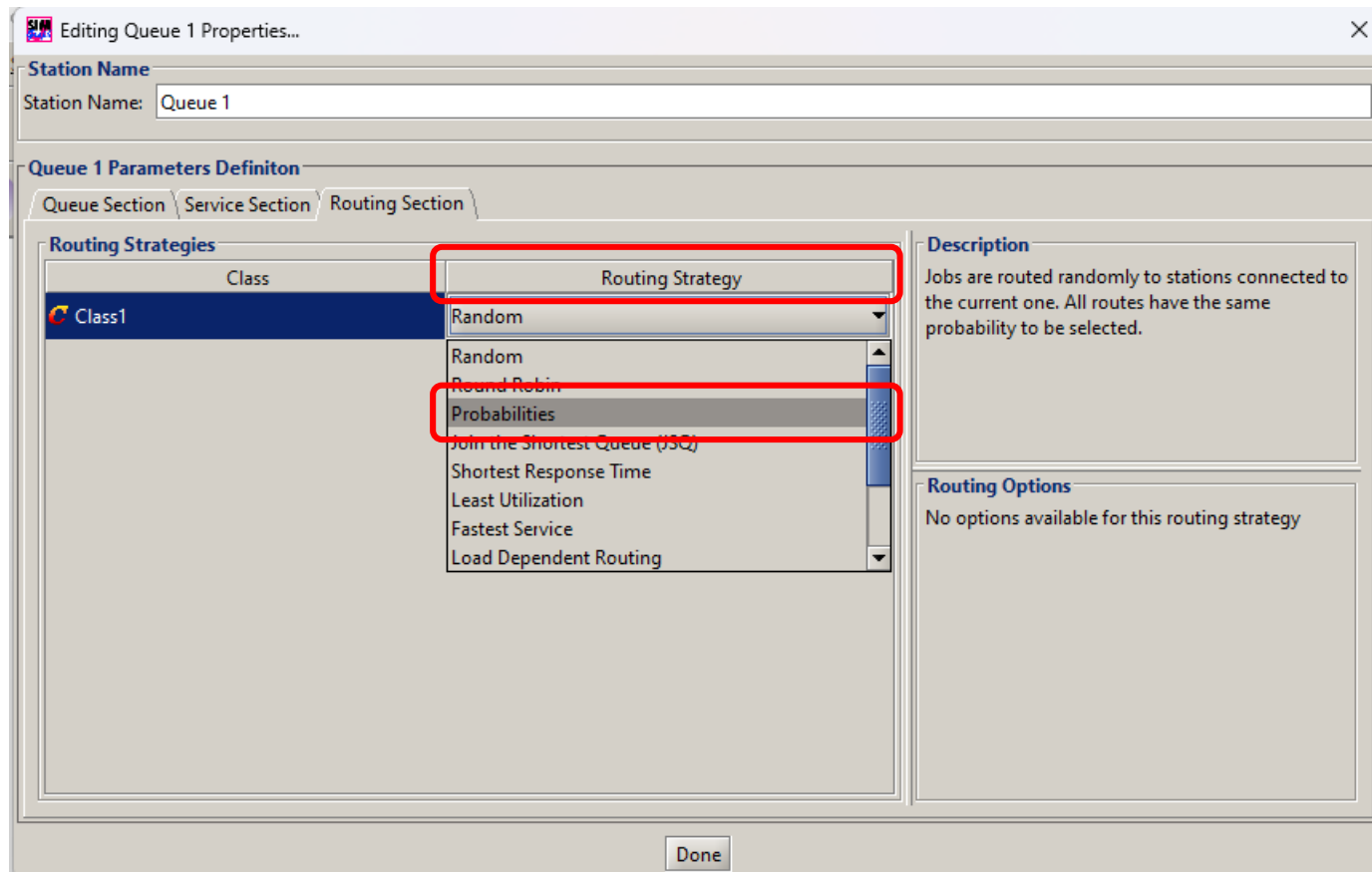
Class	Queue Policy	Drop Rule	Service Weight	Impatience	
Class1	FCFS	Infinite Capacity	--	None	[Edit]

Done



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.

Editing Queue 1 Properties...

Station Name
Station Name: Queue 1

Queue 1 Parameters Definition
Queue Section Service Section Routing Section

Routing Strategies

Class	Routing Strategy
Class1	Probabilities

Description
Jobs are routed to stations connected to the current one according to the specified probabilities. If the sum of the routing probabilities is different from 1, the values will be scaled to sum to 1.

Routing Options

Destination	Probability
Queue 2	0.5
Queue 3	0.2
Queue 4	0.3

Done



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).

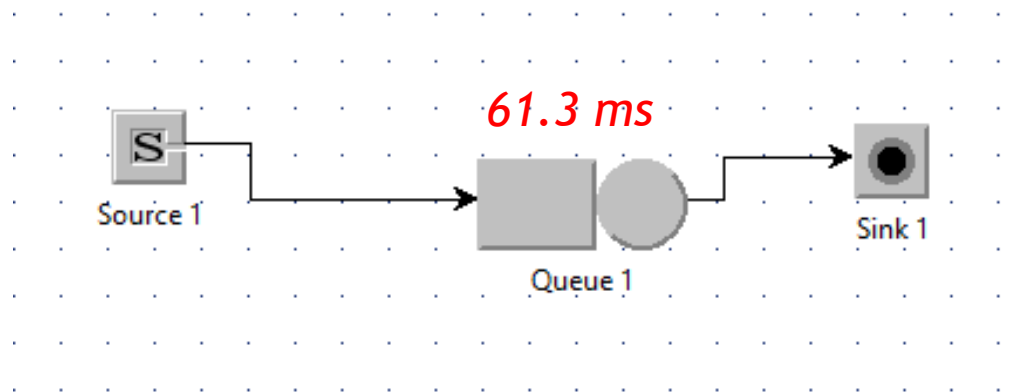


61.3 ms



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.

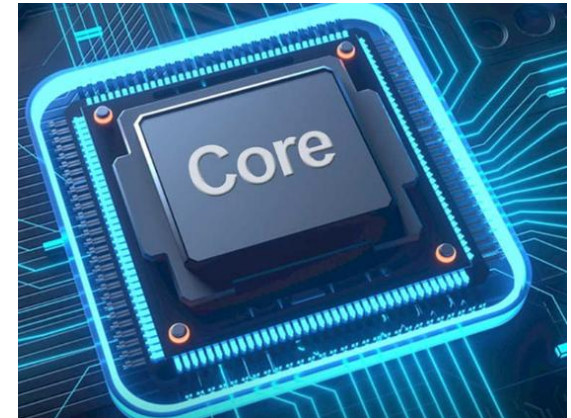




Modelling a web service

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

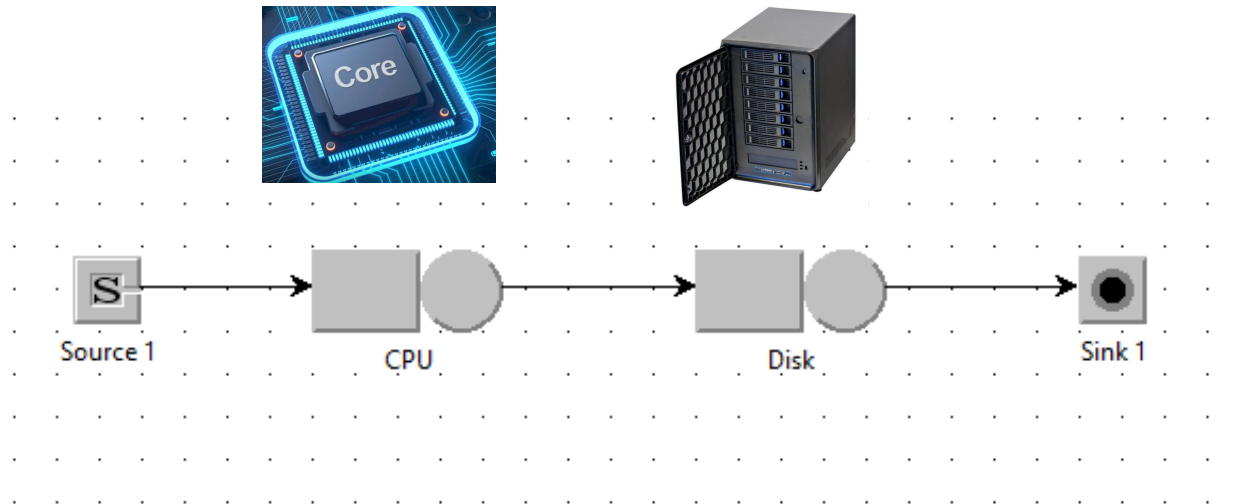
- A CPU
- A storage component





Modelling a web service

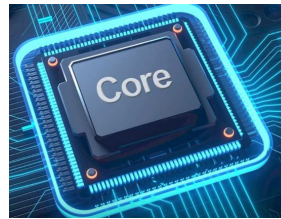
While the a Job is waiting for receiving data from the storage, another can run in parallel on the CPU, thus providing a higher throughput.





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

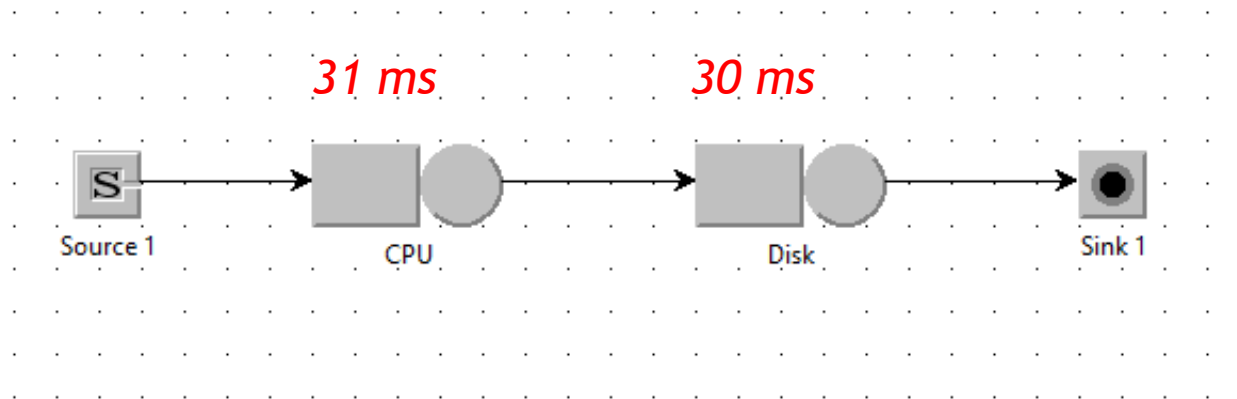


30 ms



Modelling a web service

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

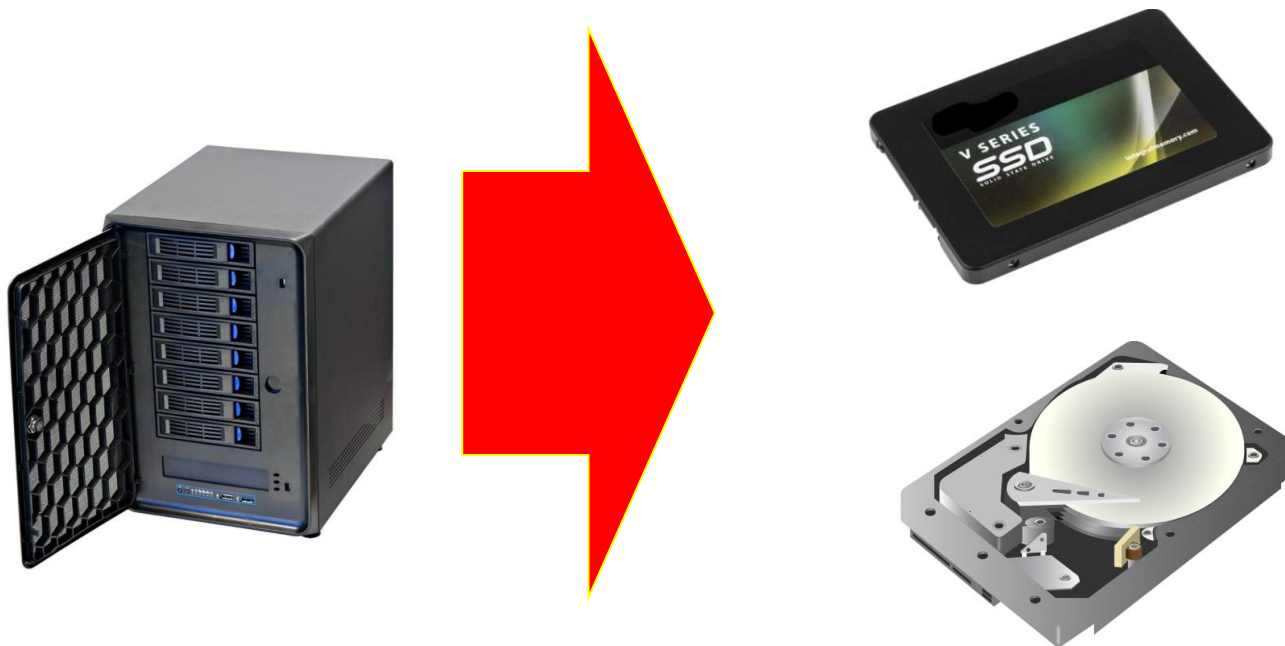


Modelling a web service

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

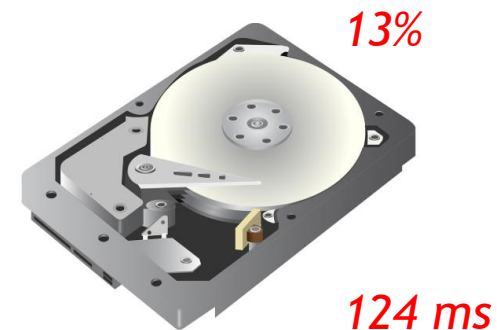
The former is very slow, and used rarely. The latter is much faster, and used more frequently.



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





Modelling a web service

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

