

# Performance Evaluation and Applications



POLITECNICO DI MILANO



## Closed models

POLITECNICO DI MILANO



## Motivating example

An *automated warehouse* has 3 robots that takes the goods from the shelves to the delivery area. Each robot has its own charging station, that requires a total of 30min to fully recharge its battery. Only one robot at a time can operate either in the shelves or delivery area, requiring respectively 6min and 4min to complete their tasks. Travelling between the zones can be done in parallel, and takes an average of 2min. Each robot has to return to the charging station on the average every 25 deliveries.



Management is interested in determining:

1. Average time between recharges for each robot.
2. System throughput (how many goods are delivered per hour).



## The response time law

In time-sharing systems, the "think time" is usually not considered in the system response time.

$$R_{Tot} = R_{Sys} + Z \quad N = X \cdot R_{Tot} \quad N = X \cdot (R_{Sys} + Z)$$

In particular, Little's law becomes the so-called "*Response Time Law*" which explicitly excludes the think time from the usual response time.



$$\text{The Response Time Law: } R = \frac{N}{X} - Z$$

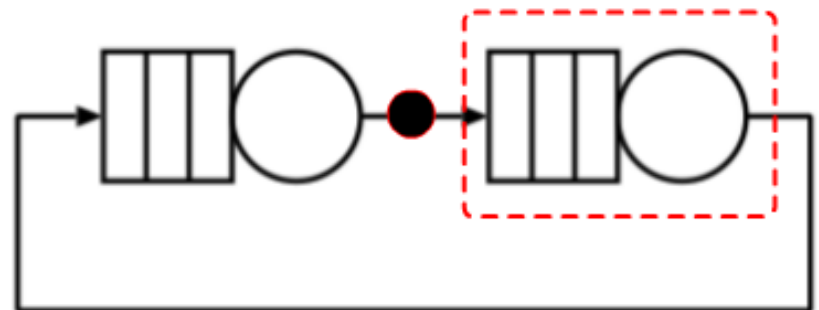
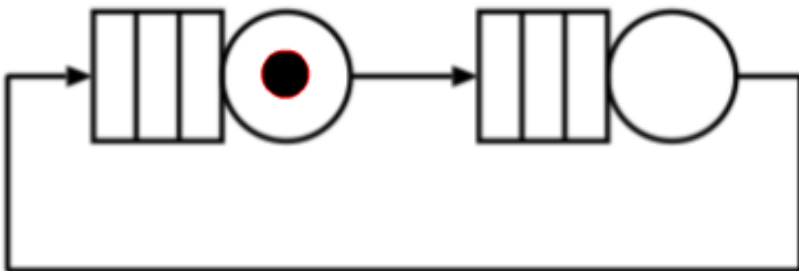
## Analysis of closed models

As for separable open queueing network models, the solution of closed ones is based on the computation of  $A_k(N)$ , the average number of jobs at the arrival.

However, the expression of  $A_k(N)$  in *closed models* is different from the one used in open models.

If we consider a network with two stations, identical demand and a single job, we can see immediately that the average number of jobs is equal to one half:  $N_k(N) = 0.5$ .

However, since there is just one job, the number of jobs that are found when the customer enters a station is always  $A_k(1) = 0$ .





## Analysis of closed models

In closed models we can use the "*Arrival Theorem*" which states *that the number of jobs that a costumer finds in the queue at its arrival is equal to the average queue length of the system with one less job.*

$$A_k(N) = N_k(N - 1)$$



## Mean Value Analysis (MVA)

The performance indices can then be computed in an iterative way, starting from an empty system and adding one job per iteration.

This technique is called "*Mean Value Analysis*" (MVA).

Let us imagine that the system has been studied up to a workload of  $N-1$  jobs. Using the arrival theorem we can determine the residence time at each station when the population increases of one job (  $N$  jobs ).

$$R_k(N) = (1 + A_k(N)) \cdot D_k = (1 + N_k(N - 1)) \cdot D_k$$



## Mean Value Analysis (MVA)

Summing up the residence time at all the stations, we can determine the *system response time*.

$$R(N) = \sum_k R_k(N)$$

Inverting the *response time law*, the system throughput can be determined.

$$R(N) = \frac{N}{X(N)} - Z$$



$$X(N) = \frac{N}{R(N) + Z}$$



## Mean Value Analysis (MVA)

Using *Little's law*, the queue length of each station can finally be determined.

$$N_k(N) = X(N) \cdot R_k(N)$$

The algorithm can then increase the population to  $N+1$  jobs, since for the arrival theorem we have that  $A_k(N+1) = N_k(N)$ , and the process can be repeated.

$$A_k(N + 1) = N_k(N)$$

$$R_k(N + 1) = (1 + A_k(N + 1)) \cdot D_k = (1 + N_k(N)) \cdot D_k$$





## Mean Value Analysis (MVA)

The starting point considers an empty system.

$$N_k(0) = 0$$

$$A_k(1) = N_k(0) = 0$$

$$R_k(1) = (1 + A_k(1)) \cdot D_k = D_k$$



## Mean Value Analysis (MVA)

To summarize we have:

```
for  $k \leftarrow 1$  to  $K$  do  $Q_k \leftarrow 0$ 
for  $n \leftarrow 1$  to  $N$  do
begin
    for  $k \leftarrow 1$  to  $K$  do  $R_k \leftarrow \begin{cases} D_k & \text{(delay centers)} \\ D_k (1 + Q_k) & \text{(queueing centers)} \end{cases}$ 

    
$$X \leftarrow \frac{n}{Z + \sum_{k=1}^K R_k}$$


    for  $k \leftarrow 1$  to  $K$  do  $Q_k \leftarrow X R_k$ 
end
```



## Mean Value Analysis complexity

The mean value analysis computes the solution of a model with complexity  $O(N \cdot K)$ .

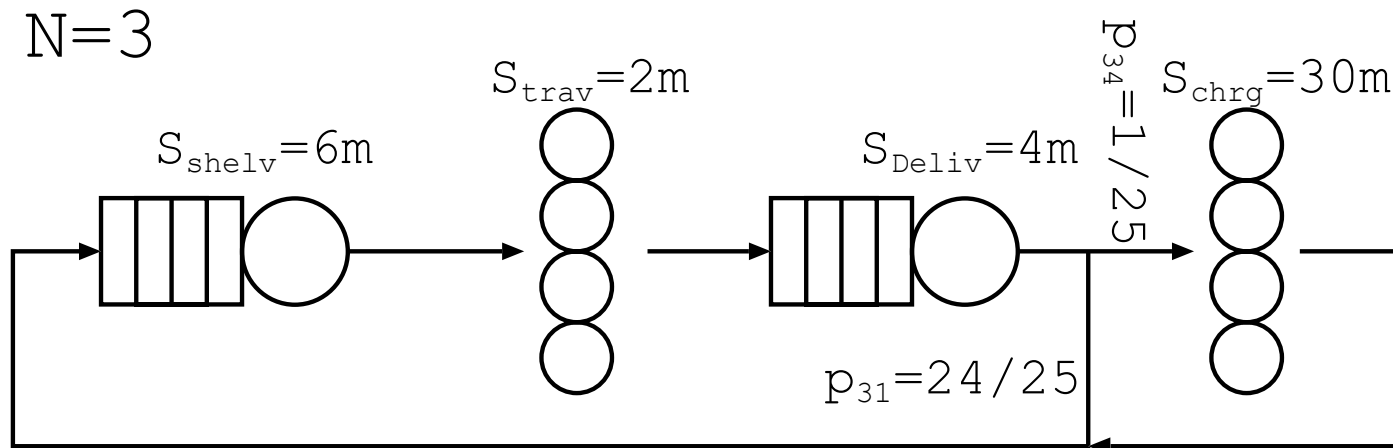
However, as a by-product, it computes the solution for the models with population sizes  $1$  to  $N-1$ .

This can be particularly effective when performing sizing studies, where the evolution of the performance against the population is required.



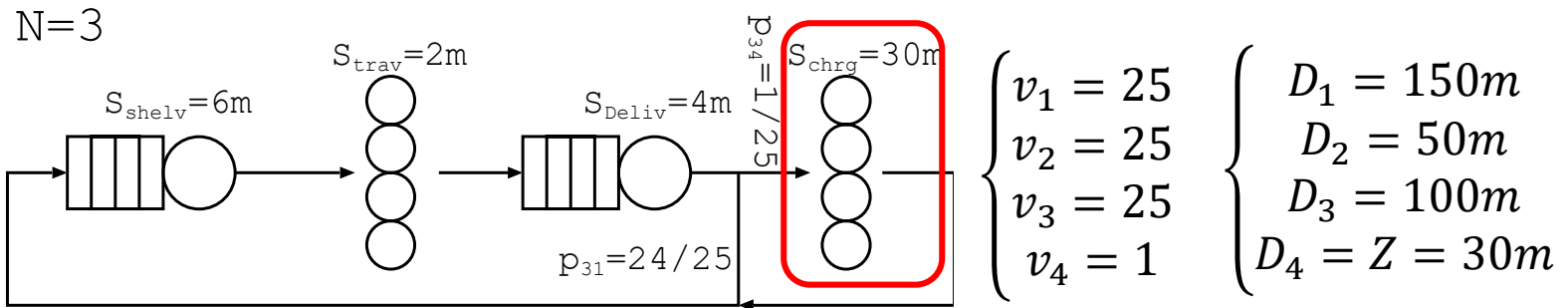
## Analysis of Motivating Example

*The considered automated warehouse can be modelled with a closed queuing model, with  $N=3$  jobs circulating inside.*



## Analysis of Motivating Example

To determine the average time between charges, the model is a time-sharing system where station 4 is both the reference and the terminal station, and we compute the system response time with MVA.



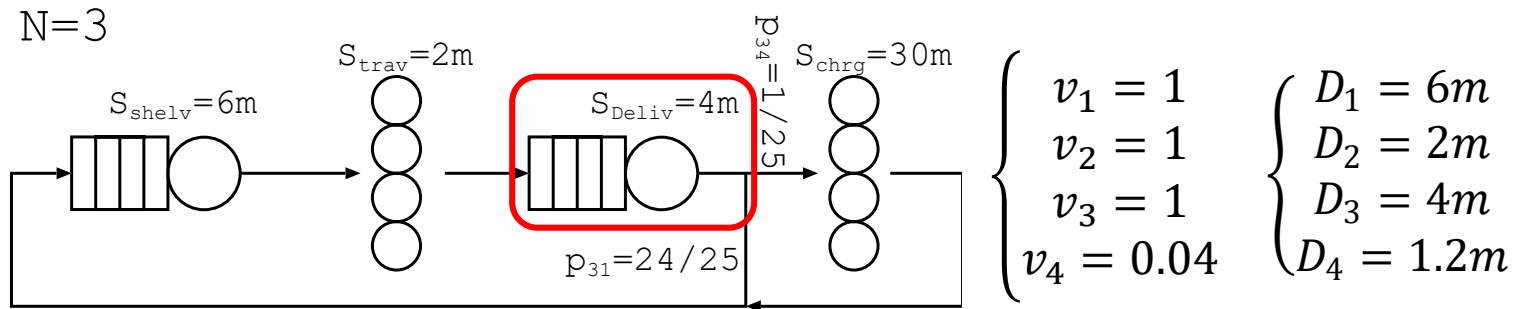
D	150	50	100	Z	30				
N	R1	R2	R3	R	X	N1	N2	N3	
0							0	0	0
1	150		100	300	0,0030303	0,45454545	0,15151515	0,3030303	
2	218,181818		130,30303	398,484848	0,00466761	1,01838755	0,23338048	0,60820368	
3	302,758133		160,820368	513,578501	0,00551898	1,67091671	0,2759491	0,88756473	

8,55964168 H



## Analysis of Motivating Example

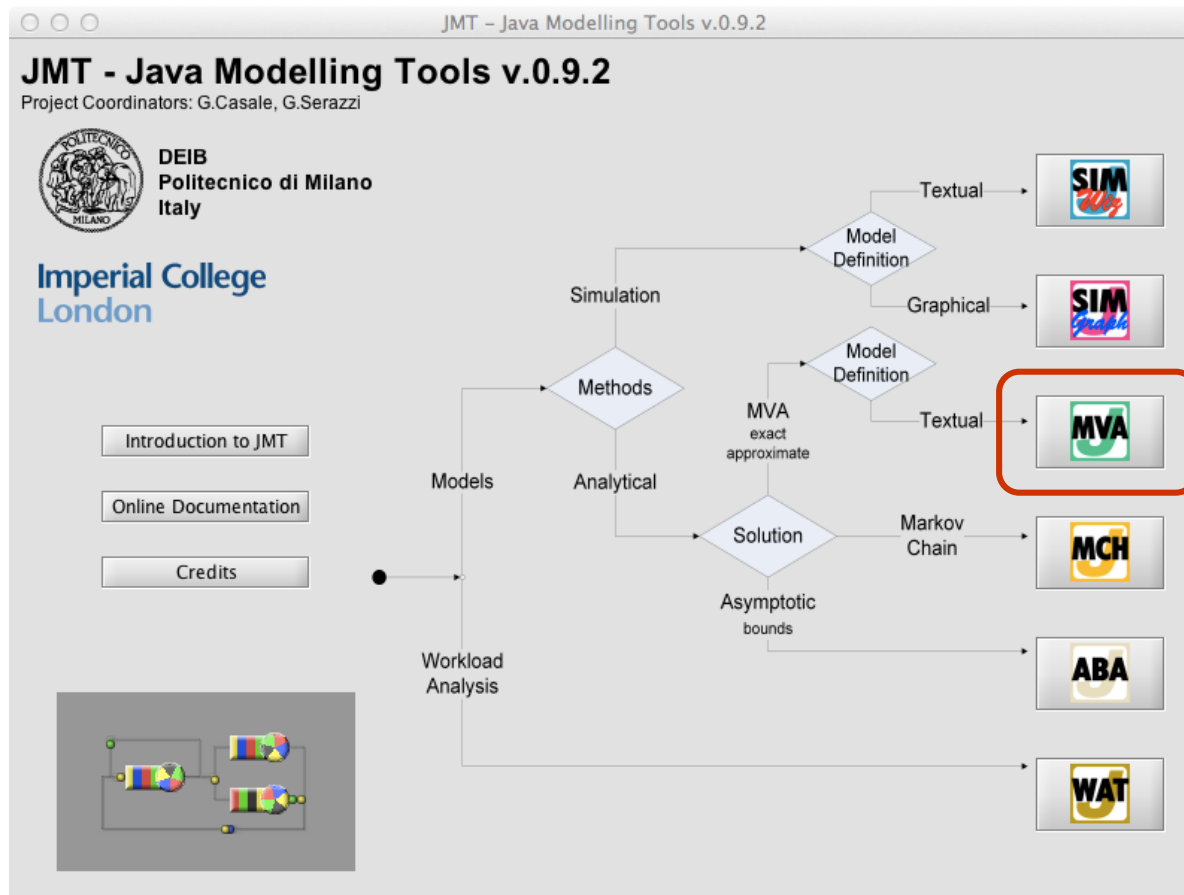
To determine the system throughput, the model is a batch system where station 3 is the reference station. With MVA we determine  $X$ .



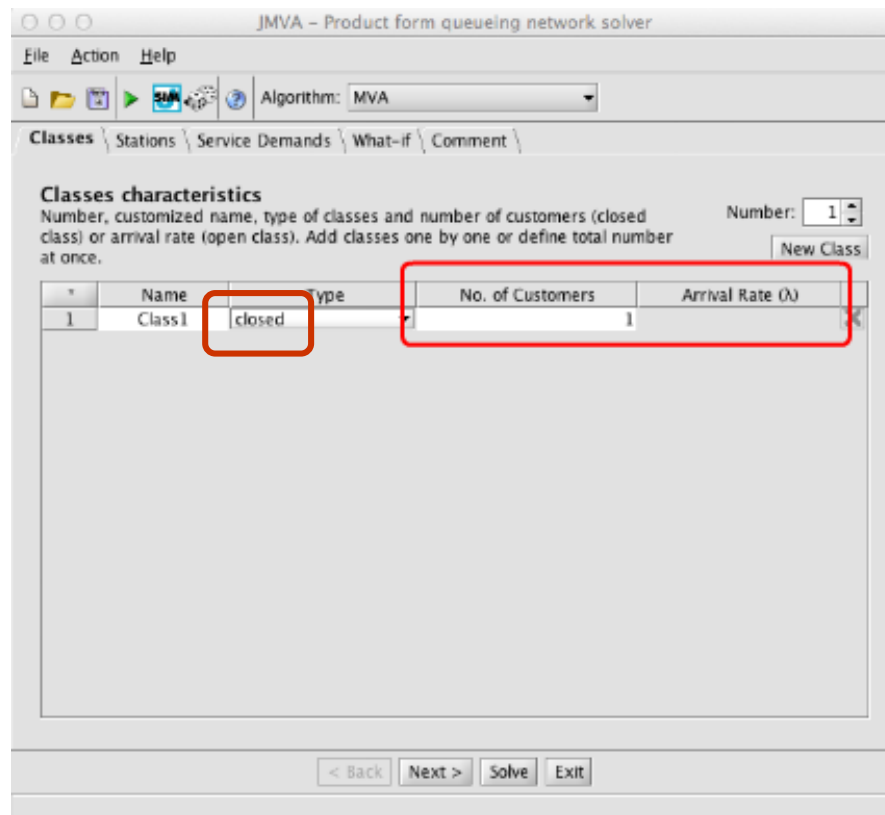
D	6	2	4	1,2						
N	R1	R2	R3	R4	R	X	N1	N2	N3	N4
0								0	0	0
1	6		2	4	1,2	13,2	0,07575758	0,45454545	0,15151515	0,3030303
2	8,72727273		2	5,21212121	1,2	17,1393939	0,11669024	1,01838755	0,23338048	0,60820368
3	12,1103253		2	6,43281471	1,2	21,74314	0,13797455	1,67091671	0,2759491	0,88756473

8,27847311 jobs/h

The JMT tool includes a component to perform analysis of open and closed separable models called *JMVA*.



The user starts selecting the classes used in the model (we will return on this later), and defining whether they are open or closed. In the same page the user specifies either the arrival rate (open models), or the total population (closed models).



JMVA - Product form queueing network solver

File Action Help

Algorithm: MVA

Classes \ Stations \ Service Demands \ What-if \ Comment \

**Classes characteristics**  
Number, customized name, type of classes and number of customers (closed class) or arrival rate (open class). Add classes one by one or define total number at once.

Number: 1

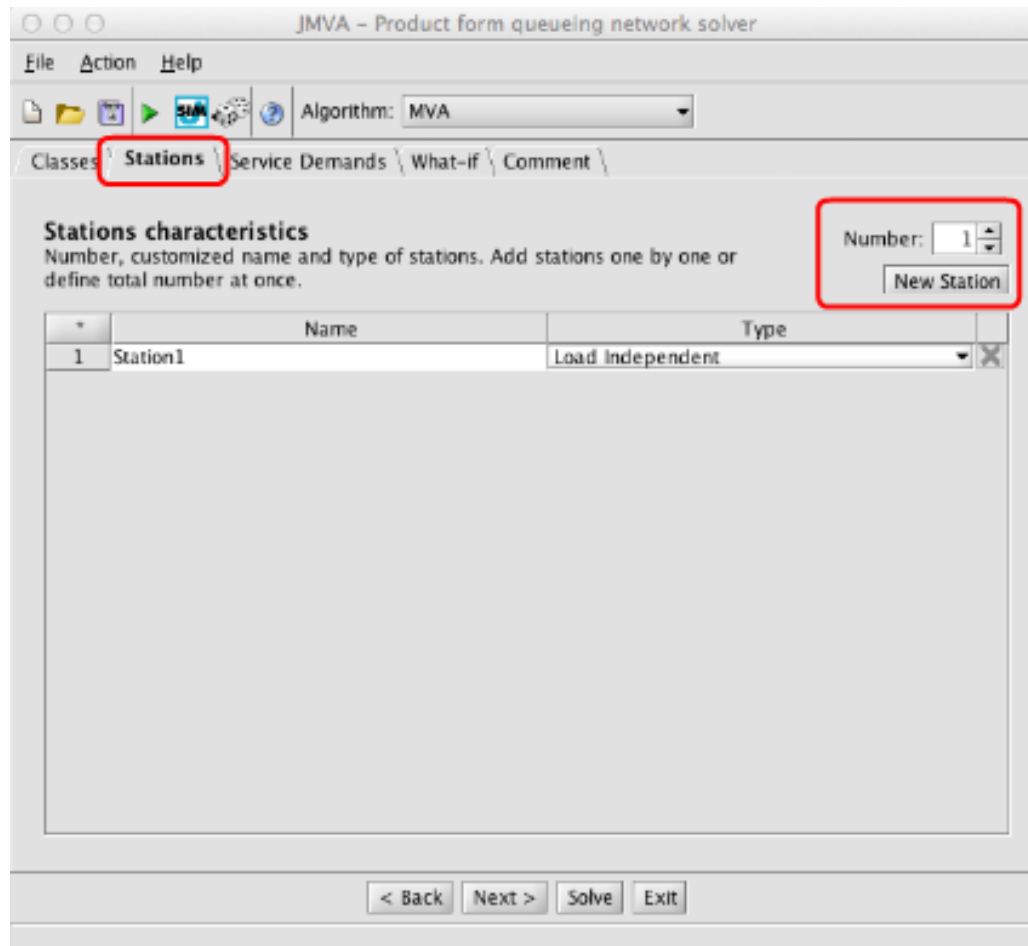
New Class

#	Name	Type	No. of Customers	Arrival Rate (λ)
1	Class1	closed	1	

< Back Next > Solve Exit



Since separable models are fully characterized by their demand, the tool does not ask us the topology of the network, but it just allows us to insert as many stations as needed.



JMVA - Product form queueing network solver

File Action Help

Algorithm: MVA

Classes **Stations** Service Demands What-if Comment

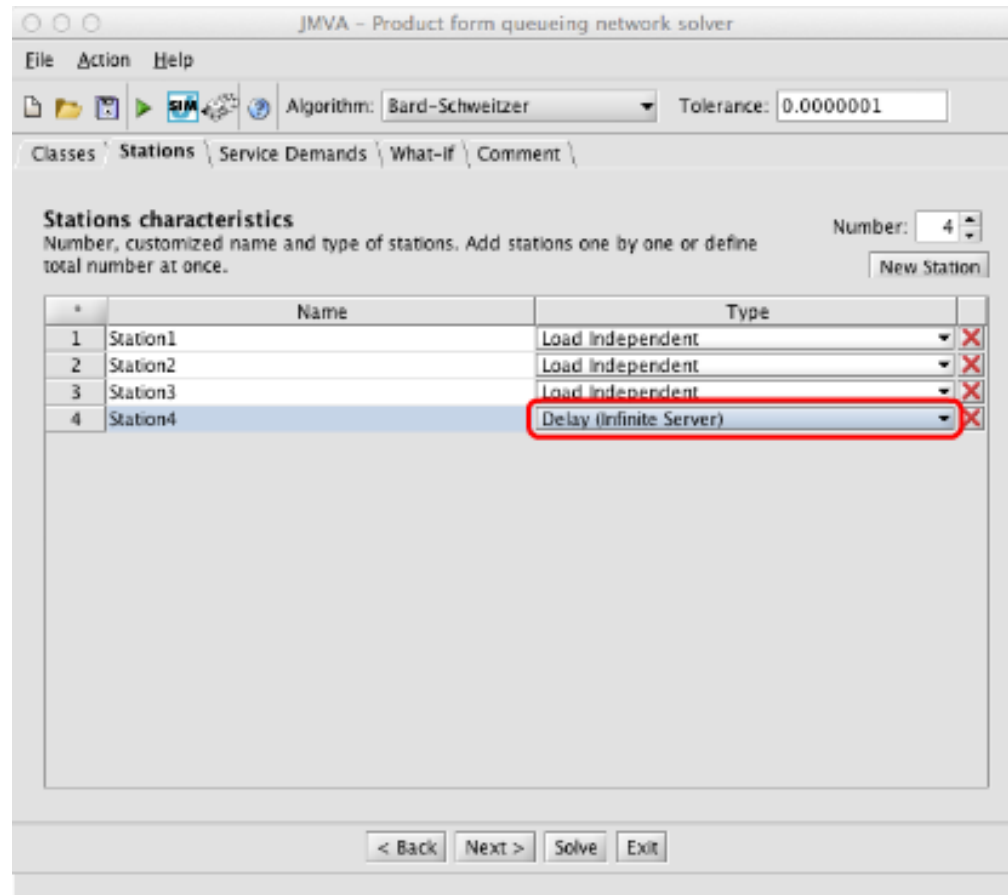
**Stations characteristics**  
Number, customized name and type of stations. Add stations one by one or define total number at once.

Number: 1

	Name	Type
1	Station1	Load Independent

< Back Next > Solve Exit

The tool supports both single server queues (Load independent), and infinite server stations. The latter can be used to insert the terminal station in time-sharing models.



JMVA - Product form queueing network solver

File Action Help

Algorithm: Bard-Schweitzer Tolerance: 0.0000001

Classes Stations Service Demands What-if Comment

**Stations characteristics**

Number, customized name and type of stations. Add stations one by one or define total number at once.

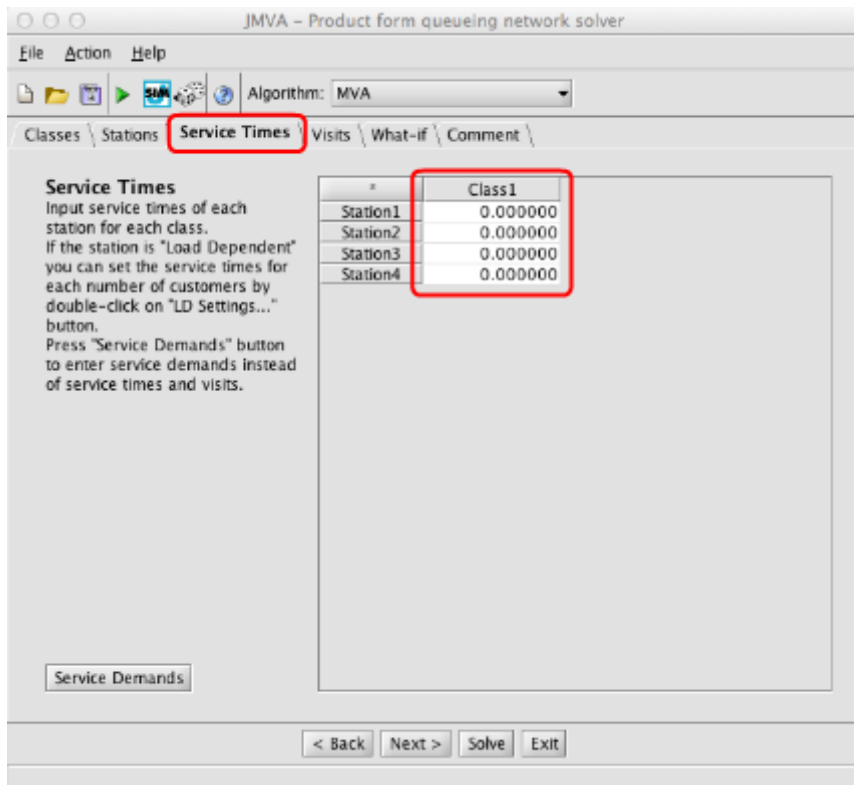
Number: 4 New Station

*	Name	Type
1	Station1	Load Independent
2	Station2	Load Independent
3	Station3	Load Independent
4	Station4	Delay (Infinite Server)

< Back Next > Solve Exit

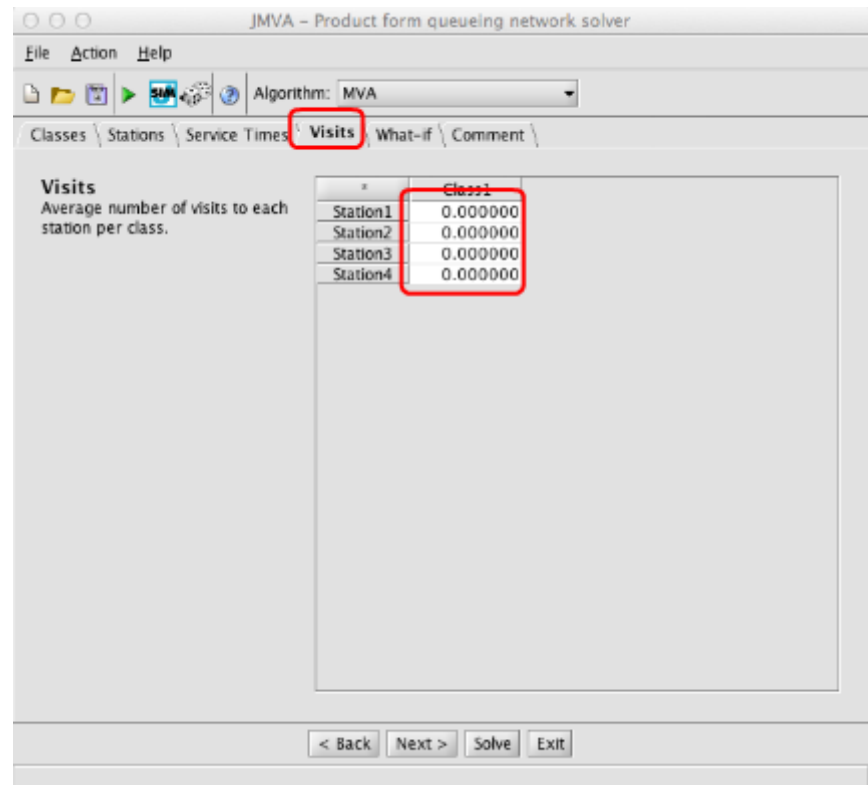
Load dependent stations can be used to add multiple server queues. However, since their definition is a bit complex, we will not consider them in this course.

For each station we then have to specify the average service time, and the visits in the corresponding tabs. Note that in time sharing models, this is also used to specify the think time of the terminal station.



The screenshot shows the 'Service Times' tab in the JMVA software. The 'Algorithm' is set to 'MVA'. The 'Service Times' section contains instructions: 'Input service times of each station for each class. If the station is "Load Dependent" you can set the service times for each number of customers by double-click on "LD Settings..." button. Press "Service Demands" button to enter service demands instead of service times and visits.' Below this is a table with columns 'Station' and 'Class1'. The table contains four rows: Station1 (0.000000), Station2 (0.000000), Station3 (0.000000), and Station4 (0.000000). A 'Service Demands' button is located at the bottom left. Navigation buttons at the bottom include '< Back', 'Next >', 'Solve', and 'Exit'.

Station	Class1
Station1	0.000000
Station2	0.000000
Station3	0.000000
Station4	0.000000



The screenshot shows the 'Visits' tab in the JMVA software. The 'Algorithm' is set to 'MVA'. The 'Visits' section contains instructions: 'Average number of visits to each station per class.' Below this is a table with columns 'Station' and 'Class1'. The table contains four rows: Station1 (0.000000), Station2 (0.000000), Station3 (0.000000), and Station4 (0.000000). Navigation buttons at the bottom include '< Back', 'Next >', 'Solve', and 'Exit'.

Station	Class1
Station1	0.000000
Station2	0.000000
Station3	0.000000
Station4	0.000000

We also have to define the reference station in the corresponding panel.

JMVA - Product Form Queueing Network Solver

File Action Help

Algorithm: MVA

Classes Stations Service Times Visits **Reference Station** What-if Comment

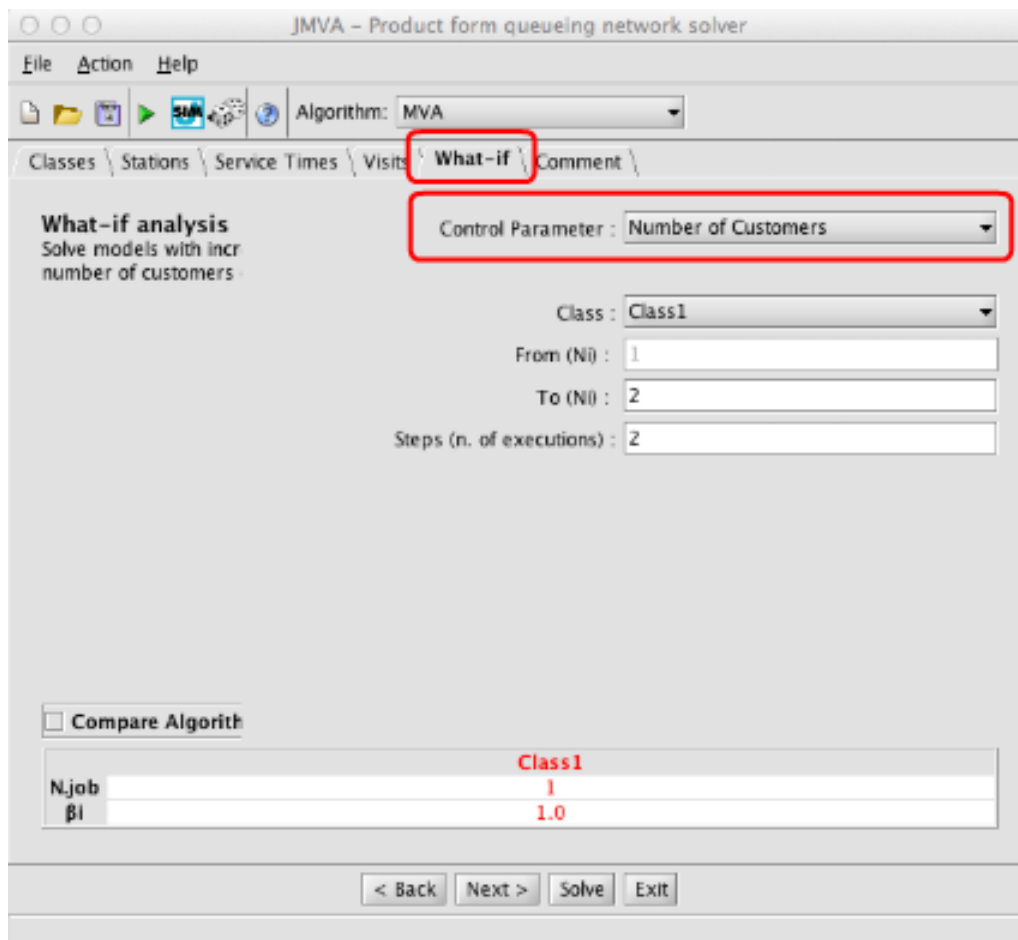
**Reference Station**

The station is used to compute the system throughput and the system response time for each **closed class**.  
Performance metrics of **open classes** are always computed with respect to the **arrival process**. Visits at the Reference station can not be Zero.  
**WARNING:** the reference station for all closed classes is forced to be the same station.

Class	Station
Class1	Station1

< Back Next > Solve Exit

If we want to perform several studies changing the workload, we can setup a range of experiments in the *What-if* tab.



JMVA - Product form queueing network solver

File Action Help

Algorithm: MVA

Classes \ Stations \ Service Times \ Visits \ **What-if** \ Comment

**What-if analysis**  
Solve models with incr number of customers

Control Parameter : Number of Customers

Class : Class1

From (N) : 1

To (N) : 2

Steps (n. of executions) : 2

☐ Compare Algorithm

	Class1
Njob	1
β1	1.0

< Back Next > Solve Exit

JMVA always computes all the performance indices, and present them in a set of panels.

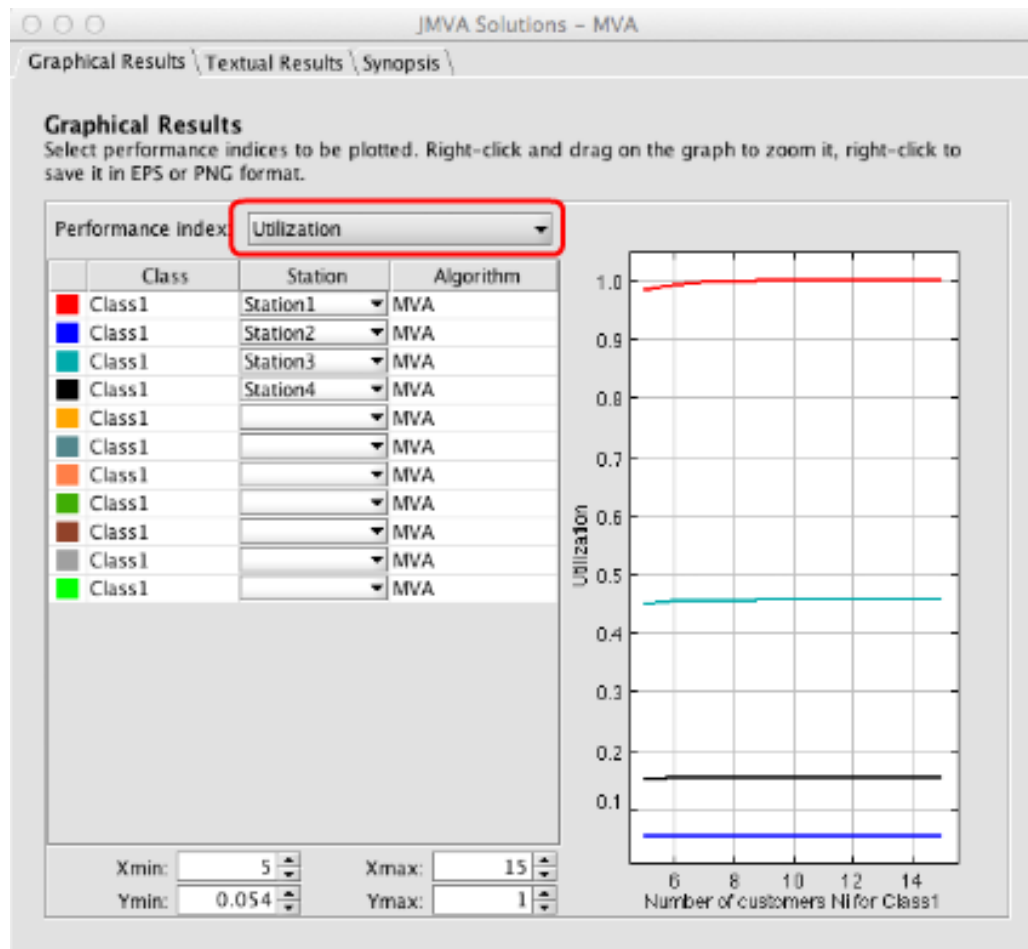
JMVA Solutions - MVA

Throughput \ Number of Customers \ Residence Times \ Utilization \ System Power \ Synopsis \

**Throughput**  
Throughput for each class at each station.

*	Aggregate	Class1
Aggregate	0.017066	0.017066
Station1	0.017066	0.017066
Station2	0.017066	0.017066
Station3	0.017066	0.017066
Station4	0.017066	0.017066

If *What-if* analysis has been enabled, the tool allows to plot and compare the various performance indices.





## Advanced Queuing Network Features

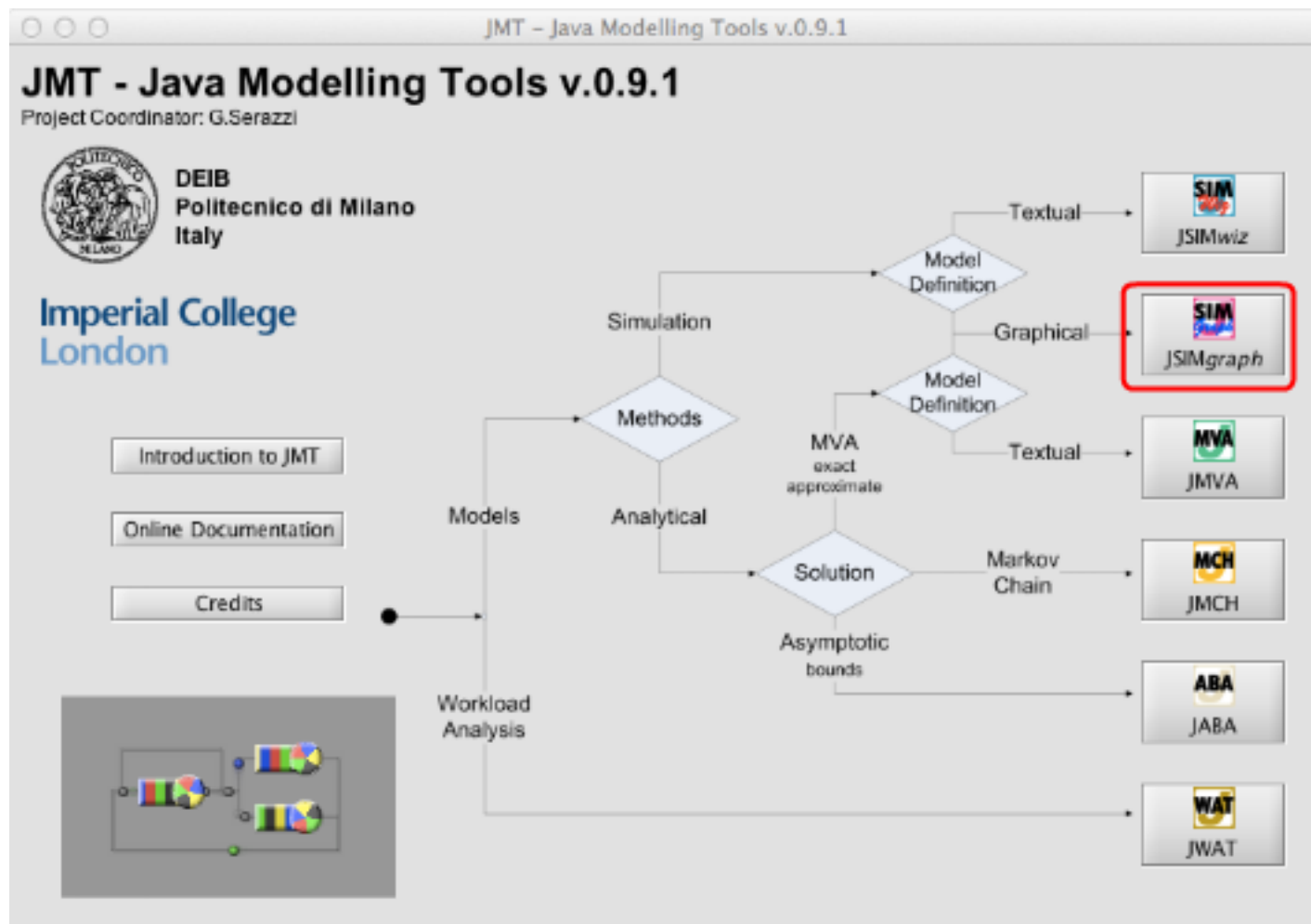
When a model is not separable, it very rarely enjoys analytical solutions or numerical techniques to compute the relevant performance indices.

In most of the cases, models exploiting these properties are solved via *discrete event simulation*.



# Advanced Queuing Network features in JMT

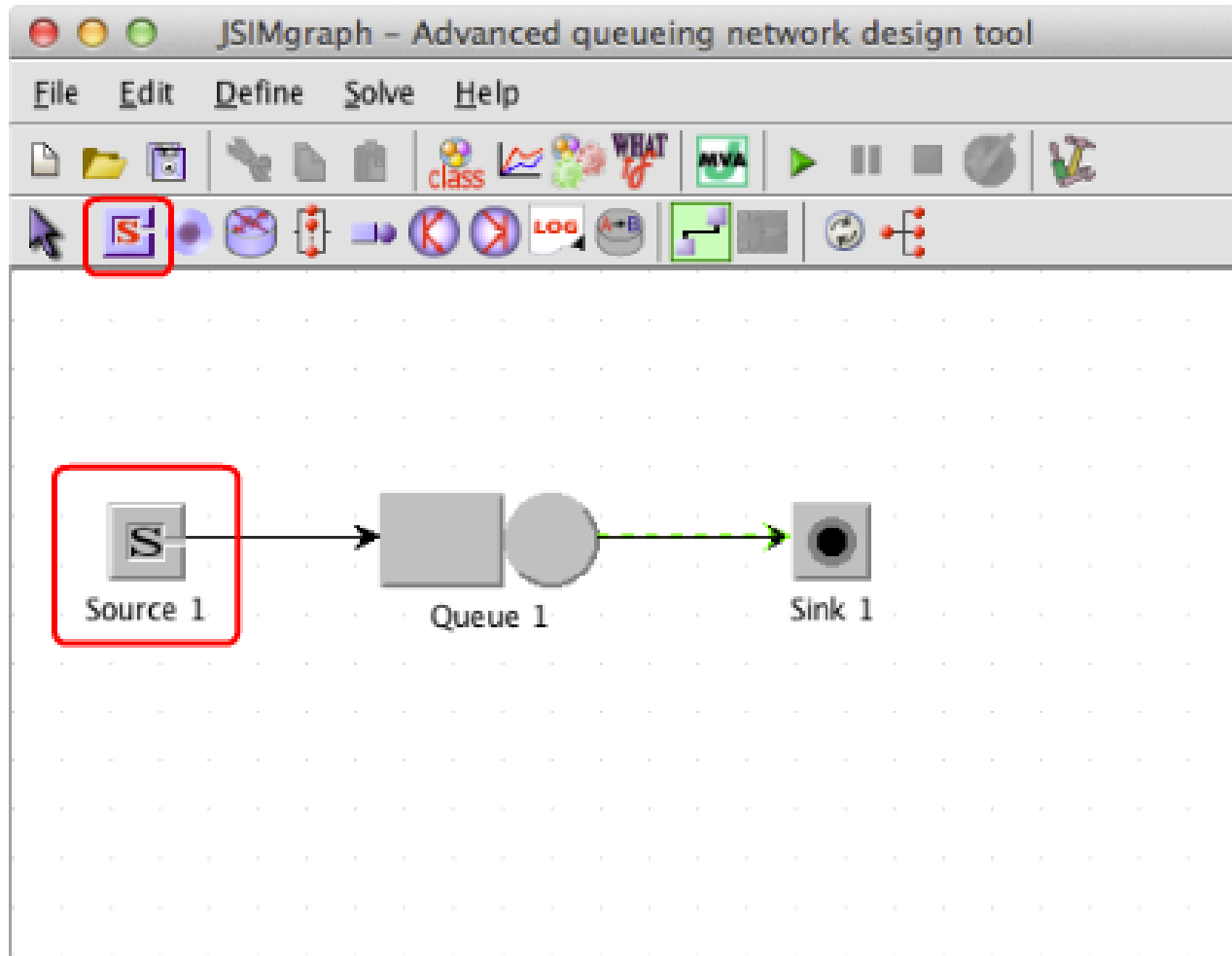
We will show these features in the *JSIMgraph* tool of JMT.





## Single queues in JMT

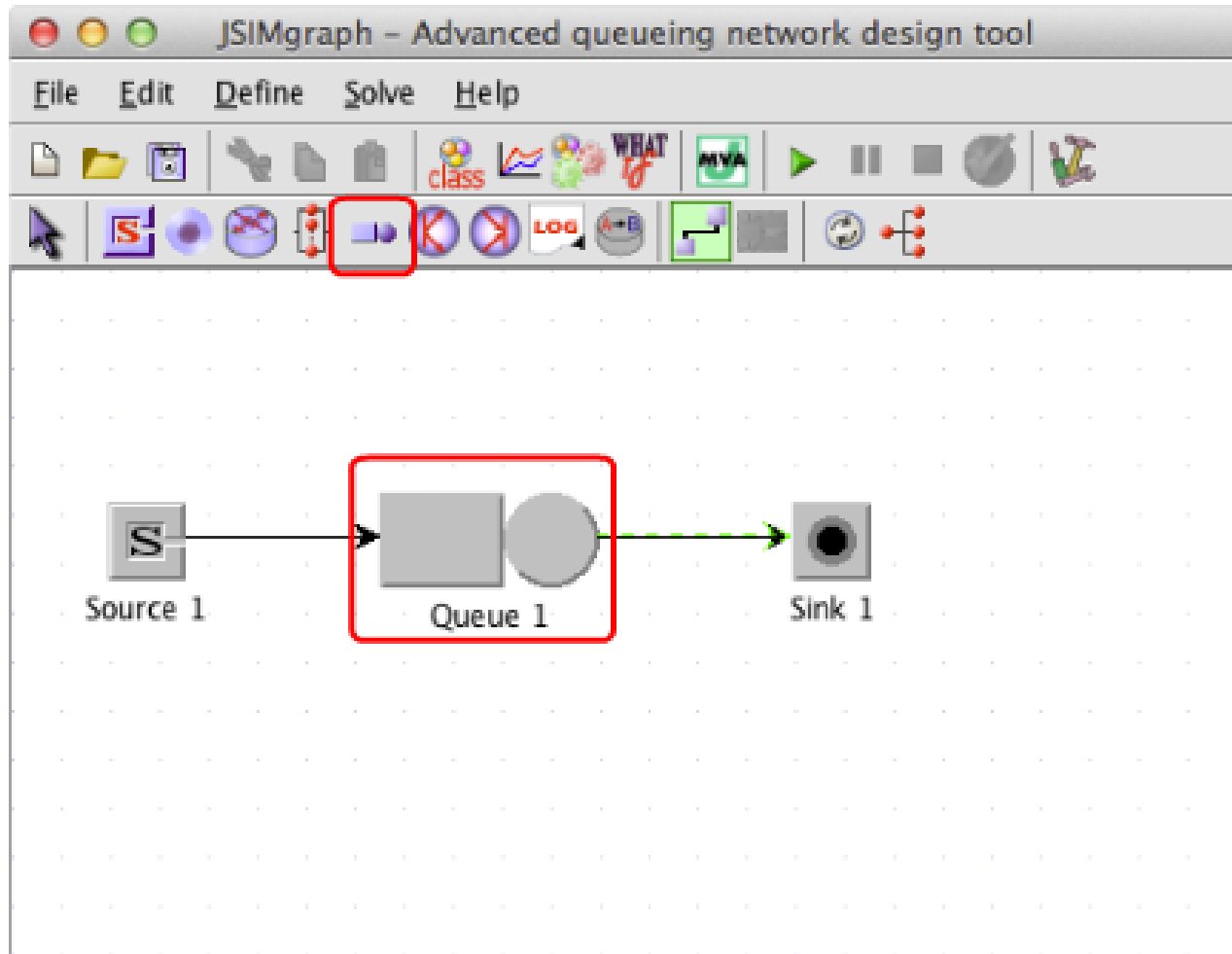
Let's start with a simple (open) single queue model. It is composed by three elements: *a Source node*, representing the arrivals.





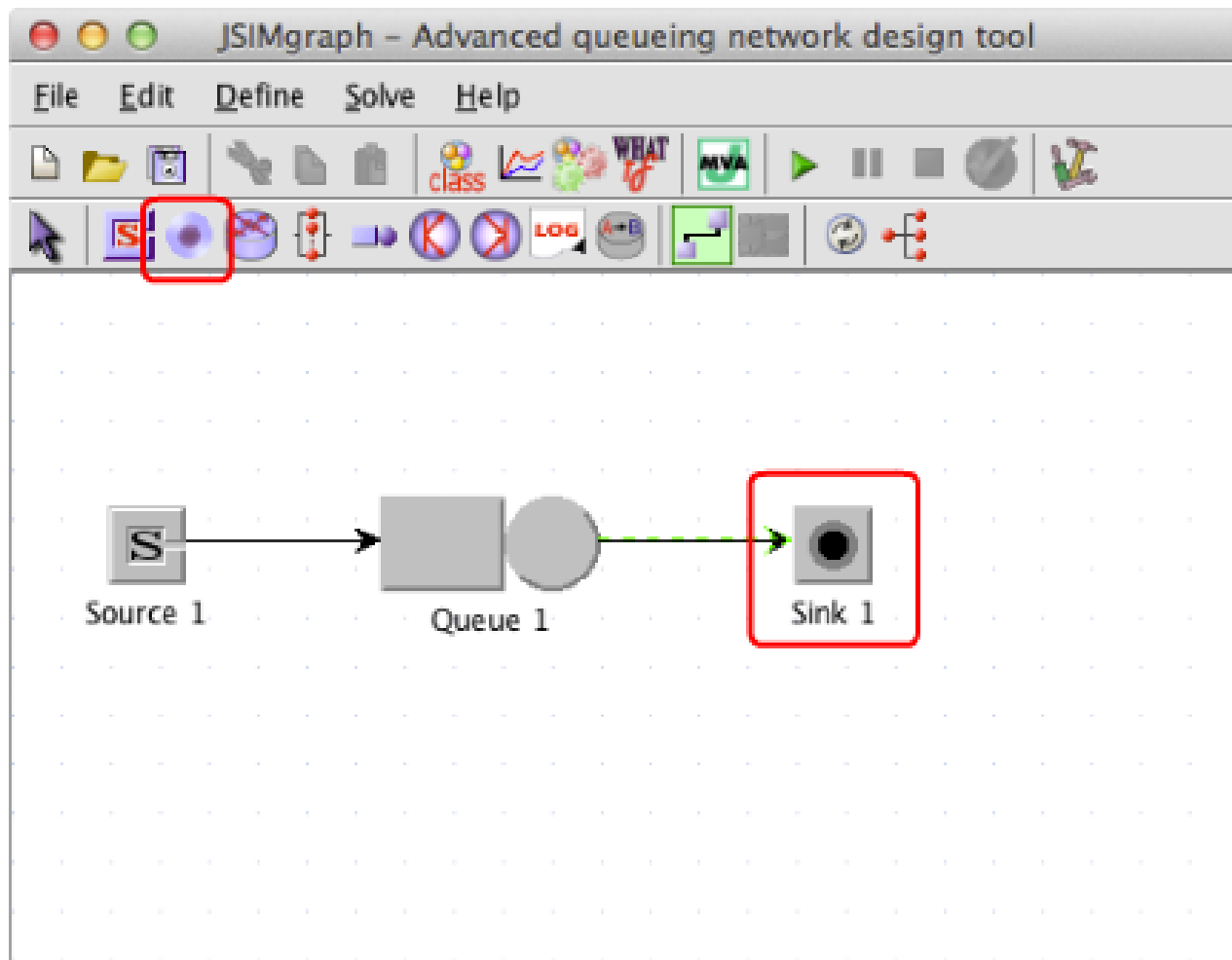
## Single queues in JMT

A *Queue node* represents the single queuing station.



## Single queues in JMT

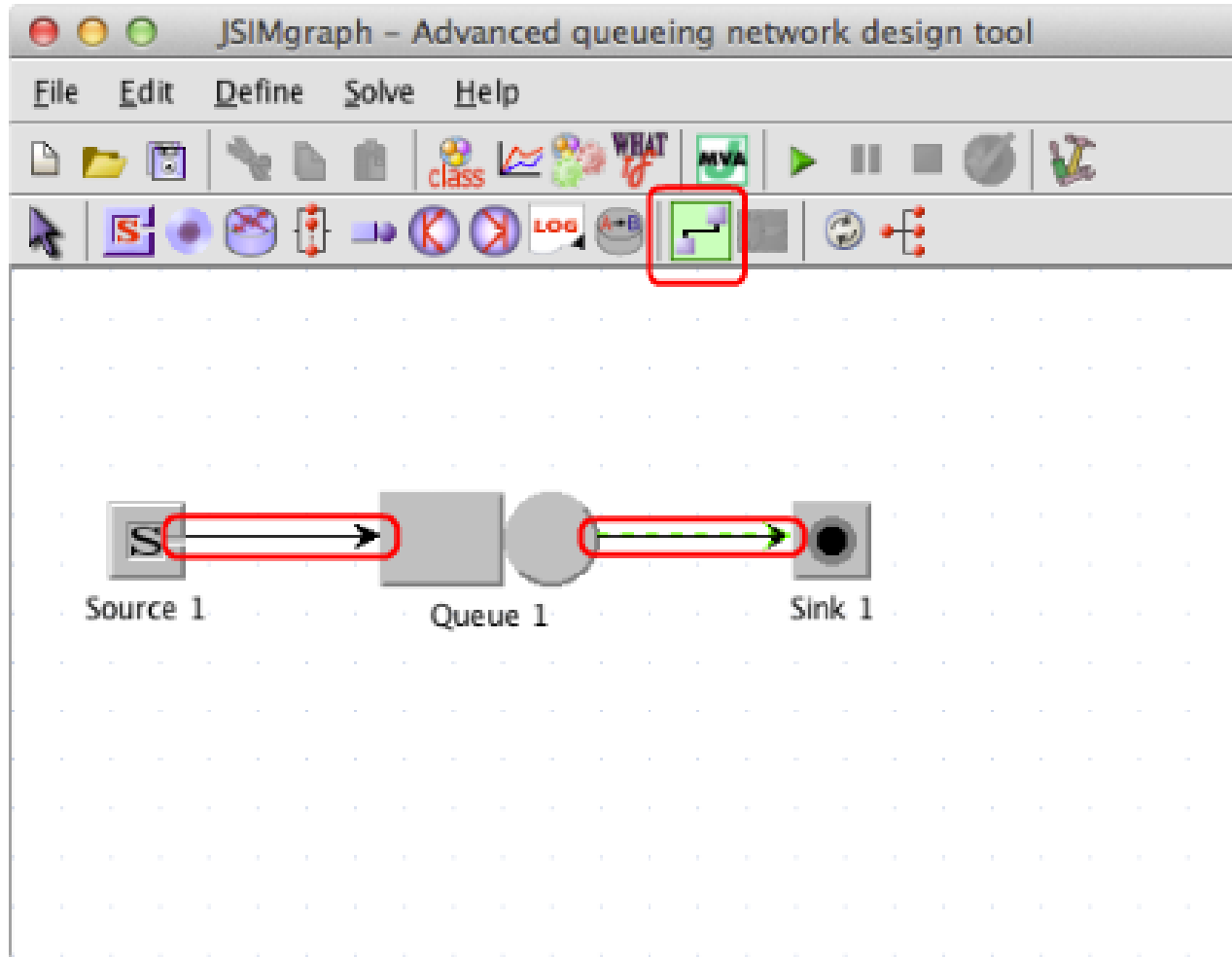
Finally, a Sink node is required to allow jobs, which have finished their service, to leave the system.





## Single queues in JMT

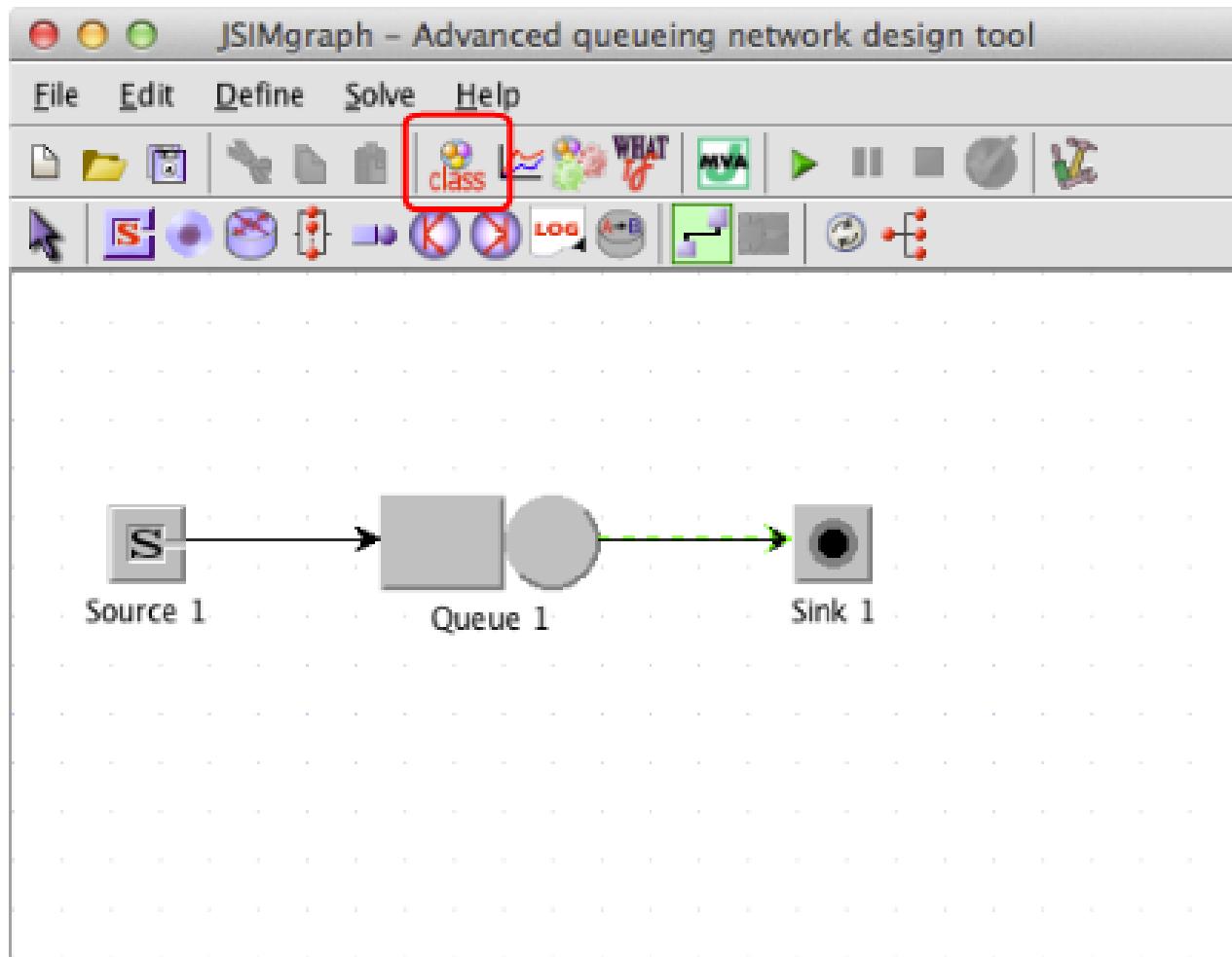
Connections define the flow of jobs from the source to the queue, and from the queue to the sink.





## Single queues in JMT

Since JMT can support different types of jobs (we will return on this later), a new customer class must be defined.





## Single queues in JMT

For jobs arriving from an external source, the type of this class must be defined as *Open*.

Define customer classes

**Classes Characteristics**  
Define type, name and parameters for each customer class.

Add Class

Classes: 1

Color	Name	Type	Priority	Population	Interarrival Time Distribut...		Reference Station	
	Class1	Open	0		exp(1)	Edit	Source 1	X

Done



## Single queues in JMT

The properties of the arrival process must be defined in the *Inter-arrival time distribution* column of the corresponding class. We will see how to define the arrival process pressing the Edit button in a few minutes.

Define customer classes

**Classes Characteristics**  
Define type, name and parameters for each customer class.

Classes: 1

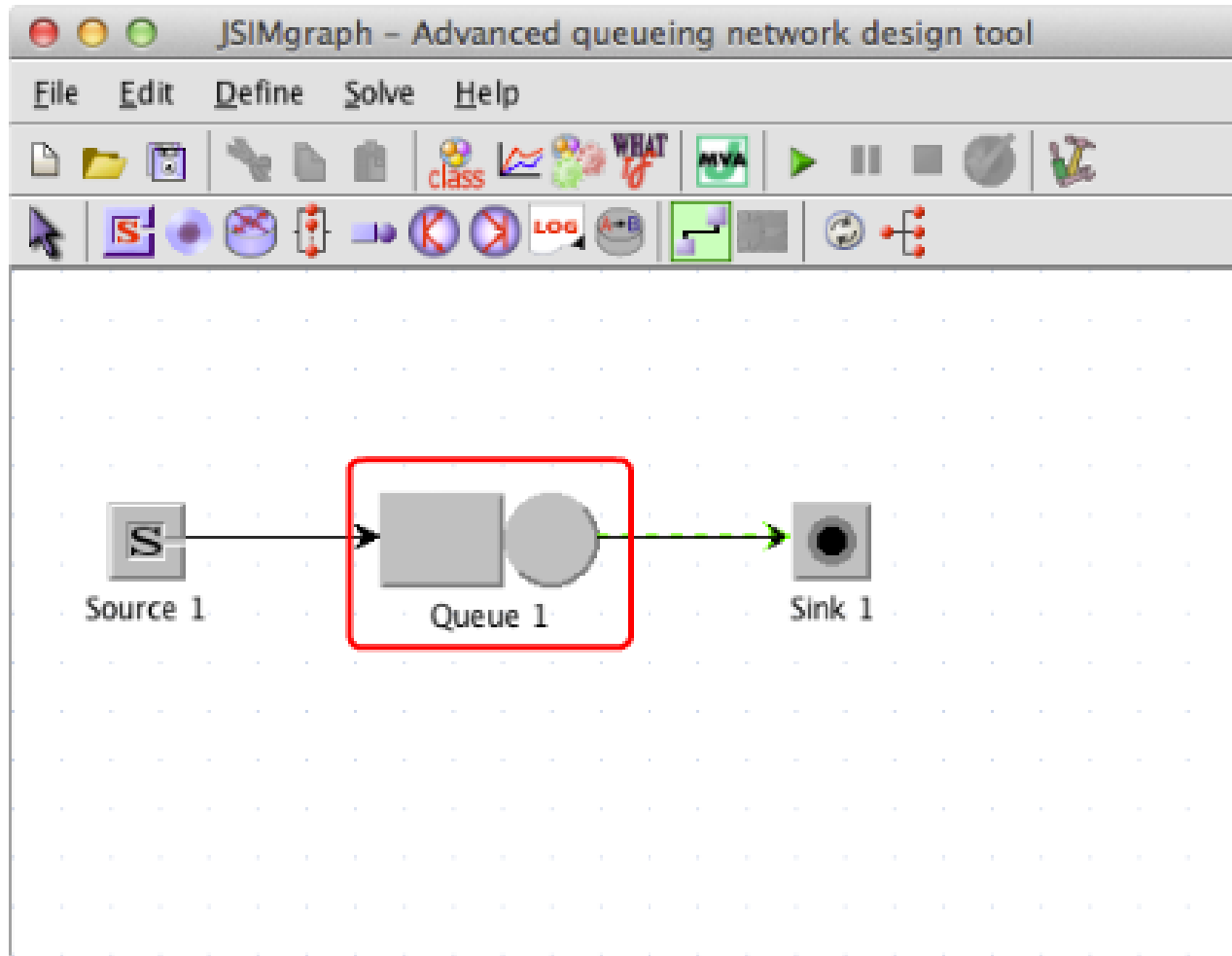
Color	Name	Type	Priority	Population	Interarrival Time Distribut...	Reference Station
	Class1	Open	0		exp(1) <b>Edit</b>	Source 1

Done



## Single queues in JMT

Properties for both the queue and service section can be defined by double-clicking on the queue ...





## Single queues in JMT

then working respectively in the *Queue Section*

Editing Queue 1 Properties...

Station Name  
Station Name: Queue 1

Queue 1 Parameters Definition

Queue Section Service Section Routing Section

Capacity

☒ infinite

☐ finite

max no. customers (queue+service) 1

Queue Policy

Station queue policy: Non-preemptive Scheduling

Class	Queue Policy	Drop Rule
Class1	FCFS	Infinite Capacity

Done



## Single queues in JMT

and in the *Service Section*.

Editing Queue 1 Properties...

Station Name  
Station Name: Queue 1

Queue 1 Parameters Definition

Queue Section **Service Section** Routing Section

Number of Servers  
Number: 1

Service Time Distributions

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

Done



## Single queues in JMT

Service time distribution can be defined pressing the *Edit* button.

Editing Queue 1 Properties...

Station Name  
Station Name: Queue 1

Queue 1 Parameters Definition

Queue Section \ Service Section \ Routing Section \

Number of Servers  
Number: 1

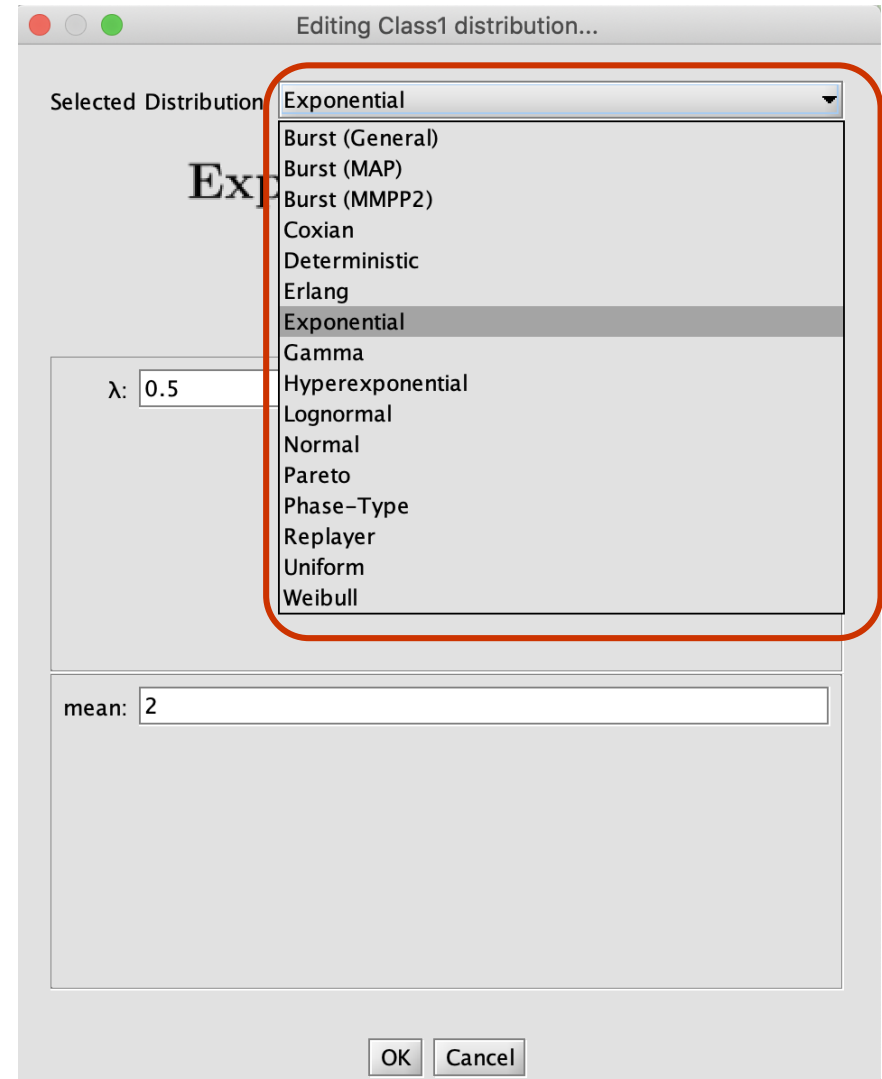
Service Time Distributions

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

Done



Several inter-arrival time and service distributions are possible:





Conventional distributions are characterized by their parameters, and are the following.

Editing Class1 distribution...

Selected Distribution: Exponential

Exp

- Burst (General)
- Burst (MAP)
- Burst (MMPP2)
- Coxian
- Deterministic
- Erlang
- Exponential
- Gamma
- Hyperexponential
- Lognormal
- Normal
- Pareto
- Phase-Type
- Replayer
- Uniform
- Weibull

$\lambda$ : 0.5

mean: 2

OK Cancel

Editing Class1 distribution...

Selected Distribution: Exponential

Exponential [exp( $\lambda$ )]:

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

$\lambda$ : 0.5

mean: 2

OK Cancel

Editing Class1 distribution...

Selected Distribution: Hyperexponential

Hyperexponential [hyp( $p, \lambda_1, \lambda_2$ )]:

$$f(x) = p \lambda_1 e^{-\lambda_1 x} + (1 - p) \lambda_2 e^{-\lambda_2 x}$$

$p$ : 0.2

$\lambda_1$ : 0.1

$\lambda_2$ : 0.4

mean: 4

$c$ : 1.457737973711

OK Cancel



Some of them allow alternative parameterizations: for example the Exponential distribution can be defined through either the rate (useful for the arrivals), or with their average (better suited for services).

Editing Class1 distribution...

Selected Distribution: Exponential

Exponential [exp( $\lambda$ )]:

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

$\lambda$ : 0.5

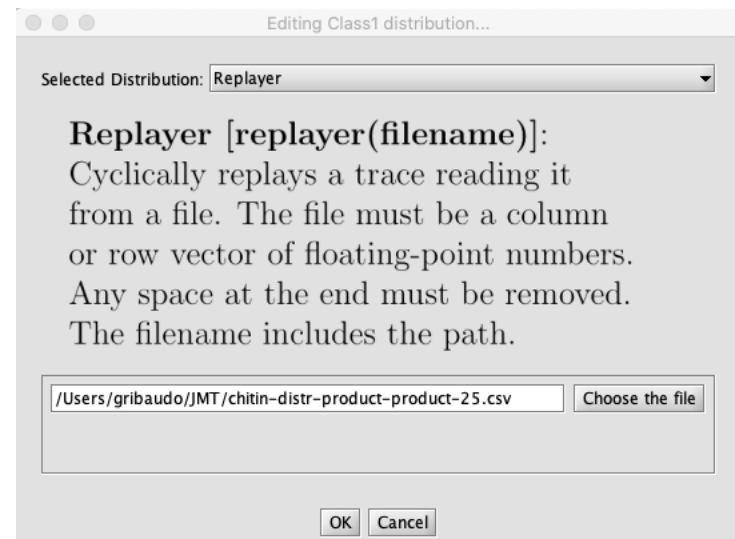
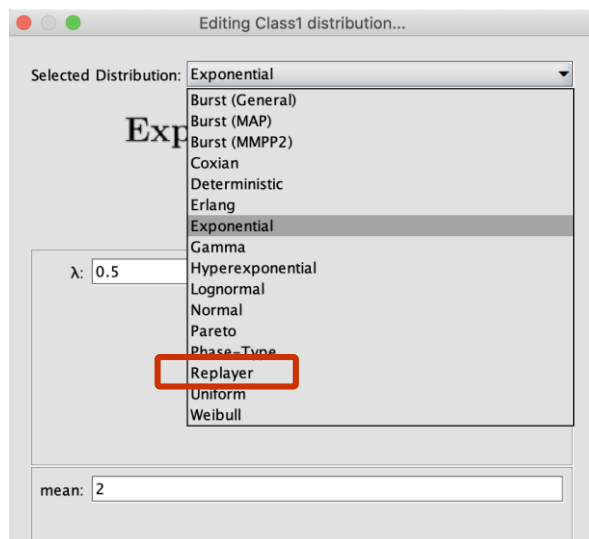
mean: 2

OK Cancel



Repeater allows to read samples from a file. It can be used for:

- Simulating a real system using data directly taken from its logs.
- Including unsupported distributions which have been previously generated by an external tool, and that have been saved in a file as a large set lot of samples.



Please note that using directly log files, instead of fitting the distribution and generating an arbitrarily large set of samples, has a lot of limitations. It must thus be generally avoided and ***used only in very special cases*** (i.e. don't do it for the exam).





## Multiple servers

In JMT, multiple servers are defined by an appropriate parameter in the service section of a station.

Editing Queue 1 Properties...

Station Name  
Station Name: Queue 1

Queue 1 Parameters Definition

Queue Section Service Section Routing Section

Number of Servers  
Number: 1

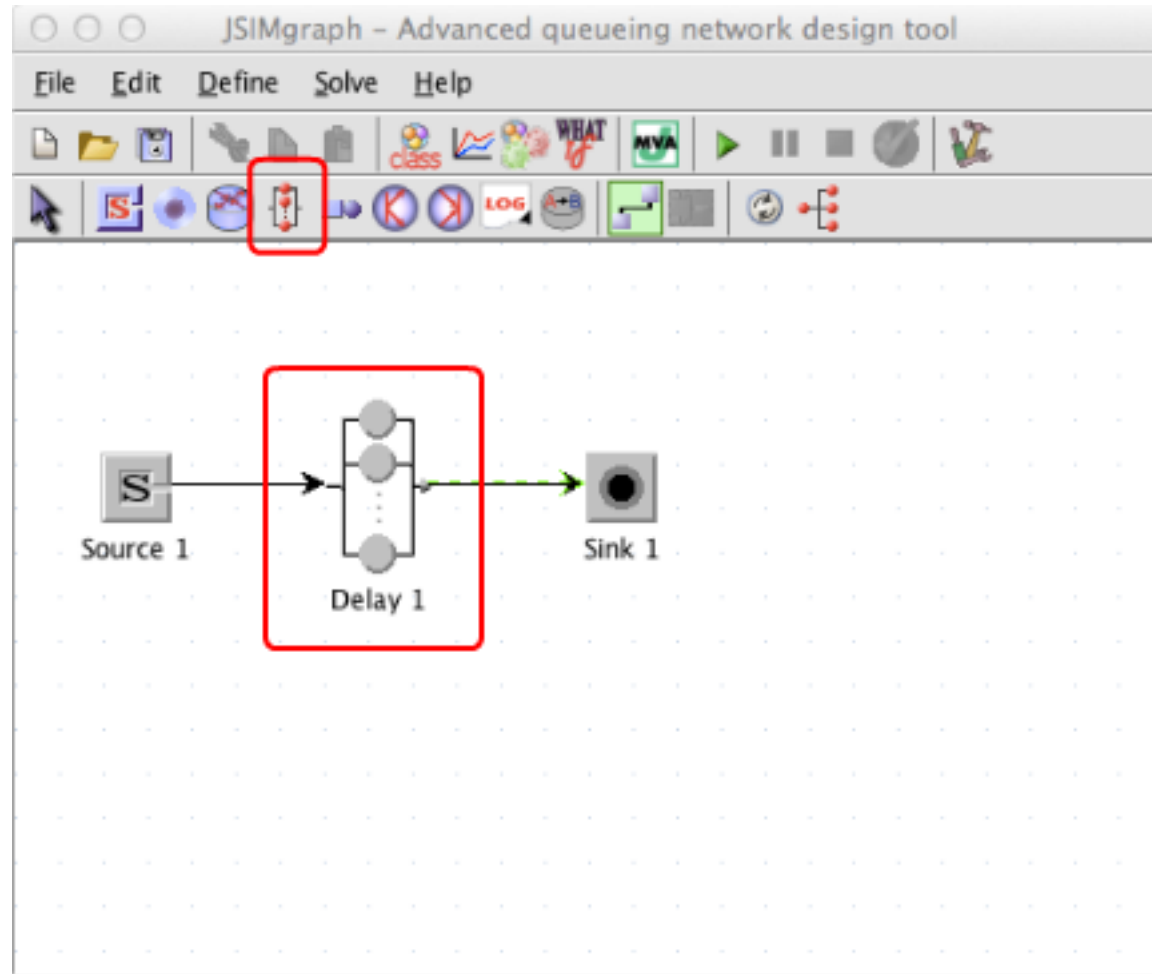
Service Time Distributions

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

Done

# Infinite servers

Infinite servers are inserted using a specific primitive called "*Delay station*".





## Infinite servers

Delay stations are characterized only by their service time (*waiting time*).

Editing Delay 1 Properties...

Station Name  
Station Name: Delay 1

Delay 1 Parameters Definiton

Service Section \ Routing Section \

Number of Servers  
Number: ∞

Service Time Distributions

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

Done



## Finite capacity

Losses and blocking are supported by specifying a finite capacity in the queuing section of node.

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) 1

Queue Policy

Station queue policy: Non-preemptive Scheduling

Class	Queue Policy	Drop Rule
Class1	FCFS	Drop

Drop Rule dropdown: Drop, Waiting Queue (no drop), BAS blocking, Drop

Done



## Finite capacity

The choice between loss or blocking is performed in the appropriate column on the table in the central part of the station properties window.

JMT supports blocking for models in which the source is another station, and it mainly focus on the BAS mechanism.

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) 1

Queue Policy

Station queue policy: Non-preemptive Scheduling

Class	Queue Policy	Drop Rule
Class1	FCFS	Drop

Done



The queuing disciplines supported by JMT includes FCFS, LCFS (both preemptive and non-preemptive) and various flavors of PS.

Editing Queue 1 Properties...

Station Name  
Station Name: Queue 1

Queue 1 Parameters Definition

Queue Section \ Service Section \ Routing Section \

Capacity

☐ infinite

☒ finite

max no. customers (queue+service) 1

Queue Policy

Station queue policy: Non-preemptive Scheduling

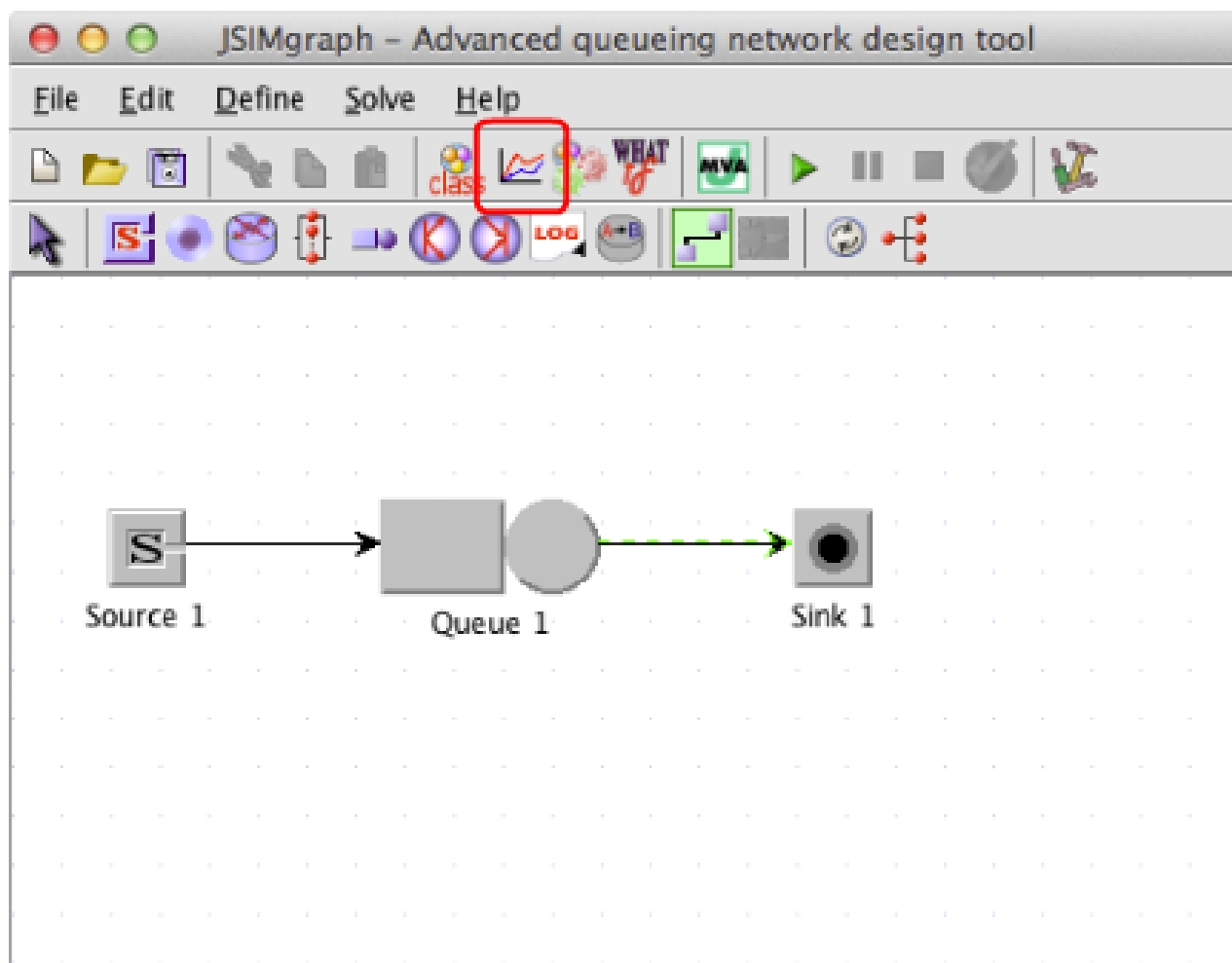
Class	Queue Policy	Drop Rule
Class1	FCFS	Drop

Done



## Single queues in JMT

In order to produce the results, *Performance Indices* must be defined.





## Single queues in JMT

Performance indices include average number of jobs, response time, utilization and throughput.

Define performance indices

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

---Select an index---

Performance Index	Class	Station/Region	Conf.Int.	Max Rel.Err.	
Number of Customers	Class1	Queue 1	0.99	0.03	X
Response Time	Class1	Queue 1	0.99	0.03	X
Utilization	Class1	Queue 1	0.99	0.03	X
Throughput	Class1	Queue 1	0.99	0.03	X

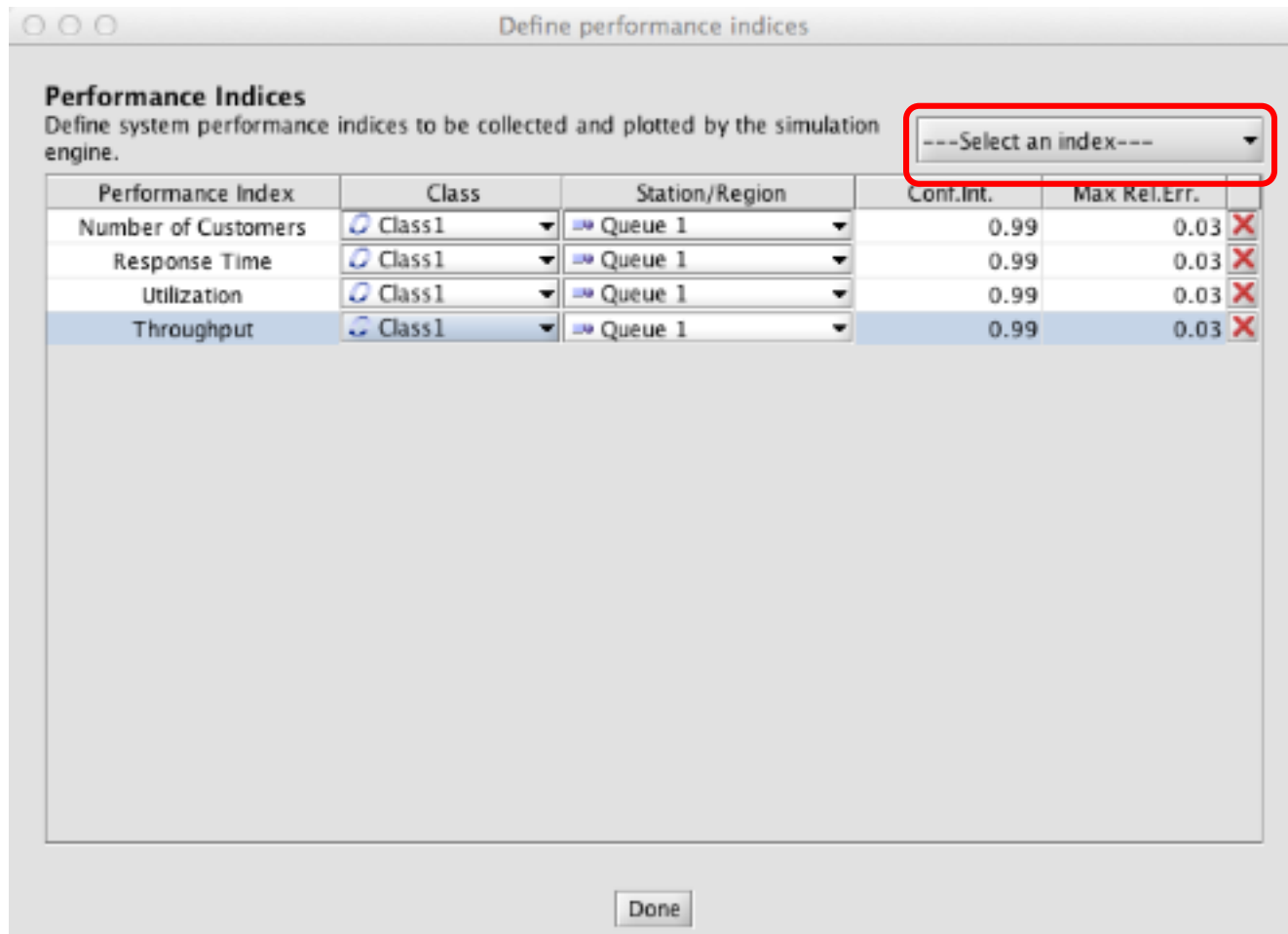
Done





## Single queues in JMT

They are selected from the drop down menu on the top-right corner of the window...





## Single queues in JMT

... and further configured in the central columns. In particular, the centre column can limit the metrics to the selected nodes only.

Define performance indices

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

---Select an index---

Performance Index	Class	Station/Region	Conf.Int.	Max Rel.Err.	
Number of Customers	Class1	Queue 1	0.99	0.03	X
Response Time	Class1	Queue 1	0.99	0.03	X
Utilization	Class1	Queue 1	0.99	0.03	X
Throughput	Class1	Queue 1	0.99	0.03	X

Done



## Single queues in JMT

For each index, a different *Confidence Level* and *Maximum Relative Error* can be defined.

Define performance indices

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

---Select an index---

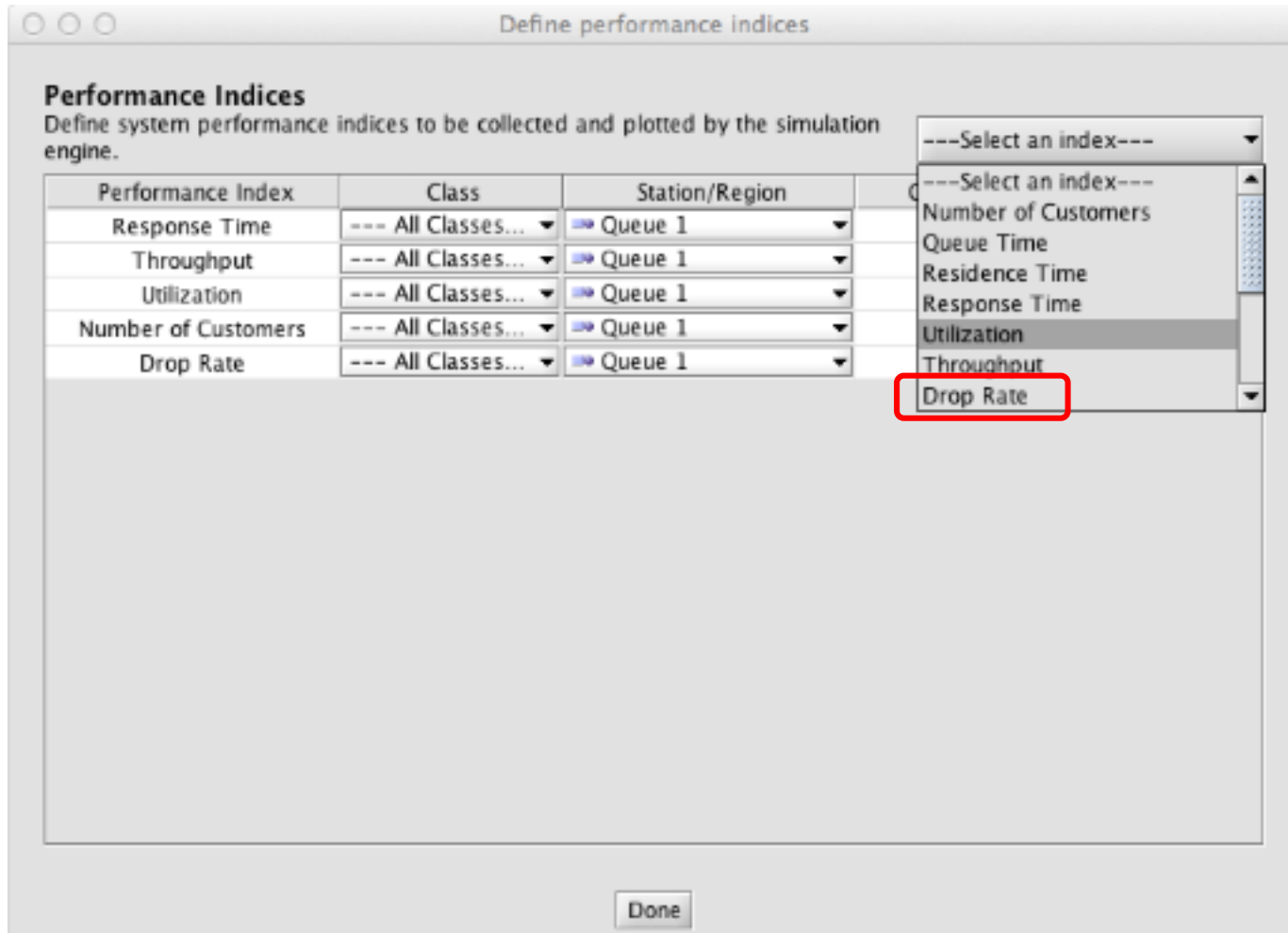
Performance Index	Class	Station/Region	Conf.Int.	Max Rel.Err.	
Number of Customers	Class1	Queue 1	0.99	0.03	X
Response Time	Class1	Queue 1	0.99	0.03	X
Utilization	Class1	Queue 1	0.99	0.03	X
Throughput	Class1	Queue 1	0.99	0.03	X

Done



## Finite capacity: losses

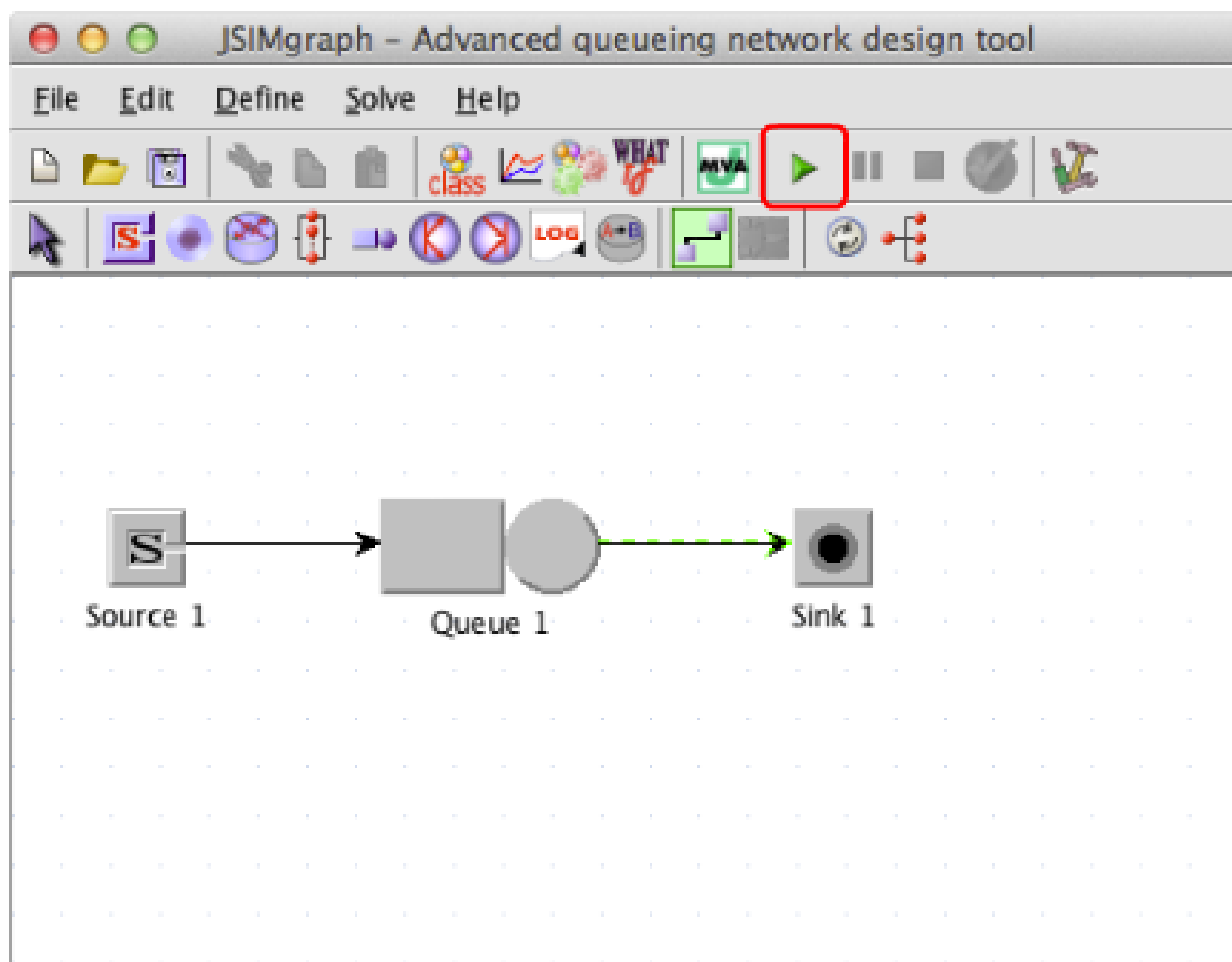
For model model with losses, the **drop rate** can be computed as well.





## Single queues in JMT

Simulation can be run with the “*Play*” button.

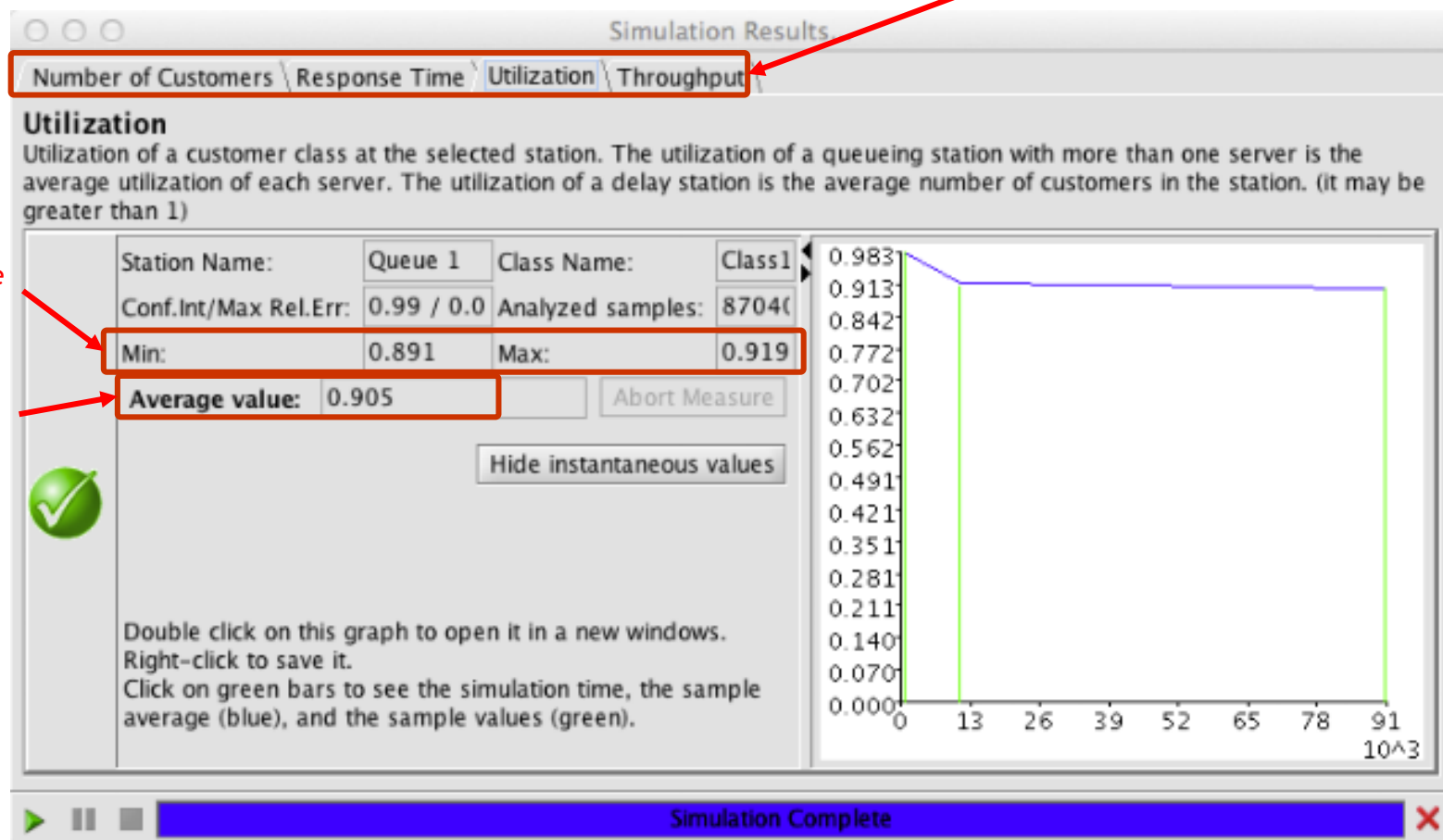




## Single queues in JMT

At the end of the simulation, JMT outputs the values of the selected performance indices.

Tabs on the top selects the different types of performance indices

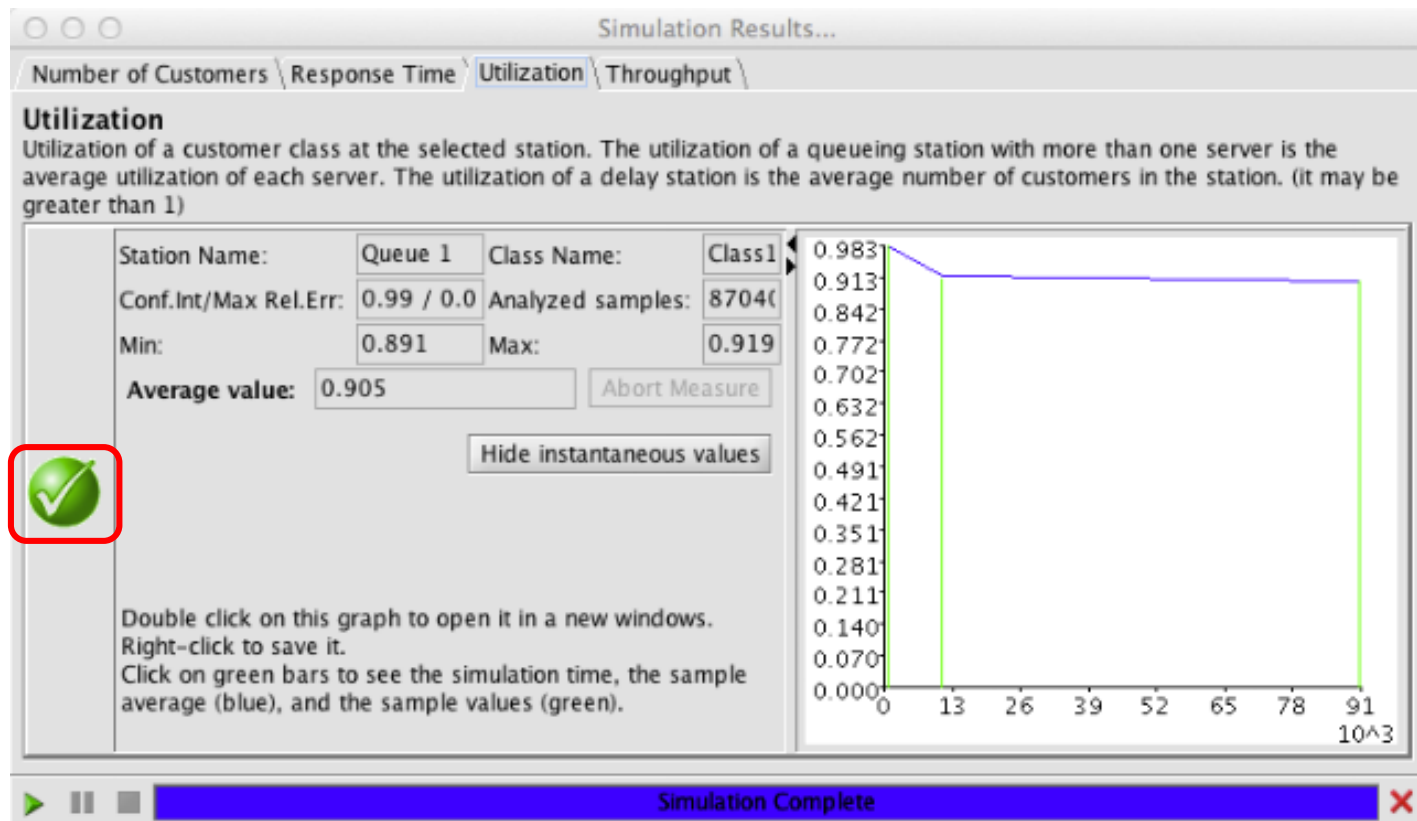




## Single queues in JMT

Simulation ends either when all the confidence intervals have reached the desired level of accuracy, or when a limit condition (user definable) has been reached.

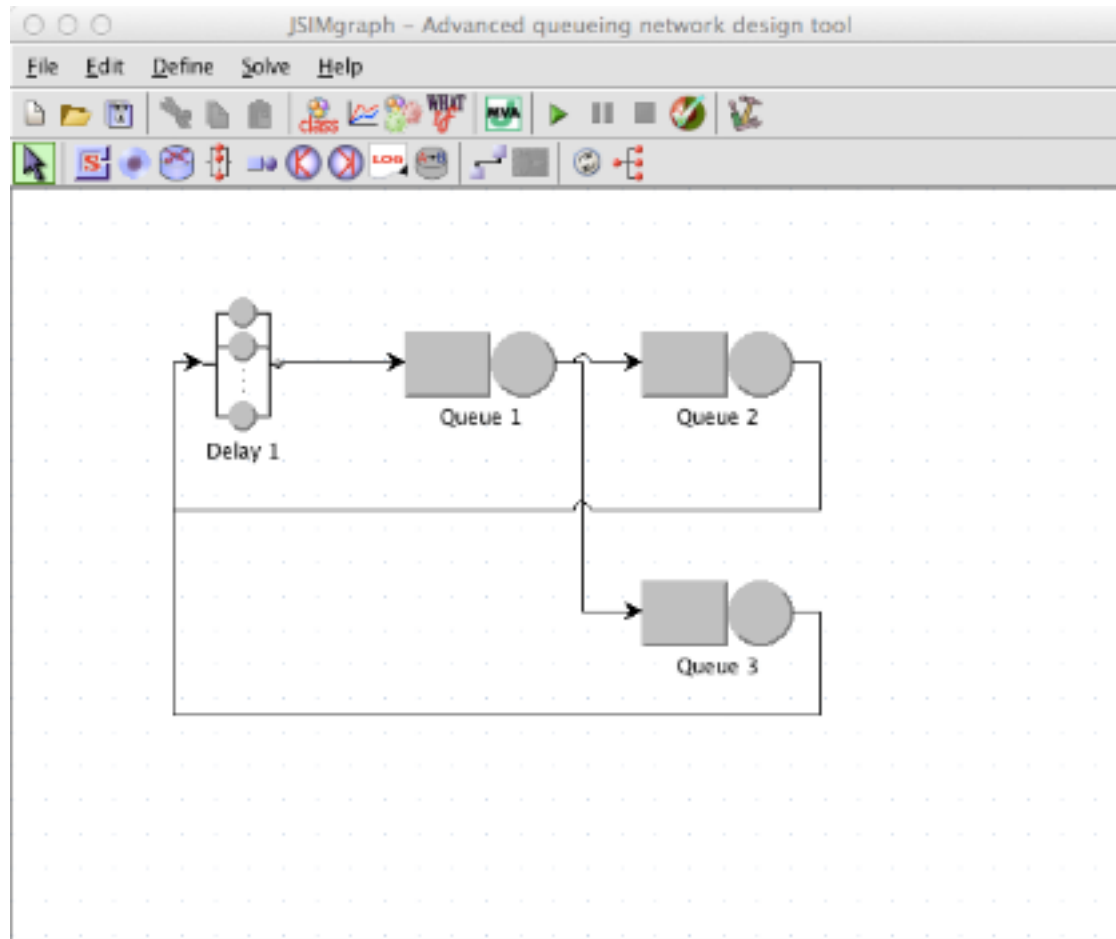
A green icon denotes that the desired accuracy was reached.





## Closed models in JMT

Closed models do not use sources or sinks. Instead they are characterized by arcs that creates loops between the stations.







## Closed models in JMT

In this case, the type “closed” should be specified for their corresponding class, and the total population is defined in the class definition tab of the model.

The screenshot displays the JMT (Java Modeling Tool) interface. The main window shows a queueing network diagram with a 'Delay 1' block, 'Queue 1', 'Queue 2', and 'Queue 3'. A red box highlights the 'class' icon in the toolbar. Overlaid on the diagram is the 'Define customer classes' dialog box. The 'Classes Characteristics' tab is active, showing a table with columns: Color, Name, Type, Priority, Population, Interarrival Time Distribution, and Reference Stat... The table contains one row for 'Class1' with a blue color, 'Closed' type, and a population of 1. Red boxes highlight the 'Closed' type and the 'Population' value. The 'Add Class' button is at the top right, and the 'Done' button is at the bottom right.

Color	Name	Type	Priority	Population	Interarrival Time Distribution	Reference Stat...
Blue	Class1	Closed		1		Delay 1



## Closed models in JMT

Reference station must also be defined in the corresponding column of the class panel.

The screenshot displays the JMT (Java Modeling Tool) interface. The main window shows a queueing network diagram with a 'Delay 1' block, 'Queue 1', 'Queue 2', and 'Queue 3'. A red box highlights the 'class' icon in the toolbar. Overlaid on this is the 'Define customer classes' dialog box. The dialog box has a title bar 'Define customer classes' and a button 'Add Class'. Below the title bar is the text 'Classes Characteristics' and 'Define type, name and parameters for each customer class'. A table lists the characteristics for 'Class1':

Color	Name	Type	Priority	Population	Interarrival Time Distribution	Reference Stat...
Blue	Class1	Closed		1		Delay 1

Red boxes highlight the 'Population' column (value 1) and the 'Reference Stat...' column (value Delay 1). The dialog box also includes a 'Classes' dropdown set to 1 and a 'Done' button at the bottom.



## Closed models in JMT

For time-sharing systems, the "terminal station" is inserted by a delay station, whose service time corresponds to the think time  $Z$ .

The screenshot shows the JMT (JMT - Advanced queueing network design tool) interface. On the left, a queueing network diagram is displayed on a grid. It features a 'Delay 1' station (represented by a rectangle with four dots) highlighted with a red box. An arrow points from 'Delay 1' to a 'Queue 1' station (represented by a rectangle and a circle). From 'Queue 1', an arrow points to another station, and another arrow points back to 'Delay 1', forming a loop. On the right, the 'Editing Delay 1 Properties...' dialog box is open. It has a 'Station Name' field containing 'Delay 1'. Below this, the 'Delay 1 Parameters Definition' section is visible, with tabs for 'Service Section' and 'Routing Section'. The 'Service Section' is active, showing a 'Number of Servers' field with the value '00'. Below that, the 'Service Time Distributions' table is shown, with a red box highlighting the 'Service Time Distribution' column. The table has one row for 'Class1' with a 'Load Independent' strategy and an 'exp(0.25)' service time distribution. The text 'Think time distribution' is written in red below the table. At the bottom of the dialog box is a 'Done' button.

JMTgraph - Advanced queueing network design tool

File Edit Define Solve Help

Editing Delay 1 Properties...

Station Name: Delay 1

Delay 1 Parameters Definition

Service Section Routing Section

Number of Servers

Number: 00

Service Time Distributions

Class	Strategy	Service Time Distribution	Edit
Class1	Load Independent	exp(0.25)	

Think time distribution

Done



## Response times in closed models

Please note that system response times in time-sharing closed models, always include the time spent in the terminal station. To obtain the same system response time proposed in the *Response Time Law* from the book of Lazoswka, you have to manually remove the average think time.

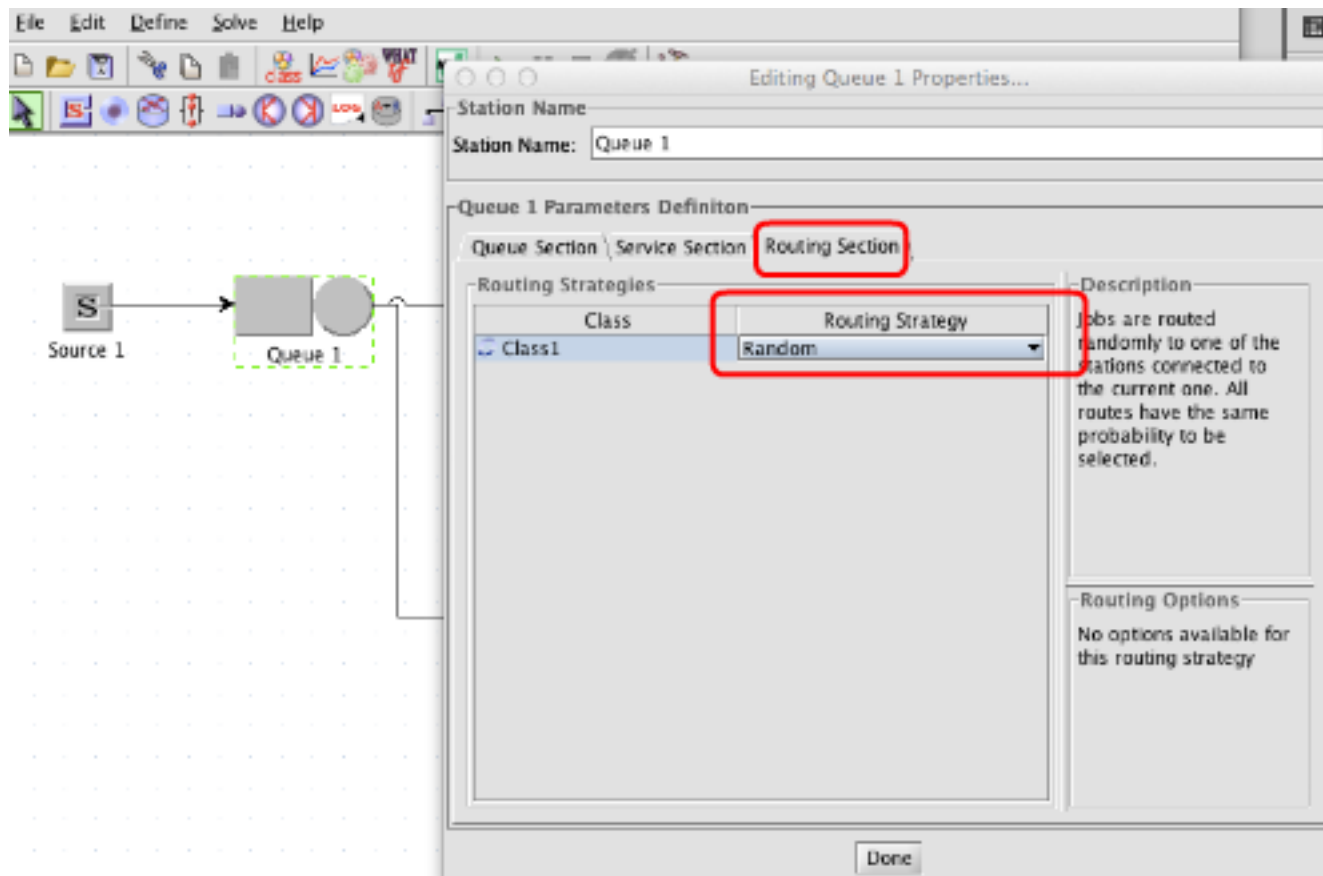
$$R_{Tot} = R_{Sys} + Z \quad N = X \cdot R_{Tot} \quad N = X \cdot (R_{Sys} + Z)$$

$$R_{Sys} = R_{Tot} - Z$$



## Routing in JMT

Every node has a property page called *routing section*. This allows embedding the definition of the routing policy in queues and other modeling elements, and it is used whenever more than one output arc is connected.





# Routing in JMT

Several routing mechanisms are available:

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	Random

Description  
Jobs are routed randomly to stations connected to the current one. All routes have the same probability to be selected.

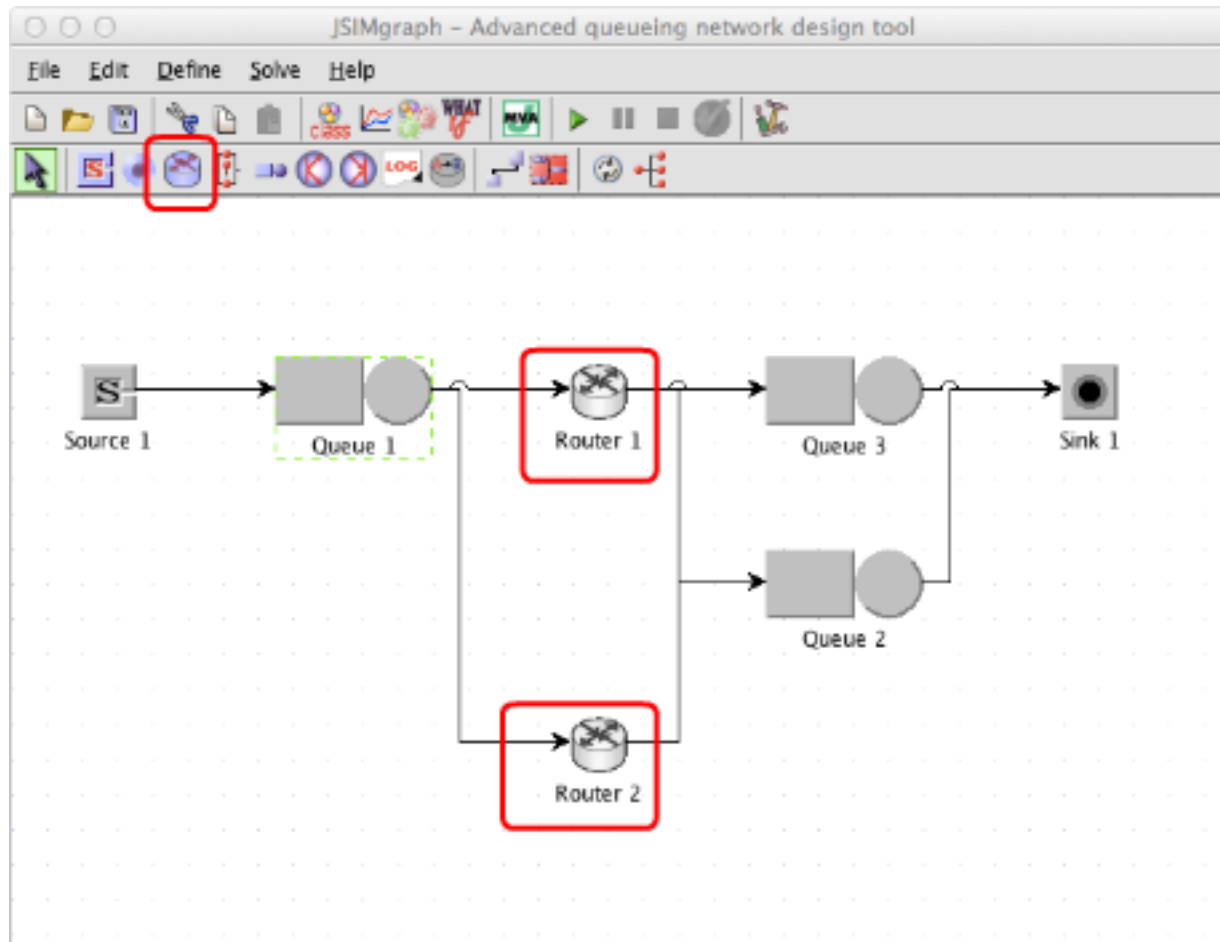
Routing Options  
No options available for this routing strategy

Done



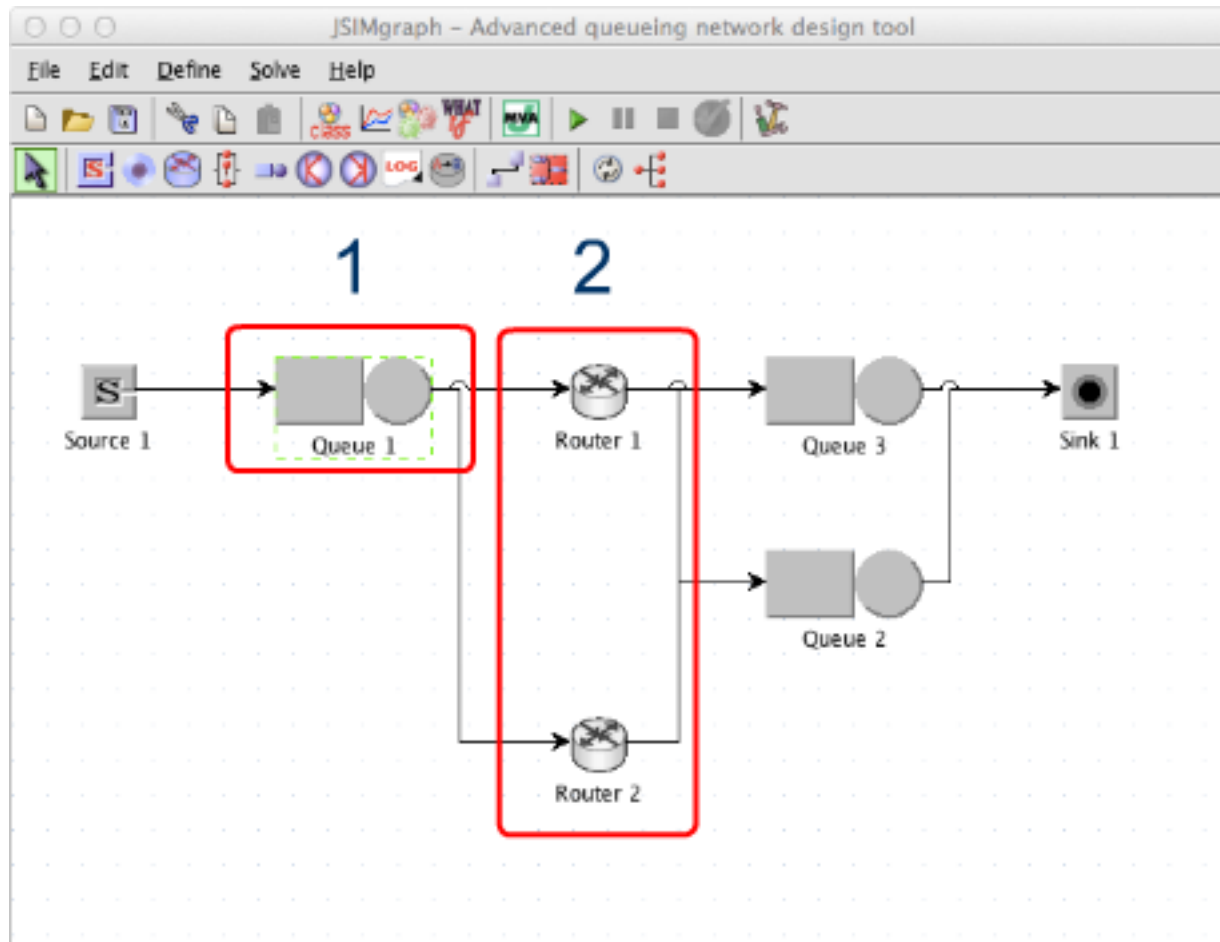
## Routing in JMT

JMT also allows including in a model specific components called *Routers*, whose properties are characterized only by the routing component page.



# Routing in JMT

The inclusion of the routing component allows creating hierarchical policies, where different algorithms can be combined to create more complex job assignments.

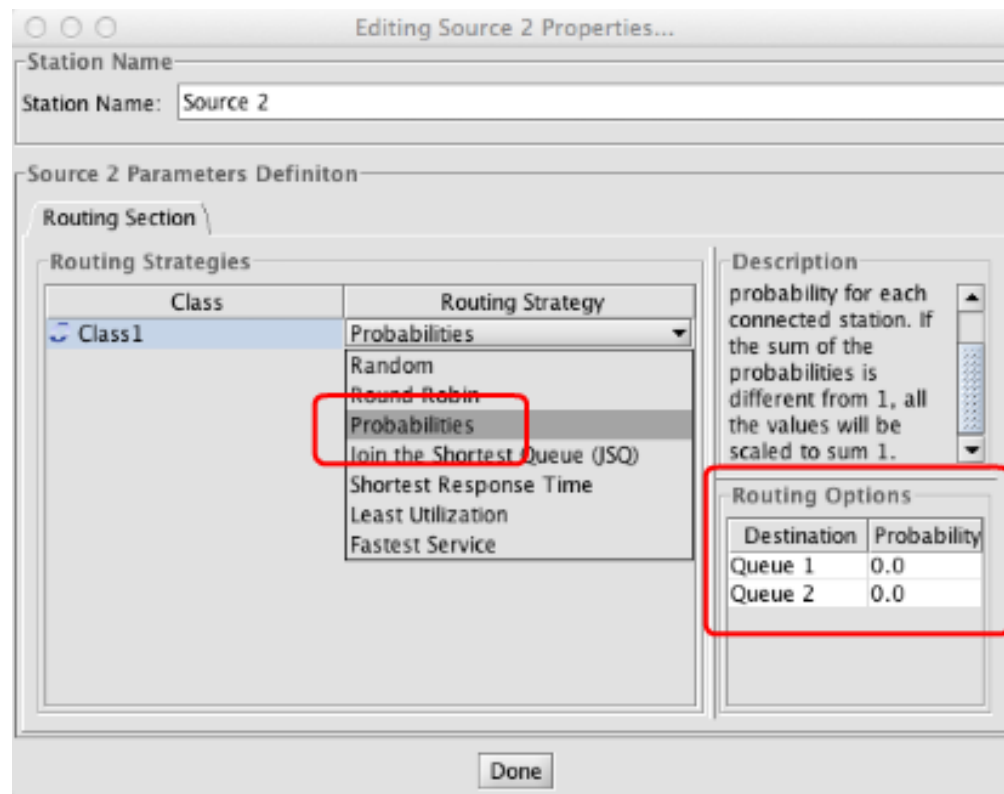






## Probabilistic routing

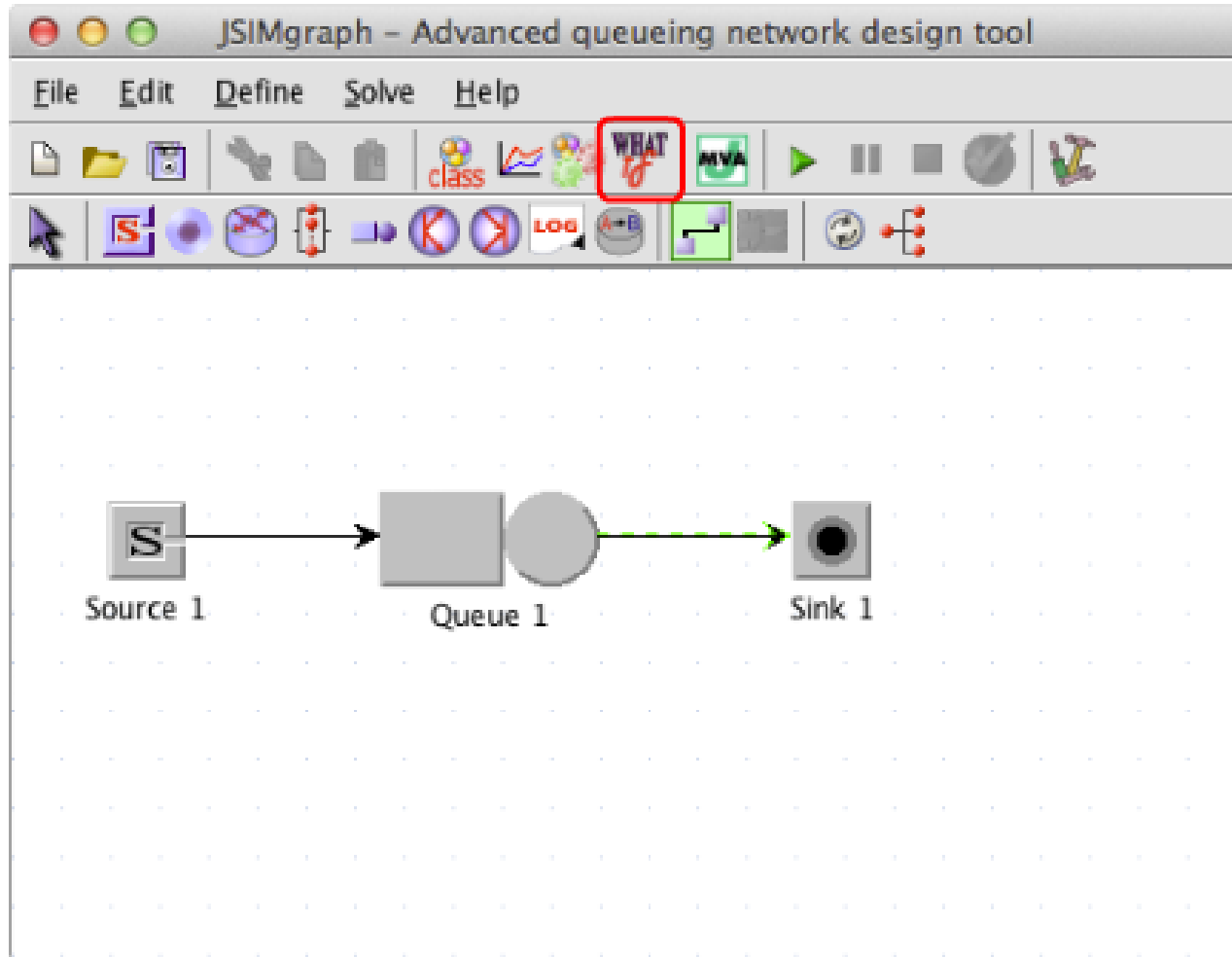
The “probabilities” routing strategy corresponds to probabilistic routing. An additional panel, which appears in the bottom right of the window, allows the user to associate a different probability to each of the nodes connected to the output of the selected queue.





## What-if analysis

What-if analysis is available also in *JSimGraph*, with a specific button.





## What-if analysis

What-if must be activated, and the user must select: the value that will be varied, its range and the number of steps that will be performed.

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

Parameter selection for the control of repeated executions

Service times

Type of service time growth

☐ Change service time of all classes

☒ Change service time of one class

From (s): 0.05

To (s): 0.45

Steps (n. of ex...): 9

Station: Queue 1

Class: Class1

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



# What-if analysis

In this case results can be plotted against the parameter that has been varied during the experiment.

