

Chapter Mqt

M/M/1 Queue Tutorial



Mqt.0 Defining the Task

This chapter presents a practical example of creating a simulation executable, running simulations, and analyzing simulation results. The exercises in this chapter demonstrate particular uses of the Node Editor, Network Editor, Probe Editor, Simulation Tool, and Analysis Tool.

The task in this chapter is to use OPNET to simulate the behavior of an **M/M/1** queue system. An **M/M/1**¹ queue consists of a first-in-first-out (**FIFO**) buffer with packets arriving randomly in accordance with a **Poisson process**, and a processor, called a server, which retrieves packets from the buffer at a specified service rate.

The performance of an **M/M/1** queue system depends on three parameters: packet arrival rate, packet sizes, and service capacity. If the combined effect of the average packet arrival rate and the average packet size exceeds the service capacity, the queue size will grow indefinitely. The task is to construct an **M/M/1** queue model to allow an analyst to observe the performance of the queuing system with varying packet arrival rates, packet sizes, and service capacities. The following statistics shall be measured:

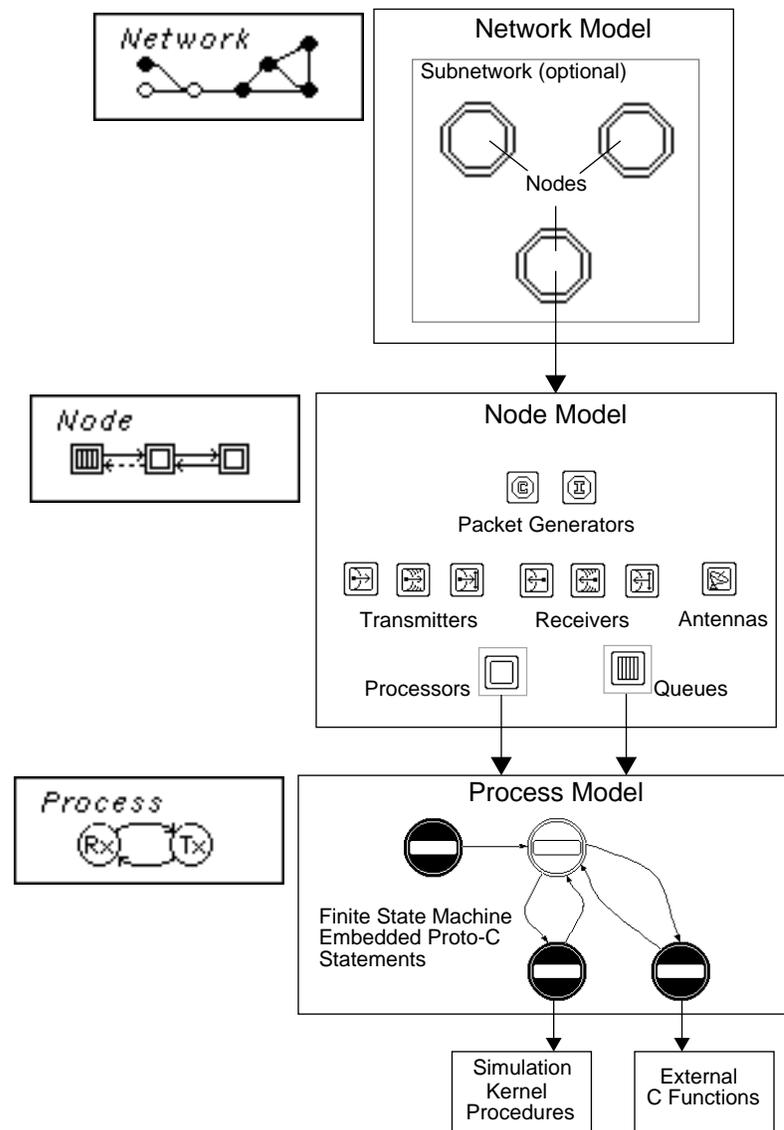
- Average delay (waiting time) experienced by packets in the queue
- Number of packets in the queue at any one time
- Average number of packets in the queue over time

1. The name “M/M/1” follows the Kendall notation for general queuing systems. The first symbol represents the packet arrival distribution, the second represents the service distributions, and the number represents the number of servers. The two M’s (from the Markov process) in **M/M/1** denote the **Poisson process** or the equivalent exponential distribution.

Mqt.1 Designing the Model

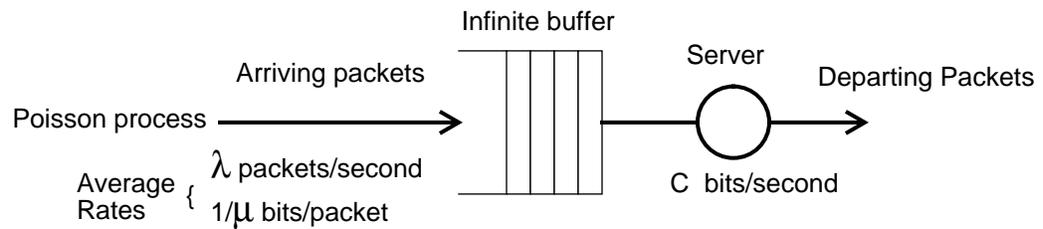
OPNET models are hierarchical. At the lowest level, the behavior of an algorithm or protocol is encoded by a state/transition diagram with embedded code based on C language constructs. At the middle level, discrete functions such as buffering, processing, transmitting, and receiving data packets are performed by separate *objects*, some of which rely on an underlying process model. These objects, called *modules*, are created or modified using the Node Editor within `opnet`, and connected to form a higher level node model. At the highest level, node objects based on underlying node models are deployed and connected by links to form a network model. The network model defines the scope of a simulation, and it is used as a “table of contents” when the simulation executable is bound together from its discrete components (i.e., the Simulation Kernel, process models, and the simulation C file containing the *main()* function).

OPNET Tools and Modeling Structure



The objective is to create a node model which will emulate an **M/M/1** queue system. An **M/M/1** queue system is generally depicted as follows:

M/M/1 Queue System



λ , $1/\mu$, and C are representative of mean packet arrival rate, mean packet size (service requirement), and service capacity, respectively.

At a minimum, the model will require a means of generating, queuing, and serving packets. All of these capabilities are adequately provided by existing modules and models supplied with OPNET.

An OPNET *ideal generator* module will be used to create packets. The **M/M/1** queue's **Poisson process** is expressed in terms of the average number of arriving **packets/second**. An alternate form of specifying this process is via an *exponential* distribution of packet *interarrivals*. This is the form used in the model, since the generator module characterizes the arrival process using the interarrival distribution.

The **M/M/1** model also requires an exponentially distributed service time. A suitable method of creating the required processing delay is to assign exponentially distributed packet lengths to the packets. Exponentially distributed packet lengths can be specified as an attribute of the generator module. Subsequent stages in the model must make use of the packet length in order to create the desired effects.

OPNET provides several queue process models, each with differing capabilities. In this problem, the packets must be serviced with a **FIFO** discipline at a specified rate defined in **bits/second** in order to achieve the exponentially distributed service times. A queue module using the OPNET-supplied `acb_fifo` process model will provide this function. The name of this process model reflects its major queuing characteristics: 'a' indicates that it is active (i.e., it acts as its own server); 'c' indicates that it can concentrate multiple incoming packet streams into its single internal queuing resource; 'b' indicates that the service time is a function of the number of bits in the packet (i.e., "bit-oriented" service); and 'fifo' indicates the service ordering discipline.

A modeling detail not directly related to the **M/M/1** queue system will be included to dispose of serviced packets. Packets no longer needed should be destroyed to free the memory allocated for them. A processor module using the OPNET-supplied `sink` process model will provide this function.

Packet streams will enable the transfer of packets between the generator module, the `acb_fifo` queue module, and the `sink` processor module. Packet streams are paths through which packets move from one module to another. Models are usually designed to send packets out on a stream when the packets should be transported to another module, and to respond to packet arrivals on a stream. The generator module, and the process models `acb_fifo` and `sink` automatically take care of the sending and receiving of packets on streams.

The network model will contain one instance of the **M/M/1** node model described above. No other nodes or links are required by the modeling goals, and a network model consisting of a single “stand-alone” node is completely valid.

Mqt.2 Entering the Model

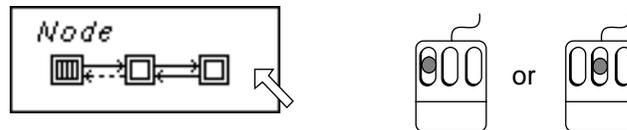
The lower level objects required for an **M/M/1** queue model are supplied with OPNET, but they need to be combined to form a node model.

TRY IT...*Getting started*

- 1) If `opnet` is not running, start the program.

`% opnet`

- 2) Open the Node Editor by clicking the Node tool button.



(Remember that a middle-click opens a full-size workspace window and a left-click allows you to specify the window size by dragging.)

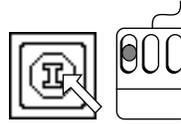
Creating the Node Model

The first step is to define the module which will randomly produce packets. An OPNET ideal generator module creates packets according to user-specified probability distributions.

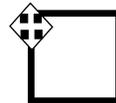
TRY IT...Create an ideal generator module

1) Create an ideal generator module as follows:

a) Activate the **Create Ideal Generator** action button.

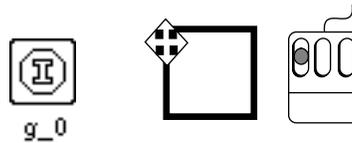


When you move the cursor into the tool window, it changes and shows the outline of a box.



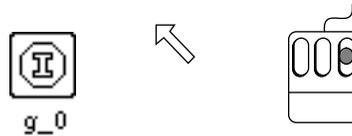
b) When the box is where you want the module icon to be, left-click the mouse.

➔ The icon appears in the tool window. When you move the mouse, notice that the cursor still shows the outline box, in case you want to create another ideal generator module.

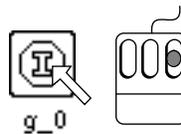


c) Right-click to end the operation.

➔ The cursor changes to the arrow shape.



2) Open the attribute dialog box of the module by right-clicking on the icon.



3) Change the attributes of the module as follows:

(src) Attributes		
Attribute	Value	Units
name	src	
interarrival pdf	exponential	
interarrival args	promoted	
pk size pdf	exponential	
pk size args	9000	
field (0) pdf	constant	
field (0) args	0.0	
packet format	NONE	

extended attrs.

Also Apply Changes to Selected Objects

View Properties Promote Cancel OK

- a) Change the **name** attribute to “**src**” (for **source**). Left-click in the **Value** column, enter the text, and press <Return>.

Note: When you type into a text entry box, press <Return> to ensure that the new data is entered.

- b) Change the **interarrival pdf** attribute to “**exponential**”. This sets the interarrival times of generated packets to be exponentially distributed, as in a **Poisson process**.
- c) Do one of the following:
- Move the cursor over the **interarrival args** attribute and press the *middle* mouse switch
 - Select the **interarrival args** attribute and left-click on the **Promote** button

The value of this attribute now displays the word “**promoted**”, indicating it has been promoted to the network level.

“Promoting” an attribute allows the value to be specified at a higher model level. In this tutorial, the mean interarrival time ($1/\lambda$) will be specified at the simulation level.

- d) Set the **pk size pdf** attribute to “**exponential**”. This sets the sizes of generated packets to be exponentially distributed.
- e) Set the **pk size args** attribute to “**9000**” and press <Return>. This sets the mean size of generated packets, also known as the mean service requirement of the queue (measured in units of **bits per packet**).

- 4) Close the dialog box by clicking on the **OK** button.

The **interarrival pdf** and **interarrival args** attributes specify the packet interarrival distribution. “PDF” stands for *probability density function*. A PDF deter-

mines the distribution of stochastic numbers. The range of possible outcomes from an exponential **PDF** extends between zero and infinity with more of the outcomes clustered near or below a specified mean value.

The **interarrival args** attribute has been promoted to the network level so that it may be specified at simulation run-time. Promoting an attribute is useful because it allows you to run multiple simulations with different attribute values without modifying the model itself.

The **pk size pdf** and **pk size args** attributes specify the packet size distribution. The exponentially distributed packet size settings cause the range of packet sizes to extend from zero to infinity with a mean size of **9000 bits**. The use of exponential **PDF's** in the **M/M/1** model is consistent with the **Poisson process** originally specified for the service time and packet arrival probabilities.

The ideal generator definition for **src** is complete. During simulations, **src** will randomly generate packets according to the interarrival **PDF** values assigned.

The next step is to create the queue module which will receive and service the packets generated by **src**.

TRY IT...*Create a queue module*

- 1) Activate the **Create Queue** action button and place a queue module to the right of the generator module.



src

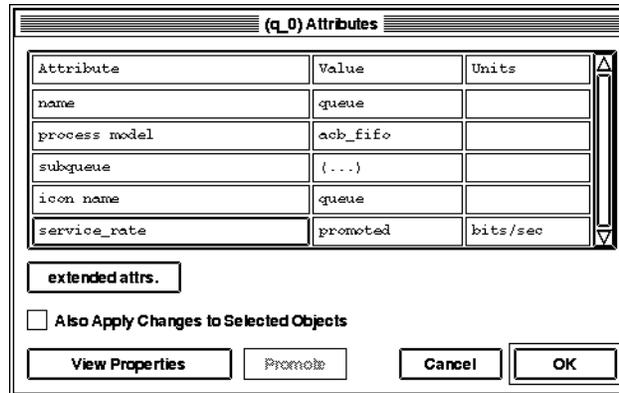


q_0

Remember to end the operation by right-clicking anywhere in the tool window.

- 2) Open the attribute dialog box of the queue module by right-clicking on the icon.

- 3) Change the attributes of the module as follows:
- Change the **name** attribute to “**queue**”.
 - Change the **process model** attribute to “**acb_fifo**”. (You might need to scroll the list of files to find this item.)
 - Left-click the **service_rate** attribute name, then click the **Promote** button.
- ➔ The completed dialog box looks like this:



- 4) Close the dialog box by clicking on the **OK** button.

When a process model is assigned to a module, the process model’s attributes appear in the module’s menu. The **acb_fifo** process model has an attribute called **service_rate**. When you select the **acb_fifo** process model, its **service_rate** attribute appears in the queue module’s attribute menu with the default value 9,600 bits per second.

The **service_rate** attribute represents the service capacity of the queue process model’s built-in server. To specify different values for **service_rate** at simulation time, you can set it to “**promoted**” in the queue module’s attribute menu (as you did in step 3c above).

The **acb_fifo** model operates as a *first-in-first-out* queue.

For proper memory management, packets should be destroyed when they are no longer needed. The OPNET-supplied **sink** process model destroys packets sent to it.

TRY IT...Create a processor module

- 1) Activate the **Create Processor** action button and place a processor module to the right of the queue module.

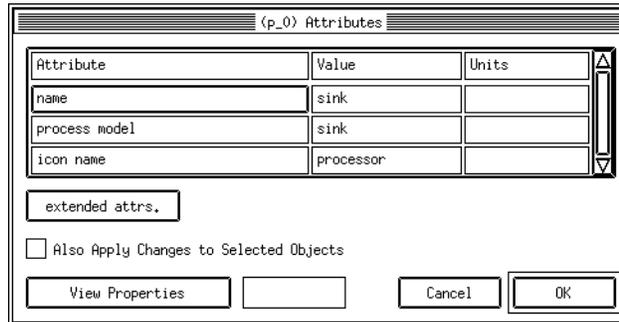


Remember to end the operation by right-clicking anywhere in the tool window.

- 2) Open the attribute dialog box of the processor module by right-clicking on the icon.

Notice that the default value of the **process model** attribute is "sink".

- 3) Change the **name** attribute to "sink".

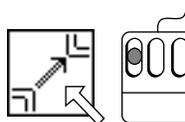


- 4) Close the dialog box by clicking on the **OK** button.

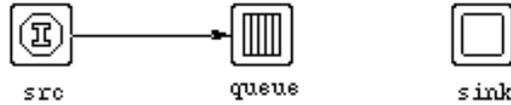
Packet streams provide a path for the transfer of packets between modules. They serve as one-way paths and are depicted by solid arrowed lines.

TRY IT...Connect the modules with packet streams

- 1) Activate the **Create Packet Stream** action button.

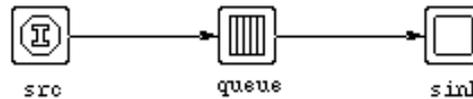


- 2) Connect the **src** module with the **queue** module by clicking on the **src** icon, then clicking on the **queue** icon.



Notice that the first module you click on becomes the source and the second becomes the destination.

- 3) Connect the **queue** module with the **sink** module by clicking on the **queue** icon, then clicking on the **sink** icon



Remember to end the **Create Packet Stream** operation by right-clicking anywhere in the tool window.

- 4) Open the attribute dialog box of a packet stream by right-clicking on either arrow.

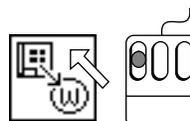
Notice that the default value of the **dest stream** attribute is "0". **opnet** automatically chooses connection ports.

- 5) Close the dialog box by clicking on the **OK** button.

Stream sources and destinations are assigned by default. Stream numbers (or indices) allow for multiple streams to be connected to a module. In this configuration, the packets from the generator leave its only output and enter stream **0** of the queue. The queue has only one *subqueue* (a subqueue is a distinct operational queue within a single queue module). In some scenarios, queues may contain multiple subqueues, allowing them to manage different classes of packets separately. The subqueue takes input packets from input stream **0** and sends out serviced packets on output stream **0**. The **sink** processor receives the serviced packets on input stream **0**.

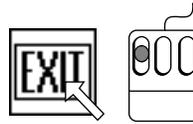
TRY IT... *Write the node model and exit*

- 1) Activate the **Write Node Model** action button.



2) Name the file "my_mm1node" and press <Return>.

3) Activate the **Exit Node Editor** action button.



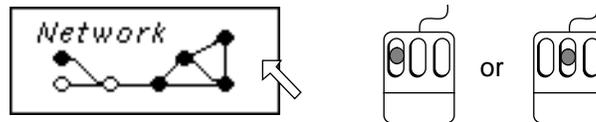
The **M/M/1** node model definition is complete. The node model will be referenced by a node object at the network level.

Creating the Network Model

The **M/M/1** network model will consist of a single node object based on the **my_mm1node** node model. The Network Editor is used both to define the network model and create the simulation executable.

TRY IT...Open the Network Editor

1) Open the Network Editor by clicking on the Network Editor tool button.



Normally, one would create a subnet and place nodes within the subnet. But because the **M/M/1** model requires only a single non-communicating node, the node location is completely irrelevant. In this tutorial the node will be placed at the global level view (also known as the **top** subnet).

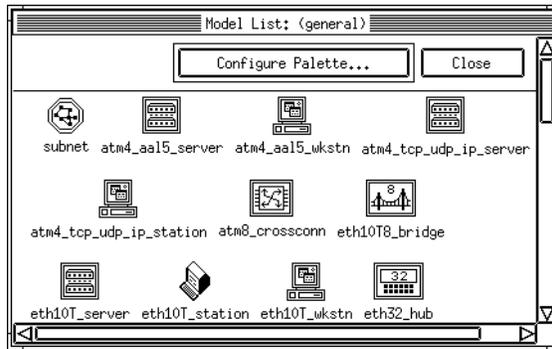
TRY IT...Create a node and exit

1) Create a fixed communication node as follows:

a) Activate the **Open Object Palette** action button.

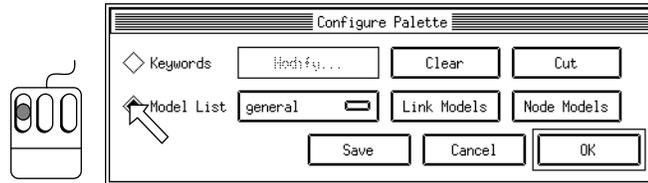


b) The Object Palette dialog box appears.



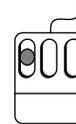
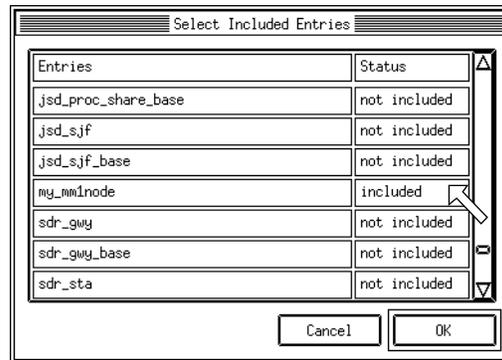
c) The palette that appears contains many objects that are not needed for this project. To reduce the set of available objects to those of interest, click on the **Configure Palette** button. This will allow you to create a *Custom Model List.*, or *CML*

- ➔ A Configure Palette dialog box appears that allows you to customize the palette.
- ➔ Make sure that the **Model List** radio button is enabled, if it is not already.

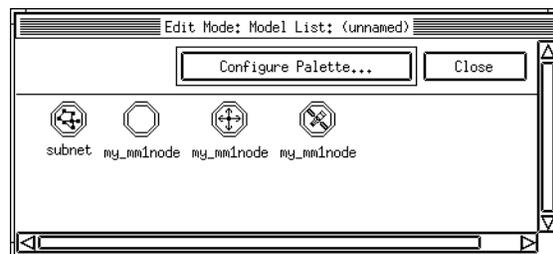


d) Press the **Clear** button in the Configure Palette dialog box to remove all currently present models from the set offered by the palette. Only the subnet icon remains.

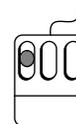
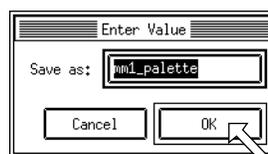
- e) Use the **Node Models** button to select the node models of interest for this project.
- ➔ A table appears showing you which node models are included in the palette. Because the **clear** operation was just used, no models are initially included.
- f) Scroll the table until you find the **my_mm1node** and click in the corresponding cell of the **status** column so that the model becomes included.



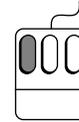
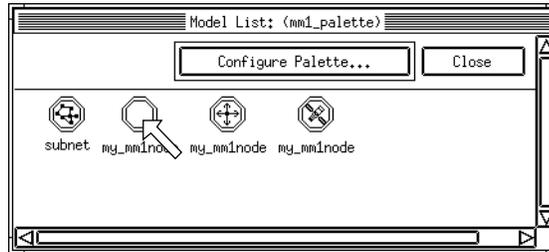
- g) Click on the table's **OK** button. The palette now contains an icon for the newly included node model.



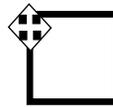
- h) Click on the **save** button in the **Configure Palette** dialog box to preserve this configuration of the palette. Enter the name **mm1_palette** at the prompt and click on the **OK** button.



- i) In the Palette, click on the fixed communication node and continue to hold the mouse button down.

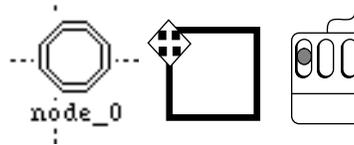


- ➔ The cursor changes and shows the outline of a box.



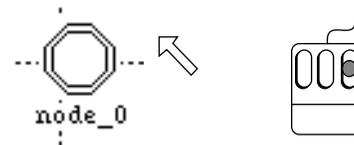
- j) Drag the box into the tool window. When the box is where you want the node icon to be, release the mouse button.

- ➔ The icon appears in the tool window. When you move the mouse, notice that the cursor still shows the outline box, in case you want to create another node.



- k) Right-click to end the operation.

- ➔ The cursor changes to the arrow shape.

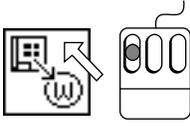


2) Change the attributes of the node:

- a) Open the attribute dialog box of the node by right-clicking on the icon.
- b) Change the **name** attribute of the node to "m1".
- c) Notice that the **model** attribute is "my_mm1node".
- d) Close the dialog box by clicking on the **OK** button.

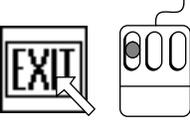
3) Write the network model:

a) Activate the **Write Network Model** action button.



b) Name the file `my_mm1net` and press `<Return>`.

4) Activate the **Exit Network Editor** action button.



The **M/M/1** model is now completely defined.

Specifying Probes

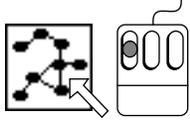
Section *Mqt.0, Defining the Task*, mentions a number of statistics to be monitored during simulations. These must be selected using the Probe Editor. Specifying appropriate probes causes the statistics to be recorded at simulation time.

TRY IT...Set a statistic probe

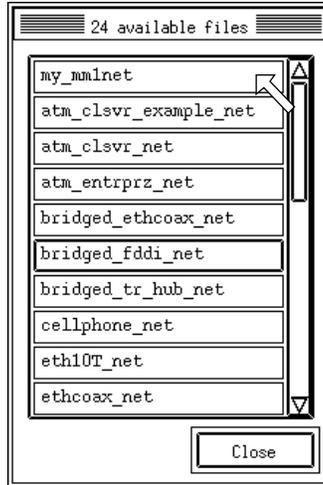
1) Open the Probe Editor by clicking on the Probe button.



2) Activate the **Set Network Model** action button.

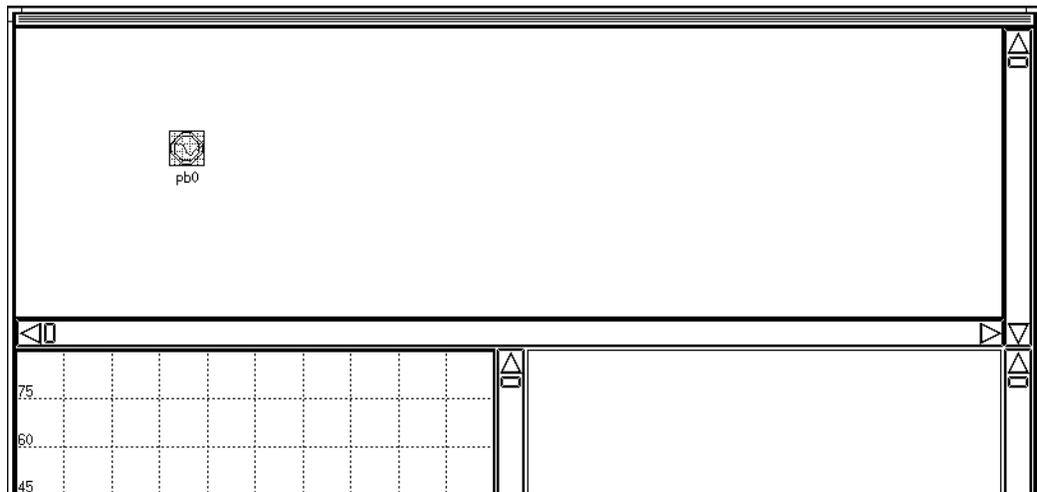


3) Select `my_mm1net` from the menu of available files.



➔ This displays the network diagram in the Network Subwindow (you might need to scroll to view the diagram).

4) Activate the **Create Node Statistic Probe** action button and place a probe anywhere in the Probe Area Subwindow.

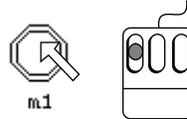


Remember to end the operation by right-clicking anywhere in the tool window.

5) Open the attribute dialog box of the probe by right-clicking on the icon.

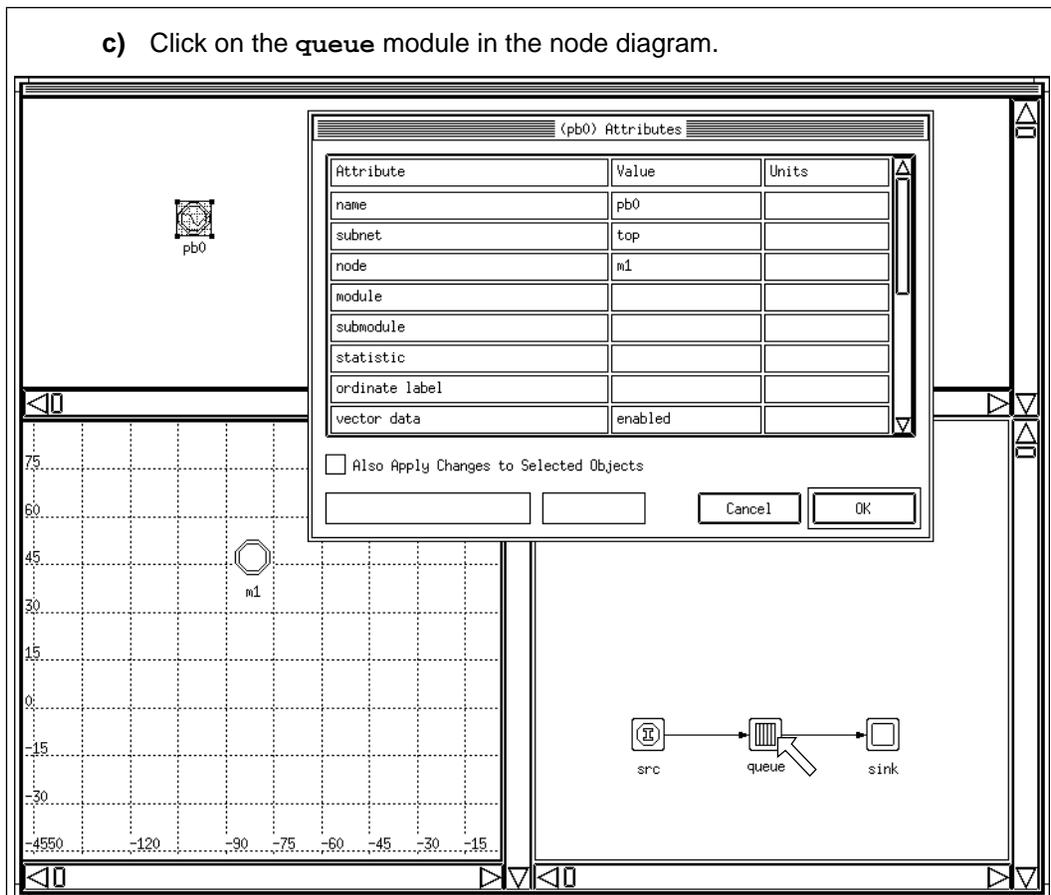
6) Use the selection method to change the attributes of the probe as follows:

- a) Click on the probe icon to select it.
 - ➔ Four corner markers appear around the icon.
- b) Click on the m1 node in the Network Subwindow.



- ➔ This adds the subnet and node information to the probe's attributes, and displays the node model diagram in the Node Subwindow (you might need to scroll to view the diagram).

c) Click on the **queue** module in the node diagram.

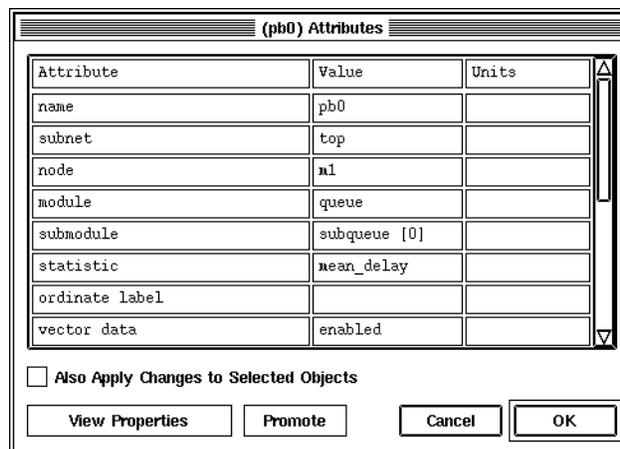


➔ The probe's attribute dialog box now specifies "queue" as the module to be probed.

d) Change the **submodule** attribute to "subqueue [0]".

e) Change the **statistic** attribute to "mean_delay".

➔ The completed dialog box looks like this:



- 7) Close the dialog box by clicking on the **OK** button.
- 8) Deselect **pb0** by clicking in a blank portion of the tool window.

Next specify the remaining statistic probes **pb1** and **pb2** as shown in the following steps. Before selecting a new probe, make sure to deselect the previous probe to avoid multiple selected probes.

TRY IT...Set more statistic probes

- 1) Activate the **Create Node Statistic Probe** action button and place a second probe anywhere in the Probe Area Subwindow.



pb0



pb1

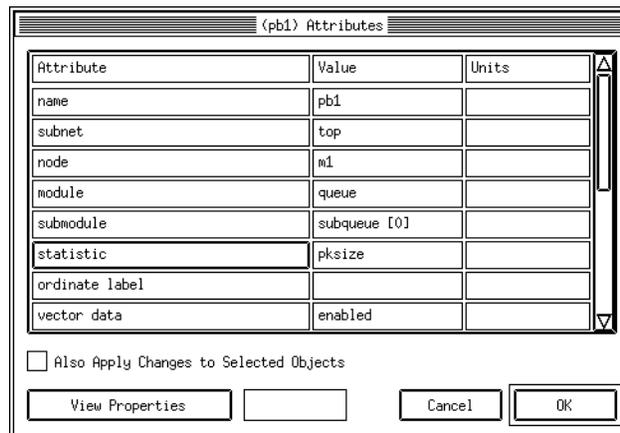
Remember to end the operation by right-clicking anywhere in the tool window.



- 2) Change the attributes of the probe as follows:
 - a) Open the attribute dialog box of the probe by right-clicking on the icon.
 - b) Click on the probe icon to select it.
 - c) Add subnet and node information by clicking on the **m1** module in the Network Subwindow.
 - d) Specify the module to probe by clicking on the **queue** module in the node diagram.
 - e) Change the **submodule** attribute to “**subqueue [0]**”.
 - f) Change the **statistic** attribute to “**pksize**”.

Notice that these instructions are the same as Steps 3 through 5 in the previous procedure, except for the value assigned to the **statistic** attribute.

➔ The completed dialog box looks like this:



- 3) Close the dialog box by clicking on the **OK** button.
- 4) Deselect **pb1** by left-clicking anywhere in the Probe Subwindow.

➔ The four corner markers disappear .
- 5) Activate the **Create Node Statistic Probe** action button and place a third probe anywhere in the Probe Area Subwindow.



pb0



pb1



pb2

6) Set the third statistic probe using the same instructions as for the second probe (Step 2), with the following exception:

f) Change the **statistic** attribute to “**mean_pktsize**”.

➔ The completed dialog box looks like this:

Attribute	Value	Units
name	pb2	
subnet	top	
node	n1	
module	queue	
submodule	subqueue [0]	
statistic	mean_pktsize	
ordinate label		
vector data	enabled	

Also Apply Changes to Selected Objects

View Properties Promote Cancel OK

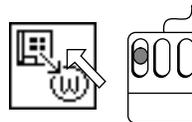
7) Close the dialog box by clicking on the **OK** button.

The **mean_delay** probe keeps track of the average delay that packets experience in the queue. The **pktsize** probe records the instantaneous number of packets in the queue at all times during the simulation. The **mean_pktsize** probe records the average number of packets in the queue during the simulation.

These probe settings must be saved to a file for the simulation program to reference.

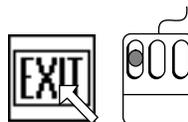
TRY IT... Write the probe file and exit

1) Activate the **Write Probe Model** action button.



2) Name the file “**mm1_probe_file**” and press <Return>.

3) Activate the **Exit Probe Editor** action button.

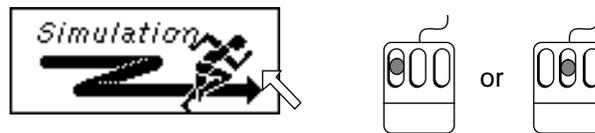


Mqt.3 Executing the Simulation

The Simulation Tool of the `opnet` program provides a convenient way to specify simulation parameters and execute simulations from within the `opnet` environment. To execute simulations using a command shell, see the *OPNET External Interfaces Manual*.

TRY IT...*Open the Simulation Tool*

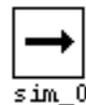
- 1) Open the Simulation Tool by clicking on the Simulation button.



To specify simulation parameters in the Simulation Tool, you first create a simulation object and then modify the object's attributes. Most of the parameters are model-independent, such as the simulation duration or the random number seed. But specific models might have attributes which were left undefined when the model was created, such as the *promoted interarrival args* attribute in the `src` module of `my_mm1node`. You can specify the values of such attributes in the Simulation Tool.

TRY IT...*Specify the simulation parameters*

- 1) Activate the `Create simulation set Object` action button and place an object anywhere in the tool window.



Remember to end the operation by right-clicking anywhere in the tool window.

- 2) Open the attribute dialog box of the simulation set object by right-clicking on the icon.

- 3) Change the attributes on the left of the dialog box as follows. Remember to press <Return> after typing data into a text entry box.

Simulation At

Name: my_mm1net

Sim program: op_runsim

Network: my_mm1net

Probe file: mm1_probe_file

Vector file: mm1_out

Scalar file:

Seed: 431

Duration: 4000

Update intvl: 250

“mm1_out” is a new file, used to store the statistics generated during the simulation. The **seed** can be any positive integer. **Duration** is the number of simulated seconds the simulation will run. **Update Intvl** is the number of simulated seconds between progress updates in the message display.

- 4) Change the attributes in the table on the right of the dialog box as follows:

- a) Click on the **Add...** button to build the list of available attributes.

Add...	Expand...	Delete
Update	View Props	Values...

➔ The **Add Attributes** dialog box displays.

- b) Click in the **Add?** column to select each attribute. You will need to click twice, once in each row.

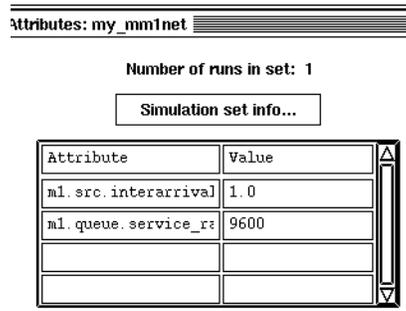
These attributes were left undefined when the model was created. Leaving them undefined in the model allows you to specify the values at simulation time so values can be changed without changing the model.

Add Attribute: sim_0

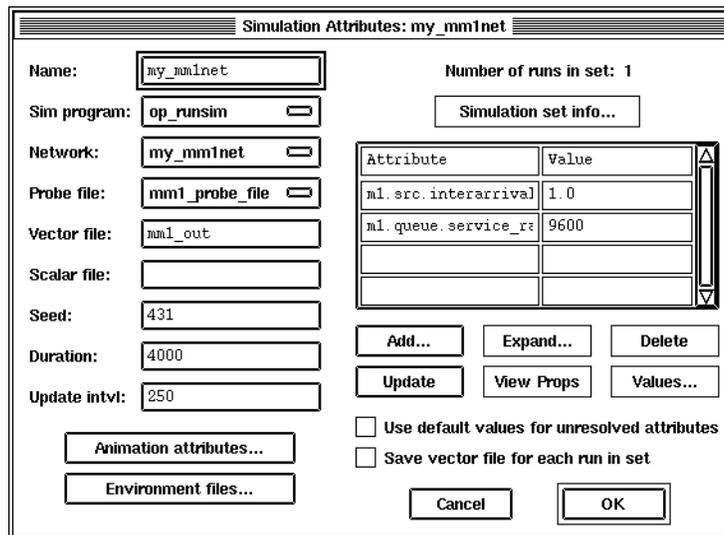
Add?	Unresolved Attributes
<input checked="" type="checkbox"/>	m1.src.interarrival_args
<input checked="" type="checkbox"/>	m1.queue.service_rate
<input type="checkbox"/>	

Wildcard... Cancel OK

- c) Close the **Add Attributes** dialog box by clicking on the **OK** button.
- d) Enter values for the two attributes in the table as shown:



➔ The completed Simulation Attributes dialog box should look like this:



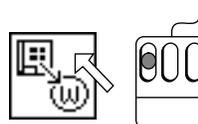
- 5) Close the **simulation Attributes** dialog box by clicking on the **OK** button.

The simulation set object represents a single simulation run. In later lessons, you will vary attribute values and create simulation set objects that represent multiple simulation runs. One or more simulation set objects constitute a *simulation sequence*.

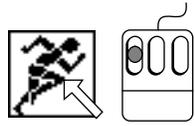
Now you can save the simulation sequence in a file and execute it.

TRY IT...Write the simulation file and execute the simulation

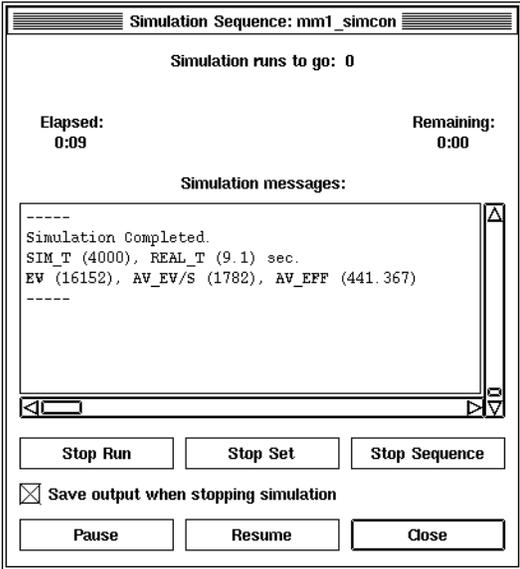
- 1) Activate the **Write Simulation Sequence** action button.



- 2) Name the file "mm1_simcon" and press <Return>.
- 3) Activate the **execute simulation sequence** action button.



Depending on the speed of the host machine, the simulation will last from several seconds to a minute to complete. The **simulation sequence** dialog box shows the progress of the simulation:



The following examples are other messages that you might see:

```
Beginning simulation at Tue Nov 14 13:04:25 1995
-----
Loading Phase
Reading network model (my_mmlnet)
-----
```

...

```
Loading Phase
Loading distribution
exponential (1.0000000000000000e+00)
-----
```

...

```
SIM_T (100.118), REAL_T (0.05199) sec.
EV (384), AV_EV/S (7386), IN_EV/S (7386)
AV_EFF (1925.72), EST_COMPLT (1 sec.)
-----
```

A common error is to leave an attribute undefined. If the simulation seems to pause indefinitely with the dialog box stating, "Loading Phase/Installing

modules and connections”, it might be prompting for values in the parent shell. Follow these steps.

TRY IT...*Fixing undefined attributes*

- 1) Drag on the header strip of the `opnet` window to move it out of the way.
- 2) Make the parent shell window the active window.
- 3) If the problem is an undefined attribute, you will see a prompt similar to this:


```
----- Unset Environment Attribute -----
* top.m1.src.interarrival args: <string>
* Desc:
* Press <return> to select default or to end list:
* Default:      1.0

Enter value> 1.0
-----
```
- 4) Press `<return>` to accept the default value and continue the simulation.
- 5) After the simulation completes, check the simulation sequence for attributes that have not been assigned values.

An audible beep occurs when the simulation is complete. For convenience, you can leave the Simulation Tool open for further runs after analyzing the results of the first run. For purposes of this Tutorial, you can exit the Simulation Tool now.



If other problems occur after starting the simulation, refer to the *Troubleshooting Chapter*.

TRY IT...*Exit the Simulation Tool*

- 1) Activate the `Exit Simulation Tool` action button.

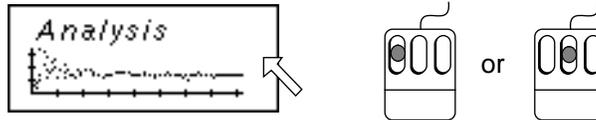
Mqt.4 Analyzing the Results

The Analysis Tool offers the capability of presenting simulation statistics in a variety of two-dimensional graph formats. Statistics monitored during the course of a single simulation are written to an *output vector file*. The default name of an

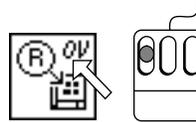
output vector file is based on the name of the corresponding simulation; however, in the previous section of this tutorial, the name of the **output vector file** generated by the simulation was named `mm1_out` in the Simulation Tool. Data in the **output vector file** consists of time and value pairs. The Analysis Tool can plot selected statistics as a function of time.

TRY IT...Open the Analysis Tool

- 1) Open the Analysis Tool by clicking the Analysis button.



- 2) Activate the Open Output Vector File action button.



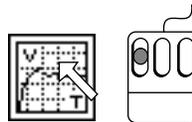
- 3) Select `mm1_out` from the menu of output vector files that pops up.

The **output vector file** must be specified before viewing statistics.

The simplest graph is that of a single vector.

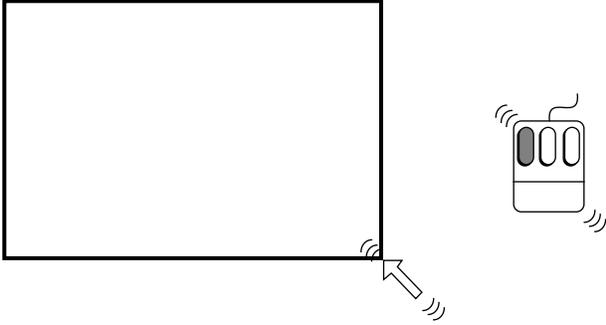
TRY IT...Plot a single vector

- 1) Activate the Create Single-vector Panel action button.



- 2) Select `top.m1.queue.subqueue[0].mean_delay` from the menu of available vectors that pops up.

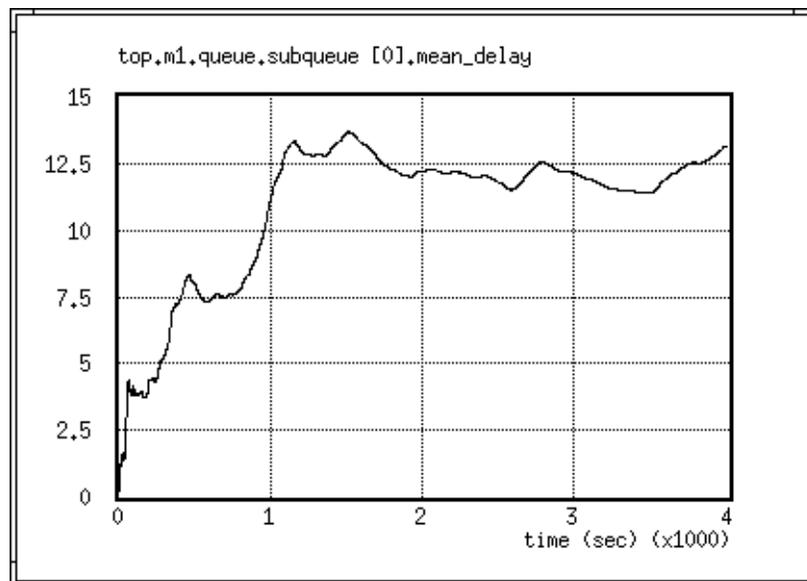
3) Specify where you want the panel by dragging a box in the tool window.



→ The graph appears in the panel.

This graph shows the mean delay experienced by packets in the queue during the course of the simulation. The horizontal axis depicts simulation time in **seconds** and the vertical axis depicts the **mean delay in seconds**.

Single Vector Graph (mean_delay)



The large change in the **mean delay** early on in the simulation reflects the sensitivity of averages to the relatively small number of samples collected. Toward the end of longer duration simulations the average will tend to stabilize.

Recall the attributes set for this simulation. Packets averaging **9000 bits** in length arrive on average once every second. The packets are serviced at **9600 bits/second**. Using the traditional symbology these parameters are represented as:

mean arrival rate:
$$\lambda = \frac{1.0}{\text{mean interarrival time}} = \frac{1p}{1.0s} = 1 (p/s)$$

$$\text{mean service requirement: } \frac{1}{\mu} = 9000 (b/p)$$

$$\text{service capacity: } C = 9600 (b/s)$$

$$\text{mean service rate: } \mu C = \left(\frac{1}{9000 (b/p)} \right) (9600 (b/s)) \cong 1.067 (p/s)$$

$$\text{mean delay: } \bar{W} = \frac{1}{\mu C - \lambda} = \frac{1p}{1.067 (p/s) - 1 (p/s)} = \frac{1p}{0.067 (p/s)} = \boxed{15s}$$

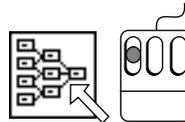
The results for this simulation show an ending **mean delay** of less than the predicted **15 seconds**, but a simulation of sufficient duration will produce the expected result. As in any simulation where steady-state results are desired, it must be verified that the simulation has indeed achieved stability before results are extracted. In this tutorial, the measured delay statistic tends to converge to the expected delay between **105 and 106 seconds** of simulation time (**3 - 30 minutes** of real time, depending on **CPU** speed and simulation duration).

Another statistic of interest is the **time-averaged queue size**. It may be noted that the menu of available output vectors used earlier included a statistic called **mean_pktsize**. This statistic computes a simple running average queue size by summing the instantaneous queue size with any previously accumulated sum whenever a queue insertion or removal event occurs, and then dividing by the number of events. This statistic, however, does not represent the **time-averaged queue size**. To properly reflect the diluting effects of time on a time-averaged statistic, the integral of the vector must be divided by the elapsed time -- an operation absent from the built-in statistic.

The desired **time-averaged queue size** may be obtained by applying an Analysis Tool *filter* to the existing **pktsize** vector.

TRY IT...*Filter a vector*

- 1) Activate the **Execute Filter Model** action button.

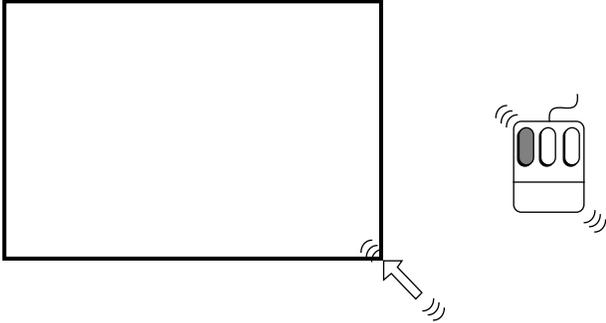


- 2) Select **time_average** from the menu of available filters that pops up (you might need to scroll).

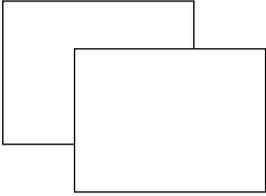
- 3) In the **Enter Value** dialog box, type **time averaged pktsize** and press **<Return>**.

4) Select `top.m1.queue.subqueue[0].pksize` from the menu of available vectors that pops up.

5) Specify where you want the panel by dragging a box in the tool window.

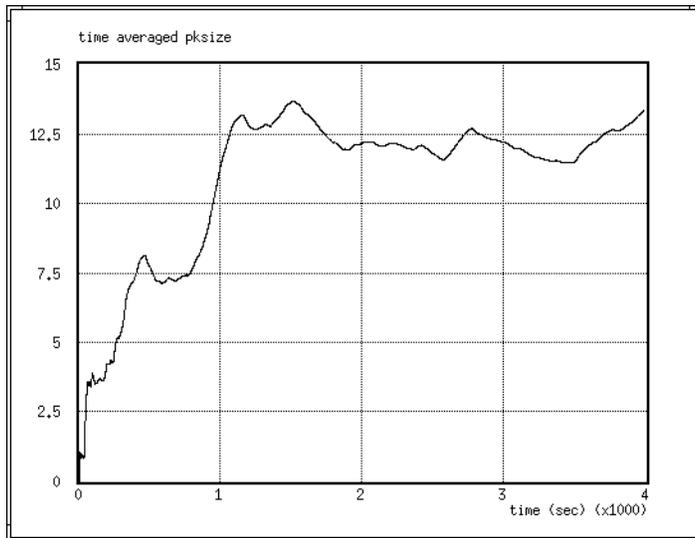


➔ The filtered graph appears in the panel. Note that panels may overlap; pressing the left mouse switch within a partially obscured panel will bring it to the foreground of the window.



The new graph should appear as follows:

Single Vector Graph (filtered)



The final value of the `time averaged pksize` vector seen here can be compared with the expected value obtained from the traditional formula: $\rho / (1-\rho)$, where

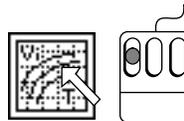
$\rho = \lambda / (\mu C)$. The formula yields **15.0** as the expected average number of packets in the queue. Once again, the results indicate a value less than expected -- a symptom of the relatively short simulation duration.

It turns out for this model that because of the large number of well dispersed queue insertions and removals that occur during the course of the simulation, the difference between the final value of the **mean_pksize** and the **time averaged pksize** vectors is negligible.

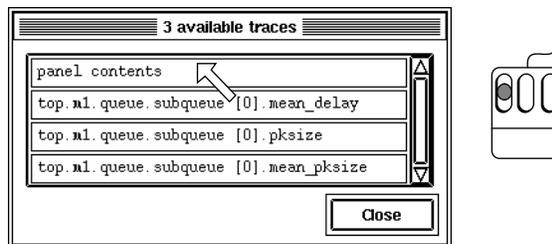
It is often desirable to have multiple vectors plotted on the same graph.

TRY IT...Plot multiple vectors

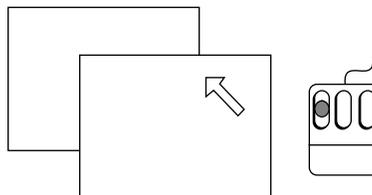
- 1) Activate the **Create Multi-vector Panel** action button.



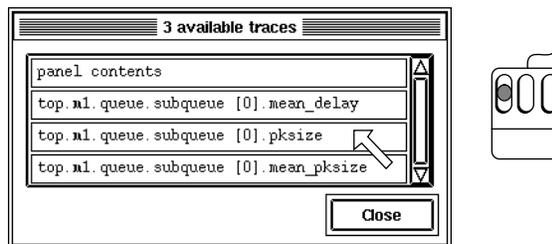
- 2) Select **panel contents** from the menu of available traces that pops up.



- 3) Click on the **time averaged pksize** panel to select the first vector.



- 4) Select **top.m1.queue[0].pksize** from the menu of available vectors to select the second vector in the graph.

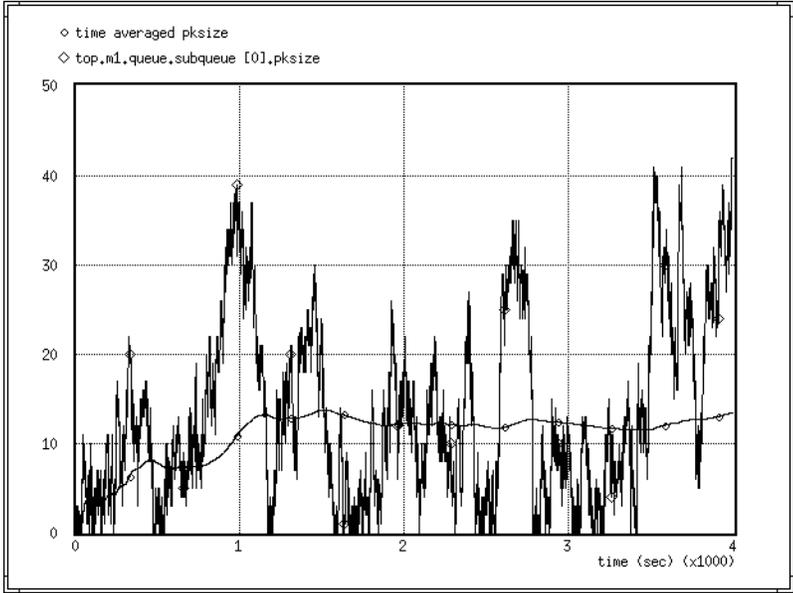


- 5) Close the dialog box by clicking on the **Close** button.

- 6) Specify where you want the panel by dragging a box in the tool window.
 - ➔ The multiple-vector graph appears in the panel.

The new panel displays two traces. The `pksize` trace shows the instantaneous number of packets in the queue during the course of the simulation. The `time averaged pksize` trace depicts the average number of packets in the queue over time.

Multiple Vector Graph (pksize, time averaged pksize)



More techniques and features of the Analysis Tool will be presented in the remaining chapters.

If desired, return to the Simulation Tool and change the `interarrival args` or `service_rate` values and re-execute the simulation. Then compare the results with the values observed above. Remember to change the name of the `vector File` if you want to retain results from previous simulations.

When finished, exit all open tools and proceed to the next chapter.

TRY IT...Exit all tools

- 1) Activate the `Exit Analysis Tool` action button.

Also activate the `Exit` button for any other tools still open.