# ***Packet Switching I***

## Introduction

In this tutorial, you use OPNET to simulate the behavior of a packet switching network. As part of creating the network, you will learn how to use the packet and link editors, new Kernel Procedures, and how to define your own statistics.

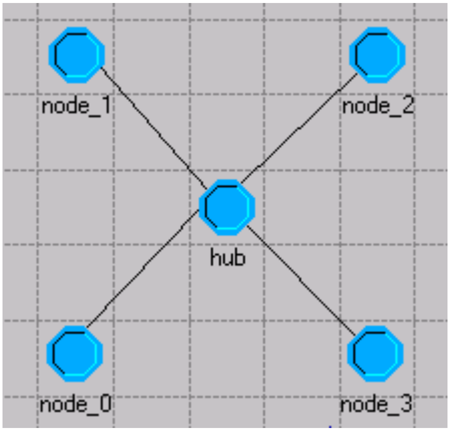
You will also

- Learn some principles of model design
- Become more familiar with process and node models and how they function in a network model
- Do parametric studies of various statistics

In a simple packet switching network, the performance of the network can be measured by the end-to-end delay experienced by traffic on the network.

In this lesson, your goal is to model a simple network in which four peripheral nodes generate traffic while a central hub node relays the traffic to the appropriate destination within the network.

**A Small Packet Switching Network**



## Getting Started

There are several design concepts you should consider before you start building the network model:

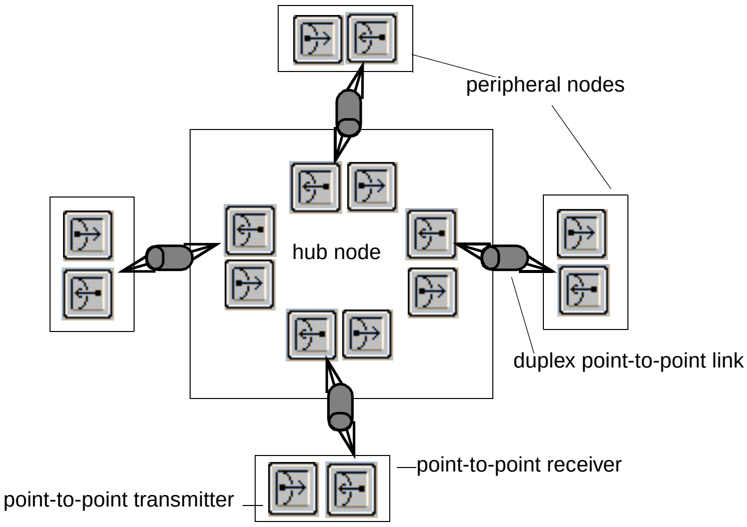
- Network topology and the physical communication medium
- The functions of the different node types
- The method the process model uses to determine which point-to-point transmitter addresses a particular peripheral node
- The role of peripheral nodes

## Network Topology

The initial network will consist of four peripheral nodes connected to a hub node by point-to-point links.

Point-to-point links can either be simplex (unidirectional) or duplex (bidirectional). In this case, you will use custom duplex links to connect transmitter-receiver pairs. Looking only at the communications medium as represented in OPNET, the network would look like this:

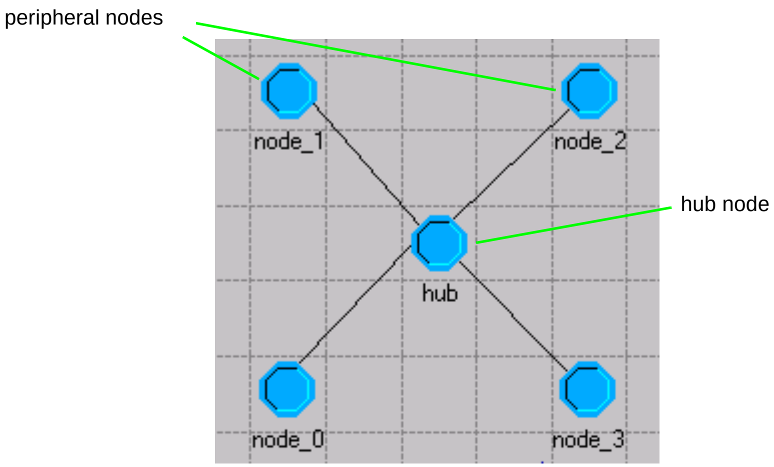
### Communications Medium of the Network



## Functions of the Different Node Types

The purpose of the model is to simulate packets traveling from one peripheral node to another peripheral node through the packet switching hub node. In the hub node, you can assume that packets containing destination addresses will arrive randomly on the four incoming links from the four peripheral nodes. The destination address is an integer value specifying a destination peripheral node. The hub node must contain a process model that can retrieve the incoming packets, read the destination address, and send the packets to the appropriate point-to-point transmitter.

### Different Node Types

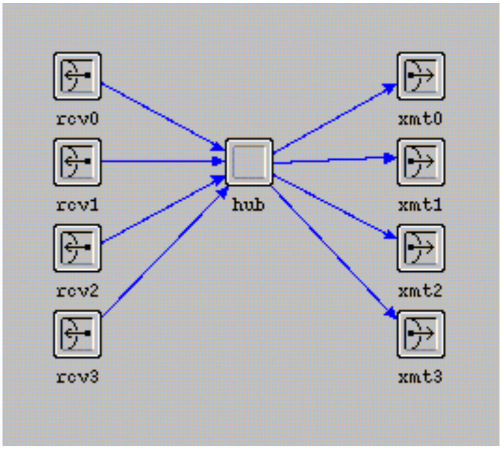


## The Role of the Hub Node Model

Packet streams each have a unique index. The easiest method is to set up a direct association between the hub process outgoing packet stream indices and the peripheral destination address values. In a more adaptive model, the hub process model could maintain a table for translating destination address values to transmitter stream indices. In this tutorial, a direct correspondence between designating addresses and packet stream indices is appropriate.

In summary, the hub node model will consist of a point-to-point transmitter/receiver pair for each peripheral node, and a process model used to relay packets from a receiver to the appropriate transmitter.

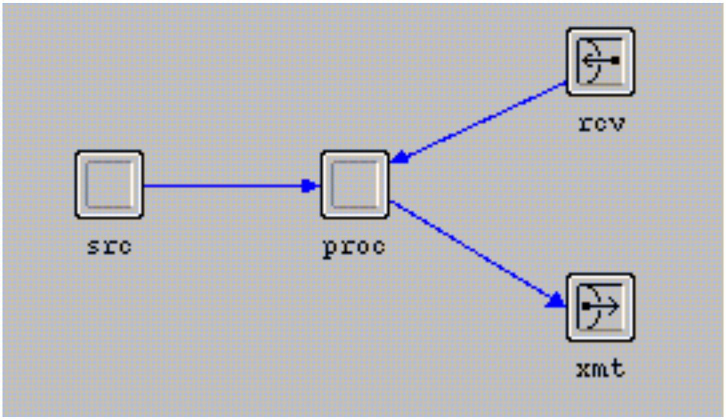
### The Hub Node



## The Role of Peripheral Nodes

The peripheral node model must generate packets, assign destination addresses, and process received packets. The peripheral node model will employ an OPNET generator module to create packets. It will employ a user-defined process model to assign destination addresses to packets and send them to the node's point-to-point transmitter. This process model will retrieve packets arriving from the point-to-point receiver. Upon receiving a packet, the same process model will calculate the packet's end-to-end delay and write the value to a global statistic (a global statistic is accessible to multiple processes throughout the system).

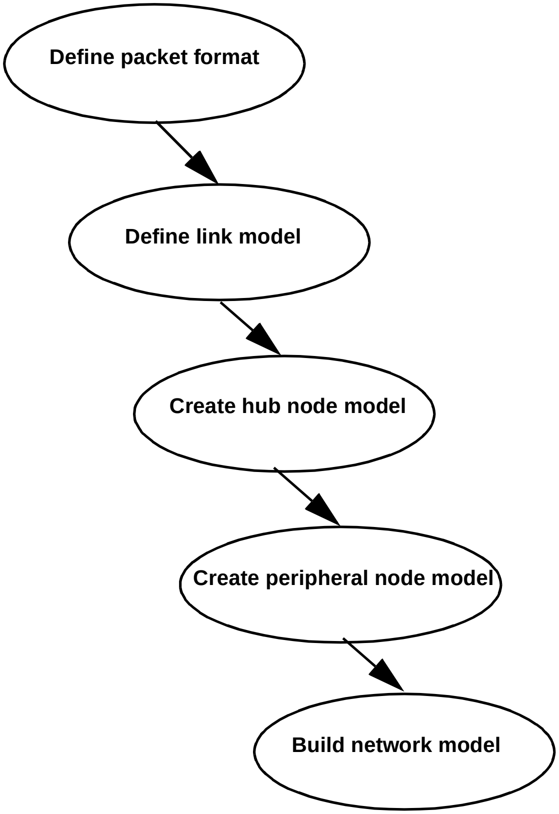
### The Peripheral Node



## Building the Models

To build this network model, you must do the following steps:

**Building the Packet Switching Network**



## Creating a New Packet Format

The **Packet Format Editor** can create packets which contain any number of different fields that affect packet size.

The packets in this network contain a single field with the destination address of the packet. Once the packet format has been created, it can be specified as an attribute in a generator so that packets created by the generator will be formatted accordingly.

To create a new packet:

**1** If OPNET Modeler is not already running, start it.

**2** Close any open projects.

**3** Choose **File** > **New...** Select **Packet Format** from the pull-down list, then click **OK**.

**4** Click on the **Create New Field** toolbar button, then move the cursor into the editor window.

**5** Left-click to place a field in the editor window, then right-click to end the field creation operation.

➥ A new field is created in the editor window.

## Setting the Packet Field Attributes

To set the packet field attributes:

1 Right-click on the packet field and select **Edit Attributes** to open its **Attributes** dialog box.

2 Set the **name** attribute to **dest_address**.

3 Make sure the **type** attribute is set to **integer**.

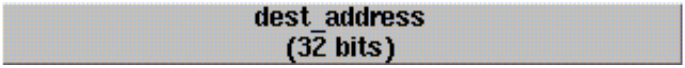
4 Note that the **size** attribute is **32**. This defines the size of the actual field in bits.

5 Change the **set at creation** attribute to **unset**. This ensures that the field will not be assigned a default value when the packet is created.

6 Click **OK** when you are finished.

➥ The packet field in the editor window displays its name and field size.

**Packet Field**



7 Choose **File** > **Save**. Name the packet **<initials>_pksw_format**, then click **OK**. Close the Packet Format Editor when you are done.

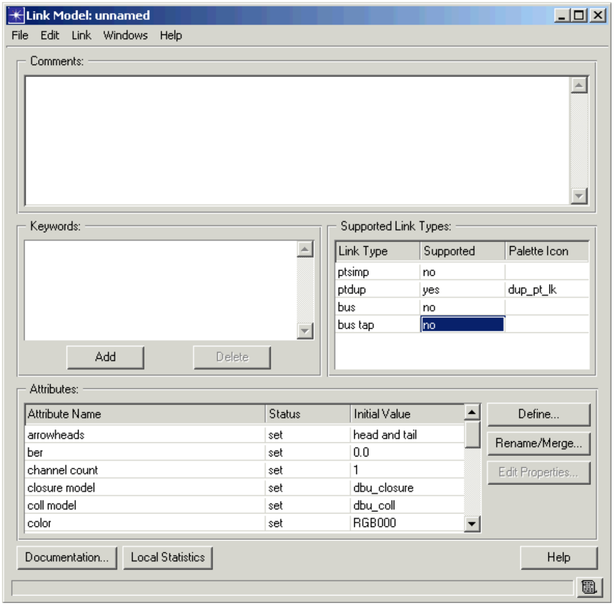
## Creating a Link Model

Use the **Link Model Editor** to create custom links.

Create a link model that will connect the hub and peripheral nodes. This duplex link model should support your packet format.

1 Choose **File** > **New...** Select **Link Model** from the pull-down list, then click **OK.**

**Link Model Editor**



| Link Type | Supported | Palette Icon |
|---|---|---|
| ptsimp | no | |
| ptdup | yes | dup_pt_lk |
| bus | no | |
| bus tap | no | |

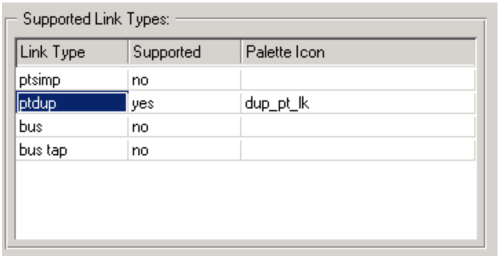
| Attribute Name | Status | Initial Value |
|---|---|---|
| arrowheads | set | head and tail |
| ber | set | 0.0 |
| channel count | set | 1 |
| closure model | set | dbu_closure |
| coll model | set | dbu_coll |
| color | set | RGB000 |

## Setting the Supported Packet Format

Because the link you are creating will support the packet format you just defined, there are several attributes that must be set:

1 In the **Supported Link Types** table, change the value under the **Supported** field to **no** for everything except **ptdup**. The model only needs to support point-to-point duplex.

Link Model Editor (Detail)



Supported Link Types:

| Link Type | Supported | Palette Icon |
|---|---|---|
| ptsimp | no | |
| ptdup | yes | dup_pt_lk |
| bus | no | |
| bus tap | no | |

2 In the **Attributes** table, scroll down to the **packet formats** attribute and click under the **Initial Value** field.

➥ The **Select Supported Packet Formats** dialog box opens.

3 Unselect both checkboxes to turn off **Support all packet formats** and **Support unformatted packets**.

4. Change the **<initials>_pksw_format** packet format’s status value to **supported**.

5. Click **OK** to close the dialog box.

## Setting Other Attributes

In addition to setting the supported packet format type, you must set the following attributes in the attribute table:

1 Set the **data rate** attribute to **9600**.

2 Set the **ecc model** to **ecc_zero_err** (error correction model).

3 Set the **error model** to **error_zero_err**.

4 Set the **propdel model** to **dpt_propdel** (point-to-point propagation delay model).

5 Change **txdel model** attribute to **dpt_txdel** (point-to-point transmission delay).

You may also want to add a comment to the link model to describe it.

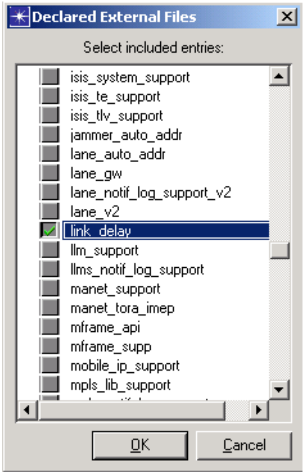
Include the link_delay external file:

1 Choose **File** > **Declare External Files...**

➥ The Declare External Files dialog box appears.

2 In the Entries column, look for **link_delay**.

## Completed Select Included Entries Dialog Box



- isis_system_support
- isis_te_support
- isis_tlv_support
- jammer_auto_addr
- lane_auto_addr
- lane_gw
- lane_notif_log_support_v2
- lane_v2
- ✓ link_delay
- llm_support
- llms_notif_log_support
- manet_support
- manet_tora_imep
- mframe_api
- mframe_supp
- mobile_ip_support
- mpls_lib_support

**3** Select the link_delay entry.

**4** Click OK to close the dialog box.

**5** Choose **File** > **Save**, name the link model **<initials>_pksw_link**, then click **Save**.

**6** Close the Link Model Editor.

## Creating the Hub Node

Now that you have defined a packet format and link model to be used in the model, you can create the hub and peripheral nodes. This process requires two steps for each node type: defining the node model and defining the process model.

Start by defining the node model for the hub. The hub needs four sets of transmitters and receivers for incoming and outgoing packets (one set per peripheral node), as well as a central processor to distribute the packets correctly.

## Creating the Node Model

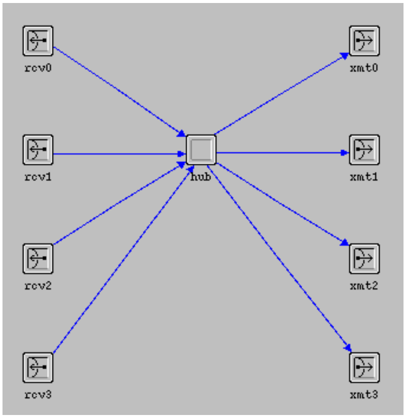
1 Choose **File** > **New...** Select **Node Model** from the pull-down list and click **OK**.

2 Place a processor, four point-to-point transmitters, and four point-to-point receivers in the editor window as shown in the following figure.

3 Change the **name** attribute on each object as shown.

4 Create packet streams to connect the modules as shown. Start with top transmitter and receiver and work your way down. Be sure to draw the streams from the receivers to the hub, then from the hub to the transmitters.

**Hub Node Model**



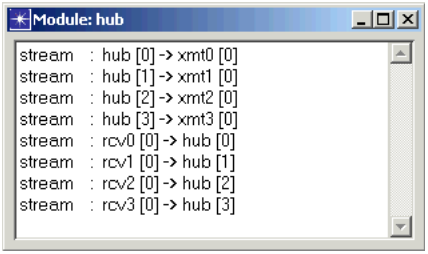
## Checking the Packet Stream Assignments

To check the packet stream assignments, do the following:

**1** Right-click on the **hub** process module and select **Show Connectivity** from the Object pop-up menu.

➥ The list of streams connecting to the **hub** module appears.

Viewing the List of Packet Streams Connected to the Hub



```
stream  : hub [0] -> xmt0 [0]
stream  : hub [1] -> xmt1 [0]
stream  : hub [2] -> xmt2 [0]
stream  : hub [3] -> xmt3 [0]
stream  : rcv0 [0] -> hub [0]
stream  : rcv1 [0] -> hub [1]
stream  : rcv2 [0] -> hub [2]
stream  : rcv3 [0] -> hub [3]
```

**2** Close the dialog box.

## Setting Other Attributes

Now that the hub node model has been created, you need to set the channel data rate and supported packet format for each receiver and transmitter:

1 Hold down the **Shift** key and left-click on each transmitter and receiver module icon to select them. Make sure the packet streams are unselected.

2 When all the modules are selected, right-click on one of the modules and select **Edit Attributes** to open its Attributes dialog box.

3 Click on the value of the **channel** attribute. In the channel (table) dialog box, change the **data rate** to **9600**.

4 Support the **<initials>_pksw_format** format.

4.1 Click on the value of the **packet formats** attribute.

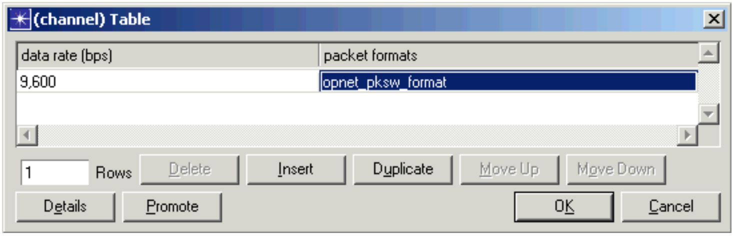
4.2 Unselect both checkboxes to turn off **Support all packet formats** and **Support unformatted packets**.

**4.3** Click on the status field for **<initials>_pksw_format** and change the status to **supported**.

**4.4** Click **OK** in the **Select Supported Packet Formats** dialog box.

**5** Verify that the correct data rate and packet formats display in the **(channel) Table** dialog box, then click **OK**.

**Supporting the Custom Packet Format**



| data rate (bps) | packet formats |
|---|---|
| 9,600 | opnet_pksw_format |

1 Rows

**6** To apply changes to all selected objects, click on the **Apply changes to selected objects** checkbox, then click **OK**.

If the **Undoable operations** dialog box appears, click **OK** to continue.

**7** Left-click in a blank area of the workspace to deselect all modules.

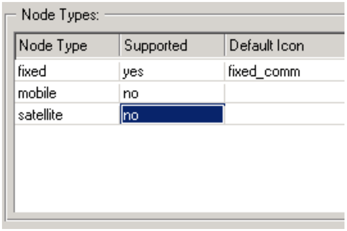
## Setting Interfaces

The last step to defining the hub node module is to specify that only fixed node types are supported:

1 Choose **Interfaces > Node Interfaces**.

2 In the Node Types table, change the **Supported** value to **no** for both **mobile** and **satellite** node types.

**Node Types Table**



Node Types:

| Node Type | Supported | Default Icon |
|---|---|---|
| fixed | yes | fixed_comm |
| mobile | no | |
| satellite | no | |

3 You may wish to add a comment to describe the node. When you are finished, click **OK** to close the dialog box.

The hub node model is now complete, except for specifying the process model for the **hub** processor module (which you can do once the process model has been created). For now, save the model as **<initials>_pksw_hub** BUT DO NOT CLOSE IT.

## Creating the Process Model of the Hub Node

In this model, the hub receives a packet and, based on the destination address, forwards it to the correct transmitter, which sends it to the destination node.

Now, create the hub process model.

In your node model, the hub processor module is connected to the transmitters and receivers by packet streams. Because each packet is associated with an interrupt, the hub process model receives an interrupt whenever a packet arrives from a receiver. Because this is the only expected type of interrupt, the hub process FSM (finite state machine) can be defined using two states: an unforced idle state to rest between events, and a forced state containing the code for processing packets.

To create the process model of the hub:

1 Choose **File** > **New...** Select **Process Model** from the pull-down list, then click **OK**.

2 Using the **Create State** toolbar button, place one state in the editor window.

3 Change the **name** attribute of the state to **idle**.

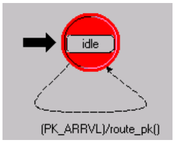
When a packet arrives from a point-to-point receiver, the process model is invoked by a stream interrupt. The transition between the two states must test for this condition.

1 Use the **Create Transition** toolbar button to draw a transition from the **idle** state to itself.

2 Right-click on the transition and change the **condition** attribute to **PK_ARRVL,** and the **executive** attribute to **route_pk()**.

3 Click **OK** to close the attribute dialog box for the transition.

Initial State and Transition



Next, define the **PK_ARRVL** macro:

1 Click on the **Edit Header Block** toolbar button.

2 Enter the following macro definition for **PK_ARRVL:**

```
#define PK_ARRVL (op_intrpt_type () == OPC_INTRPT_STRM)
```

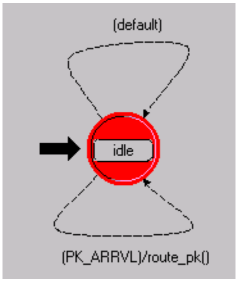
3 Choose **File > Save** when you are finished.

The **PK_ARRVL** condition compares the delivered interrupt type with the OPNET predefined symbolic constant **OPC_INTRPT_STRM**, which indicates a stream interrupt. Although this is the only expected type of interrupt for this model, it is good to safeguard against run-time missing transition errors. To do this, you can create a default transition loop on unforced states.

To create a default transition on the unforced state:

1 Create a transition from the **idle** state back to itself.

2 Change the **condition** attribute to **default** and click **OK** to close the dialog box.



You also must define the enter executives for the **route_pk** state:

1 Click on the **Edit Function Block** toolbar button.

2 Enter the following code:

```
static void route_pk(void)
    {
    int dest_address;
    Packet * pkptr;
    FIN(route_pk());
    pkptr = op_pk_get(op_intrpt_strm ());
    op_pk_nfd_get_int32 (pkptr, "dest_address",
        &dest_address);
    op_pk_send (pkptr, dest_address);
    FOUT;
    }
```

3 Choose **File > Save** when you are finished.

This code executes when the **FSM** follows the transition.

The first line after FIN(route_pk()) has two effects: it retrieves the arriving packet from the appropriate input stream (whose index is determined by **op_intrpt_strm()**). **op_pk_get()** then uses the packet-stream index argument to return a pointer to the packet.

Next, the process must obtain the destination address contained in the packet. The destination address is held in the packet's **dest_address** field, which is an integer data type. The second line of code assigns the destination address to the **dest_address** local variable.

Recall that the destination address corresponds directly to the outgoing packet stream index of the hub node. The final line of code sends the packet to the correct point-to-point transmitter based on the destination address.

Finally, you can define the model interface attributes for the process and compile the model:

1. Select **Interfaces > Process Interfaces**.

2. Set the begsim interrupt attribute to enabled.

3 Add a comment if you want, then click **OK** when you are finished with the Process Interfaces.

Next, compile the model:

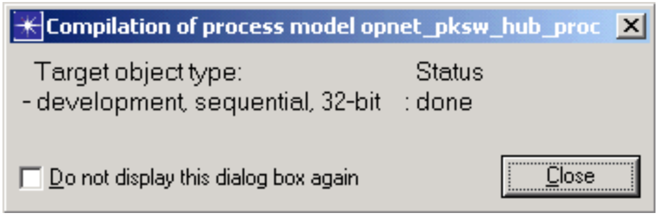
1 Click on the **Compile Process Model** toolbar button.

2 Name the file **<initials>_pksw_hub_proc**, then click **Save**.

➥ The compilation status dialog box appears.

3 When the Status changes to **done**, click **Close**.

**Compilation Status Dialog Box**



Compilation of process model opnet_pksw_hub_proc

```
Target object type:                    Status
- development, sequential, 32-bit    : done
```

4 Select **File > Close** to close the Process Editor.

After you define the hub process model, you can set the **process model** attribute of the **hub** processor module in the Node Editor.

1. Choose **Windows > Node Models > <initials>_pksw_hub**.

   ➥ The Node Model Editor window becomes active.

2. Right-click on the **hub** module and select **Edit Attributes**. Change the **process model** attribute to **<initials>_pksw_hub_proc**.

3. Click **OK** to close the Attributes dialog box.

4. Save the node model.

## Creating the Peripheral Node Model

When a peripheral node generates a packet, it must assign a destination address to the packet, then transmit it to the hub. When it receives a packet, the node must record the packet's end-to-end delay. To accomplish these tasks, a peripheral node model must consist of a generator module, a processor module, and a point-to-point transmitter and receiver.

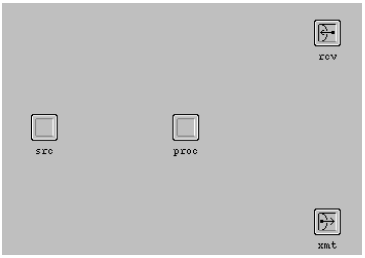
To create the peripheral node model:

**1** Make sure you have saved your hub model, then choose **Edit > Clear Model**.

➥ The workspace is cleared. This avoids the step of closing and reopening the Node Editor. Remember to use **Save As...** when you save the model you are about to construct.

**2** Place and name the modules as shown.

**Initial Modules**



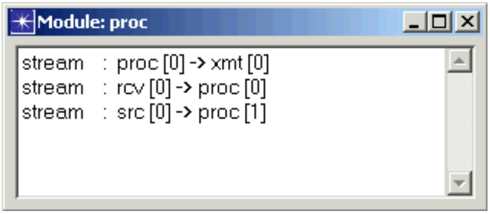
3 Open the Attributes dialog box of the **src** module, change the **process model** attribute to **simple_source**, then click **OK** to close the **Attribute** dialog box.

4 Create packet streams to connect the different modules in the following order:

- **rcv** to **proc**
- **proc** to **xmt**
- **src** to **proc**

Verify the assignment of packet streams between the modules. Right-click on the **proc** module and choose **Show Connectivity** from the pop-up menu to see the following table.

**Show Connectivity Dialog Box (Detail)**



```
stream   : proc [0] -> xmt [0]
stream   : rcv [0] -> proc [0]
stream   : src [0] -> proc [1]
```

If the streams are not correct, the simulation might not generate results. If necessary, correct the streams by deleting all links and recreating them as specified in Step 4.

After you create the network models, you will want to run simulations for different packet interarrival times, setting the **Packet Interarrival Time** attribute just before running the simulations. To make it possible to set the attribute then, you need to promote the attribute. To do this, you need to edit the attributes of the **src** module:

**1** Right-click on the **src** module to open its Attributes dialog box.

**2** Click on **Packet Interarrival Time** in the left column to highlight the Attribute name, then right-click and select **Promote Attribute to Higher Level** from the pop-up menu.

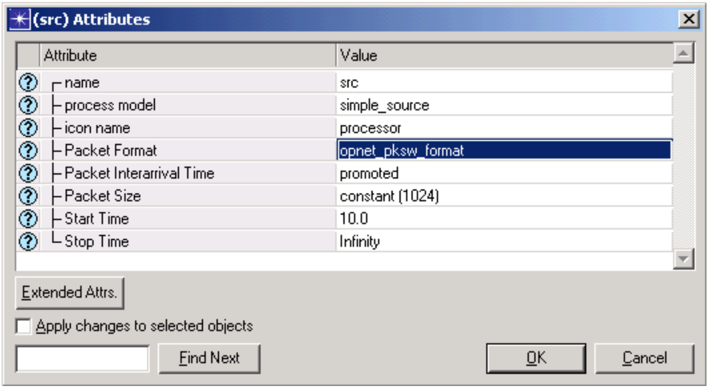
➥ The **Packet Interarrival Time** attribute is promoted so that its value can be set later.

We also want the processor to create packets with the previously-defined format.

**3** Specify **<initials>_pksw_format** as the value for the **Packet Format** attribute.

**4** Verify that your **(src) Attributes** table is similar to the one in the following figure, then click **OK**.

**Packet Format Attribute is <initials>__pksw_format**


(src) Attributes

| Attribute | Value |
|---|---|
| name | src |
| process model | simple_source |
| icon name | processor |
| Packet Format | opnet_pksw_format |
| Packet Interarrival Time | promoted |
| Packet Size | constant (1024) |
| Start Time | 10.0 |
| Stop Time | Infinity |

Next, we want to change the data rate and supported packet formats for the receiver and transmitter. You did this same operation earlier, in the hub module:

1 Hold down the Shift key and left-click on the receiver and transmitter modules.

➥ This selects both modules.

Make sure the packet streams are unselected. This should happen when you select the first receiver.

2 Right-click on one module and select **Edit Attributes**.

**3** Click on the value of the **channel** attribute. In the **(channel) Table** dialog box, set the **data rate (bps)** to **9600**.

**4** Support the **<initials>_pksw_format** format.

**4.1** Click on the value of the **packet formats** attribute.

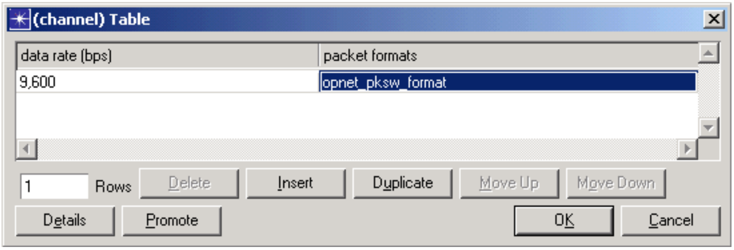
**4.2** Unselect both checkboxes to turn off **Support all packet formats** and **Support unformatted packets**.

**4.3** Click on the status field for **<initials>_pksw_format**.

**4.4** Click **OK** to close the **Select Supported Packet Formats** dialog box.

**5** Verify that the correct data rate and packet formats display in the **(channel) Table** dialog box, then click **OK**.

## Supporting the Custom Packet Format


(channel) Table

| data rate (bps) | packet formats |
|---|---|
| 9,600 | opnet_pksw_format |

6 In the open Attributes dialog box, click on the **Apply changes to selected objects** checkbox, then click **OK**. (If a confirm dialog box appears, click **Yes**.)

➥ The message “2 objects changed” appears in the status bar.

Both selected modules now have a data rate of 9600 and use the **<initials>_pksw_format** packet format.

The next step in defining the peripheral node model is to specify that only fixed node types are supported. You did this same operation earlier, in the hub module.

Because the node is fixed, not mobile or satellite, you need to edit the model attribute interfaces:

1 Choose **Interfaces** > **Node Interfaces**.

2 In the Node Types table, change the **Supported** value for **mobile** and **satellite** types to **no**, and add a comment to describe the model, if you want.

3 Do not close the **Node Interfaces** dialog box. You still need to use it.

---

Renaming attributes allows you to simplify complex attribute names or to expand terse ones.

---

You may want to simplify attribute names, especially when long hierarchical names are confusing and unnecessary. You may also want to expand a highly-abbreviated attribute name to make it more understandable.

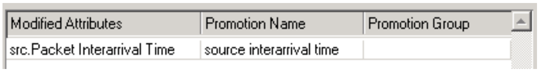
For this model, you will rename the interarrival time attribute:

1 In the **Node Interfaces** dialog box, click on the **Rename/Merge...** button.

2 In the Unmodified Attributes pane, select the **src.Packet Interarrival Time** attribute, then click on the **>>** button to move it to the **Modified Attributes** pane.

3 Change the text in the **Promotion Name** column to **source interarrival time.**

**Renaming an Attribute**



| Modified Attributes | Promotion Name | Promotion Group |
|---|---|---|
| src.Packet Interarrival Time | source interarrival time | |

4 Click **OK** to close the **Rename/Merge Attributes** dialog box.

You can assign a set of symbolic values to an attribute to provide a simpler user interface. The values you assign are displayed as options in a list.

There are several advantages to assigning symbolic names to attribute values:

- Options can be limited to appropriate values.
- Options can be assigned descriptive names, making it easier for a user to select the correct value.
- Users can select values from a list, avoiding typographical errors.

Add symbolic values to the **source interarrival time** attribute:

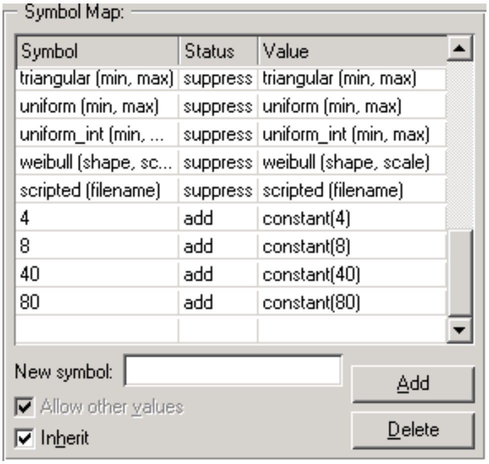
**1** In the Attributes table of the **Node Interfaces** dialog box, select the newly-renamed **source interarrival time** attribute, then click on the **Edit Properties...** button.

➥ The **Attribute: source interarrival time** dialog box appears.

**2** In the **Symbol Map** table, change the status of all symbols to **suppress** by clicking in the status column of each row.

**3** Add entries in the **Symbol Map** as shown, then click **OK** to close the dialog box.

## Adding Symbolic Values


Symbol Map:

| Symbol | Status | Value |
|---|---|---|
| triangular (min, max) | suppress | triangular (min, max) |
| uniform (min, max) | suppress | uniform (min, max) |
| uniform_int (min, ... | suppress | uniform_int (min, max) |
| weibull (shape, sc... | suppress | weibull (shape, scale) |
| scripted (filename) | suppress | scripted (filename) |
| 4 | add | constant(4) |
| 8 | add | constant(8) |
| 40 | add | constant(40) |
| 80 | add | constant(80) |
| | | |

New symbol:

[✓] Allow other values

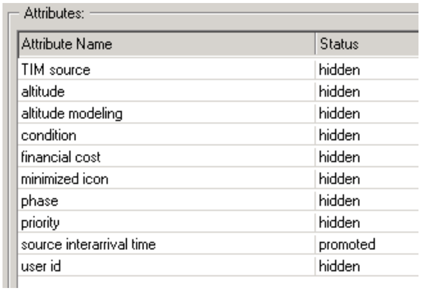
[✓] Inherit

Hiding attributes simplifies an interface. This is desirable when the end user does not need to change an attribute value. Hiding an attribute does not affect its value during simulation.

Many attributes that appear in the peripheral node are not relevant to this model. To avoid confusion and clutter, these attributes can be hidden:

1 In the Attributes table of the Node Interfaces dialog box, change the status of all attributes except **source interarrival time.**

**Node Interfaces Dialog Box**



| Attribute Name | Status |
|---|---|
| TIM source | hidden |
| altitude | hidden |
| altitude modeling | hidden |
| condition | hidden |
| financial cost | hidden |
| minimized icon | hidden |
| phase | hidden |
| priority | hidden |
| source interarrival time | promoted |
| user id | hidden |

**2** Click **OK** to close the Node Interfaces dialog box.

**3** Choose **File** > **Save As...**, name the node model **<initials>_pksw_node**, then close the Node Editor.

## Creating the Process Model of the Peripheral Node

The peripheral node process model does two main functions: (1) assigns destination addresses and sends packets, and (2) keeps track of end-to-end delay.

To complete its tasks, the peripheral node process model needs two states:

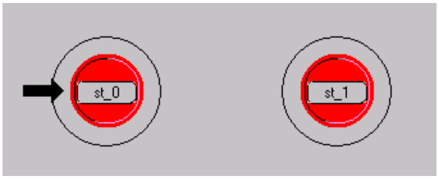
- an initial state
- an idle state

To create the process model:

1 Choose **File > New...**, select **Process Model** from the pull-down list, and click **OK**.

2 Put two states in the editor window as follows:

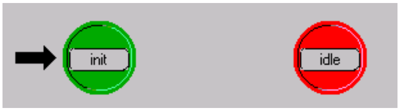
**Initial States**



**3** Change the attributes of the states, as indicated:

**3.1** For the initial state, change the **name** attribute to **init** and the **status** to **forced**.

**Changed States**



**3.2** For the other state, change the **name** attribute to **idle**. (Leave the **status** as **unforced**.)

In the **init** state, the process model will load a uniform PDF in a range of 0 to 3.

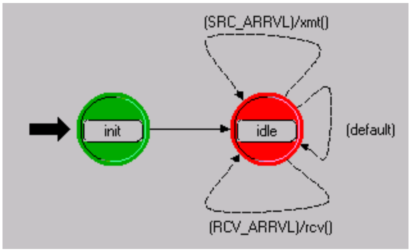
Next, create transitions between the states in the process model:

**1** Draw the transitions between the states and specify the transition conditions and executives as shown below.

The **xmt()** transition executive will generate and assign destination addresses to packets as they arrive from a generator. Packets will then be sent on to the point-to-point transmitter.

The **rcv()** transition executive is entered when a packet arrives. In the executive, the process model will determine the packet's end-to-end delay, update the global statistic, and destroy the packets.

**Drawing Transitions**



2 Next, click on the **edit header block** button to define the transition macros for **SRC_ARRVL** and **RCV_ARRVL** in the header block.

3 Enter the following code:

```
/* packet stream definitions */
#define RCV_IN_STRM 0
#define SRC_IN_STRM 1
#define XMT_OUT_STRM 0

/* transition macros */
#define SRC_ARRVL (op_intrpt_type () == \
   OPC_INTRPT_STRM && op_intrpt_strm () == SRC_IN_STRM)

#define RCV_ARRVL (op_intrpt_type () == \
   OPC_INTRPT_STRM && op_intrpt_strm () == RCV_IN_STRM)
```

Note the use of constant definitions of stream indices (`RCV_IN_STRM`, `SRC_IN_STRM`, and `XMT_OUT_STRM`) for easier code interpretation. The stream indices correspond to the values set in the node model.

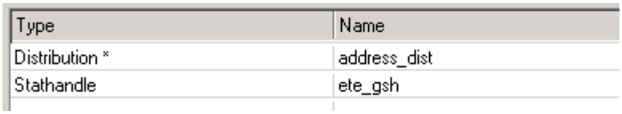
4 Choose **File > Save**.

Next, define the state and temporary variables:

1 Click on the **Edit State Variables** toolbar button.

2 Enter the following information in the table:

**State Variable Dialog Box (Detail)**



| Type | Name |
|---|---|
| Distribution * | address_dist |
| Stathandle | ete_gsh |

3 Click **OK** when you are finished with the state variables.

Next, you need to define the end-to-end delay statistic that will be collected when each packet arrives at its destination.

1 Choose **Interfaces > Global Statistics**.

2 Enter the **Stat Name** as **ETE Delay**.

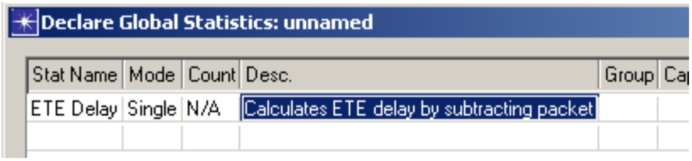
3 Click in the **Desc.** column and enter the following text in the dialog box:

```
Calculates ETE delay by subtracting packet
creation time from current simulation time.
```

4 Choose **File > Save**.

5 Verify that the completed **Declare Global Statistics** dialog box looks like this:

Declare Global Statistics Dialog Box (Detail)



| Stat Name | Mode | Count | Desc. | Group | Ca |
|---|---|---|---|---|---|
| ETE Delay | Single | N/A | Calculates ETE delay by subtracting packet | | |
| | | | | | |

6 Close the **Declare Global Statistics** dialog box by clicking on the **OK** button.

Finally, you need to add the enter and exit executives for each of the states in the process model. First add the enter execs for the **init** state:

**1** Double-click on the top half of the **init** state to open its **enter execs**, then enter the following code:

```
address_dist = op_dist_load ("uniform_int", 0, 3);
ete_gsh = op_stat_reg ("ETE Delay",
    OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
```

**2** Choose **File** > **Save**.

The **xmt()** transition executive acts when a packet arrives from the generator module. A destination address must be assigned before the packet is sent to the point-to-point transmitter.

**3** In the **Function Block**, enter the following code:

```
static void xmt(void)
    {
    Packet * pkptr;
    FIN(xmt());
    pkptr = op_pk_get (SRC_IN_STRM);
    op_pk_nfd_set_int32 (pkptr, "dest_address",
        (int)op_dist_outcome (address_dist));
    op_pk_send (pkptr, XMT_OUT_STRM);
    FOUT;
    }
```

The first statement obtains a pointer to the packet arriving from the generator. The next statement sets the value of the **dest_address** field, which is an integer data type, to the value returned by the **op_dist_outcome()** procedure.

**op_dist_outcome()** returns a random number according to the distribution given as an argument. In this case, **address_dist** is a pointer to the uniform integer distribution loaded in the **init** state.

The last statement sends the packet to the output stream, which is connected to the point-to-point transmitter.

The **rcv** transition executive is entered when a packet arrives on the stream connected to the point-to-point receiver. The purpose of the **rcv** executive is to calculate and record the end-to-end delay of the packets using a global statistic.

**4** In the **Function Block**, enter the following code:

```
static void rcv(void)
    {
    Packet *pkptr;
    double ete_delay;
    FIN (rcv());
    pkptr = op_pk_get (RCV_IN_STRM);
    ete_delay = op_sim_time () -
        op_pk_creation_time_get (pkptr);
    op_stat_write (ete_gsh, ete_delay);
    op_pk_destroy (pkptr);
    FOUT;
    }
```

The first statement in the code after FIN (rcv()) obtains a pointer to the packet arriving from the point-to-point receiver.

The next statement calculates the end-to-end delay by subtracting the packet's creation time from the current simulation time.

The third statement writes the end-to-end delay to a global statistic, and the fourth statement destroys the packet.

**5** Choose **File** > **Save** to close the pad.

Enable the beginning simulation interrupt. This interrupt starts the process.

1. Choose **Interfaces > Process Interfaces**.
2. In the **Process Interfaces** dialog box, change the initial value of the **begsim intrpt** attribute to **enabled**.

   You may want to add a comment to describe the process.
3. When you are finished, click **OK** to close the dialog box.

Finally, compile the process model, as follows:

1 Click on the **Compile Process Model** toolbar button.

2 Supply the file name **<initials>_pksw_nd_proc** and click **Save**.

3 When the process model has finished compiling, close the **Compilation** dialog box then close the Process Model Editor. (If the model does not compile, check for typographical errors.)

Now that the peripheral node process model is complete, you can update the process model attribute of the **proc** processor module:

1 If a Node Editor is not already open with the **<initials>_pksw_node** node model, select **File > Recent Files > <initials>_pksw_node**.

2 Right-click on the **proc** module and select **Edit Attributes** from the Object pop-up menu, then change the **process model** attribute to **<initials>_pksw_nd_proc**.

3 Click **OK** in the Edit Attribute dialog box once you have changed the **process model** attribute.

4 **Save** the changes made to the node model, then close the Node Editor.

You are now ready to build the network model.

## Building the Network

For the packet switching network, neither the subnet extent nor the grid properties are important.

Now that you have built the underlying node, process, and link models, you can build the network model. Remember that the initial network contains one hub node and four peripheral nodes. To begin:

1 Choose **File** > **New...** Select **Project** from the pull-down list, then click **OK**.

2 Enter the project name as **<initials>_pksw_net**, and the scenario name as **baseline**, then click **OK**.

3 When the Startup Wizard appears, click **Quit**. You will specify the scale of the network and a palette manually.

The first thing to do is create an object palette that contains the necessary models.

1 Click on the **Open Object Palette** toolbar button.

2 Click on the **Configure Palette...** button in the object palette.

3 In the **Configure Palette** dialog box, click on the **Clear** button, then click on the **Node Models** button.

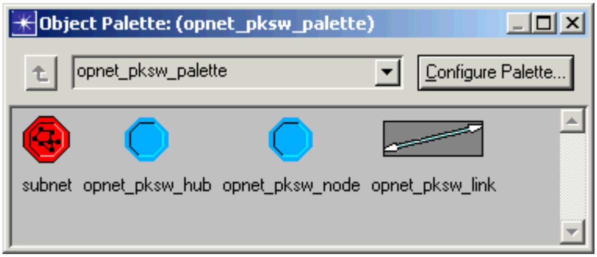
4 Include **<initials>_pksw_hub** and **<initials>_pksw_node**, then click **OK**.

5 In the **Configure Palette** dialog box, click on the **Link Models** button.

6 Include the **<initials>_pksw_link** model, then click **OK**.

7 Click **OK** in the **Configure Palette** dialog box and name the palette **<initials>_pksw_palette**. You will not see mobile or satellite nodes unless you are using the Wireless module, but your palette should look similar to this:

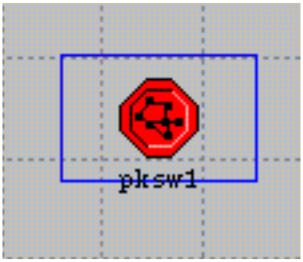
**Customized Object Palette**



You are now ready to build the network:

1 Place a subnet in the editor window and name it **pksw1**.

**Network Model**



2 Double click on the subnet to move to the subnet view.

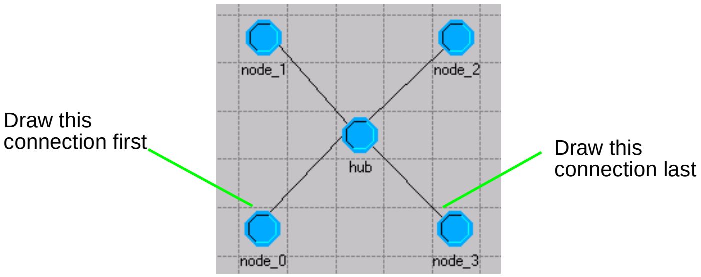
3 Place four **<initials>_pksw_node** objects in the window.

4 Place one **<initials>_pksw_hub** in the window. Name that node **hub**.

5 Beginning at node_0, and using **<initials>_pksw_link**, connect each peripheral node to the hub as shown in the following figure.

Drawing connections in the order node_0. node_1, node_2, and node_3 ensures that your statistics match those shown in this lesson. If you make connections in a different order, your simulation results may differ.

**Connecting Peripheral Nodes to the Hub**

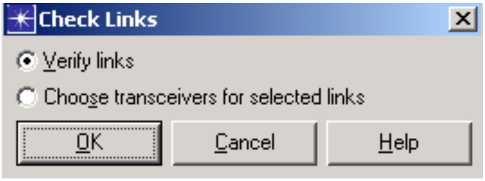


Before you save the project, it is a good idea to get into the habit of checking the links in the network for consistency:

1 Click on the **Verify Links** toolbar button.

2 Make sure that the **Verify Links** option is checked, then click **OK**.

**Check Links Dialog Box**


Check Links

[● Verify links]

○ Choose transceivers for selected links

**3** If a red **X** appears over a link, the link is not consistent. In that case:

**3.1** Click on the **Verify Links** toolbar button again.

**3.2** Click on the **Choose transceivers for selected links** checkbox, then click on the **OK** button.

➥ The red **X** disappears when the link inconsistency is resolved.

For a link to be consistent, the packet formats and data rates of the transmitters and receivers within the objects must match those defined for the link.

## Choosing, Collecting, and Analyzing Results

Now that you have built the model, you are ready to simulate network behavior.

To see the effect of the packet generation rate on the performance of the network, you will use the Simulation Editor to create two simulations sets for different packet interarrival times.

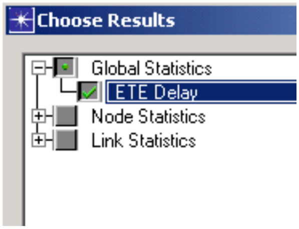
## Choosing Results

First, choose the results to be collected during the simulation: end-to-end delay and link utilization.

1 Right-click in a blank area of the editor window, then select **Choose Individual DES Statistics** from the pop-up menu.

2 Under **Global Statistics**, choose **ETE Delay**. This was the statistic you defined in the peripheral nodes' process model. When you are finished, click **OK**.

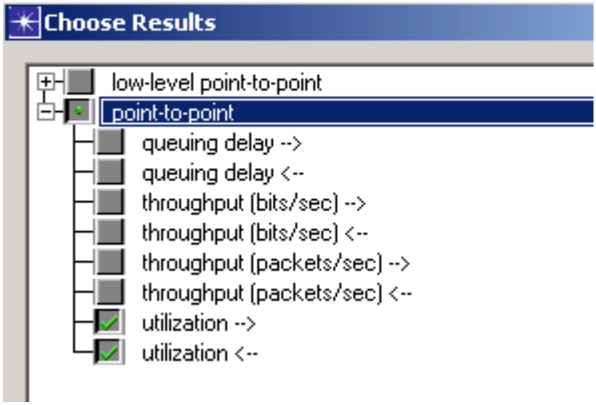
Choose Results Dialog Box



3 Right-click on the link between **node_0** and the **hub**, then select **Choose Individual DES Statistics** from the object pop-up menu.

4 Under **point-to-point**, choose both **utilization** statistics. Click **OK** when you are finished.

## Utilization Statistics Selected


Choose Results

- low-level point-to-point
- point-to-point
  - queuing delay -->
  - queuing delay <--
  - throughput (bits/sec) -->
  - throughput (bits/sec) <--
  - throughput (packets/sec) -->
  - throughput (packets/sec) <--
  - ✓ utilization -->
  - ✓ utilization <--

**5** Save the project with the default name.

## Configuring the Simulation

In this simulation, the source processor creates packets of a constant size. This setting, in combination with the fixed data rate of the point-to-point transmitters and receivers, results in a fixed end-to-end delay for the packets.

However, if packets are sent to a transmitter quickly enough, some of the packets will be delayed in the transmitter's queue. If the packet interarrival time (**source interarrival time** attribute) is varied, the end-to-end delay will be affected. To model this, configure two simulations with different packet interarrival times:

1 Choose **DES** > **Configure/Run Discrete Event Simulation (Advanced)**.

➥ The Simulation Editor opens, showing a single simulation set.

2 Right-click on the simulation set and select **Edit Attributes** from the pop-up menu.

Now you will define two simulations for two different packet interarrival times. Recall that we promoted the generator's **interarrival time** attribute so we could define it now, at simulation time.

To define the first simulation with a packet interarrival time of 4:

1. In the **Common** tab, change the **Simulation set name** to **<initials>_pksw_sim1**.

2. Change the **Duration** to **1,000 seconds** and the **Seed** to **21**.

3. Define the **interarrival time** attribute, as follows:

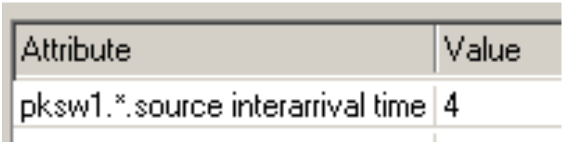
   **3.1** Click on the **Object Attributes** tab.

   **3.2** Click the **Add...** button.

   **3.3** In the Add Attribute dialog box, click in the **Add?** column next to the **pksw1.*.source interarrival time** attribute and click **OK**.

   **3.4** In the Simulation Set dialog box, click in the **Value** column, then choose **4** from the pull-down list.

**Simulation Set Dialog Box (Detail) for the First Simulation**



| Attribute | Value |
|---|---|
| pksw1.*.source interarrival time | 4 |

4 Click on the **Advanced:Files** tab, then change the name of the **Vector File** to **<initials>_pksw_sim1**.

5 Click **Apply** to save your changes, then close the dialog box by clicking **Cancel**.

To define the second simulation:

1 Select the existing simulation set, copy it, then paste it below the first set.

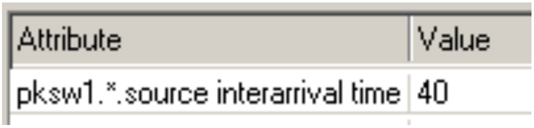
➥ The new set is automatically named **<initials>_pksw_sim2**.

2 Right-click on the simulation set and select **Edit Attributes** from the pop-up menu.

3 Under the **Common** tab, verify that **Duration** is set to **1000 seconds**.

4 Follow the method you used earlier to change the value of the **source interarrival time** to **40**.

Simulation Set Dialog Box (Detail) for the Second Simulation



| Attribute | Value |
|---|---|
| pksw1.*.source interarrival time | 40 |

5 Click on the **Advanced:Files** tab, then change the name of the **Vector File** to **<initials>_pksw_sim2**.

6 Click **Apply** to save, then close the dialog box.

7 Select **File > Save** to save the simulation sequence.

Make sure that the **repositories** preference is set correctly before running the simulation.

1 Choose **Edit** > **Preferences**.

2 Make sure the **repositories** preference has the value **<empty>**.

## Running Both Simulations

To run the simulations, do the following:

1. Click the **Execute Simulation Sequence** toolbar button to begin simulation execution.

   ➥ The simulation takes approximately 1 minute to run, depending on the speed of your machine.

2. After the simulation completes, close the simulation message window and exit the simulation sequence. If you had problems, see *"Troubleshooting Tutorials"*.

## Analyzing Results

Because you ran two separate simulations using two simulation sets created in the Simulation Editor, you must use the Analysis Tool to view the results of your simulations.

To compare the end-to-end delay and utilization between the two scenarios:

1. Choose **File** > **New...** Select **Analysis Configuration** from the pull-down list and click **OK**.

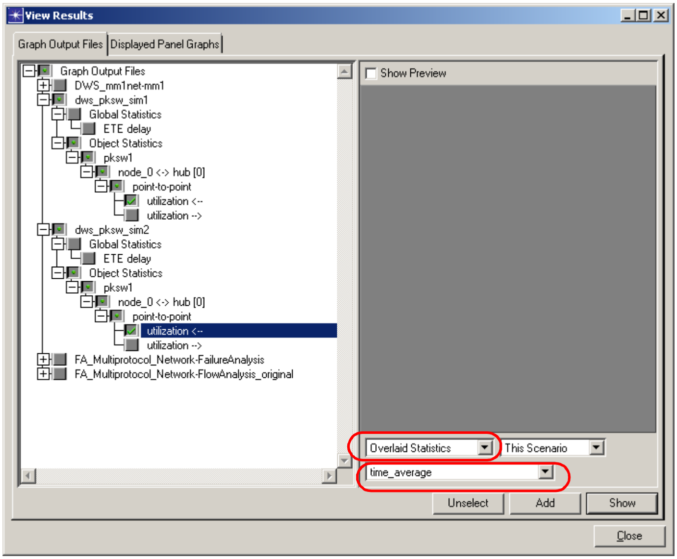
2. Click on the **Create a Graph of a Statistic** toolbar button.

3. The **View Results** dialog box opens. It lists results either by output vector file name or by project-scenario name. Look for the output vector files you specified (**<initials>_pksw_sim1** and **<initials>_pksw_sim2**.)

4. Select the box for **Object Statistics: pksw1: node_0 <-> hub[0]: point-to-point: utilization <- -**.

   Do this for BOTH output vector files (**<initials>_pksw_sim1** and **<initials>_pksw_sim2**).

## Utilization <-- Statistic is Checked for Both Output Vector Files



- Graph Output Files
  - DWS_mm1net-mm1
  - dws_pksw_sim1
    - Global Statistics
      - ETE delay
    - Object Statistics
      - pksw1
        - node_0 <-> hub [0]
          - point-to-point
            - [✓] utilization <--
            - utilization -->
  - dws_pksw_sim2
    - Global Statistics
      - ETE delay
    - Object Statistics
      - pksw1
        - node_0 <-> hub [0]
          - point-to-point
            - [✓] utilization <--
            - utilization -->
  - FA_Multiprotocol_Network-FailureAnalysis
  - FA_Multiprotocol_Network-FlowAnalysis_original

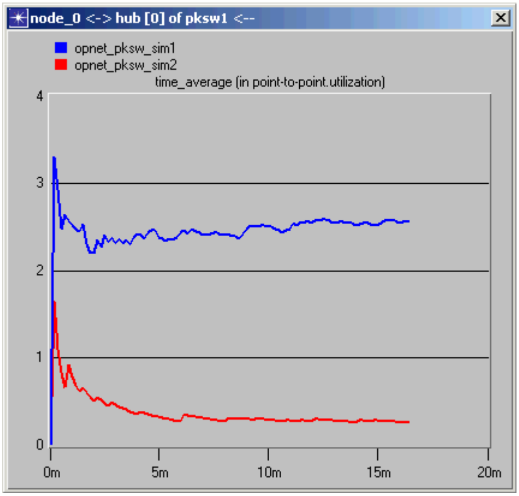
Overlaid Statistics | This Scenario | time_average

5 Make sure the graph is set to show **Overlaid Statistics**, then select **time_average** from the list of available filters.

6 Click **Show**.

The time-averaged graph of utilization for this lesson is shown in the following figure.

**Time-averaged Utilization for Both Simulations**



A large reduction in packet generation results in a corresponding reduction in link utilization.

You can change the x-axis to display seconds by right-clicking in the panel (not the graph), then selecting **Time Axis > seconds** from the pop-up menu.

Close and delete the graph when you are finished.

You also want to look at the end-to-end delay of the packets to see if any queuing occurred.

1. Click the **Unselect** button in the **View Results** dialog box.

2. Change the filter type back to **As Is**.

3. From **<initials>_ pksw_sim1**, choose **Global Statistics: ETE Delay**, then click **Show**.

4. Move the graph so that it does not overlap the **View Results** dialog box.

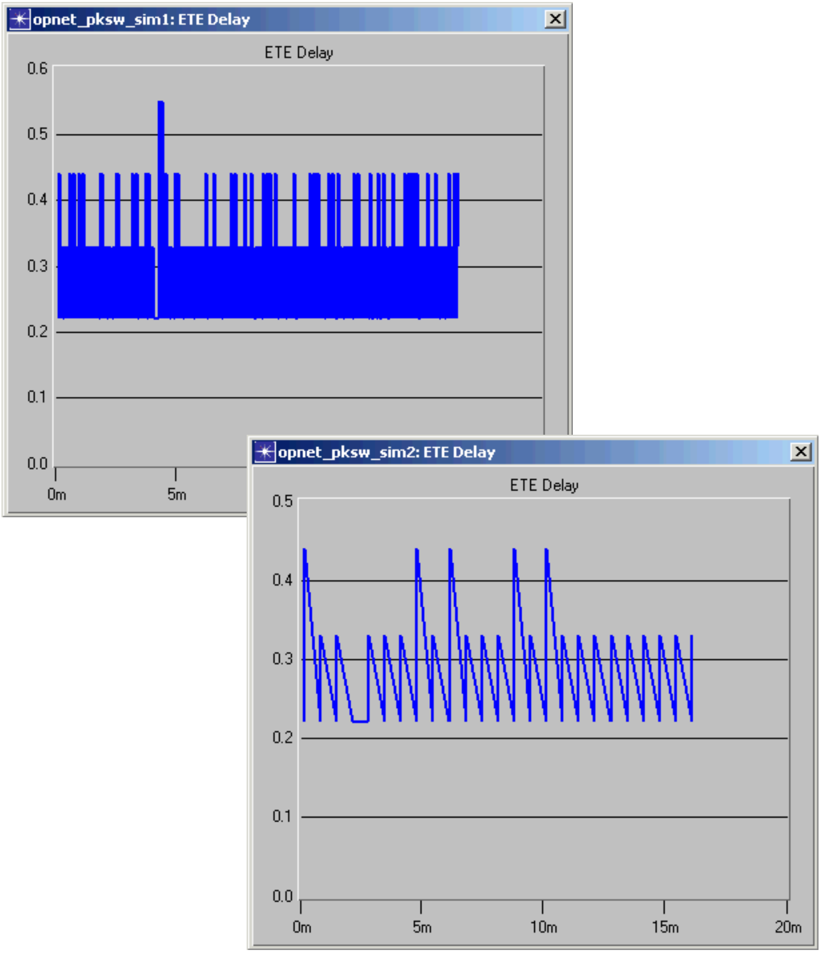
5. Click the **Unselect** button in the **View Results** dialog box.

6. From **<initials>_ pksw_sim2**, choose **Global Statistics: ETE Delay**, then click **Show**.

   ➥ The two graphs are shown in the following figure.

The graphs for end-to-end delay should resemble the ones in the following figure.

**End-to-End Delay for Both Simulations (Linear Draw Style)**



In linear draw style, the graphs do not clearly show the delay experienced by individual packets. To show this, you can change the draw style to discrete.

To change the draw style of one of the ETE delay graphs to discrete:

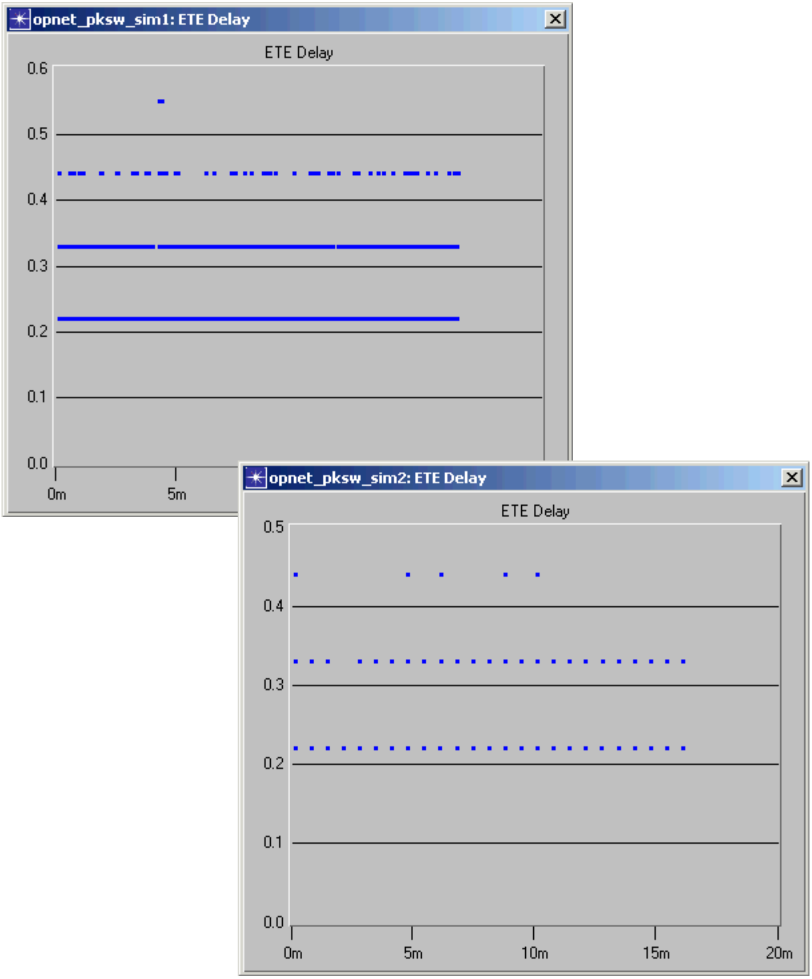
1. Right-click anywhere in the graph, and select **Draw style > Discrete** from the pop-up menu.

   ➥ The graph changes from a linear to discrete draw style.

2. Perform the same operation for the other ETE Delay graph.

The discrete graphs of ETE delay should resemble those in the following figure:

**End-to-End Delay for Both Simulations (Discrete Draw Style)**



In the next tutorial, you enhance the current network model by adding a second star network. Return to the main tutorial menu and choose **Packet Switching II** from the list of available lessons.