5. Enhancing the Model with Deference (CSMA)

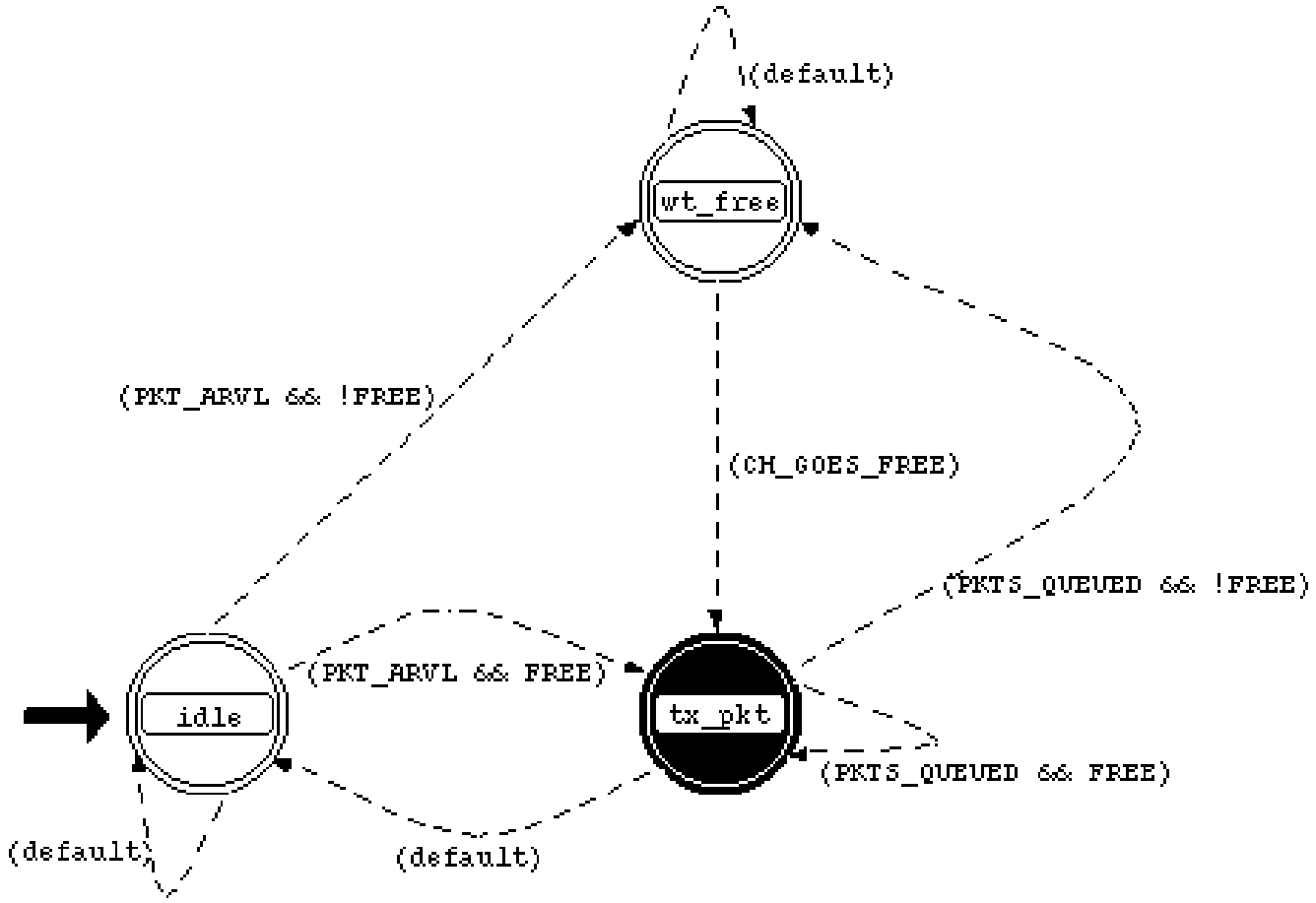
The performance of the Aloha random access protocol can be enhanced by adding a carrier sense capability. The carrier sense capability is employed in the classical CSMA protocol, which requires a source node to sense the channel and determine that it is free before committing to a transmission.

You can enhance the existing **<initials>_aloha_tx** process model so that the process waits until the channel is free before transmitting a packet.

1) Select **Open** from the **File** menu. Select **Process Model** from the pull-down menu.

2) Select the **<initials>_aloha_tx** model, then click **OK**.

Modify the states and transitions so that the model appears as shown (detailed instructions are given below and on the next page):



1) Create a new state and name it **wt_free**.

2) Create a transition from **wt_free** to **tx_pkt**, and change the condition to **CH_GOES_FREE**.

3) Create a transition from the **wt_free** state back to itself and set the condition to **default**.

Directions for enhancing the **<initials>_aloha_tx** model continue on this page:

4) Create a transition from the **idle** state to **wt_free** and change the condition to **PKT_ARVL && !FREE**.

5) Add a transition from the **idle** state back to itself with a condition of **default**.

6) Change the condition on the transition from **idle** state to the **tx_pkt** state to **PKT_ARVL && FREE**.

7) Change the unconditional transition from **tx_pkt** to **idle** to conditional by setting the condition attribute to **default**.

8) Create a transition from **tx_pkt** back to itself, and set the condition to **PKTS_QUEUED && FREE**.

9) Finally, create a transition from **tx_pkt** to **wt_free** and set the condition to **PKTS_QUEUED && !FREE**.

Remember, you can move a condition label by left-clicking on the label and dragging it to a new position.

You also need to edit the header block:

1) Add the following lines to the end of the process model header block.

```
/*  Input statistic indices  */
#define CH_BUSY_STAT 0

/*  Conditional macros  */
#define FREE (op_stat_local_read (CH_BUSY_STAT) == 0.0)
#define PKTS_QUEUED (!op_strm_empty (IN_STRM))
#define CH_GOES_FREE (op_intrpt_type () == OPC_INTRPT_STAT)
```

These changes cause the process to verify that the channel is free before transmitting. If the process wishes to send a packet, it must first confirm that the channel is free by using the Kernel Procedure **op_stat_local_read()** to read the channel's **busy** statistic. If the channel is not free, the process enters the state **wt_free** until a "channel goes free" interrupt is received. At the node level, the underlying statistic wire is triggered when the **busy** statistic changes to 0.0. This triggering is activated by enabling the wire's **falling edge trigger** attribute.

1) Choose **Save As** from the **File** menu and rename the model **<initials>_csma_tx**.

2) **Compile** the model, then **close** the Process Editor.

Enhancing the Generic Transmitter Node Model

You can enhance the generic transmitter node model so that the bus receiver module delivers a falling edge statistic interrupt to the processor module whenever the receiver **busy** statistic changes from “busy” (1.0) to “free” (0.0).

To enhance the generic transmitter node model so that it supports CSMA:

1) Open the **<initials>_cct_tx** node model.

2) Right-click on the **statistic wire** to open its attributes dialog box, then set the **falling edge trigger** attribute to **enabled**.

3) Open the attribute dialog box for the **tx_proc** processor module and change the **process model** attribute to **<initials>_csma_tx**. Click **OK**.

   ➡ The processor now uses a process model which acts on **channel busy** statistic interrupts delivered by the receiver module.

4) Select **Save As** from the **File** menu, and rename the file **<initials>_cct_csma_tx**.

5) Select **Close** from the **FIle** menu.

Redefining the Network Model

Now that you have modified the appropriate models to support CSMA, you need to change the network model to use the new models. Instead of creating an entirely new model, you can duplicate the existing scenario (including the network model) and make the appropriate changes.

1) In the Project Editor, choose **Duplicate Scenario** from the **Scenarios** menu. Name the new scenario **CSMA**.

The only changes that need to be made to the network model is to change the transmitter nodes to the new CSMA transmitter nodes.

2) Add the **<initials>_cct_csma_tx** node model to your object palette.

3) Right-click on one of the transmitter nodes and choose **Select Similar Nodes** from the object pop-up menu.

   ➡ All 20 transmitter nodes are selected.

4) Right-click on any of the selected nodes and choose **Edit Attributes** from the pop-up menu.

5) Change the **model** attribute to **<initials>_cct_csma_tx**.

6) Check **Apply Changes to Selected Objects**, then click **OK**.

   ➡ The node models are changed to **<initials>_cct_csma_tx.**

The next step is to set up a series of simulations to be executed using the CSMA model.

1) Save the changes you made to the project.

2) From the **Simulation** menu, choose **Configure Simulation (advanced)**.

3) Right-click on the simulation set to open its attribute dialog box.

4) Change the **Scalar file** to **<initials>_cct_c**.

5) Change the **Seed** to **11**, then click **OK**.

6) Save the simulation sequence (note that you do not have to rename it, as the simulation sequence file was duplicated and renamed along with the scenario).

7) Be sure to delete any existing output scalar files with the name **<initials>_cct_c**.

8) Execute the simulation. As there are 12 simulations in the set, it may take a few minutes. When you are finished, **close** the Simulation Editor.

Analyzing the CSMA Results

Now that the simulation is complete, you can view the results. Again, since there are scalar results, you must use the Analysis Configuration Editor.

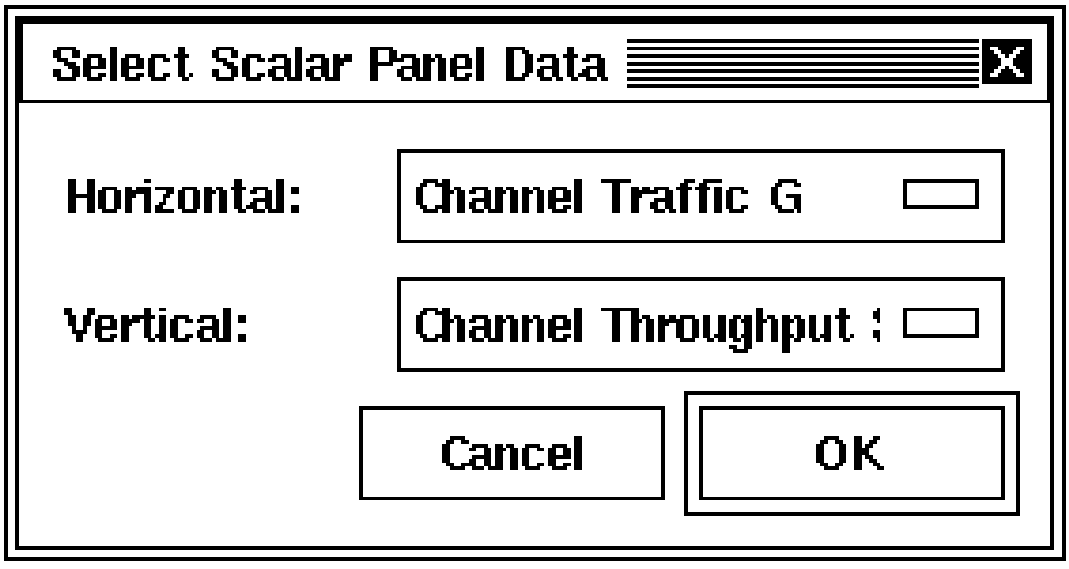
1) In the Project Editor, choose **View Results (Advanced)** from the **Results** menu.

2) In the Analysis Configuration Editor, choose **Load Output Scalar File** from the **File** menu.

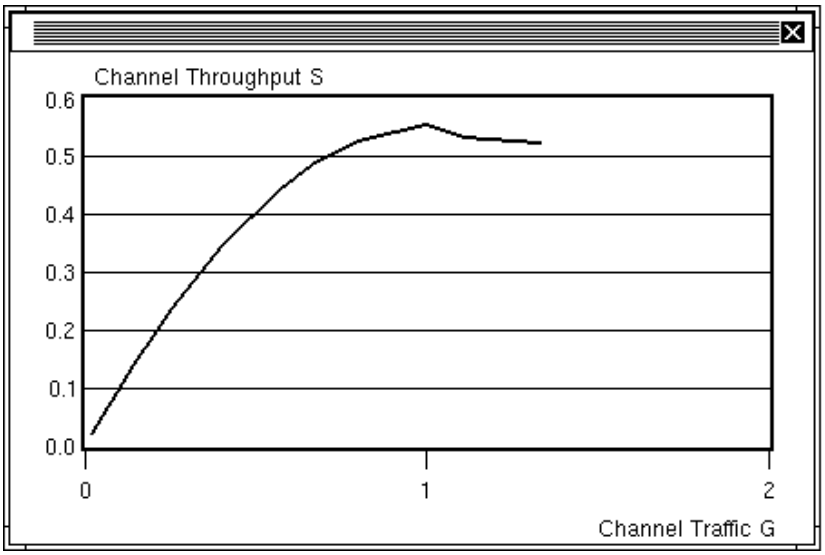
3) Select **<initials>_cct_c** from the list of available files.

4) Click on the **Create a graph of two scalars** action button.

5) In the **Select Scalar Panel Data** dialog box, select the horizontal variable **Channel Traffic G** first, then select **Channel Throughput S** as the vertical variable, then click **OK**.



The resulting graph should resemble the one below:



The CSMA protocol is seen to achieve a maximum channel throughput in excess of 0.5. However, to compare the performance of the Aloha and CSMA protocols directly, it is easier to display both curves simultaneously on the same graph.

To view the results from both simulations on the same graph:

1) Select **Load Output Scalar File** from the **File** menu.

2) Select **<initials>_cct_a** from the menu.

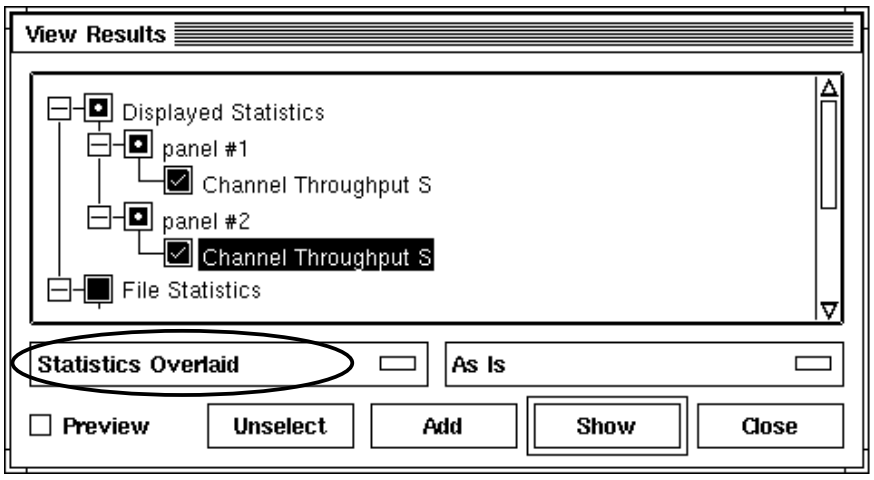
3) Click on the **Create a graph of two scalars** action button.

4) Select the horizontal variable **Channel Traffic G** first, then select the vertical variable **Channel Throughput S** from the menu of available scalars that pops up. Click **OK**.

5) From the **Panels** menu, choose **Create Vector Panel**.

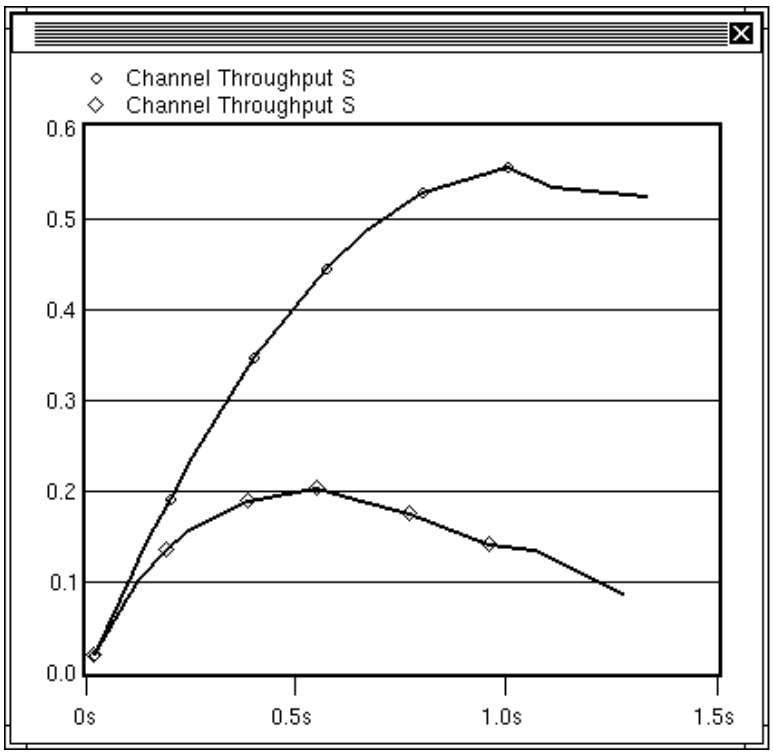
6) Open the **Displayed Statistics** menu, and select both displayed statistics.

7) Change the display mode to **Statistics Overlaid**, then click **Show**.



➡ The resulting graph is shown on the next page.

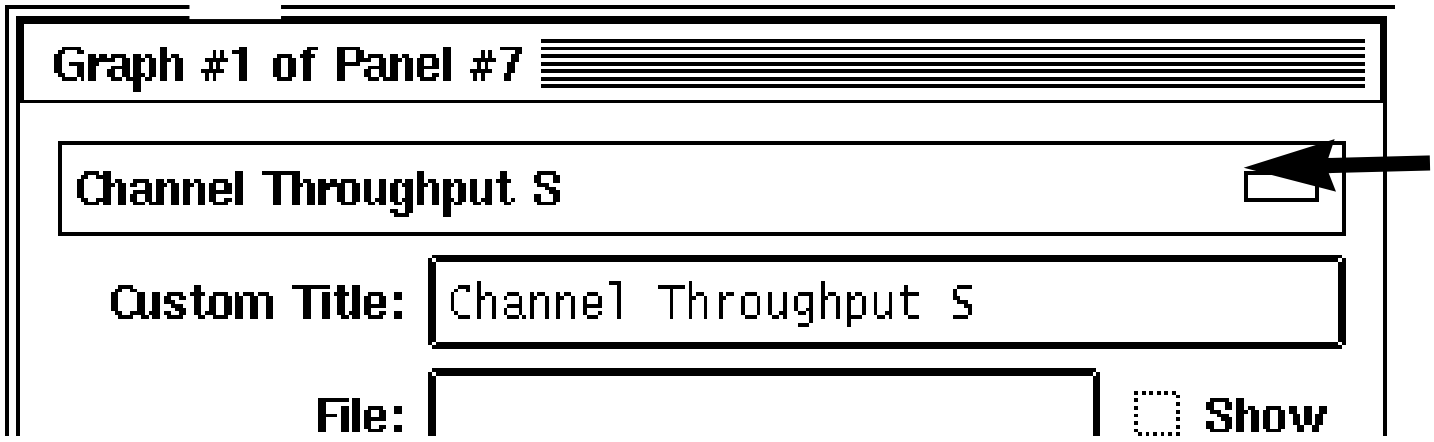
The graph of the two scalars should resemble the graph below:



Because both the Aloha and CSMA simulation executables used the same label in the **op_stat_scalar_write()** Kernel Procedure, only one ordinate label appears. For clarity, the axes labels can be adjusted.

1) Display the **Edit Graph properties** dialog box by right-clicking on the multiple vector graph and selecting **Edit Graph Properties** from the graph area. (Note: If Edit Graph Properties does not appear in the pop-up menu, be sure you clicking in the graph area and not in the area around the graph.)

   ➡ Notice **Active trace** in the top section of the dialog box:



In the **Active trace** menu, the traces are listed in the order in which they were added to the multiple vector graph. Information for the first trace (CSMA) is shown automatically when this dialog box appears. To change the labels for the CSMA trace:

1) Change the **Custom Title** to "**CSMA Channel Throughput**".

2) Click on the **Apply** button at the bottom of the dialog box.

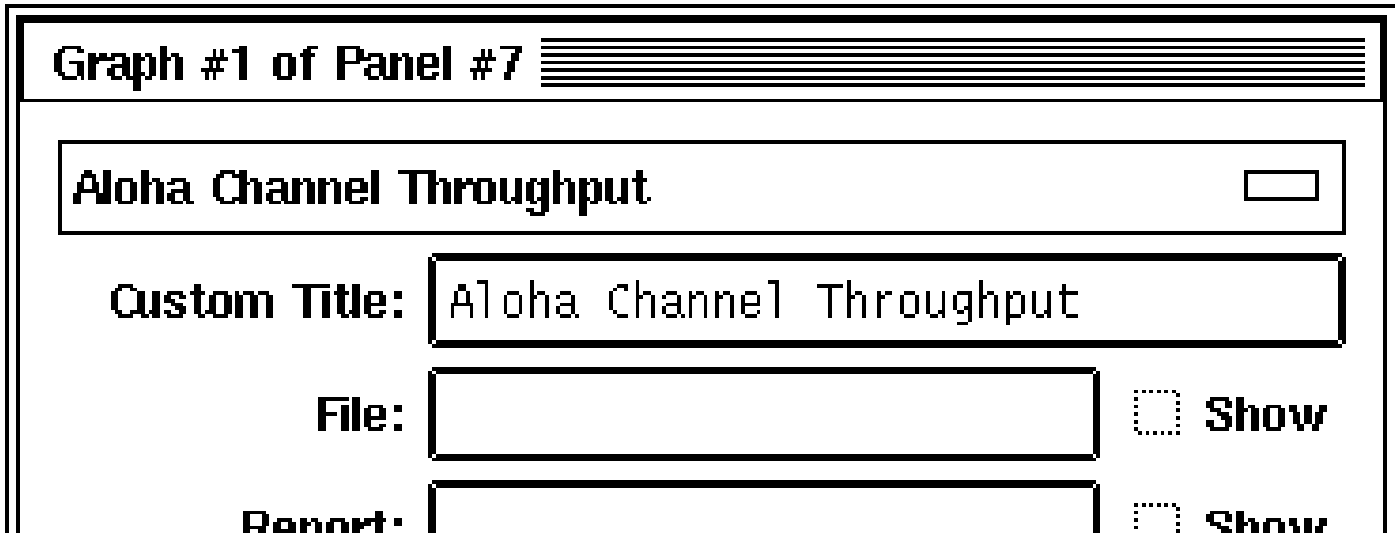
You can also change the labels for the Aloha trace:

1) Change the **Active trace** to **Channel Throughput S**.

2) Change the **Custom Title** to **Aloha Channel Throughput**.

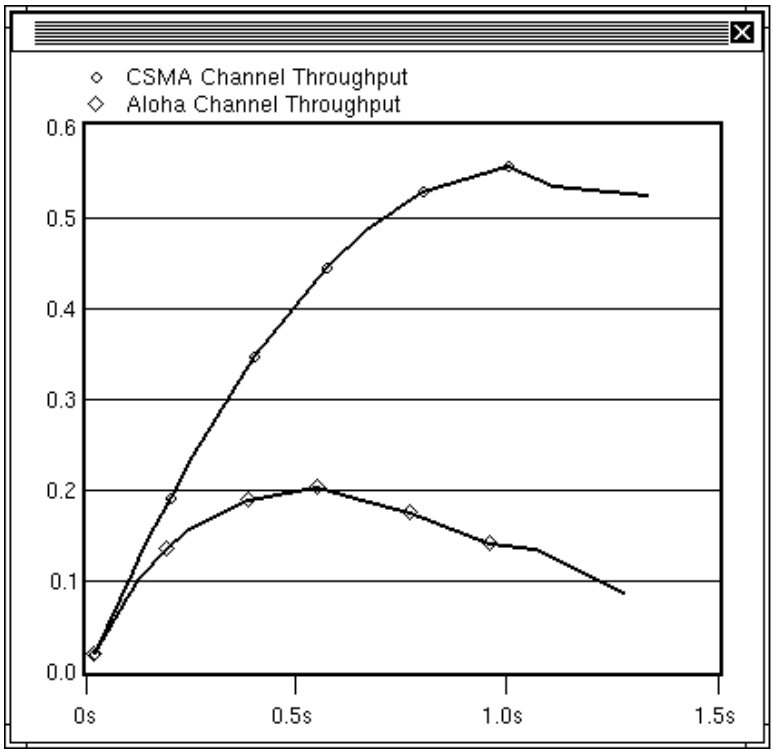
3) Click on the **Apply** button at the bottom of the dialog box.

➡ The top section of the dialog box now looks like this:



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

The graph should appear as shown below:



The CSMA protocol is seen to be superior to the Aloha protocol at all channel traffic loads. The theoretical channel throughput S as a function of channel traffic G in a 1-persistent CSMA channel with negligible propagation delay is given by **S=G(1+G) e-G/(G+e-G)**. This formula predicts a maximum throughput of approximately 0.54 at a channel traffic of approximately 1.0. Although the simulations in this tutorial are brief and limited, the results agree with this prediction.

6. Enhancing the Model with Collision Detection and Backoff (Ethernet)

If a node has full-duplex capability, it can both transmit and monitor a connected bus link at the same time. This capability can be modeled using the Ethernet protocol.

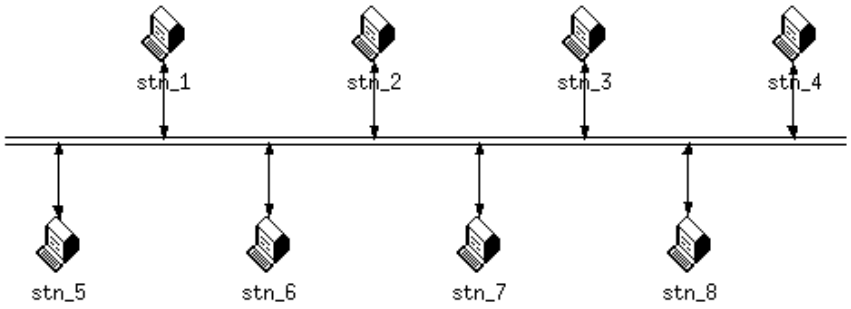
A node with full-duplex capability can both transmit and 'listen' on the line to determine whether a collision condition exists. This operational mode is commonly referred to as Carrier-Sense Multiple Access with Collision Detection (or CSMA/CD). This is practiced by the commercial protocol Ethernet, and is accurately modeled by an OPNET-supplied example model.

Since Ethernet is a fairly sophisticated model, you will not be building it from scratch. Instead, the next several pages provides you with a guided tour of the standard Ethernet process, node, and network models.

The ethcoax_net Network Model

The **ethcoax_net** network model consists of a multi-tap bus network populated by eight nodes. The nodes employ the node model **ethcoax_station**.

The ethcoax_net Network Model

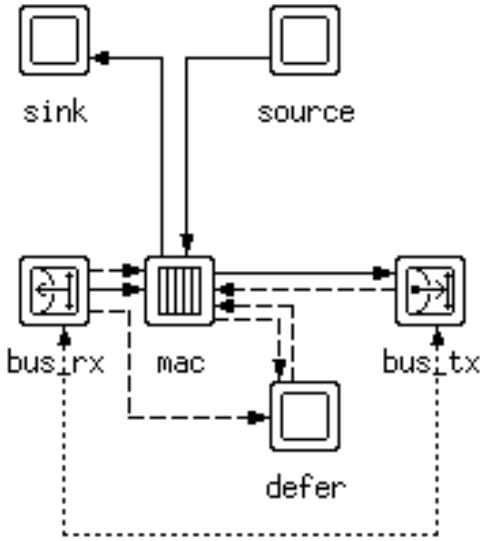


The ethcoax_station Node Model

The **ethcoax_station** node model is significantly more complicated than the Aloha or CSMA node models. It has three processor modules, a queue module which performs the bulk of the channel access processing, and a pair of bus receiver and transmitter modules.

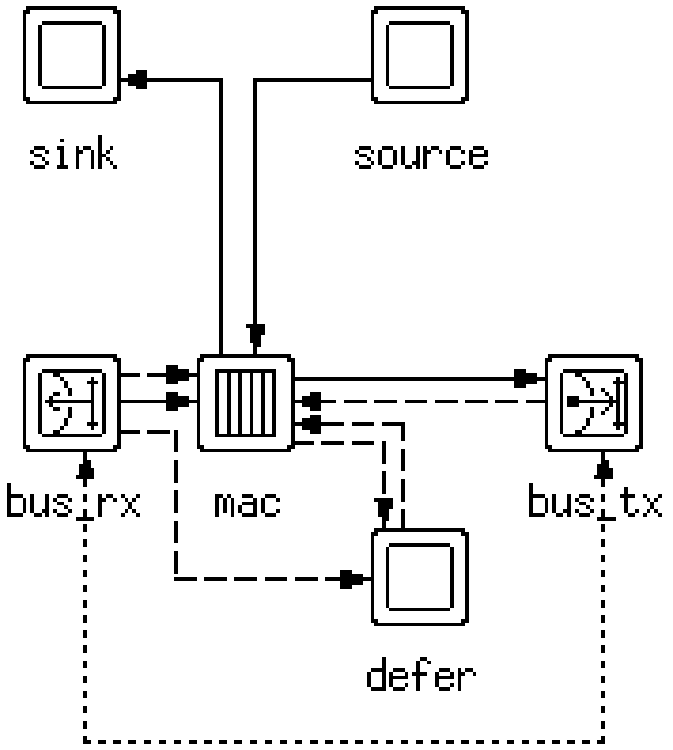
The **ethcoax_station** node model provides part of the functionality associated with the OSI Data Link Layer called the Media Access Control (MAC) sublayer. The functions of the individual modules in the model are discussed on the next page.

The ethcoax_station Node Model



The **bus_tx** and **bus_rx** modules serve as the bus link interface. These modules are set to transmit and receive at a data rate of 10 Mbits/second, the standard data rate used in an Ethernet network.

The **sink** processor represents higher layers and simply accepts incoming packets which have been processed through the **mac** process.



The **defer** processor independently monitors the link's condition and maintains a deference flag which the **mac** process reads over a statistic wire to decide whether transmission is allowed.

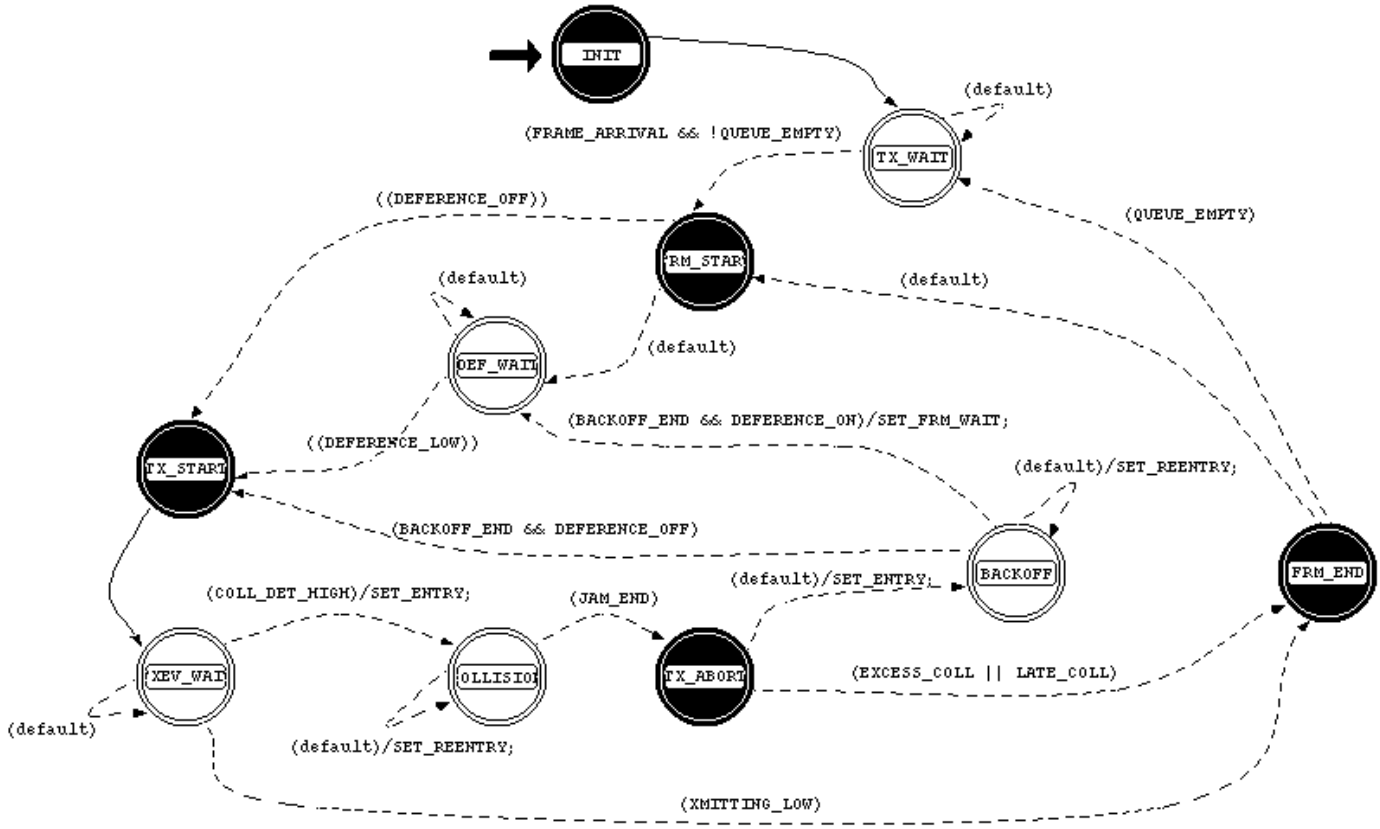
The **source** module represents higher layer users who submit data for transmission.

The **mac** process handles both incoming and outgoing packets. Incoming packets are decapsulated from their Ethernet frames and delivered to the **sink** process. Outgoing packets are encapsulated within Ethernet frames and when the deference flag goes low, a frame is sent to the transmitter. This process also monitors for collisions, and if one occurs, the transmission is appropriately terminated and rescheduled for a later attempt.

The eth_mac_v2 Process Model

The **eth_mac_v2** process model manages the transmission and reception of packets. These tasks have been decomposed into three basic functions: encapsulating and queuing outgoing packets, decapsulating and delivering incoming packets, and managing an ongoing transmission.

The eth_mac_v2 Process Model



The eth_gen_v2 Process Model

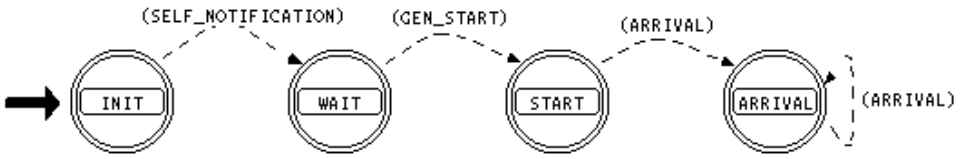
The **eth_gen_v2** process generates packets representing the application data.

The **init** state waits for a begin simulation interrupt and schedules a self interrupt for time 0. By waiting for a self interrupt, **init** allows other process models to receive and process their begin simulation interrupts.

When the first self interrupt is delivered, the **wait** state establishes the valid address ranges and specific distributions for address generation and packet interarrival times. After the first interrupt occurs, **wait** writes some global statistics and schedules a self interrupt.

When the interrupt occurs, the **ARRIVAL** enter executives are executed. A legitimate destination address is associated with the packet, the packet is sent to the **mac** processor, and the next packet generation self interrupt is scheduled.

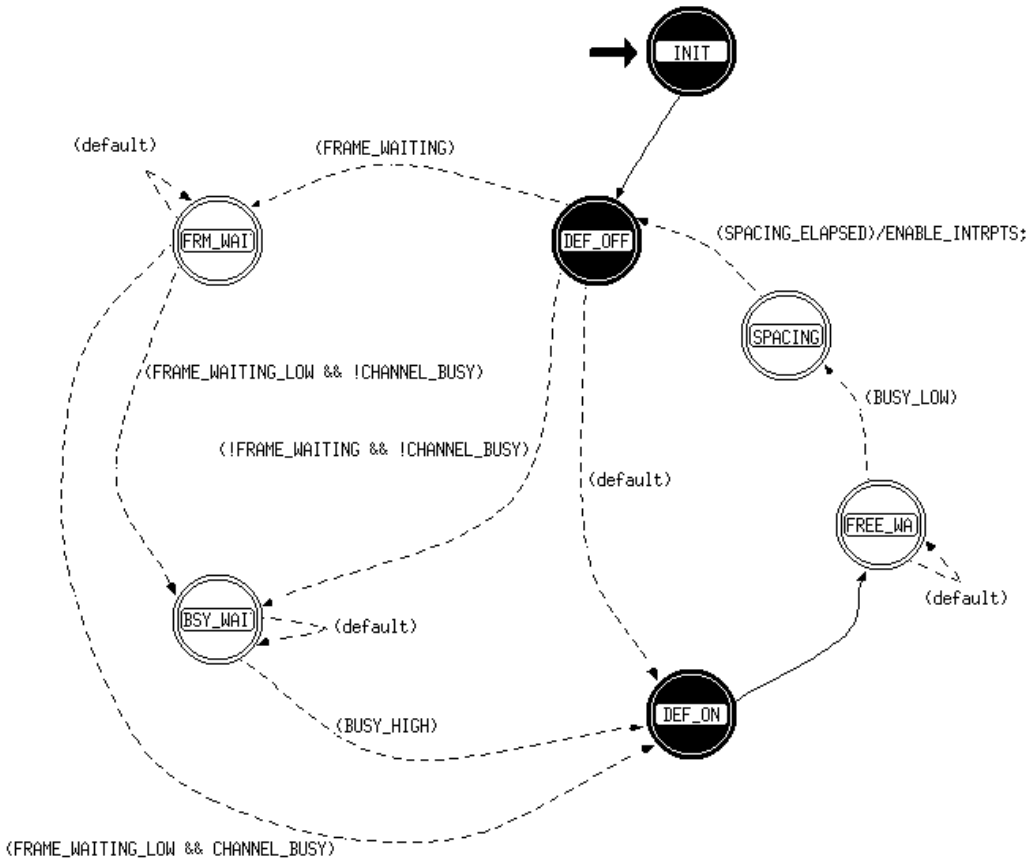
The eth_gen_v2 Process Model



The eth_defer_v2 Process Model

The **eth_defer_v2** process determines whether the deference flag should be raised or lowered. The deference flag is read by the **eth_mac_v2** process to decide whether a transmission is permissible or whether the channel must be deferred to another user.

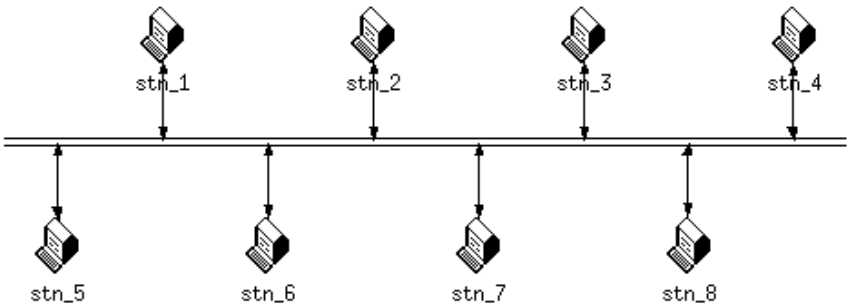
The eth_defer_v2 Process Model



Executing the ethcoax_net simulation

1) **Open** the **ethcoax_net** Project.

   ➡ The ethcoax_net model opens in the workspace.



2) From the **Simulation** menu, choose **Configure Simulation (Advanced)**.

3) In the Simulation Sequence editor, click on the **Execute Simulation Sequence** action button.

   ➡ The simulation takes about 8 minutes to complete, depending on the speed of your machine.

4) When the simulation is complete, click **Close**.

5) Close the Simulation Sequence editor.

7. Analyzing the Ethernet Results

Because the default Ethernet model collects results differently than the Aloha and CSMA simulations, you need to use a slightly different approach to view the results from this simulation.

1) In the Project editor, select **View Results** from the **Results** menu.

2) Select **Object Statistics: ethcoax_net: bus_0[0]: utilization**.

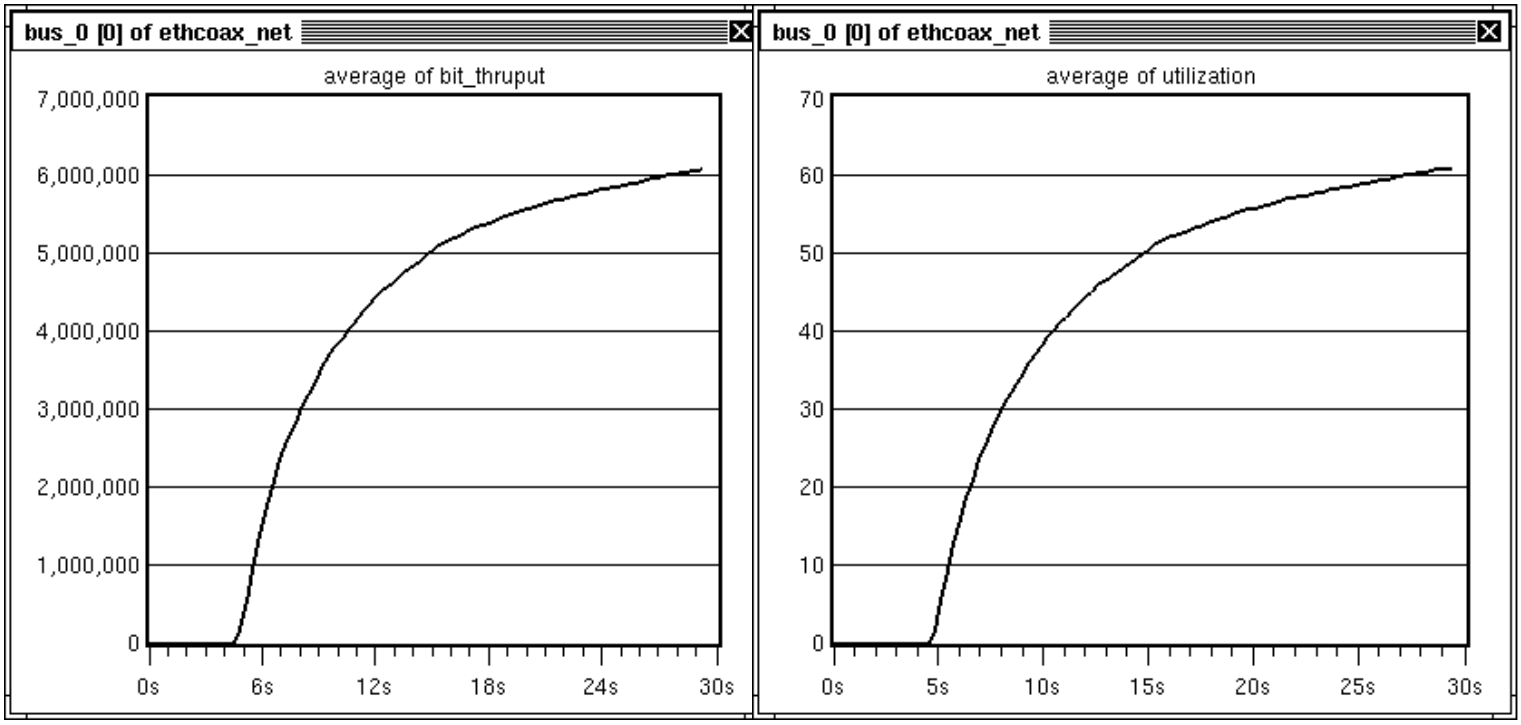
3) Change the filter type from **As Is** to **Average**, then click **Show**.

4) Click the **Unselect** button in the **View Results** dialog box, then select **Object Statistics: ethcoax_net: bus_0[0]: bit_thruput**.

5) Click on **Show**.

6) Place the graph in a workspace location such that both graphs are in full view and consider the results (graphs are shown on the next page).

The graphs from ethcoax_net should resemble those shown below:



Though the general trend of both graphs is the same, note that the graphs have radically different ordinate bounds.

The **bit_thruput** statistic measures the average number of bits successfully received by the receiver per unit time. By definition, this statistic only counts the bits associated with collision-free packets and can reach a maximum value of no more than the 10 Mbits/second data rate assigned to the channel. As you can see, the throughput after 30 seconds of simulation stabilizes near 6.1 Mbits/second. To get a more accurate reading of the actual throughput, you can view the vector graph as a set of data points.

1) Verify that the **bit_thruput** panel is the active window.

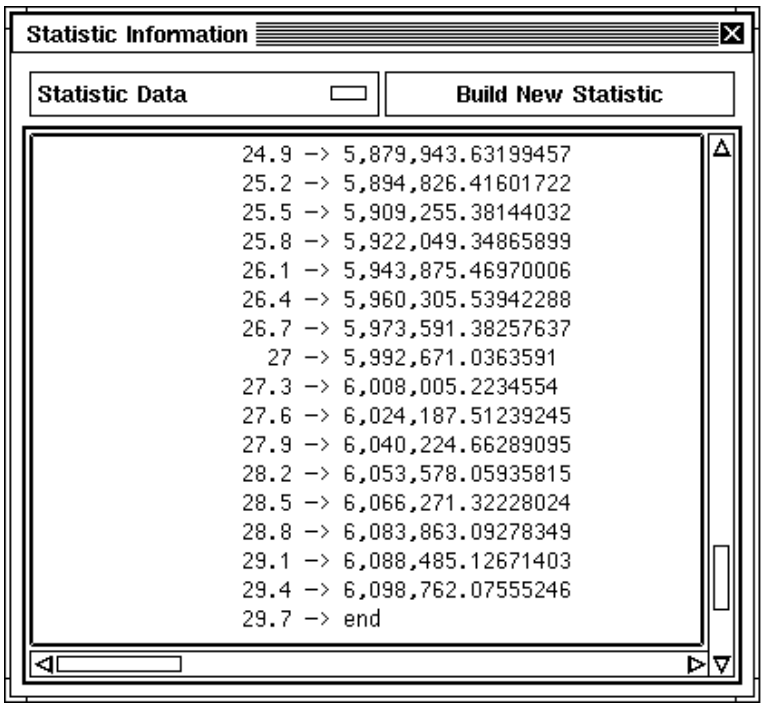
2) Right-click on the panel border and select **Show Statistic Data**.

   ➡ A window opens, showing the sequence of ordinate and abscissa pairs for the vector, in ASCII format.

3) From the pull-down menu at the top of the **Statistic Information** dialog box, select **Statistic Data**.

4) Scroll to the bottom of the editing pad to read the final value of the vector (as shown on the next page).

The ASCII data should resemble that shown below:

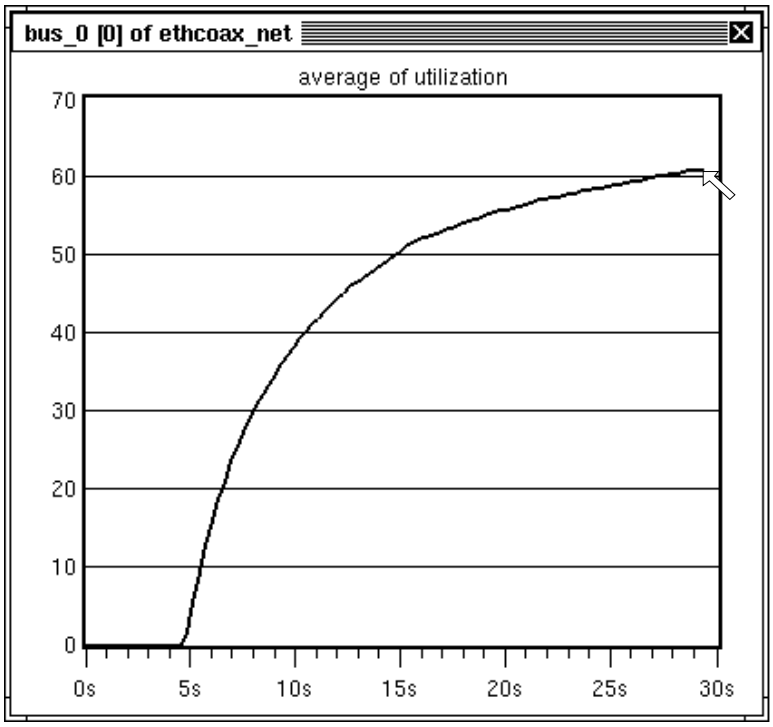


At the end of the simulation, the receiver's **bit_thruput** statistic is indeed almost exactly 6.1x10^6 bits/second. When divided by the channel capacity, this raw level of bit throughput results in a normalized channel throughput of 0.61 (i.e., channel utilization of 61 percent).

When you are finished viewing the data, close the edit pad by clicking on the **X** in the upper-right hand corner of the dialog box.

1) Click on the **average of utilization** graph to activate that window.

2) Move the cursor to the far right of the vector and let the cursor come to rest. The tool-tip will read the final value of the channel utilization.



Here, an average channel utilization near 60 should be seen. This value is the percentage of the 10 Mbits/second channel that the probed transmitter uses.

This indicates that even at the relatively high channel traffic of 61 percent, the Ethernet protocol is able to carry essentially all of the submitted load. This also demonstrates the superiority of the carrier sensing, collision detection, and backoff strategies.

Ethernet uses, over the less sophisticated methods that the pure Aloha and CSMA protocols use.

When you are finished experimenting, close all active Editors.

Congratulations! You have completed all of the basic tutorial lessons. By now, you should be able to build your own network model, collect statistics, run a simulation, and analyze the results on your own. You may wish to continue by looking at the lessons in the Modeler XE tutorial, which cover other aspects of Modeler's capabilities such as topology and traffic import. If you have options such as the radio version or Xpress Developer, look at the corresponding tutorials.

From time to time, you may have questions regarding certain aspects of Modeler. We strongly urge you to consult the online documentation provided (under the Help menu), and you can also contact MIL 3's Technical Support either via phone (202.364.8889), e-mail (opnet@mil3.com), or our web page (www.mil3.com).

Good luck model building!