Modeling a multi-queue multi-server system:

It was a fairly straightforward process to extend a single-queue, single-server model to a single-queue, multi-server model. You either add servers in the form of Activity Delay blocks (Discrete Event.lix) or you replace the Activity Delay block that models the single server with an Activity, Multiple block (Discrete Event.lix) for modeling a "pool" of servers. The single-queue, multi-server model with 2 separately modeled servers looked like:

The variant using a "server pool" looked like:
The single-queue, single-server model under **Extend** naturally fits the context of a "timed" simulation. To reconfigure the simulation to stop on a criteria other than time, a **Select DE Output** block (**Discrete Event.lix**) was used to divert those **items** generated while the final customers were being served out of the model. This tactic (initiated by a "door locked" criteria") effectively shut down the generator so that remaining **items** in the model could be drained out.

The **Select DE Output** block (**Discrete Event.lix**) is a natural mechanism for routing an input **item** to the shortest queue. When its **select** input is connected, the **Select DE Output** block routes **items** to its **a** output connector or to its **b** output connector depending on the input value and how the block has been configured in its dialogue. Normally, it is set up to route to the **a** connector on input 0 and to the **b** connector on input 1, but the routing criteria can be shifted to **x** and **x+1**; e.g., route to the **a** connector on input 3 and route to the **b** connector on input 4.

For a multi-queue model, when a new **item** is generated, the **select** value simply is determined in accord with a model criteria specifying which queue the **item** enters. One obvious criteria is that the item enters the shortest queue (or the first among the shortest, if there are several of this length). This means that the queue lengths must be monitored and used in establishing the queue to be selected.

In **Extend**, it is easy to acquire the length, since it is one of the outputs (L) of the **Queue FIFO** Block (**Discrete Event.lix**). A **Max & Min** block (**Generic.lix**) configured to receive the line lengths of two or more queues as inputs, produces the "number" of the connector receiving the minimum input value at the **Con** output associated with the **Min** output. This value identifies which queue has the shortest line.
Now if the Select DE Output block (*Discrete Event.lix*) is configured to route to \textit{a} when \textit{select} is 1 (and so route to \textit{b} when \textit{select} is 2), then a generated \textit{item} is directed to the queue of minimum length by connecting the \textit{select} input to "min line" (from the Max & Min block (*Generic.lix*)).

If there are more than 2 queues, then the Select DE Output blocks simply need to be set up in sequence with the select criteria following "1 or 2" being "3 or 4" then "5 or 6", and so forth. For example, for 4 queues:

If the minimum length is at connector 3 or 4; i.e., \textit{min line} is 3 or 4, then the input item is rejected at the first Select DE Output block and sent on to the second one, which routes the item to the queue with the shortest line.

The upper Select DE Output block (*Discrete Event.lix*) has been configured to route to output connector \textit{a} on \textit{select} connector value 1 and to output connector \textit{b} on \textit{select} connector value 2. Likewise, the lower Select DE Output block has been configured to route to output connector \textit{a} on \textit{select} connector value 3 and to output connector \textit{b} on \textit{select} connector value 4.
The model is finalized by simply connecting each Queue FIFO block (*Discrete Event.lix*) to an Activity Delay block (*Discrete Event.lix*).

This setup provides for 4 servers (each modeled by and Activity Delay block), each having its own queue (modeled by a Queue FIFO block, one for each server).

Specific values can be monitored (for debugging or display purposes) by using a ReadOut block (*Generic.lix*) or a Status block (*Discrete Event.lix*). Care must be taken in reading values from the Input Random block (*Generic.lix*) since the ReadOut block must pull a random value to read it, causing an extra random value to be consumed. In this case it is better to use a Timer block (*Discrete Event.lix*). Its typical use is to time how long an item is held in an Activity Delay block (*Discrete Event.lix*) where either the discrete service times can be output or the running mean service time can be output.

The Status block provides several output values when used for viewing items but when sensing a value is limited to the display of the content of a value. The ReadOut block can read and pass on the value (except note the exception with Input Random blocks), making it useful for debugging.
The **Status** block when used to sense items can optionally absorb them (set in the block’s dialogue). Among other pieces of information, the Status block provides the *interarrival time* between the items it views. It is less useful for “on the spot” debugging, but is extremely useful for the display of “model reality check” information.

The **Max & Min** block (*Generic.lix*) has 5 input connectors. If more are needed, the block can be cascaded (automatically via the dialogue, and otherwise manually). For example, for 10 inputs, the configuration is

The value of **min line** will be the connector with the minimum length (numbered 1 through 10).