Chapter 2
Building a Model with Elementary Components

For ease in building and maintaining models, the QSIM Application promotes hierarchical model building, user component construction, and component reuse by providing a comprehensive set of primitive components and the ability to assemble components into compound components. For example, you can build a queue-server network, then encapsulate it, identify an image to represent it, use it to define a template for additional replication, and save the template in a SAS data set to be shared in other models and by other users. This chapter discusses the details of the elementary components used in model building.

There are several types of elementary components: sources, servers, queues, logic, holders, charts, and connectors. Although each of these has a special role, they have much in common, as is evident from their pop-up menus (displayed by pressing the right mouse button while pointing to the component).

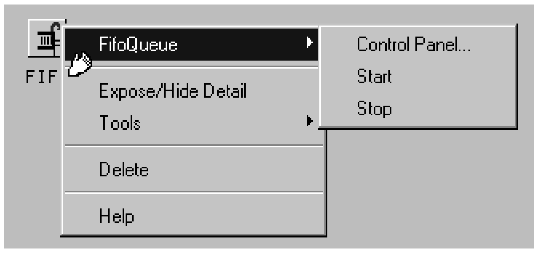


Figure 2.1. The Pop-Up Menu on the FIFO Queue Component

Figure 2.1 shows a typical pop-up menu. The first entry in this pop-up menu is the name of the component. The submenu from that entry has **Control Panel...** as the first choice. Selecting it displays the component's control panel, which enables you to set component parameters. Other choices include **Expose/Hide Details**, **Tools**, **Delete**, and **Help**. The **Expose/Hide Details** entry toggles the display of the component between an icon that represents the component, which you can change, and a drawn representation of the component. For some of the components, the drawn representation shows state information while the simulation is in progress and animating. For example, the family of queues slowly fills as transactions arrive and queue up for service.

Many of the components also have internal state information that changes as the result of transaction arrival and other kinds of message sending. There are five general types of actions that either change component state or return information about the component state: *transaction arrival, request for transaction, are you busy message, query message*, and *trigger message*. The *query message* is sent from the Modifier components and formulas. These messages make requests about the state of the component, for example, the number of transactions waiting in a queue. The *trigger message* is sent when a transaction arrives at a Trigger component, and it is used to change the state of the component. For example, since the Sampler services the "start" message, a transaction arriving at a Trigger component can start a Sampler. The sections that follow document the elementary components and show in tables the types of state and information messages that the components service.

Each elementary component has a Control Panel associated with it. This panel provides you access to parameters that control the behavior and appearance of the component.

The system assigns a unique component id to each of the elementary components. This is done so that you can unambiguously identify each component that appears in a list box. For example, when you instantiate a Server all that appears in the Simulation window is the icon . If you have several of these icons and you look at a Trigger control panel, such as the one shown in Figure 2.14, you cannot distinguish them unless you give each a unique label. By default, each will have a unique id, which is appended to the name of the object and displayed in the list box. As an alternative, you can give the component a label, which will be displayed in the list box. By default, the iconic representation of a component in the Simulation window, includes the unique id.

Source Components

The source components are sources of transactions for the simulation network. There are two types of sources: Sampler and Transaction Pool.

Icon	Component	Description
	Sampler	generates transactions with prescribed inter-arrival times
	Transaction Pool	a source of transactions

The Sampler generates a transaction, and then waits a specified time interval before generating another transaction. The time between transactions, called the inter-arrival time, can be a sample of a random variable (from one of several distributions), a fixed amount, or the value of a variable read from a SAS data set. By default, the inter-arrival time is an observation of an exponential random variable with parameter 1. This means that by default the Samplers follow a Poisson arrival process.

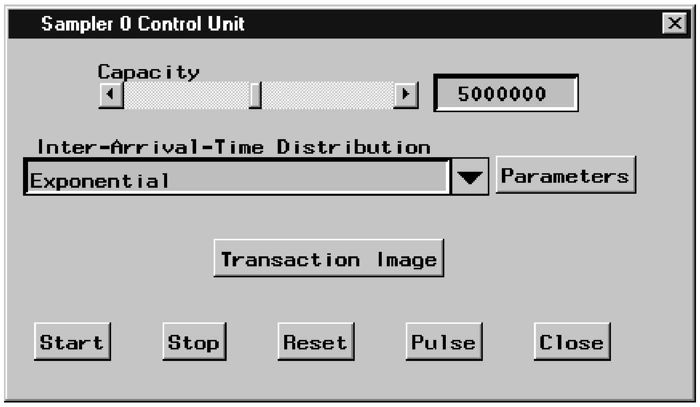


Figure 2.2. The Sampler Control Panel

Figure 2.2 shows the Control Panel for the Sampler. The combobox labeled “Inter-Arrival-Time Distribution” enables you to specify the type of distribution. If you press the down arrow a list of distributions is displayed. You select the inter-arrival time distribution by selecting one of these. See Chapter 6, “Random and Exogenous Variation in the Model,” for information on the choices of distributions.

The control panel also has a slider for setting the capacity of the Sampler, that is, the number of transactions that will be generated before the Sampler shuts off. The “Transaction Image” button on the control panel enables you to choose a bitmap image that would flow through the network when the simulation is being animated.

The Transaction Pool (the other source component) differs from the Sampler only in that it does not create transactions unless it receives a *request for transaction* message. In other ways, it is identical to the Sampler.

The following documents the logic of the source components:

Request for Transaction

A Sampler passes requests to arcs leading into it. A Transaction Pool generates a transaction and initiates its flow.

Are You Busy Message

The source components pass this request to arcs leading out of them and return their answers.

Query Message

Keyword	**Meaning**
capacity	returns the capacity.
id	the source component’s unique identifier.
on	TRUE if the source component is started, else FALSE.
remaining	returns the number of transactions remaining to be sent.
size	returns the number of transactions sent.

Trigger Message

Keyword	**Meaning**
reset	stops the source component and resets the number of transactions remaining to be sent to 1.
setCapacity	sets the capacity from the transaction attribute “capacity.”
setDistribution	sets the distribution from the transaction attribute “distribution.” This attribute should be a character string whose value is one of the distributions: Exponential, Gamma, Erlang, Uniform, IUniform, and Deterministic.
setParameter1	sets the first parameter in the distribution from the transaction attribute “parameter1.”
setParameter2	sets the second parameter in the distribution from the transaction attribute “parameter2.”
start	starts the source component.
stop	stops the source component.

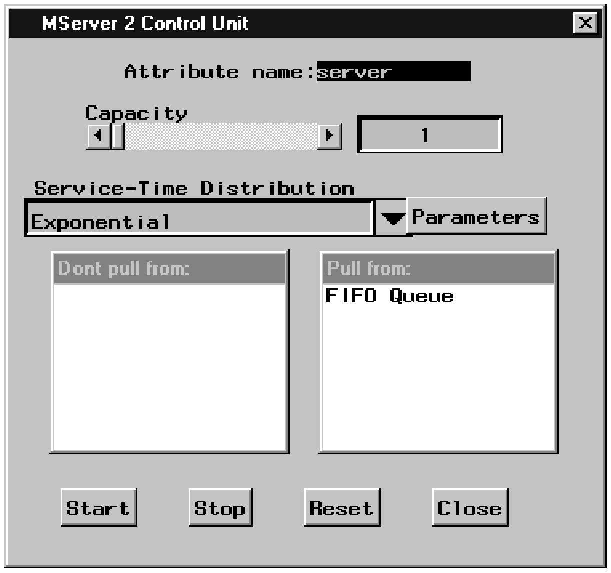
Server Components

Server components model a resource used by a transaction for a specified amount of time. There are two types of servers: Server and MServer.

Icon	Component	Description
	Server	provides service for a transaction
	MServer	provides service simultaneously for multiple transactions

The Server holds the transaction while it is served. The service time can be a sample of a random variable (from one of several distributions), a fixed amount, or the value of a variable read from a SAS data set. By default, the service time is an observation of an exponential random variable with parameter 1.

An MServer, or multiple-server, can service multiple transactions simultaneously. The capacity of an MServer is set using the slider labeled “Capacity” on its control panel.



MServer 2 Control Unit

Attribute name: server

Capacity: 1

Service-Time Distribution: Exponential

Dont pull from:

Pull from: FIFO Queue

Figure 2.3. The Multiple-Server Control Panel

Notice that in the lower-right corner of the server components, there is a small rectangle. This is a Balk node. If a transaction arrives at a Server when it is busy or at an MServer when it is at capacity, the transaction will flow out the Balk node. Consider, for example, a situation where transactions are either serviced upon arrival by server 1 or, if server 1 is not free, wait for service from server 2. This is modeled by the network in Figure 2.4.

When the transaction leaves the Server or MServer, it has an attribute as named in the control panel that contains the time that the transaction spent in the server. This attribute can be used for controlling the simulation logic and for measuring the performance of the simulation by displaying it in one of the chart components or saving it in a SAS data set.

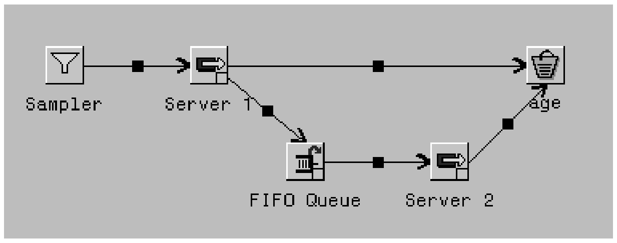


Figure 2.4. Server Balk Model

The following documents the logic of the server components:

Transaction Arrival

If the server is busy, at capacity, or stopped, the transaction flows out the Balk node; otherwise, service is scheduled. On service completion, a *request for transaction* message is sent to arcs directed into the server. If a transaction is found, then its flow is initiated. Regardless, the transaction that just finished service flows on each of the arcs directed out of the server.

Request for Transaction

If the server is not busy or stopped, then pass on the request to all arcs directed into the server. The order in which the requests for service are issued is determined by the order of the components in the "Pull from" list box on the Sever Control Panel. Also, if a component is not included in the "Pull from" list box, then the request for transaction message is not propagated on the arc leading to that component.

Are You Busy Message

If the server is not busy and not stopped, then return FALSE; otherwise, return TRUE.

Query Message

Keyword	Meaning
busy	returns TRUE if the Server is busy or the MServer is at capacity or either is stopped; else, returns FALSE.
capacityIs	returns the capacity of the MServer
full	returns TRUE if the Server is busy or the MServer is at capacity; else, returns FALSE.
id	returns the server's unique identifier.
off	returns TRUE if the Server or MServer is stopped.
sizeIs	returns the number of multiple-server units that are busy.
space	returns TRUE if the Server is free or the MServer is not at capacity; else, return FALSE.

Trigger Message

Keyword	Meaning
preempt	removes all the transactions that are being served. They flow out of the Balk node.
preemptContinue	removes all the transactions that are being served. They flow out of the Balk node. The server requests transactions from upstream components.
removeIt	removes the transaction at the Trigger, if it is also being served. It flows out of the Balk node.
reset	resets the Sever and MServer destroying all waiting transactions.
seize	attempts to obtain service for the transaction that arrived at the Trigger.
setCapacity	sets the MSever capacity from the transaction attribute "capacity."
setDistribution	sets the distribution from the transaction attribute "distribution."
setParameter1	sets the first parameter in the distribution from the transaction attribute "parameter1."
setParameter2	sets the second parameter in the distribution from the transaction attribute "parameter2."
start	starts the Server component.
stop	stops the Server component. Transactions in service have normal completion.

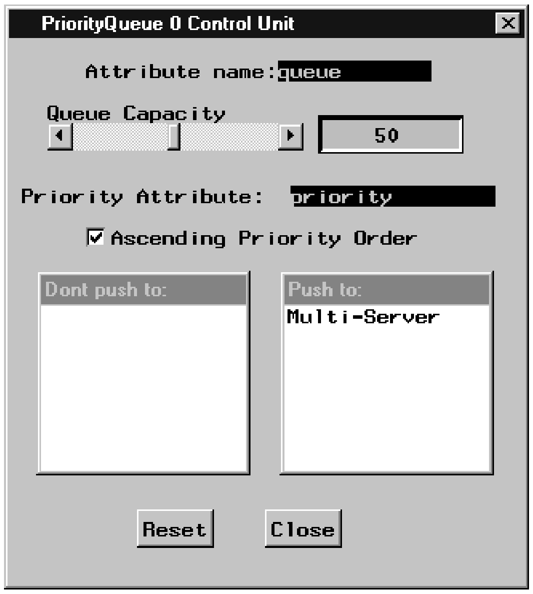
Queue Components

Queue components are transient storage for transactions. There are three types of queues: FIFO Queues, LIFO Queues, and Priority Queues.

Icon	Component	Description
	FIFO Queues	first-in-first-out
	LIFO Queues	last-in-first-out
	Priority Queues	priority

Each type of queue can behave as a buffer. This means that when the transaction first arrives, the queue will not try to route it to a nonbusy component but will wait for a *request for transaction* message from a downstream component before sending it on. In addition, you can have the queue behave as a buffer for some downstream components and as a standard queue for others. Those components in the Don't push to listbox in the Queue Control Panel (see Figure 2.5) define components for which the queue acts as a buffer. Those components in the Push to listbox define components for which the queue acts as a standard queue.

The LIFO and FIFO queues order transactions according to their arrival time. The Priority Queue uses the value of the numeric transaction attribute named “priority” to determine placement location in the queue. This default name can be changed. The priority attribute can be assigned to a transaction by the Modifier component, discussed in the section “Logic Components” on page 19. By default, the *smaller* the value of the attribute, the *higher* placement in the queue and the sooner the element will leave the queue. Although this is the default priority order, it can be changed by unselecting the “Ascending Priority Order” check box on the control panel shown in Figure 2.5.


PriorityQueue 0 Control Unit

Attribute name: queue

Queue Capacity 50

Priority Attribute: priority

☑ Ascending Priority Order

Dont push to:

Push to:
Multi-Server

Figure 2.5. The Priority Queue Control Panel

When each transaction leaves the queue, it has an attribute with the time it spent in the queue. The name of this attribute can be specified in the queue control panel. See Figure 2.5 for where to give the attribute name. By default the attribute name for all queues is “queue.”

The following documents the logic of the Queue components.

Transaction Arrival

If the queue is off or at capacity, the transaction flows out the Balk node; otherwise, it sends the message *are you busy* to the nodes on arcs directed away from the queue and listed in the Push to list box. If FALSE is returned, then route the transaction there; otherwise, queue the transaction.

Request for Transaction

If the queue is not empty (size > 0), then remove the next transaction according to the type of queue and send it out the arc directed to the component that made the request; otherwise, return FALSE.

Are You Busy Message

always returns FALSE.

Query Message

Keyword	Meaning
capacity	returns the queue’s capacity
id	returns the queue’s unique identifier.
releaseType	returns a string naming the way that the last transaction was released from the balk node. Possible values are: “balk,” “empty,” “filter,” “filterOne,” and “releaseOne.”
size	returns the number of transactions that are in the queue.
space	returns TRUE if there is unused capacity in the queue.

Trigger Message

Keyword	Meaning
balk	causes the transaction at the Trigger to leave the queue from the Balk node.
empty	empties the queue of all transactions. Note that the transactions do not leave via the Balk node.
filter	evaluates a formula for each transaction in the queue. If the formula evaluates to TRUE, the transaction balks; otherwise, it maintains its place in the queue. The formula that is evaluated should be in an attribute named ”formula” in the triggering transaction.
filterOne	evaluates a formula for each transaction in the queue. The first transaction for which the formula evaluates to TRUE balks. The formula that is evaluated should be in an attribute named “formula” in the triggering transaction.
insert	inserts the transaction at the Trigger into the queue.
releaseOne	releases one transaction from the queue via the the Balk node.
reset	destroys all transactions in the queue.
start	starts the queue.
stop	stops the queue.

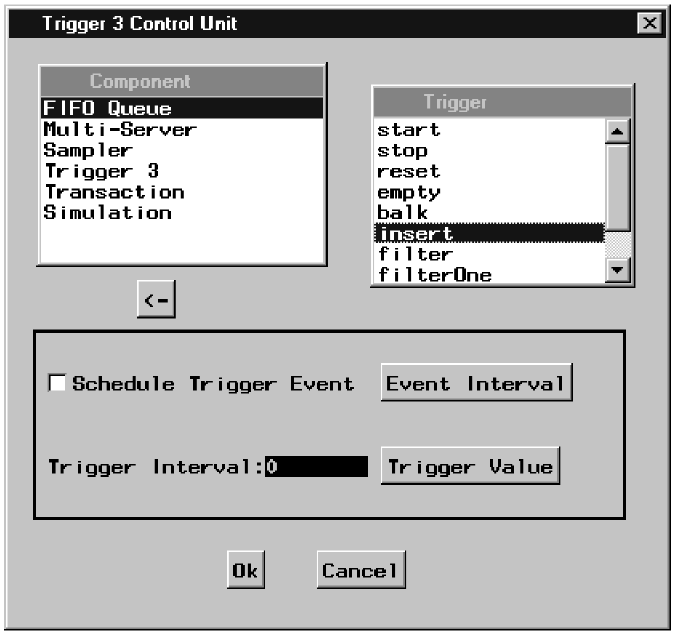
Logic Components

The logic components fall into two categories: those that control the flow of transactions Adder, Splitter, Router, and Switch; and those that change the state of the simulation Modifier and Trigger. Descriptions follow.

Icon	Component	Description
	Adder	assemble multiple transactions
	Splitter	split single transactions
	Modifier	assign an attribute to transactions
	MModifier	assign multiple attributes to transactions
	Trigger	change a component’s state
	MTrigger	change multiple components’ state
	Router	direct flow as a function of system state
	Switch	direct flow as a function of system state

Trigger Component

When a transaction arrives at a Trigger component, it initiates a message being sent to another component. For example, Figure 2.6 shows the control panel for a Trigger component. Notice that the `FIFO Queue` component is selected and that the `insert` trigger is also selected.



Trigger 3 Control Unit

Component: FIFO Queue (selected), Multi-Server, Sampler, Trigger 3, Transaction, Simulation

Trigger: start, stop, reset, empty, balk, insert (selected), filter, filterOne

☐ Schedule Trigger Event

Trigger Interval: 0

Figure 2.6. The Trigger Control Panel

When a transaction arrives at this Trigger, the “insert” message is sent to the specific queue selected in the control panel. As documented in the section “Queue Components” on page 17, the transaction that arrives at the Trigger is the one inserted into the queue named “FIFO Queue.”

Notice the check box labeled `Schedule Trigger Event` in the trigger control panel. You select this check box to delay execution of the trigger event. You can specify the length of the delay by pressing the Event Interval button. This opens a Distribution window (like the one shown in Figure 4.3) from which you can choose a distribution, a fixed interval, or a numeric variable in a SAS data set.

The `Trigger Interval` field provides a mechanism for disabling the trigger for some transactions. If the trigger interval is one, then every other transaction will activate the trigger logic. If the trigger interval is two, then every third transaction will activate the trigger logic, and so on.

The Trigger Value push button provides a mechanism for associating a value with the trigger. This is used with the “setFromTrigger” trigger message in Holders.

The following documents the logic of the Trigger component:

Transaction Arrival

executes the trigger; then the transaction flows down each arc directed away from the component.

Request for Transaction

the request is sent up each arc directed into the component.

Are You Busy Message

if any of the components on arcs directed out of the Trigger is busy, then return TRUE; else, return FALSE.

Query Message

Keyword	Meaning
id	returns the component’s unique identifier.
value	returns the value associated with the Trigger.

Trigger Message

Keyword	Meaning
reset	resets the Trigger.
start	starts the Trigger.
stop	stops the Trigger.

MultiTrigger Component

When a transaction arrives at a MTrigger component, it initiates the sending of a set of messages to a set of components, one message to each component. For example, Figure 2.7 shows the control panel for a MTrigger component.

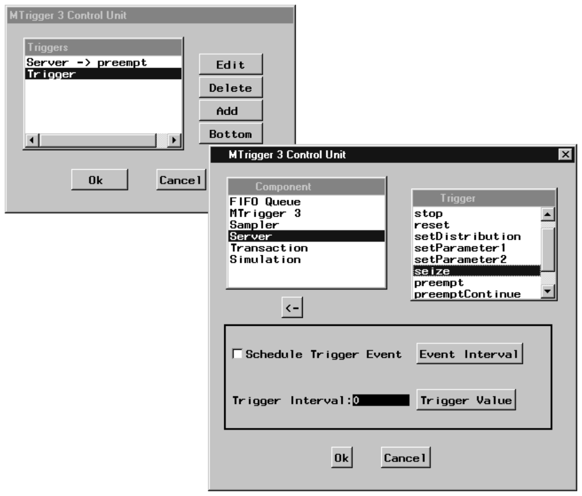


Figure 2.7. The MTrigger Control Panel

Notice that the list box labeled Triggers contains two entries, one labeled `Server -> preempt` and the other labeled `Trigger`. The first one indicates that an arriving transaction will cause the “preempt” message to be sent to Server. The second one, labeled `Trigger`, has not been specified but is selected. Selecting this and pressing the **Edit** button raises the second window which looks like the Trigger Control Panel. Notice that the Server named `Server` is selected in that window. Also notice that `seize` has been selected. This means that the second message triggered by an arriving transaction will send the “seize” message to the component labeled Server.

The following documents the logic of the MTrigger component:

Transaction Arrival

sends each message to the appropriate component, then the transaction flows down each arc directed away from the component.

Request for Transaction

the request is sent up each arc directed into the component.

Are You Busy Message

if any of the components on arcs directed out of the MultiTrigger is busy, then return TRUE; else, return FALSE.

Query Message

Keyword	Meaning
id	returns the component’s unique identifier.

Trigger Message

Keyword	Meaning
reset	resets the Multi-Trigger.
start	starts the Multi-Trigger.
stop	stops the Multi-Trigger.

Modifier Component

The Modifier component assigns an attribute to a transaction. The control panel, shown in Figure 2.8, provides a field for entering the attribute name and a set of radio buttons for specifying how a value for that attribute is calculated. It can be the result of a simple character or numeric assignment, a formula evaluation, sampling of a random variable, or the value read from a variable in a data set. Regardless, the result of evaluation is the value given to the attribute. This attribute-value combination is unique to the transaction, and the transaction carries it on its route through the simulation network.

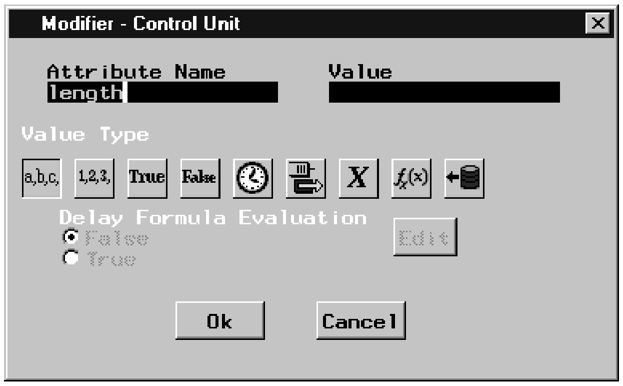


Figure 2.8. The Modifier Control Panel

By default, the value is calculated when the transaction arrives at the Modifier. However, if you select the `Delay Formula Evaluation` check box, the formula is not evaluated when the transaction arrives at the modifier but is itself the value of the attribute. This feature is used with the "filter" trigger message on queues.

Details regarding formulas are discussed in the "Formulas" section.

The following documents the logic of the Modifier component:

Transaction Arrival
assigns the attribute value pair; then the transaction flows down each arc directed away from the component.

Request for Transaction
the request is sent up each arc directed into the component.

Are You Busy Message
if any of the components on arcs directed out of the Modifier is busy, then return TRUE; else, return FALSE.

Query Message

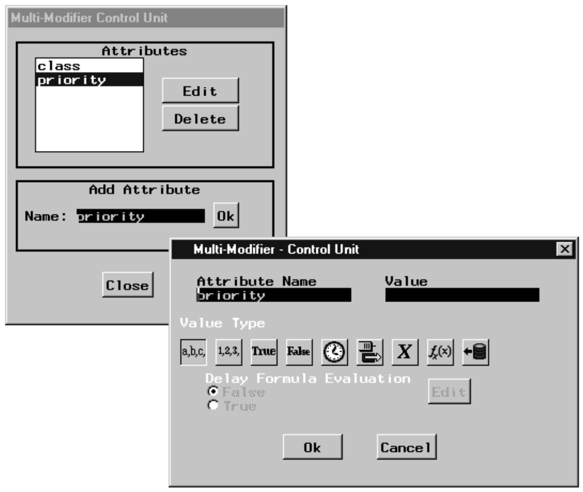
Keyword	Meaning
id	returns the component's unique identifier.

Trigger Message

Keyword	Meaning
reset	resets the Modifier.
start	starts the Modifier.
stop	stops the Modifier.

MultiModifier Component

The MModifier component assigns multiple attributes to a transaction. The control panel, shown in Figure 2.9, provides a field for entering the attribute name and an **OK** push button for adding the attribute to the attributes list.



Name: priority

Attribute Name: priority

Value:

Figure 2.9. The MModifier Control Panel

The list shows two attributes, named “class” and “priority.” By default when a transaction arrives at the component, each of the attributes is assigned a value as is done in the Modifier component. You specify the details of how each attribute is evaluated by selecting the attribute and pushing the **Edit** button. This raises a control panel like the one for the Modifier as shown above. These attribute-value combinations are unique to the transaction, and the transaction carries them on its route through the simulation network.

The following documents the logic of the MModifier component:

Transaction Arrival

assigns the attribute; then the transaction flows down each arc directed away from the component.

Request for Transaction

the request is sent up each arc directed into the component.

Are You Busy Message

if any of the components on arcs directed out of the Modifier is busy, then return TRUE; else, return FALSE.

Query Message

Keyword	Meaning
id	returns the component's unique identifier.

Trigger Message

Keyword	Meaning
reset	resets the Multi-modifier.
start	starts the Multi-modifier.
stop	stops the Multi-modifier.

Switch Component

The Router and Switch components are for controlling the flow of transactions as a function of the state of the simulation. The Router can have a formula associated with each arc directed away from it. When a transaction arrives at the Router, each formula is evaluated and the transaction flows down all arcs with formulas that evaluate to TRUE. The Switch is similar to the Router, but it has only one formula associated with it. The formula evaluation is interpreted as a case, which identifies an arc or set of arcs down which the transaction should flow. If the evaluation does not identify a valid case, the transaction flows out the Balk node.

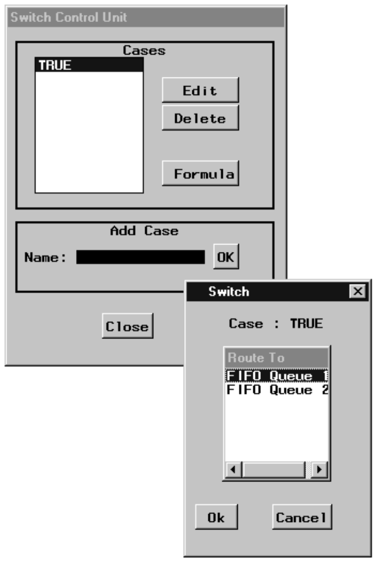


Figure 2.10. Switch Control Panel

Figure 2.10 shows the control panel for a switch connected to two queues as in Figure 2.11. Selecting the **Formula** button displays a Formula Manager Window (see Chapter 4, “Formulas,”). There you build, verify, and save the formula associated with the switch. When a transaction arrives at the switch, the formula associated with the switch is evaluated. This value is compared to each of the cases listed in the Switch control panel. The transaction flows down the arcs associated with each of the cases that match. You can associate arcs with a case by selecting a case and pressing the **Edit** button. This displays the Switch Control Panel, as shown in Figure 2.10. In this window you select one or more of the listed components. For example, a Switch can be used to direct transactions to the smaller of two queues.

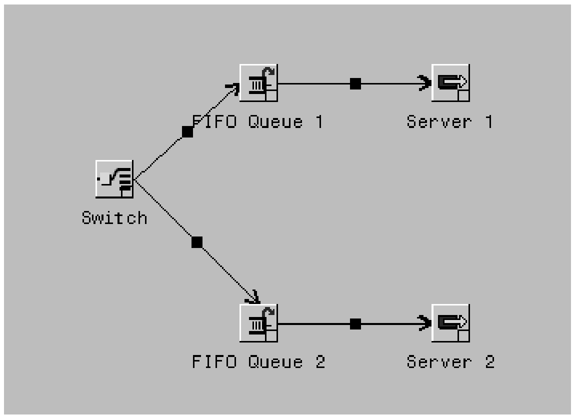


Figure 2.11. Switch Controlled Queue Selection

The following documents the logic of the Switch component:

Transaction Arrival

evaluates the formula for the switch. The transaction flows down the arcs leaving the switch that have case values matching the formula evaluation. If there is no match, the transaction flows out the balk node.

Request for Transaction

the request is sent up each arc directed into the component.

Are You Busy Message

if any of the components on arcs directed out of the Switch is busy, then return TRUE; else, return FALSE.

Query Message

Keyword	Meaning
id	returns the component's unique identifier.

Trigger Message

Keyword	Meaning
reset	resets the Switch.
start	starts the Switch.
stop	stops the Switch.

Router Component

The Router and Switch components are for controlling the flow of transactions as a function of the state of the simulation. The Router can have a formula associated with each arc directed away from it. When a transaction arrives at the Router, each formula is evaluated and the transaction flows down all arcs with formulas that evaluate to TRUE. The Switch is similar to the Router, but it has only one formula associated with it. The formula evaluation is interpreted as a case, which identifies an arc or set of arcs down which the transaction should flow. If the evaluation does not identify a valid case, the transaction flows out the Balk node.

The following documents the logic of the Router component:

Transaction Arrival

evaluates the formula for each arc leaving the router. If an evaluation returns TRUE, then the transaction flows down the associated arc.

Request for Transaction

the request is sent up each arc directed into the component.

Are You Busy Message

if any of the components on arcs directed out of the Router is busy, then return TRUE; else, return FALSE.

Query Message

Keyword	Meaning
id	returns the component's unique identifier.

Trigger Message

Keyword	Meaning
reset	resets the Router.
start	starts the Router.
stop	stops the Router.

Adder Component

The Adder component is useful when you want to model the assembly process such as putting two parts together. The Adder guarantees that this will occur only when both of the parts are available.

The following documents the logic of the Adder component:

Transaction Arrival

if all components on arcs leading into the Adder can initiate flow, then initiate a transaction from each and generate a *new* transaction to travel down each arc directed away from the Adder; otherwise, the transaction flows out the balk node.

Request for Transaction

if all components on arcs leading into the Adder can initiate flow, then initiate a transaction from each and generate a *new* transaction to travel down each arc directed away from the Adder.

Are You Busy Message

if all components on arcs leading into the Adder can initiate flow, then return TRUE; else, return FALSE.

Query Message

Keyword	Meaning
id	returns the component's unique identifier.

Trigger Message

Keyword	Meaning
reset	resets the Adder.
start	starts the Adder.
stop	stops the Adder.

Splitter Component

The transaction entering the Splitter leaves down all arcs away from the splitter. A single instance of the transaction leaves the Splitter multiple times so it is in multiple places at once. It appears that these are multiple copies but they all refer to the same transaction instance. This means that any change to the transaction can be detected in multiple places in the model. This is the same behavior as with a Port or Connector. The additional behavior that is provided by the Splitter comes when an "Are You Busy Message" is sent to it. This behavior differs from both the Port and Connector.

The following list documents the logic of the Splitter component:

Transaction Arrival

the transaction flows down each arc directed away from the component.

Request for Transaction

if none of the components on arcs directed out of the Splitter is busy, then pass on the request to all components on arcs leading into the Splitter; else, deny the request.

Are You Busy Message

if any of the components on arcs directed out of the Splitter is busy, then return TRUE; else, return FALSE.

Query Message

Keyword	Meaning
id	returns the component's unique identifier.

Trigger Message

Keyword	Meaning
reset	resets the Splitter.
start	starts the Splitter.
stop	stops the Splitter.

Holder Components

There are two types of holders: StringHolder and NumberHolder. These are used to hold strings and numbers for maintaining user defined state information.

Icon	Component	Description
■	NumberHolder	storage for a number
■	StringHolder	storage for a string

NumberHolder Components

There are various ways to both save information and retrieve information from holders. The values of attributes carried by transactions can be saved in a holder when the transaction enters the holder. Alternatively, a value can be saved in a holder when a transaction enters a trigger in some other part of the simulation model. For example, suppose that you want to save the value of a transaction attribute called “weight” in a NumberHolder.

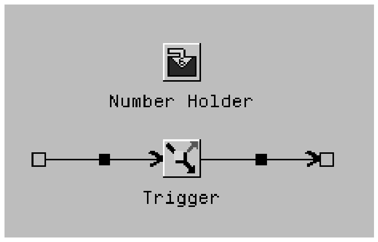


Figure 2.12. Number Holder Saving “weight” Attribute

In the model fragment in Figure 2.12, when the transaction arrives at the Trigger, the value of the weight attribute in that transaction is saved in the NumberHolder. Now, another part of the simulation can query the NumberHolder to find the current value of weight. You could also route the transaction directly to the NumberHolder and update its state that way.

You specify the name of the attribute that is stored in the NumberHolder in the NumberHolder Control Panel, which is displayed by selecting **Control Panel...** selection on the pop-up menu on the NumberHolder.

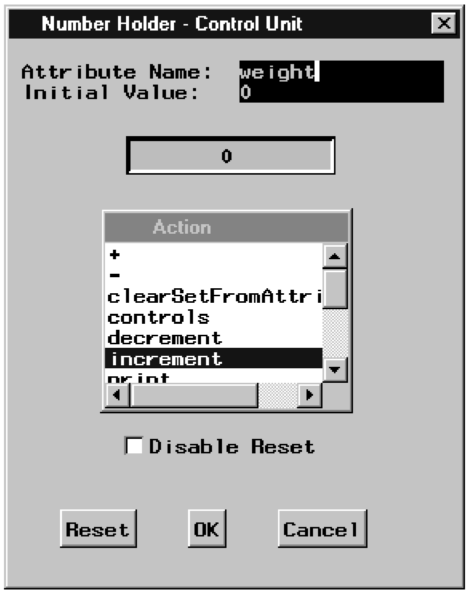


Figure 2.13. Number Holder Control Panel

Notice that the NumberHolder control panel shown in Figure 2.13 has the attribute name "weight" in the **Attribute Name** field.

The transaction sets the NumberHolder when it arrives at the Trigger because the Trigger Control Panel, as shown in Figure 2.14, has the `NumberHolder` component selected and the `setFromAttribute` selected.

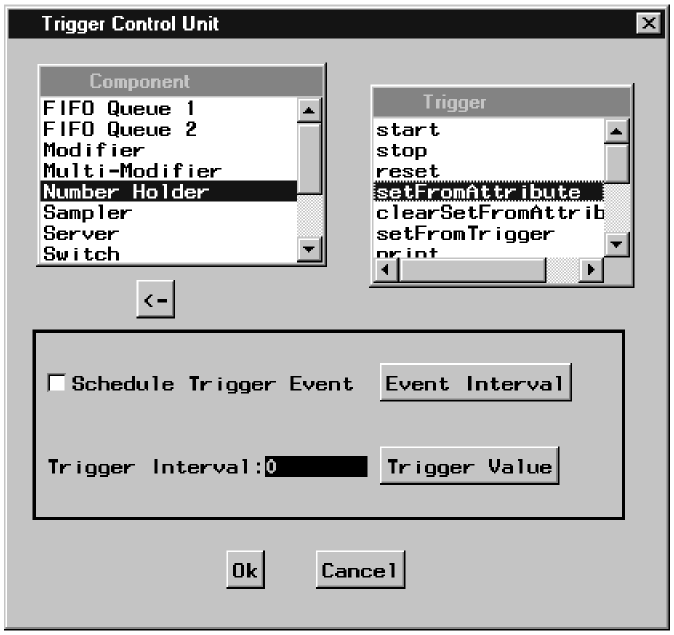


Figure 2.14. A Trigger Control Panel

The **Initial Value** field in the holder control panels provide a way of initializing the holder. This is useful when using the holder as a counter of resources. When the NumberHolder decrements, there is one less available resource. Other parts of the model may query the holder to see if there are resources available for certain activities. In this case it may be desirable to have an initial pool available.

The **Disable Reset** check box will disable the resetting of the last value in the holder when the simulation reset button is pressed. If not checked, when the reset button is pressed the holder is reset to its initial value. If checked, the reset button has no affect on the holder.

The following documents the logic of the NumberHolder component:

Transaction Arrival

sets the value as specified in the NumberHolder Control Panel; then flows the transaction to each arc directed away from the component.

Request for Transaction

passes on the request to all arcs directed into the component.

Are You Busy Message

if any component on an arc leading out of the Trigger is busy, then return TRUE; otherwise, return FALSE.

Query Message

Keyword	Meaning
currentValue	returns the value in the holder at the current time
id	returns the component's unique identifier.
value	returns the value in the holder when the transaction passed through it.

Trigger Message

Keyword	Meaning
+	adds the transaction attribute to the value.
-	subtracts the transaction attribute from the value.
clearSetFromAttribute	clears the value then sets it.
controls	displays the Holder Control Panel.
decrement	decrements the value.
increment	increments the value.
print	prints the value on the SAS Log window.
reset	resets the value and the value.
setFromAttribute	sets the value from the transaction attribute.
setFromTrigger	sets the value that is assigned with the **Trigger Value** button in the Trigger Control Panel.
setTimenow	sets the simulation time into the value.
start	starts the holder.
stop	stops the holder.

StringHolder Components

The following list documents the logic of the StringHolder component:

Transaction Arrival
sets the value as specified in the StringHolder Control Panel; then flows the transaction to each arc directed away from the component.

Request for Transaction
passes on the request to all arcs directed into the component.

Are You Busy Message
if any component on an arc leading out of the Trigger is busy, then return TRUE; otherwise, return FALSE.

Query Message

Keyword	**Meaning**
currentValue	returns the value in the holder at the current time.
id	returns the component's unique identifier.
value	returns the value in the holder when the transaction passed through it.

Trigger Message

clearSetFromAttribute	clears the value then sets it.
controls	displays the Holder Control Panel.
print	prints the value on the SAS Log window.
reset	resets the value.
setFromAttribute	sets the value from the transaction attribute.
setFromTrigger	sets the value that is assigned with the **Trigger Value** button in the Trigger Control Panel.
start	starts the holder.
stop	stops the holder.

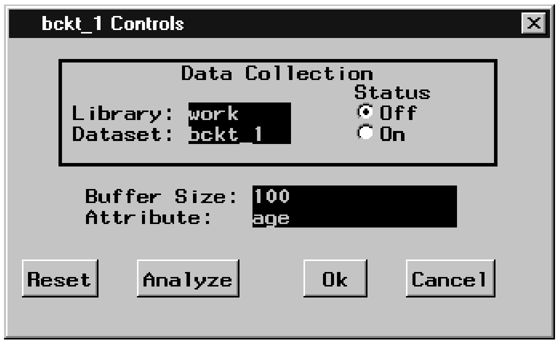
Chart Components

There are five types of charts and a Bucket component. The charts are used to display information about the performance of the system. The bucket collects data. Charts can be used in two ways: you can drag and drop a component (queue, server, or NumberHolder for example) on a chart and then choose the attribute of the component you want to display in the chart and the frequency with which to sample the component. In those cases, the chart will instantiate a bucket within it to collect the data. On the other hand, if you drop a bucket into a chart, the chart uses that instant as its data source. This is used to display attribute data in transactions.

Icon	Component	Description
	Bucket	collect value of an attribute for analysis
	VHistogram	vertical histogram of numeric data
	HHistogram	horizontal histogram of numeric data
	VBoxPlot	vertical box plot of numeric data
	HBoxPlot	horizontal box plot of numeric data
	LinePlot	plot of numeric data over simulation time

Bucket

The bucket is a component for collecting statistics on an attribute and saving its values in a SAS data set. You name an attribute for which you want to collect statistics in the Bucket control panel. The attribute "age" is the default collected. You can also specify the buffer size, which is the number of values of the attribute that is used in calculating statistics and displaying in the chart component. There is also a way to name a data set into which the attribute's values are saved and a way to start and stop data collection. In addition, the 'Reset' button empties the buffer and the 'Ok' button sets the buffer size and attribute names.


bckt_1 Controls

Data Collection

Library: work

Dataset: bckt_1

Status: [✓ Off] On

Buffer Size: 100

Attribute: age

Figure 2.15. A Bucket control panel

When you select the **Collect Data** check box, for each transaction the values of the monitored attribute are saved in a SAS data set. Chapter 7, "Saving and Restoring," discusses the details of this data set. You can analyze the data you collect by pressing the **Analyze** button which executes PROC UNIVARIATE and PROC GPLOT. Chapter 8, "Analyzing the Sample Path," discusses the details of the type of analysis.

Box Plot

The Box Plot shows the minimum, maximum, and quartiles of the attribute that it is monitoring. This attribute can be named in a bucket that is dropped on the box plot or can be one of the states of a component that is dropped on the box plot. If a bucket is dropped on the box plot then the bin controls for the box plot are those of the bucket. If a component is dropped on the box plot then there is a hidden bucket associated with the box plot and the bin controls are associated with the box plot and are accessible from the box plot control panel.

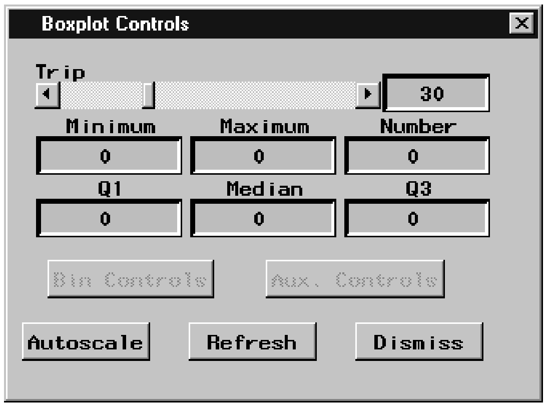


Figure 2.16. A Box Plot Control Panel

The box plot control panel has buttons to enable or disable automatic scaling; refresh the plot; raise the local bin controls; and to specify auxiliary controls. The latter two buttons apply if you drop a component on a box plot.

You can also drop box plots onto other box plots. This enables you to collect statistics on the minimum, maximum, and quartiles of the attribute. It is a type of *batch means*.

Histogram

The histogram shows a distribution histogram of the selected attribute. The Histogram control panel shows the minimum, maximum, number, mean, and standard deviation of the attribute that it is monitoring. This attribute can be named in a bucket that is dropped on the histogram or can be one of the states of a component that is dropped on the histogram. If a bucket is dropped on the histogram, then the bin controls for the histogram are those of the bucket. If a component is dropped on the histogram, then the bin controls are associated with the histogram and are accessible from the histogram control panel.

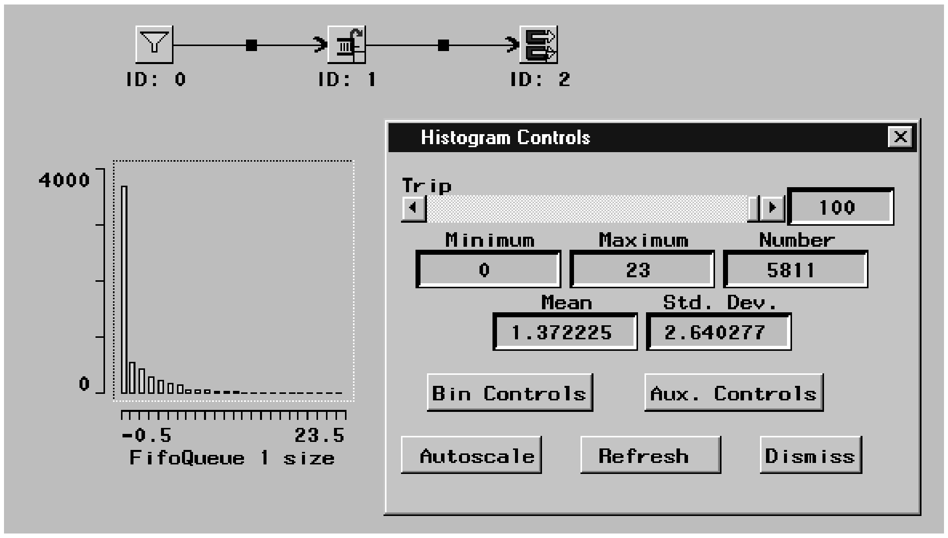


Figure 2.17. A Histogram and Control Panel

The histogram control panel has buttons to enable or disable automatic scaling; refresh the plot; raise the local bin controls; and to specify auxiliary controls. The latter two buttons apply if you drop a component on a histogram.

You can also drop histograms onto other histograms. This enables you to collect statistics on the minimum, maximum, mean, and standard deviation of the attribute. It is a type of *batch means*.

Line Plot

The line plot shows the attributes sample path. The line plot control panel shows the minimum, maximum, number, mean, and standard deviation of the attribute that it is monitoring. This attribute can be named in a bucket that is dropped on the line plot or can be one of the states of a component that is dropped on the line plot. If a bucket is dropped on the line plot, then the bin controls for the line plot are those of the bucket. If a component is dropped on the line plot, then the bin controls are associated with the line plot and are accessible from the line plot control panel.

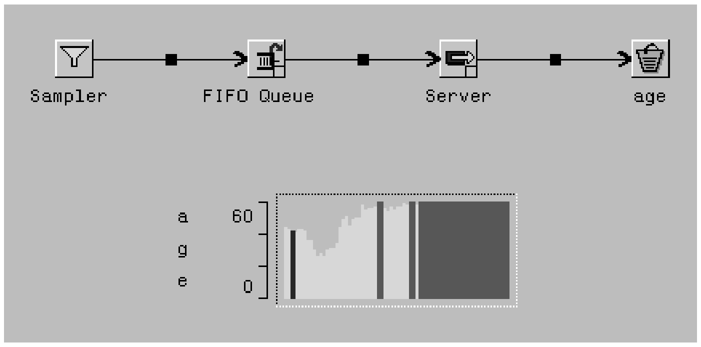


Figure 2.18. A Line Plot of Transaction Age

The line plot control panel has buttons to enable or disable automatic scaling; refresh the plot; raise the local bin controls; and to specify auxiliary controls. The latter two buttons apply if you drop a component on a line plot.

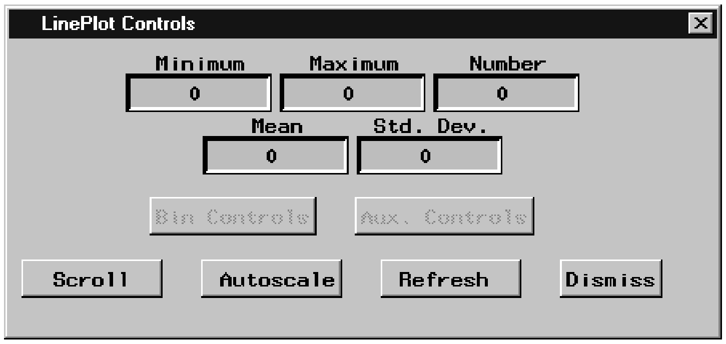


Figure 2.19. A Line Plot Control Panel

You can also drop line plots onto other line plots. This enables you to collect statistics on the minimum, maximum, mean, and standard deviations of the attribute. It is a type of *batch means*.

Port, Connector, and Label Components

Ports and Connectors aid in connecting components to each other and are useful when building hierarchical models and assembling components into larger aggregate components. You can annotate the simulation with text by using Labels. In addition, you can attach labels to many of the elementary components. You do this by selecting **Tools ➤ Add label** from the pop-up menu on the component you want to annotate. Then, type the text you want to appear in the label.

Icon	Component	Description
	Port	for connecting multiple components
	Connector	for connecting multiple components without using arcs
Label	Label	for annotating the model

Figure 2.20 shows an example with a connector labeled “a.” When a transaction flows into connector “a,” it will flow out of all other connectors “a.”

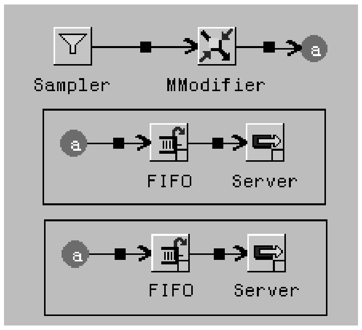


Figure 2.20. Simple Example Using a Connector

The three “a” connectors are treated as identical. An equivalent model using an instance of a Port is shown in Figure 2.21.

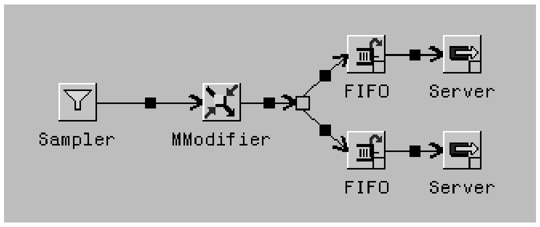


Figure 2.21. Simple Example Using a Port

Here, you see the port explicitly connects the three components, which were implicitly connected using the Connector. In addition, ports have a special role in compound components. In this setting they can be used to create special connections from the outside of compound components to the inside of compound components. See Chapter 5, “Building a Model with Compound Components,” for more details on this special function.

You duplicate a Connector by selecting **Duplicate** on the pop-up menu. If you type in the interior of the Connector, then all the duplicates of that connector will display the same text.

Connecting Components

The examples presented thus far use arcs to connect components. This section describes arcs and some of the features they provide in more detail. If you click on the right side of a component, when the cursor is in the “+” shape, a rubberband line displays from the component to the cursor. If you don’t see this line, it means that the component doesn’t support arcs directed away from it. You will not get an error message. If there is a rubberband line attached to the cursor, when you click in another component that supports arcs directed towards it, the rubberband line is replaced by a solid arc. If, while the rubberband line is connected to the cursor, you move the cursor to the right or below the window border, the window will scroll automatically. This is power scrolling, and it allows you to connect components that may not be visible in the window simultaneously. If you click outside the window border while power scrolling, then the rubberband line is dropped.

There are two types of arcs: regular arcs and segmented arcs. As the name implies, segmented arcs are composed of multiple line segments. Figure 2.22 shows the two types of arcs.

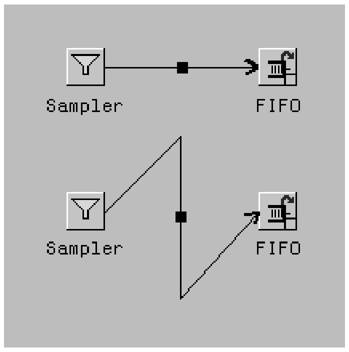


Figure 2.22. Two Types of Arcs

If you click on the simulation window background while a rubberband line is connected to the cursor, the selected point ends one line segment and begins another. In this way you can create circuitous routes between components.

Notice the rectangular handle in the center of the arc. This is the arc's "hot spot." If you click the right mouse button while the cursor is over the hot spot, a menu associated with the arc pops up.

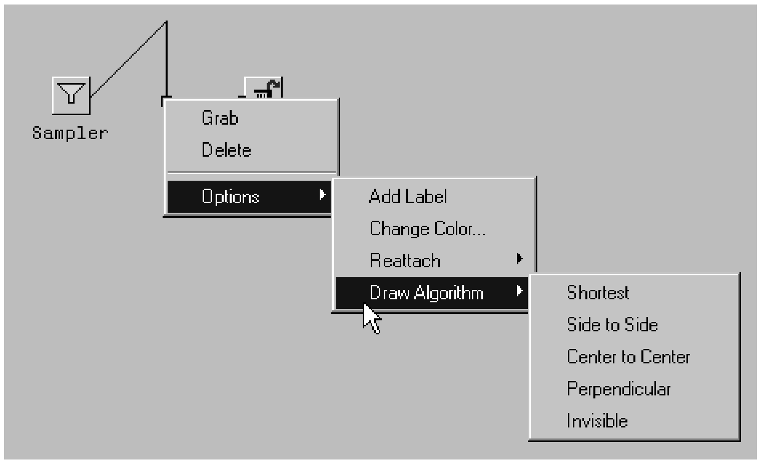


Figure 2.23. Pop-Up Menu for a Segmented Arc

In the pop-up menu for segmented arcs in Figure 2.23, there is a selection titled **Perpendicular**. This selection causes the arc to be drawn so that the line segments are perpendicular to each other. As the numerous selections in Figure 2.23 show, a full range of capabilities are available.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/OR® User's Guide: QSIM Application, Version 8,* Cary, NC: SAS Institute Inc., 1999. 110 pp.

SAS/OR® User's Guide: QSIM Application, Version 8

ISBN 1–58025–488–8

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999