

# Chapter 12

## Materials Handling

---

In the `STANDARD LIBRARY`, you have already noticed that there is a `VEHICLE` object and a `CONVEYOR` path. Perhaps the term “vehicle” already conjures up the idea of moving materials or people from place to place and your intuition would be correct. The `VEHICLE` object can pick up and delivery entities which can be used to act as a fork lift truck in a manufacturing operation, a human stock handler in a warehouse, a taxi cab for picking up and delivering people as well as a bus, a train, or other type of people mover to move entities either individually or in a group. The vehicle is an individual object whose routing is based on either “demand” or on a “fixed path”. A conveyor in SIMIO, on the other hand, provides a continuous platform for conveying items or people. It can represent a wide range of industrial devises like gravity conveyors, belt conveyors, powered overhead conveyor, and even a power and free conveyor. Often materials handling in industry will employ a variety of materials handling devises. A distinction among conveyors used by SIMIO is whether the conveyor “accumulates” or doesn’t (“non-accumulating”). An accumulating conveyor allows entities to queue up at its end. A non-accumulating conveyor will stop when an entity gets to the end and won’t start up until that entity is removed.

A key to modeling materials handling problems in SIMIO is the use of the `TRANSFERNODE` and to somewhat a lesser extent, the external, `OUTPUTBUFFER` which are attached to many of the objects like the `SOURCE` and the `SERVER`. The `TRANSFERNODE` (unlike the `BASICNODE`) provides the “*Transport Logic*” which means you can request a vehicle to move the entity to its next designation. If that transport device is not available, then the entity must wait. The `TRANSFERNODE` can be used alone and entities can wait there for pickup at that node but they will not queue up in the usual fashion. For other objects, the `OUTPUTBUFFER` provides a place for the entities to wait for transportation.

### Part 12.1: Vehicles: Cart Transfer in Manufacturing Cell

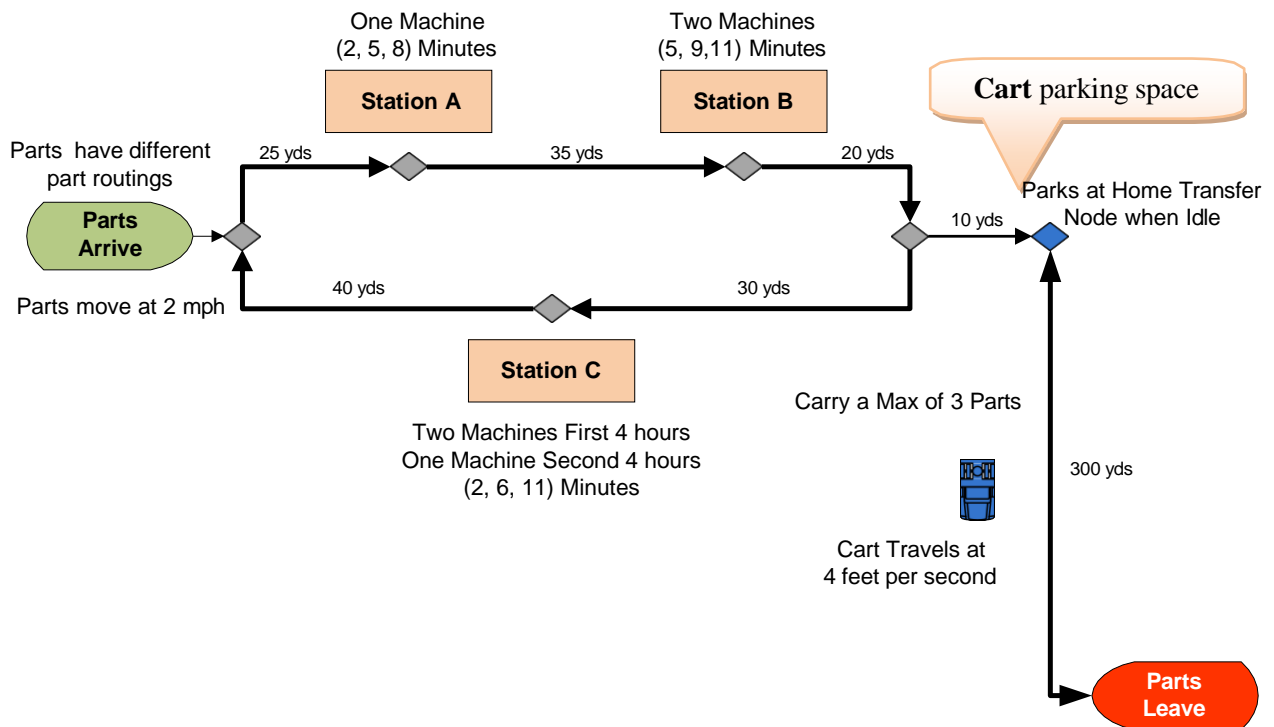
We will reconsider the original problem of Chapter 5, whose layout and information are shown in Figure 12.1. Recall there are four part types, each with its own source and each part type has its own routing through the manufacturing cell. However, we now learn that instead of leaving the system, parts actually have to be put on a cart and wheeled to the warehouse building 300 yards away (see Figure 5.9). The cart can carry a maximum of three parts and travels at four feet per second. After the cart has finished dropping off parts, it will return to the pickup station for the next set of parts or wait there until a part is ready to be taken. The cart does not have to be filled for it to be taken to the warehouse (i.e., will carry one part if it is available).

**Step 1:** Open up the last model from Chapter 5 and save it as chap12.1.

**Step 2:** To facilitate a cart taking the parts to the warehouse, delete the `CONNECTOR` between the `SINK` object and the prior `BASICNODE`. Next, insert a `TRANSFERNODE` in between the **SnkPartsLeave** named **TNodeCart** as seen in Figure 12.1.

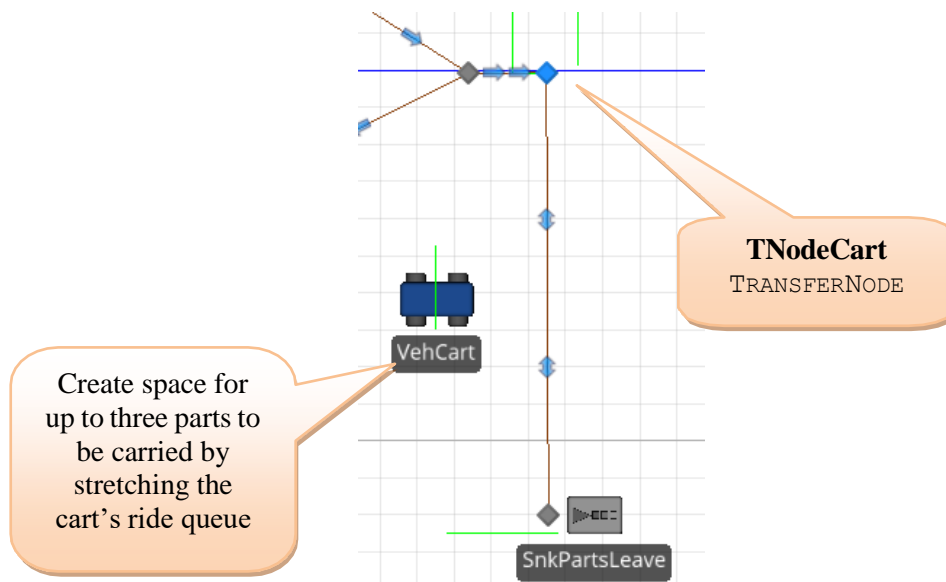
**Step 3:** Select the **Input@SnkPartsLeave** and remove the *Tally Statistics* as `VEHICLES` will be entering instead. To calculate the same statistics, we could insert a *Tally* step in the *Entered* add-on process trigger.

**Step 4:** Connect the `BASICNODE` to the `TRANSFERNODE` with a `PATH` with a logical length of 10 yards. Finally, connect the **TNodeCart** `TRANSFERNODE` to the `SINK` with a `PATH` making sure it’s *Type* is set to “*Bidirectional*” so our cart can move in both directions. Set the path’s logical length to 300 yards.



**Figure 12.1: Adding a Vehicle to Carry Part to the Exit**

**Step 5:** Drag and drop a **VEHICLE** object from the [Standard Library] to the modeling canvas in the facility window named **VehCart** similar to Figure 12.2. Make sure to increase the size of the **RIDESTATION** queue which animates the parts being transported so it can hold three parts.



**Figure 12.2: Adding a Vehicle**

**Step 6:** Change the **TNodeCart** **TRANSFERNODE**'s *Ride On Transporter* property to "True" and specify the "VehCart" as the *Vehicle Name* which was just added (see Figure 12.3). Every time an entity passes through this node, it will now generate a ride request to be sent to the vehicle. The entity cannot continue until the vehicle is there to pick it up.

<input type="checkbox"/>	<b>Crossing Logic</b>	
<input type="checkbox"/>	<b>Routing Logic</b>	
<input type="checkbox"/>	<b>Transport Logic</b>	
	Ride On Transporter	<b>True</b>
<input type="checkbox"/>	Transporter Type	Specific
	Transporter Name	<b>VehCart</b>
	Reservation Method	Reserve Closest
	Selection Goal	Preferred Order
	Selection Condition	
<input type="checkbox"/>	Transporter Status On Drop-Off	
	Keep Reserved If	

**Figure 12.3: Setting up the Node to Have Entities Utilize a Vehicle**

**Step 7:** Use Figure 12.4 as a guide to set the **VehCart**'s *Initial Desired Speed* to four feet per second and *Ride Capacity* to three. Also, make sure to set the *Initial Node(Home)* to the **TNodeCart** node which was just added, as this will be our vehicle's home node. Leave the *Routing Type* specified as "On Demand", meaning it will respond to requests for transportation from the stations as needed. Now change the **VehCart**'s *Idle Action* property to "Park At Home" so that the vehicle will always go to the home node if there are no entities with ride requests.<sup>152</sup> Otherwise the **VEHICLE** would remain at the **SINK** until a ride request is generated.

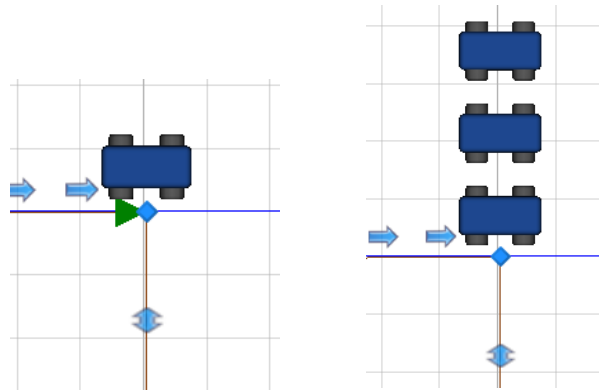
Properties: VehCart (Vehicle)	
<input type="checkbox"/>	Show Commonly Used Properties Only
<input type="checkbox"/>	<b>Transport Logic</b>
	Initial Ride Capacity <b>3</b>
	Task Selection Strategy First In Queue
<input type="checkbox"/>	Load Time 0.0
<input type="checkbox"/>	Unload Time 0.0
	Park to Load/Unload False
	Minimum Dwell Time Type No Requirement
<input type="checkbox"/>	<b>Travel Logic</b>
<input type="checkbox"/>	Initial Desired Speed <b>4.0</b>
	Units <b>Feet per Second</b>
	Initial Network Global
	Network Turnaround Method Exit & Re-enter
<input type="checkbox"/>	<b>Routing Logic</b>
	Initial Priority 1.0
	Initial Node (Home) <b>TNodeCart</b>
	Routing Type On Demand
	Idle Action <b>Park At Home</b>
	Off Shift Action Park At Node

**Figure 12.4: Vehicle Properties**

**Step 8:** Run the simulation now to see how the cart works in the animation. As you can see the **VehCart** originally is parked at the **TNodeCart** node as seen in Figure 12.5. By default, the parking animation queue is above the node<sup>153</sup> and oriented from left to right. If there was more than one vehicle they would be stacked on top of one another again facing left to right in the facility as seen in the right picture of Figure 12.5 where there are three carts in the system.

<sup>152</sup> The *Network Turnaround Method* can be changed for example to "Reverse" so the cart would go forward to the exit and then look as if it backs up to the **TNodeCart** node. By default it will turn around and move forward back to the home **TNodeCart** node.

<sup>153</sup> Note the familiar green line that represents the animated queue line is not visible in this case.



**Figure 12.5: Vehicle is parked at the Home Node using Default Parking Station**

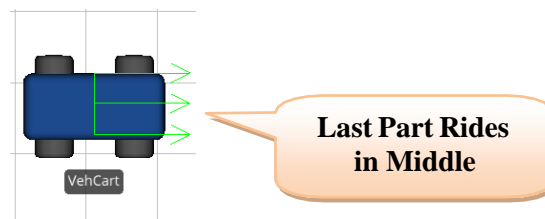
*Question 1:* Does the Cart pick up more than one part?

---

*Question 2:* What do you notice about how a single part or two parts ride on the vehicle?

---

**Step 9:** To fix the issue of a single part riding out to one side, select the `RIDE STATION` queue on the `VEHICLE` and change the *Alignment* from “None” to “Point” under the *Appearance* tab. Add an additional vertex to the queue to accommodate three parts and move the first orientation point which is on the bottom to the middle as seen in Figure 12.6.



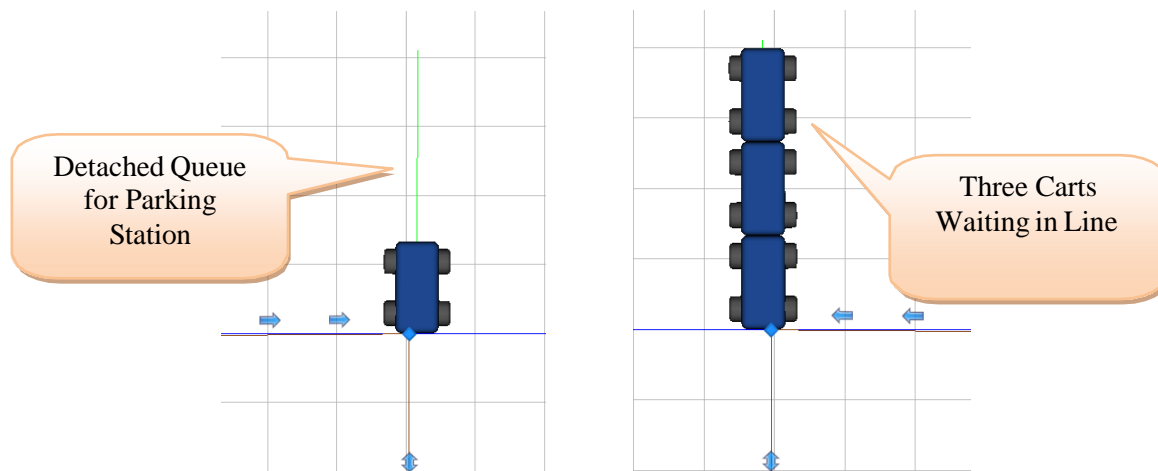
**Figure 12.6: Changing the Orientation of the Riding Queue**

**Step 10:** Run the simulation now to see how the cart works in the animation.

**Step 11:** By default, a parking location for a node is automatically added above the node as seen in Figure 12.7. However, you should generally not utilize the default one since the location and/or shape of the parking station cannot be adjusted as well as the orientation of the `VEHICLES`. Select the `TNodeCart` node by clicking it and then toggle the *Parking Queue* option in the *Appearance* tab to be clear. Next, a parking location is needed for the vehicle or the vehicle will vanish when parked.<sup>154</sup> Click on the `TNodeCart` node and select “*Draw Queue*” from the *Appearance* Tab. Click on the `ParkingStation.Contents`. A cross cursor will now appear in the facility window. Left click where you want the front of the queue to be and then right click where you want it to end (before you right click to end the queue you could also continue left clicking to add more vertices). Place it above the node so the Cart will be facing downward as seen in Figure 12.7. Click on the queue and in the properties window make sure the *Queue State* is `ParkingStation.Contents`. This queue will animate the parking station of the `TNodeCart` `TRANSFERNODE` for vehicles parked at the node. It can be oriented and placed at any location. Figure 12.7 shows the same scenario as Figure 12.5 but this time we are able to orient the vehicles to pointing in the same direction as the path.

---

<sup>154</sup> The parking queue is again just for the animation since the vehicle is physically at the node even though it cannot be seen.

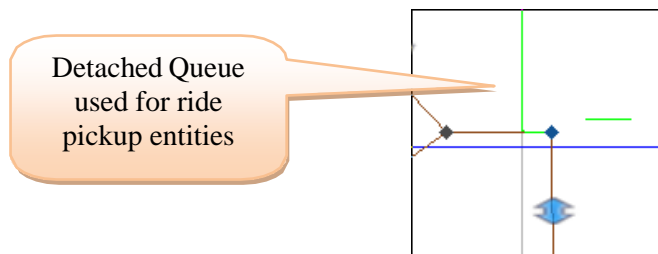


**Figure 12.7: Vehicle is parked at the Home Node using the New Station**

**Step 12:** Run the simulation now to see how the cart works in the animation.

The model does not show how many parts are waiting for the cart at the transfer node so it may be helpful to see how many are there at any given time in the simulation. To accomplish this we can add an animated “Detached Queue” for the **TNodeCart**.

**Step 13:** Click on the **TNodeCart** node and select “*Draw Queue*” from the “*Appearance*” Tab. Click on the “*RidePickupQueue*”.<sup>155</sup> A cursor will now appear in the facility window. Left click where you want the front of the queue to be and then right click where you want it to end (before you right click to end the queue you could also continue left clicking to add more vertices). As seen in Figure 12.8, arrange this detached queue in an “L” shape. Otherwise an entity is shown trying to enter the transfer node will be visible<sup>156</sup> and by overlaying the “L” it shows all the entities waiting for transportation.<sup>157</sup>



**Figure 12.8: Ride Pickup Queue**

**Step 14:** Run the simulation to see how the `DETACHED_QUEUE` works.

**Question 3:** What is the utilization of the **VehCart** during a 40 hour replication?

<sup>155</sup> If added just generic animated queue now, change the *Queue State* property to **TNodeCart.RidePickupQueue** where **TNodeCart** is the name of the transfer node where the parts are being picked up. Your transfer node’s name may be different.

<sup>156</sup> You can actually see the entity attempting to enter the transfer node if you watch that node carefully.

<sup>157</sup> These queues are for animation purposes only. The parts physically reside at the entrance of the node and one will always be visible as the others will stack up underneath. In later chapters, the notion of a `STATION` to hold entities will be discussed.

## Part 12.2: Cart Transfer among Stations

Now, suppose that the transportation of parts among the stations, including entry and exit are handled by two similar vehicles. These vehicles are different than the **VehCart** which carries the parts to the warehouse **SINK**. These carts will wait at the beginning of the circular network until they are needed.

**Step 1:** Convert the one **BASICNODE** that connects the four **SOURCES** into a **TRANSFERNODE** by right clicking and selecting the *Convert to Type* menu item. This will be necessary in order for parts entering the network from the sources the ability to request transportation on the inside carts. Next, name this new **TRANSFERNODE** **TNodeStart**.

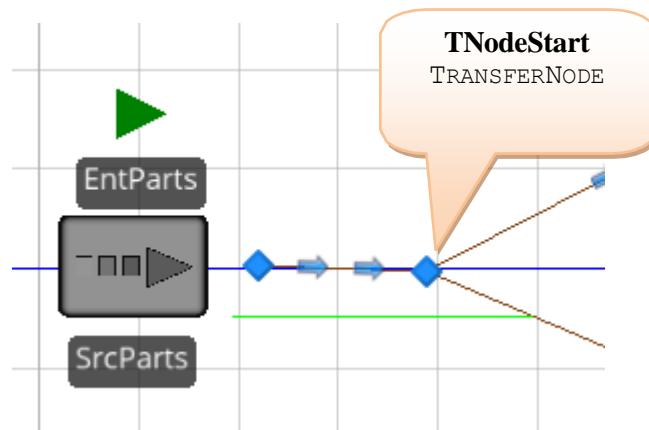


Figure 12.9: Changing the Start Node to a **TRANSFERNODE**

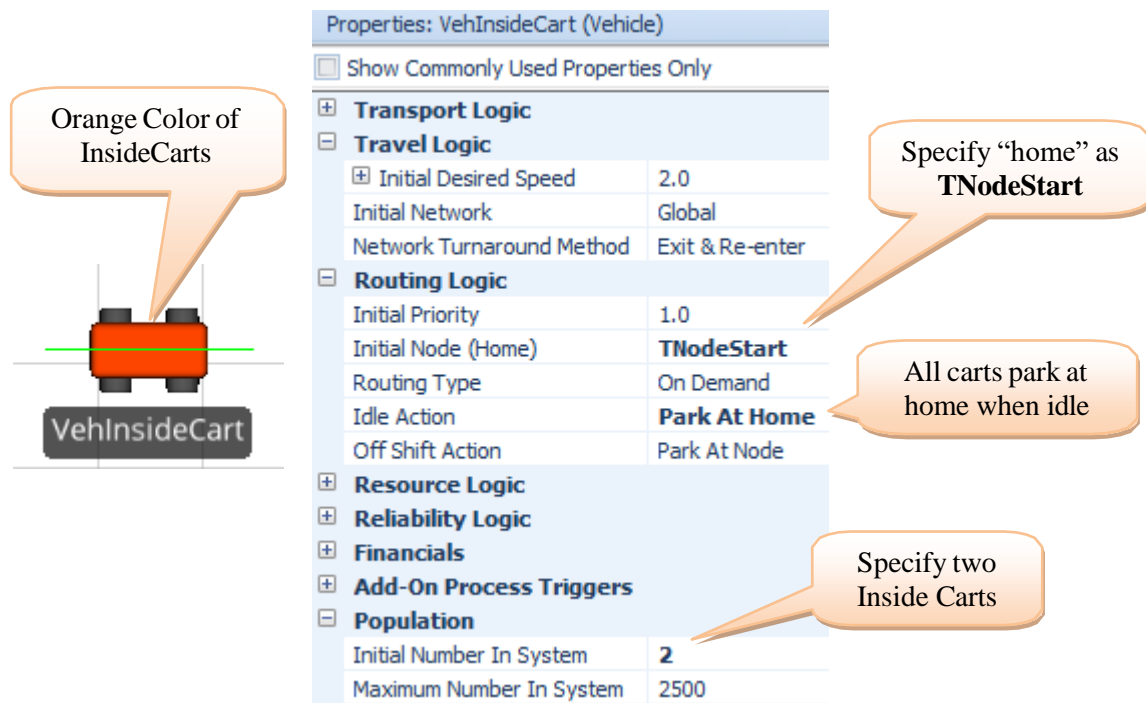
**Step 2:** Now add the inside carts by inserting another **VEHICLE** from the [Standard Library] into the model naming it **VehInsideCart**. Shrink the vehicle leaving the ride station queue in the same position.

**Step 3:** Make the **VehInsideCart** have an initial *population* of two<sup>158</sup> which will allow you to see the simultaneous behavior of multiple vehicles on the same path. The new vehicle will have the characteristics, as shown in Figure 12.10 (i.e., the *Initial Node* should be **TNodeStart** and the *Idle Action* property should be set to “Park at Home”). Select the “Transporting State” from the “Additional Symbols” and color it orange.

Notice that the **VEHICLE** object is similar to the **ENTITY** object in that multiple instances, which we call “run-time” instances can be created from the “design-time” instance of the **VehInsideCart**.<sup>159</sup> “Design-time” occurs while you are putting instances of the objects onto the modeling. All the objects in the **FACILITY** window are initialized just before the simulation begins to execute. “Run-time” occurs when the simulation executes.

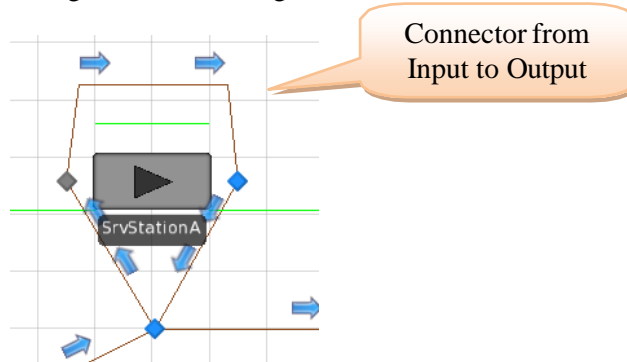
<sup>158</sup> The fact that the property description says *Initial number in System* should indicate that vehicles can be added and removed from the simulation during the simulation execution (something that is left to a later chapter).

<sup>159</sup> We will see that the **WORKER** object can also have run-time instances.



**Figure 12.10: Properties of the InsideCart**

**Step 4:** Now we need to make sure that the inside carts can drop off parts and then return back to the circular path. Since they obviously cannot go through the `SERVERS`, the inside carts would move to the input node of a `SERVER`, drop off the part and then become deadlocked (i.e., stuck).<sup>160</sup> The easiest solution is to allow the carts to maneuver around the stations (i.e., add a `CONNECTOR` from the input to the output nodes of each station as shown in Figure 12.11 making sure the direction of the link is correct).<sup>161</sup>



**Figure 12.11: Adding a Connector at a Station**

**Step 5:** Next, select the `TNodeStart` and the three output nodes (i.e., `TRANSFERNODES`) of the station servers and specify the entities leaving these nodes to ride on an `VehInsideCart` as seen in Figure 12.12.

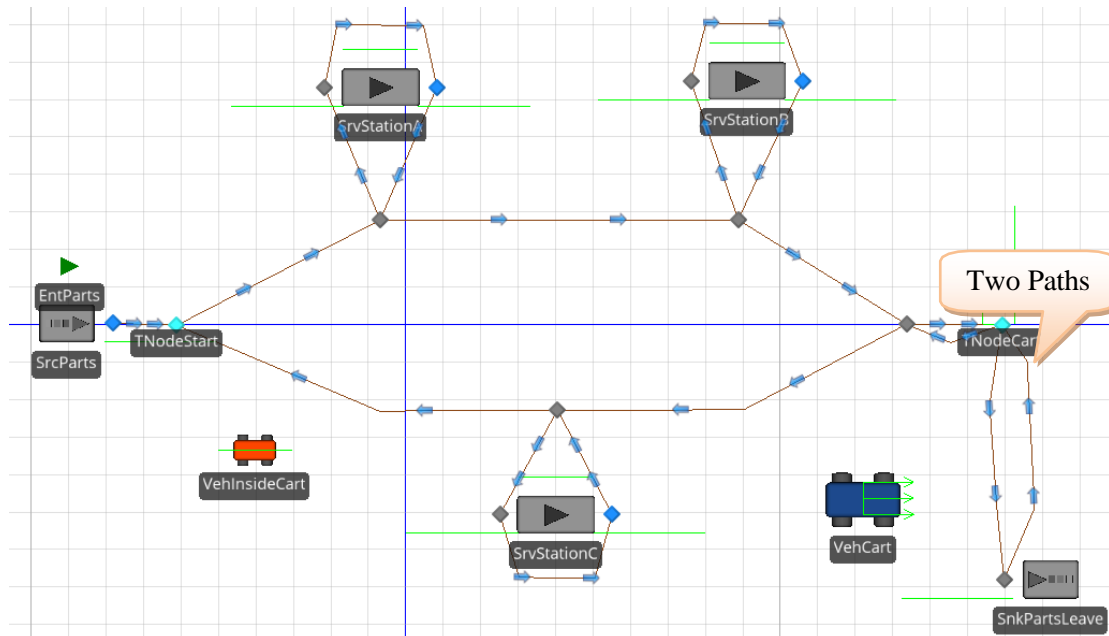
<sup>160</sup> By default, `WORKERS` and `VEHICLES` will not enter fixed objects like `SERVERS`, `SINKS`, `COMBINERS`, etc. One solution is to create a connector leaving the input node of the server to the transfer node as well as another one from the transfer node to the output node to allow pickup.

<sup>161</sup> Recall, `CONNECTORS` take up zero time as well as have zero length in the model. Basically, the inputs and outputs of the server are really the transfer nodes of the circular path.

Transport Logic	
Ride On Transporter	True
Transporter Type	Specific
Transporter Name	VehInsideCart
Reservation Method	Reserve Closest
Selection Goal	Preferred Order

**Figure 12.12: Use the Inside Cart**

**Step 6:** Finally change the path between the exit node on the inside loop and the **TNodeCart** node to two one-way paths (i.e., add an additional path from the **TNodeCart** back to the basic node making sure to make the path ten yards long). Also change the path between the **TNodeCart** and the **SnkPartsLeave** to two one-way paths as well. Now your overall model should look like the one in Figure 12.13.<sup>162</sup>



**Figure 12.13: Model with Vehicles**

**Step 7:** Also, change the **VehCart** *Population* property *Initial Number in the System* to “2” so two carts will be available for the outside transportation. The **TNodeCart** node should also be specified as shown in Figure 12.14.

Routing Logic	
Outbound Link Preference	Any
Outbound Link Rule	Shortest Path
Entity Destination Type	By Sequence
Transport Logic	
Ride On Transporter	True
Transporter Type	Specific
Transporter Name	VehCart

**Figure 12.14: Specifying TNodeCart**






**Step 8:** Save and run the model observing what happens.

**Question 4:** What did you notice about the utilization of the outside carts?

<sup>162</sup> One can use bidirectional paths but this can lead to bottlenecks and deadlock as vehicles, workers, and entities can only be going in one direction and must wait for all items to clear path before the direction can be reversed. It is always better to use to separate paths.



**Step 9:** The difficulty was the parts were not unloaded from the inside cart to be transported by one of the two outside carts to the `SINK`. The issue is once the part has finished its last processing on a station, the next sequence is the exit. Therefore, the cart takes the part to its next destination (i.e., **Input@SnkPartsLeave**). To force the inside cart to drop off the parts at the **TNodeCart** node, this node should be inserted into the four part sequences right before the **Input@SnkPartsLeave** row for each type as seen in Figure 12.15.<sup>163</sup>

Table Parts	Sequence Part		
	Sequence	ProcessingTimes (Minutes)	ID
1	Input@SrvStationA	Random.Pert(2,5,8)	 1
2	Input@SrvStationC	Random.Pert(2,6,11)	 1
3	TNodeCart	0.0	 1
4	Input@SnkPartsLeave	0.0	 1
5	Input@SrvStationA	Random.Pert(1,3,4)	 2

**Figure 12.15: Adding the Transfer Cart to Force a Drop off of the Parts**

**Step 10:** Once all four sequences have been modified, save and execute the model. Observe the interacting behavior of the carts as well as notice how the carts park.

**Question 5:** How do parts get transported from the inside workstations to the outside cart for transfer to the sink where the parts leave?

---

**Question 6:** Is the queue ride parking for the **TransferCart** node needed anymore?

---

**Question 7:** What is the utilization of both (outside) **Carts** as well as the utilization of both **InsideCarts** at the end of the simulation (24 hours)?

---

**Question 8:** How are the individual vehicles within a cart type distinguished in the output from each other?

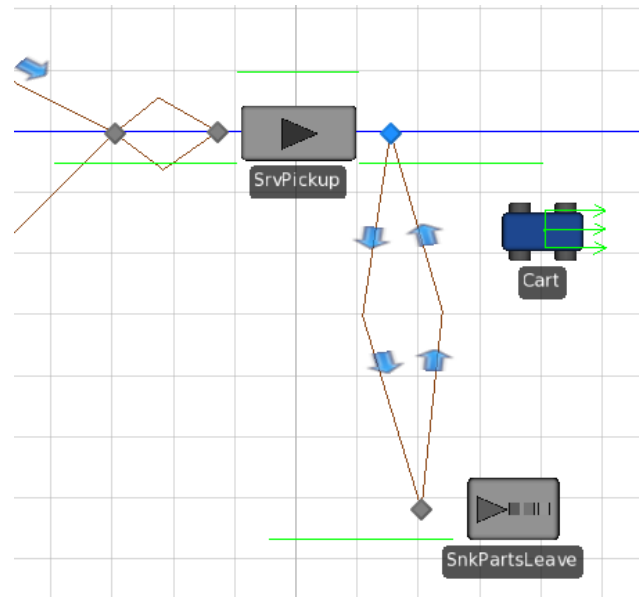
---

**Step 11:** There are a few anomalies that occur within the model. Notice how the inside carts will wait while they drop off a part when an outside cart is not available. When the part starts the drop off process, it visits the node which informs the part that it requires a vehicle to continue and therefore never completes the drop off process which releases the inside cart. To fix this anomaly along with other animation issues, delete the **TNodeCart** node and insert a new `SERVER` named **SrvPickup** with an infinite capacity and zero for the processing time as seen in Figure 12.16.

---

<sup>163</sup> The easiest way to insert the node into the sequence is to select the **Input@SnkPartsLeave** row and then click the *Insert Row* button in the *Data* section of the *Table* tab.

Properties: SrvPickup (Server)	
<input type="checkbox"/> Show Commonly Used Properties Only	
<b>Process Logic</b>	
Capacity Type	Fixed
Initial Capacity	<b>Infinity</b>
Ranking Rule	First In First Out
Dynamic Selection Rule	None
<input checked="" type="checkbox"/> Transfer-In Time	0.0
Process Type	Specific Time
<input checked="" type="checkbox"/> Processing Time	<b>0</b>
Units	Minutes



**Figure 12.16: Adding a new Server as Pickup Point**

**Step 12:** Reconnect the two 10 yard paths from the path network to the input of the **SrvPickup** and the two 300 yard paths from the output node to the **SINK**.

**Step 13:** For the sequence table, change the TransferCart node to the **Input@SrvPickup** as well as make the home node for the **Cart** the **Output@SrvPickup** node as seen in Figure 12.17.

Table Parts	Sequence Part		
	Sequence	ProcessingTimes (Minutes)	ID
1	Input@SrvStationA	Random.Pert(2,5,8)	1
2	Input@SrvStationC	Random.Pert(2,6,11)	1
3	Input@SrvPickup	0.0	1
4	Input@SnkPartsLeave	0.0	1
5	Input@SrvStationA	Random.Pert(1,3,4)	2

<b>Travel Logic</b>	
<input checked="" type="checkbox"/> Initial Desired Speed	<b>4</b>
Units	<b>Feet per Second</b>
Initial Network	Global
Network Turnaround Method	Exit & Re-enter
<b>Routing Logic</b>	
Initial Priority	1.0
Initial Node (Home)	<b>Output@SrvPickup</b>
Routing Type	On Demand
Idle Action	<b>Park At Home</b>
Off Shift Action	Park At Node

**Figure 12.17: Changing the Sequence Table and Cart Home Node**

**Step 14:** Finally, request for transportation on the **Output@SrvPickup** transfer node using the same information from Figure 12.14.

**Step 15:** Save and run the model observing what happens with the two types of carts.

**Question 9:** Now what is the utilization of both (outside) **Carts** as well as the utilization of both **InsideCarts** at the end of the simulation (24 hours)?

### Part 12.3: Other Vehicle Travel Behaviors (Fixed Route and Free Space Travel)

Now let's experiment with different vehicle specifications to create different cart behaviors. We will now explore both a fixed route travel (i.e., bus) as well as travel through free space.

## Fixed Route Travel

Currently the **VehInsideCart** travels to wherever transport is needed among the stations including entry and pickup. That *Routing Type*, specified on the vehicle definition, is called “**On Demand**.” Suppose now we want the cart to travel in a fixed sequence around all the service points in the model.

**Step 1:** From the Data tab, insert a `SEQUENCE` TABLE for the vehicle *Route Sequence* named **SeqInsideCart** as shown in Figure 12.18. Note the sequence includes both input and output nodes of the stations.

Seq Inside Cart		Table Parts
		Sequence
1		TNodeStart
2		Input@SrvStationA
3		Output@SrvStationA
▶ 4		Input@SrvStationB
5		Output@SrvStationA
6		Input@SrvPickup
7		Input@SrvStationC
8		Output@SrvStationC

Routing Logic	
Initial Priority	1.0
Initial Node (Home)	TNodeStart
Routing Type	Fixed Route
Route Sequence	SeqInsideCart
Off Shift Action	Park At Node

**Figure 12.18: New Route Sequence for InsideCart**

**Step 2:** Select the **VehInsideCart** and set the *Routing Type* property to “**Fixed Route**” and the *Route Sequence* to the **SeqInsideCart** sequence table as seen in the right picture of Figure 12.18.

**Step 3:** Run the model and observe both the behavior of the cart(s) and the statistics on the cart performance.

**Question 10:** Are both inside carts behaving as expected? What is their utilization?

---

**Question 11:** Can you see the second cart? Why is it’s utilization 100%?

---

**Step 4:** Because both carts are always active, their utilization is 100% and the two carts ride (i.e., overlap one another) in the same location. So if we are to see carts we need to interrupt the constant motion or have the two carts start at different times (i.e., have a Source create them at different times). There several ways to achieve the result.

**Step 5:** Change the *Load Time* of the **VehInsideCart** to an `Exponential(0.5)` minutes. Save and run the model observing the cart behavior.

**Question 12:** What happens to both inside carts?

---

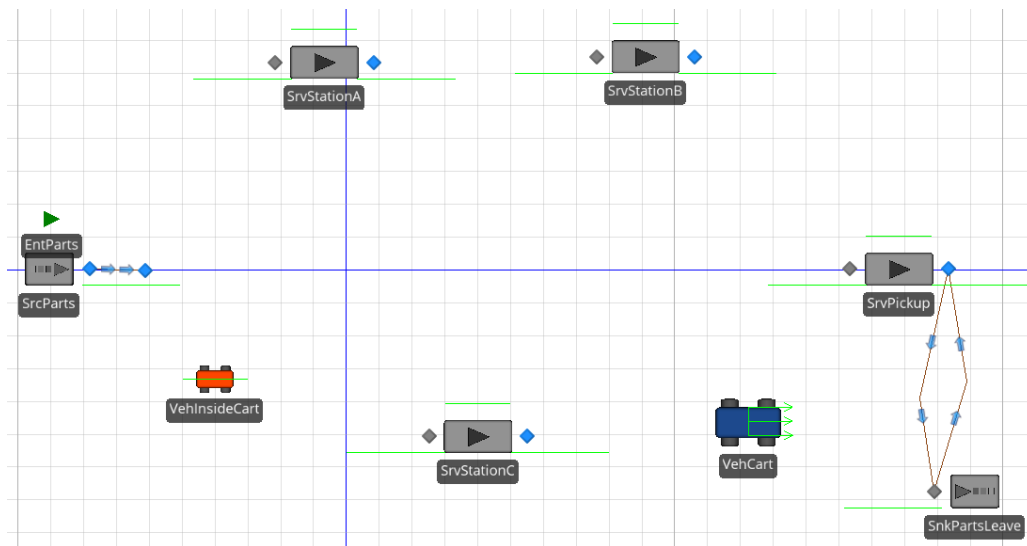
## Free Space Travel

Rather than travel on a specific path between stations, vehicles, workers and entities can travel in “free space.” To do this, SIMIO needs the “location” of the various nodes. In other words, it’s going to assume that your model is “to scale” as it might be if the animation layout is based on computer-aided drawing (CAD) software. It’s important to recognize that the time it takes parts to be moved by vehicles depends on the distance the vehicle travels and its travel speed<sup>164</sup>. To illustrate the free space travel, let’s treat our animation as though it came from a CAD drawing.

---

<sup>164</sup> When vehicles move entities, it’s the speed of the vehicle that determines how fast it traverses a distance and not the entity speed.

**Step 6:** Save the current model as Chap12.3Step6 and eliminate all the paths and nodes between the stations and the **SrvPickup** so that travel is unrestricted, as seen in Figure 12.19.



**Figure 12.19: Layout to Scale: No Paths**

**Step 7:** Modify the **VehInsideCart** “Travel Logic” and “Routing Logic” as Figure 12.20. In this case we are showing there is no travel network – meaning the vehicle must use “Free Space”. We also are routing **On Demand** and the *Idle Action* is to **Remain in Place**.

Properties: VehInsideCart (Vehicle)	
<input type="checkbox"/> Show Commonly Used Properties Only	
+ Transport Logic	
- Travel Logic	
Initial Desired Speed	2.0
Initial Network	No Network (Free Space)
Network Turnaround Method	Exit & Re-enter
- Routing Logic	
Initial Priority	1.0
Initial Node (Home)	TNodeStart
Routing Type	On Demand
Idle Action	Remain In Place
Off Shift Action	Park At Node

**Figure 12.20: Travel Logic and Routing Logic**

**Step 8:** Run the model and observe the travel of the inside carts.

**Question 13:** When would this type of travel be most appropriate and what do you observe about the parking?

## Part 12.4: Conveyors: A Transfer Line

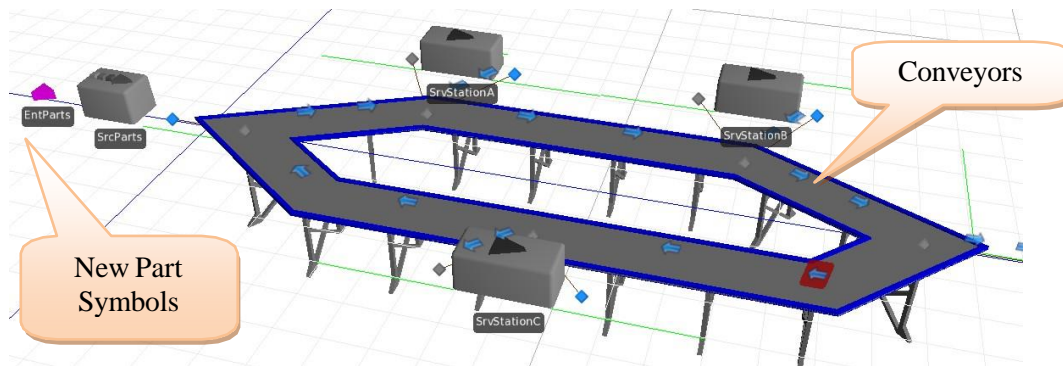
SIMIO provides for two kinds of conveyor modeling concepts: *accumulating* and *non-accumulating*. An accumulating conveyor allows parts to queue at its end, but the conveyor keeps moving. This approach might model a belt conveyor whose parts rest on “slip sheets” that allows the belt to continue to move. A non-accumulating conveyor must stop until the on/off operation completes. This concept might apply to an overhead conveyor with parts on carriers. SIMIO conveyors do not have on/off stations, so that a transfer line will have to be modeled as a series of individual conveyors. Since the transfer line stops, the non-accumulating concept applies. However the individual conveyors will need to be synchronized.

Suppose now that the transfer among stations in the manufacturing cell is handled by a transfer line which is a continuous conveyor, possibly in a loop, that has on/off stations. When a part is at an on/off station, the entire line pauses until the part has been placed on or taken off the conveyor. It takes between 0.5 and 1.5 minutes to load/unload the parts from the nodes.

**Step 1:** Reopen the last model from Chapter 5 and save it as a new name. Change the arrival rate of the source to be Exponential(15) now.

**Step 2:** Change all the paths on the interior of the model to conveyors by selecting the path and accessing the *Convert To Type* submenu via a right mouse click.<sup>165</sup> To embellish the model, add path decorators for conveyors as seen in Figure 12.21 from the *Path Decorators* tab.

**Step 3:** Also modify the symbols for the parts so they are one meter wide by one meter in length by one-half meter in height. Select the **EntParts** MODEL ENTITY and for each *Active Symbol*, change the size under the *General*→*Physical Characteristics*→*Size* properties. After selecting a different active symbol, you will need to deselect it and the reselect it to see the size for that symbol. We will need to be a little more specific about these sizes as we size the conveyors. We created these symbols from the “*Project Home*” tab in the “*Create*” section, clicking on “*New Symbol*” and selecting “*Create a New Symbol*”. When creating the symbol we paid close attention to its size (the grid is in meters).




**Figure 12.21: 3D View with Conveyors**

**Step 4:** Next, properties of the conveyors need to be specified. Here it is convenient to use the “*Spreadsheet View*” by right-clicking on one of the conveyors and selecting *Open Properties Spreadsheet View for all objects of this type*. The lengths from the previous paths (recall they were 25, 35, 20, 30, and 40 yards) do not need to be modified. All five conveyors should have a conveyor speed of 0.5 meters/second for each conveyor. Set the *Accumulating* property to **False** for each conveyor (i.e., the selection boxes should be unchecked).

**Step 5:** In the *Entity Alignment* property, choose the *Cell Location* option. Choosing this option causes the conveyor to be conceptually divided into cells. Parts going onto the conveyor will need to be placed into a cell. Therefore choose the number of cells to be equal to number associated with the conveyor length. Otherwise, if you select *Any Location* for the *Entity Alignment*, a part can go onto the conveyor at any place, even interfering with another.

---

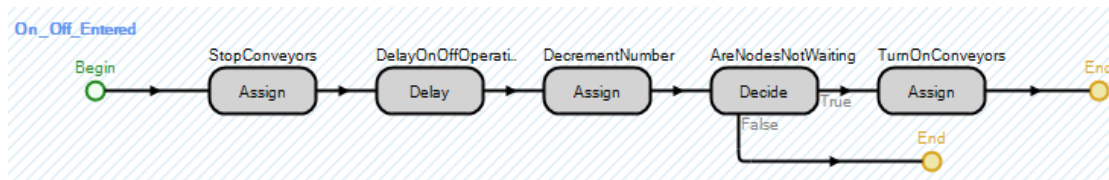
<sup>165</sup> Note you can use the *Ctrl* key to select all five paths and convert them all at once.

Properties Spreadsheet - 5 Conveyor Objects															 Click here to select an object
Travel Logic	Routing Logic	Reliability Logic	State Assignments	Financials	Add-On Process Triggers	Advanced Options	General	General/Physical Characteristics							
Instance Name	Initial Traveler Capacity	Entry Ranking Rule	Entry Ranking Expression	Initial Desired Speed	Units	Entity Alignment	Cell Spacing Type	Number Of Cells	Cell Size	Units	Auto Align Cells	Drawn To Scale	Logical Length	Units	
Conveyor1	Infinity	First In First Out	Entity.Priority	2.0	Meters per Second	Cell Location	Fixed Cell Size	1	1 Meters	With First Entity	<input type="checkbox"/>	<input type="checkbox"/>	25 Yards		
Conveyor2	Infinity	First In First Out	Entity.Priority	2.0	Meters per Second	Cell Location	Fixed Cell Size	1	1 Meters	With First Entity	<input type="checkbox"/>	<input type="checkbox"/>	35 Yards		
Conveyor3	Infinity	First In First Out	Entity.Priority	2.0	Meters per Second	Cell Location	Fixed Cell Size	1	1 Meters	With First Entity	<input type="checkbox"/>	<input type="checkbox"/>	20 Yards		
Conveyor4	Infinity	First In First Out	Entity.Priority	2.0	Meters per Second	Cell Location	Fixed Cell Size	1	1 Meters	With First Entity	<input type="checkbox"/>	<input type="checkbox"/>	30 Yards		
Conveyor5	Infinity	First In First Out	Entity.Priority	2.0	Meters per Second	Cell Location	Fixed Cell Size	1	1 Meters	With First Entity	<input type="checkbox"/>	<input type="checkbox"/>	40 Yards		

**Figure 12.22: Specifying Properties for all Five Conveyors of the TransferLine (Spreadsheet View)**

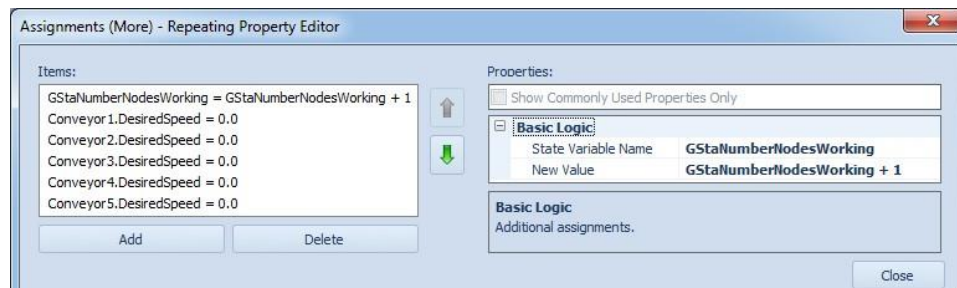
**Step 6:** Since the transfer line consists of multiple conveyors rather than a continuous one, the conveyors have to be synchronized so that when parts enter an on/off location on the transfer line, the entire line which is a set of conveyors stops to await the on/off operation.<sup>166</sup> To do this, first define a new `DISCRETE STATE VARIABLE` of type integer (from the *Definitions* tab) for the **Model** named **GStaNumberNodesWorking**. This variable will represent the number of on/off locations that are currently either putting parts onto or taking parts off the transfer line. When this state variable is zero, the transfer line is running and when this variable is greater than one, the transfer line should be stopped.

**Step 7:** Let's now insert a new "process." From the "Processes" tab, select "Create Process" and name the new process **On\_Off\_Entered**. The process will consist of the steps shown in Figure 12.23.



**Figure 12.23: Process when Entering/Leaving the Transfer Line**

- When a part reaches an on/off operation, the *Assign* step needs to increment the **GStaNumberNodesWorking** by one and all conveyors need to be stopped by setting their desired speed to zero (see Figure 12.24).<sup>167</sup>



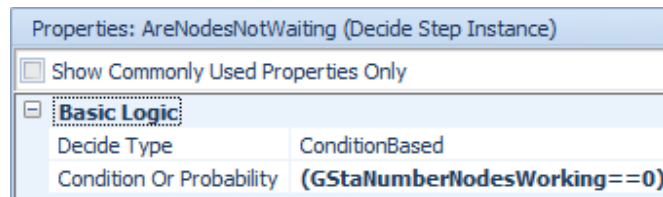
**Figure 12.24: Incrementing the Number Nodes Working and Shutting Down the Conveyors.**

- The *Delay* step will be used to model the on/off time, which is assumed to be uniformly distributed with a minimum of 0.5 and a maximum of 1.5 minutes.
- The second *Assign* step will now decrement the **GStaNumberNodesWorking** by one.
- The *Decide* step is based on the condition that `GStaNumberNodesWorking==0`, meaning no other on/off stations are busy loading/unloading the transfer line.

<sup>166</sup> The synchronization is not exact since the occupied cell of the upstream conveyor is not known to the downstream conveyor and parts from a cell may conflict with parts arriving from the upstream conveyor (i.e., **SrcParts**).

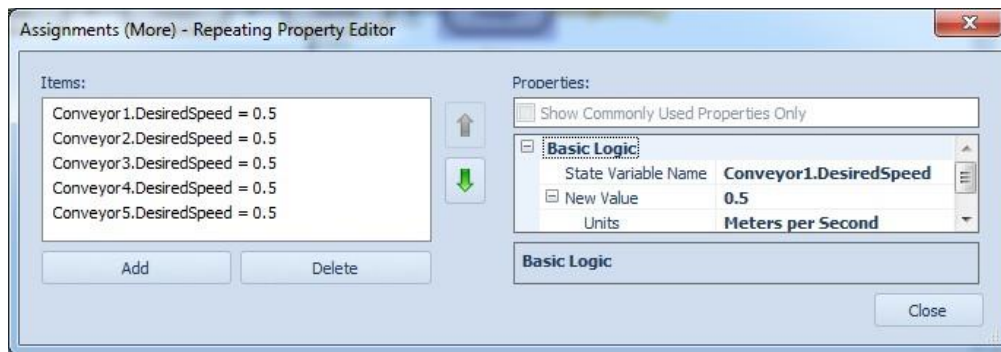
<sup>167</sup> You will need to select the *Assignments (More)* button in the *Assign* step.





**Figure 12.25: Decide Step Option**

- The last *Assign* step will turn the conveyors back on. SIMIO stores all the times internally in meters per hour. If the state variable has units these can be specified in the assignment. Make sure you change the units to meters per second or an assignment of 0.5 would be treated as 0.5 meters per hour.<sup>168</sup> You can convert the original specification of 0.5 meters per second to 1800 meters per hour to be consistent with how SIMIO will interpret the assignment and avoid having a conversion each time. Figure 12.26 shows the conveyors being turned back on.



**Figure 12.26: Turning the Conveyors Back On**

**Step 8:** Finally, the **On\_Off\_Entered** generic process needs to be invoked appropriately (i.e., every time a part is loaded or unloaded to the transfer line). Select the **On\_Off\_Entered** as the add-on process trigger for all the nodes and one path (i.e., the path to the exit **SnkPartsLeave**) as specified in Table 12.1.

**Table 12.1: All the Add-On Process Triggers that Need to Go Through the On/Off Process Logic**

Add-On Process Trigger	Which Object the Trigger is Associated
<i>Exited</i>	<b>Output@SrcParts</b> (TRANSFERNODE)
<i>Entered</i>	<b>Input@StationA</b> (BASICNODE)
<i>Exited</i>	<b>Output@StationA</b> (TRANSFERNODE)
<i>Entered</i>	<b>Input@StationB</b> (BASICNODE)
<i>Exited</i>	<b>Output@StationB</b> (TRANSFERNODE)
<i>Entered</i>	<b>Input@ SnkPartsLeave</b> (BASICNODE)
<i>Entered</i>	<b>Input@StationC</b> (BASICNODE)
<i>Exited</i>	<b>Output@StationC</b> (TRANSFERNODE)

**Question 14:** Why did we increment and decrement **GStaNumberNodesWorking** by one rather than just setting it to one or zero which would eliminate the need for the *Decide* step?

---

**Question 15:** Why not simply use the **On\_Off\_Entered** process at the BASICNODE of the entry/exit?

---

<sup>168</sup> The presumption of a standard internal time probably represents a choice of efficiency over ease of use, since determining the original time units would represent some additional computational effort.

**Step 9:** Run the simulation for 40 hours at a slow speed to be sure each of the parts follows their sequence appropriately and answer the following questions.

**Question 16:** How long are all part types in the system (enter to leave)?

---

**Question 17:** What is the utilization of each of the `SERVERS`?

---

**Question 18:** What is the average number on the **Conveyor2** link?

---

**Question 19:** What is the average time on the link for **Conveyor3** link?

---

## Part 12.5: Machine Failures in the Cell

Failures are a part of most mechanical and other systems. SIMIO provides methods to model these failures within the “*Reliability Logic*” section of the `SERVER` (and other objects). In the case of the transfer line, we also need to shut down the line while a machine is in the state of being repaired.<sup>169</sup> Let’s assume that **SrvStationC** is the unreliable machine. In particular, we will assume that the MTBF (Mean Time Between Failures) is exponentially distributed with a mean of ten hours, while the MTTR (Mean Time To Repair) is modeled with a Triangular distribution (0.5, 1, 2) hours.

**Step 1:** In the “*Reliability Logic*” section for the Station C `SERVER`, specify the *Failure Type* to be “*Calendar Time Based*”, the *Uptime Behavior* be `Random.Exponential(10)` with units of hours, and the *Time To Repair* should be `Random.Triangular(0.5, 1, 2)` with units of hours as seen in Figure 12.27. Remember that the server is failing and that means that all the capacity associated with this server fails (the capacity for this server is scheduled, but it changes between 1 and 2).

<input checked="" type="checkbox"/> <b>Reliability Logic</b>	
Failure Type	<b>Calendar Time Based</b>
<input checked="" type="checkbox"/> Uptime Between Failures	<b>Random.Exponential(10)</b>
Units	Hours
<input checked="" type="checkbox"/> Time To Repair	<b>Random.Triangular(0.5,1,0,2)</b>
Units	Hours

**Figure 12.27: Specifying a Failure and Repair**

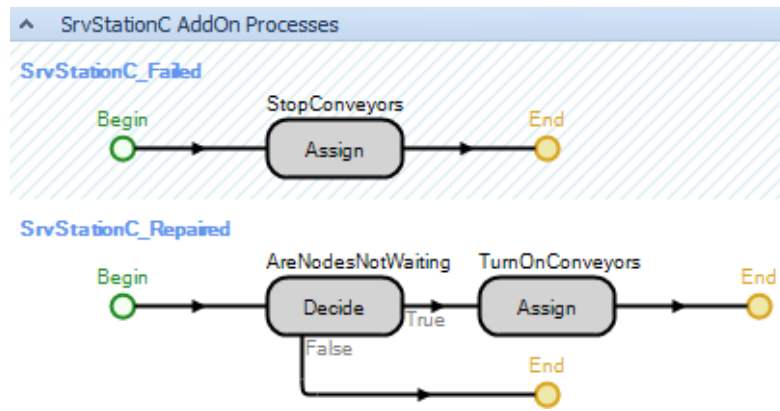
**Step 2:** When the **SrvStationC** fails, the entire transfer line should be shut down and then started back up once the station has been repaired. Select **SrvStationC** and create new Add-On Process Triggers, “*Repairing*” and “*Repaired*” to produce new processes **SrvStationC\_Failed** and **SrvStationC\_Repaired**. The “*Failed*” trigger will be invoked at start of the failure while the “*Repaired*” trigger will be invoked after the `SERVER` has been repaired. In **SrvStationC\_Failed**, use an *Assign* step to turn off all of the conveyors as by setting their `DesiredSpeed` to zero. In the **SrvStationC\_Repaired**, use an *Assign* step to turn back on the conveyors if there is currently no nodes loading/unloading (i.e., setting the `DesiredSpeed` = 0.5 meters per second for all conveyors) as seen in Figure 12.28.<sup>170</sup>

---

<sup>169</sup> The tremendous cost of downtime due to everything on the line becoming idle is one of the reasons why transfer lines and similar highly automated processes are becoming relatively rare.

<sup>170</sup> Copy the *Decide* and *Assign* steps from the **On\_Off\_Entered** process and remove the increment of the **GStaNumberNodesWorking**.





**Figure 12.28: Failed and Repairing Process Triggers**

**Step 3:** Also, the **On\_Off\_Entered** process should not start up the transfer line if the **SrvStationC** has failed and is currently being repaired. To do this change the *Decide* step in the **On\_Off\_Entered** process so the conditional expression becomes the following.<sup>171</sup>

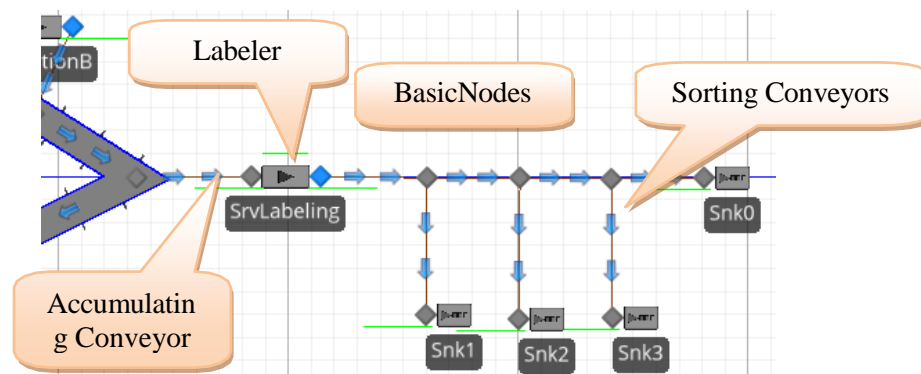
`(GStaNumberNodesWorking==0) && (SrvStationC.Failure.Active==False)`<sup>172</sup>

**Step 4:** Note that the **SrvStationC** will change to the color red when it is down, indicating that the server (and all its capacity) is in the “Failed” state. Run the model and make sure the transfer line stops when the station is also under repair.

**Question 20:** Is there anything about the final animation that troubles to you?

## Part 12.6: Sorting Conveyors

Now we find out that the output from the conveyor system is another system of conveyors which sorts the parts into destinations. The parts that exit the manufacturing cell are routed to an accumulating conveyor at a labeling machine. From the labeling machine, the parts are conveyed to one of four sinks corresponding to the specific part type as seen in Figure 12.29. The four conveyors directly in front of the sinks are accumulating conveyors. These conveyors keep moving and parts are allowed the queue, although in this case the sinks pose no barrier to exit. The rest of the conveyors are non-accumulating.



**Figure 12.29: Labeling and Sorting Conveyors**

<sup>171</sup> `Object.Failure.Active` function will return true if the current object has failed.

<sup>172</sup> Notice that SIMIO utilizes C# logical operators (i.e., `==` for equality, `&&` for “And” statement, and `||` for the “Or” statement).









**Step 1:** Remove the original **SnkPartsLeave** and replace it with a **SrvLabeling**. The labeling processing time follows a Triangular(1,2,3) minutes and has a capacity of “1.” Since the labeler will be supplied by an accumulating conveyor from the inside set of conveyors (i.e., the **BASICNODE**), set the *Input Buffer* capacity property to zero because it is not needed and there is no delay in off-loading to this conveyor.

**Step 2:** Add three **BASICNODES** to follow the **SrvLabeling** and name them **Basic1**, **Basic2**, and **Basic3**, in that order. These are conveyor diverters that will send the sorted parts to their destinations. Add three sinks, **Snk1**, **Snk2**, **Snk3**, and **Snk0** to represent the destinations for parts of type 1, 2, 3, and 0 respectively. Connect up the objects with conveyors according to Table 12.1.

**Table 12.2: Conveyor Properties**

From/To	Conveyor Type	Conveyor Length	Conveyor Speed
BasicNode4 to SrvLabeling	Accumulating	10 meters	1.0 meters per second
SrvLabeling to Basic1	Non-Accumulating	10 meters	0.5 meters per second
Basic1 to Snk1	Accumulating	10 meters	0.1 meters per second
Basic1 to Basic2	Non-Accumulating	10 meters	0.5 meters per second
Basic2 to Snk2	Accumulating	10 meters	0.1 meters per second
Basic2 to Basic3	Non-Accumulating	10 meters	0.5 meters per second
Basic3 to Snk3	Accumulating	10 meters	0.1 meters per second
Basic3 to Snk0	Accumulating	10 meters	0.1 meters per second

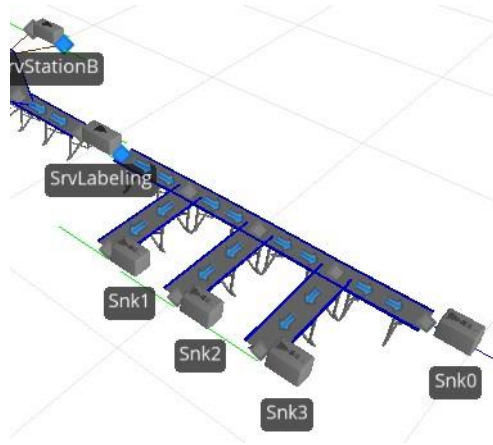
**Step 3:** For the part’s. sequence table, replace the **Input@SnkPartsLeave** entries with **Input@SrvLabeling**. Next add the sink node destinations associated with the part types. Figure 12.30 shows part of the table and you can insert the last placement at the end to maintain the sequence.

10	Input@SrvLabeling	0	 2
11	Input@SrvStationB	Random.Pert(5,9,11)	 3
12	Input@SrvStationC	Random.Triangular(6,9,11)	 3
13	Input@SrvLabeling	0.0	 3
14	Input@Snk0	0.0	 0
15	Input@Snk1	0.0	 1
16	Input@Snk2	0.0	 2
17	Input@Snk3	0.0	 3

**Figure 12.30: Changes to the Sequence Table**

Next change the **Output@SrvLabeling** node *Entity Destination* to *By Sequence*. There is no need to add intermediate nodes (the **BASICNODES**) since SIMIO can find the designations from the **SrvLabeling**.

**Step 4:** Select all the conveyors and add the conveyor path decorators as seen in Figure 12.31.



**Figure 12.31: Conveyor with Path Decorators**

**Step 5:** Save and run the simulation for a while observing the animation which may need slowing down.

**Question 21:** Are the parts being conveyed as expected and do you see accumulation at the labeler?

---

**Step 6:** Run the simulation for 40 hours.

**Question 22:** What is the average number of parts that accumulate on the conveyor in front of the labeler?

---

**Question 23:** What is the Utilization of the labeler?

---

**Question 24:** Why does the `OUTPUTBUFFER` of the labeler have content?

---

**Question 25:** What is the average number on the conveyor between **Basic2** and **Basic3**?

---

**Question 26:** Do you have any concerns about the operation of the sorting conveyor system?

---

## Part 12.7: Commentary

- Material handling can be a substantial cost of production and thus is of great interest to many companies. Vehicles and conveyors provide considerable flexibility in modeling material handling.
- The modeling of cranes has not been included in this chapter. These are a complex category of materials handling. However, SIMIO has developed a special library for cranes that can represent a wide variety of overhead and floor cranes. In particular, there is a 3D animation of the cranes to reflect the fact that cranes operate in three dimensions. This package is highly recommended.