MathWorks Hands-on Workshop

Discrete Event System Modelling with SimEvents

The aim of this workshop is to give you hands on experience modelling a discrete event system using Simulink and SimEvents. The example used throughout this workshop is a lunch line queue. Although this is a simplistic example and is likely not related to your specific application, the modelling concepts can be applied and adapted to a range of different scenarios. At the end of this workshop, you should be able to start building your own models using the knowledge you have gained from these exercises. This workshop should ideally be run under the supervision of a MathWorks engineer, however, it may also be used as a standalone tutorial.

This workshop requires the following products:

- MATLAB **(R2016a or later is required)**
- Simulink
- SimEvents
- Statistics and Machine Learning Toolbox

This workshop is not compatible with R2015b and earlier releases of MATLAB and SimEvents.

Pre-requisite Knowledge

A basic knowledge of MATLAB is highly recommended before starting this workshop. If you have never used MATLAB before, or you could do with a refresher, please complete the MATLAB Academy 2 hour on-ramp course.

https://matlabacademy.mathworks.com/

Additional Videos to help you get started can be accessed from the MathWorks website:

https://au.mathworks.com/products/matlab/videos.html

Although the above resources are helpful, it is highly recommended to enrol in a MATLAB Fundamentals training course:

https://au.mathworks.com/training-schedule/matlab-fundamentals

Getting Started with Simulink

The first thing you need to do is start Simulink. You can then open a blank model as well as the Simulink library browser. The library browser contains the block libraries you will use to build the models throughout this workshop. To begin, execute '***simulink***' at the MATLAB command prompt.

>>simulink

Alternatively, you can **click on the 'Simulink Library' button** (Figure 1) in the toolstrip under the 'Home' tab.

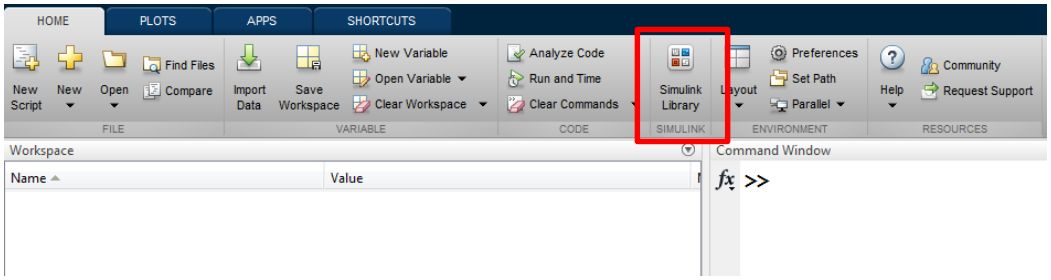

Figure 1: Simulink Library button

The Simulink Start Page will open in a separate window. This allows you to open pre-configured Simulink models for different types of applications. For this workshop, **click on the "Blank Model"** as shown in Figure 2.

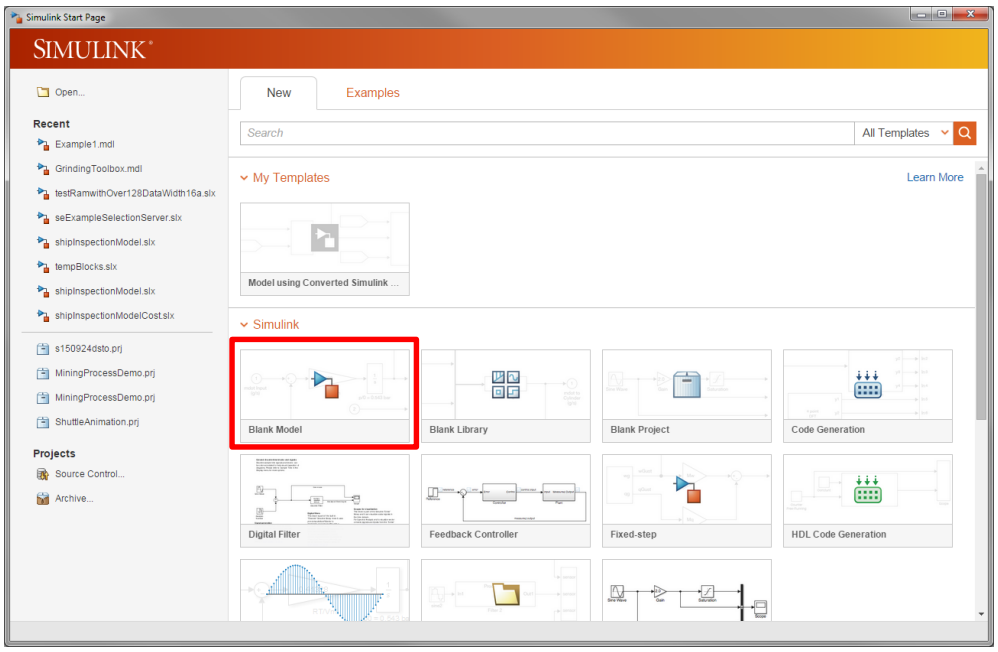

Figure 2: The Simulink Start Page

A blank model called ‘untitled’ will appear to replace the Simulink Start Page. We can use this blank canvas to layout our models by placing and connecting Simulink blocks. Simulink blocks encapsulate algorithms that you can use to graphically develop dynamic models and then simulate over time. These blocks reside in the Simulink library browser.

To open the library browser, **click on the library browser button** as shown in Figure 3, and the library browser should appear as shown in Figure 4. The browser contains a tree structure of libraries which will differ depending on which products you are licenced to access. In this workshop, you will only use the Simulink library and the SimEvents library. Ensure that you have access to these two libraries before continuing.

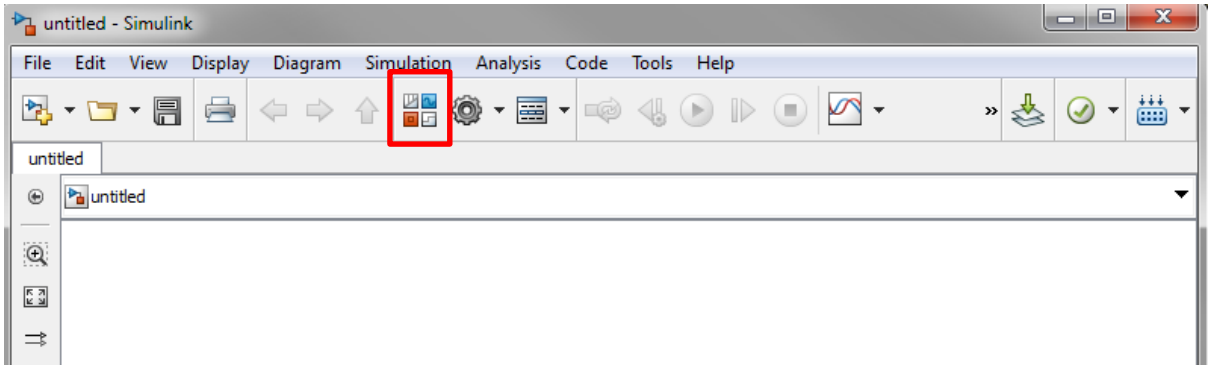

Figure 3: Open the library browser

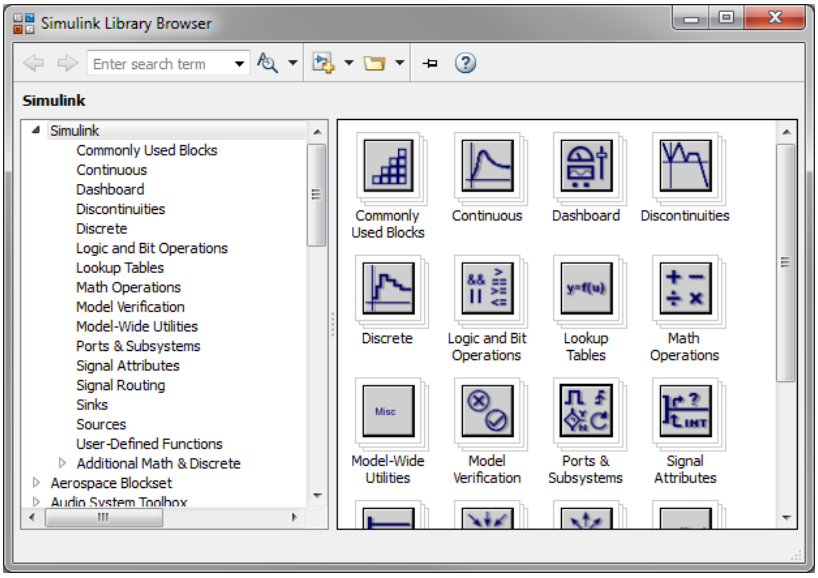

Figure 4: The Simulink Library Browser

Exercise 1: Generating Entities

In this first example you will start with a very basic model that generates entities. In SimEvents, the term "entity" is used to describe some physical thing that is moving through your simulation. In the case of the lunch line, the entity is the customer. In a data network simulation, the entity could be a packet of data.

Instructions:

1. Drag and drop the blocks from Table 1 into your model, i.e., the blank canvas in the Simulink editor. To drag and drop, click on the block with your left mouse button, hold the button down, drag the block onto the blank canvas, then release the mouse button.
2. To configure the parameters, double click on the corresponding block to open the block parameters dialog.
3. Connect up the blocks as shown in Figure 5. To connect the blocks, click on the output port of the Entity Generator with the left mouse button, hold the button down, drag the wire to the input port of the Entity Terminator, then release the mouse button. Repeat to connect the entities arrived 'a' port to the Scope block.
4. Click on the Run button to start the simulation. (Highlighted in red box in Figure 5). Double click the scope to view the arrived entities as shown in Figure 6.

Tips:

- You can quickly connect blocks using the CTRL key. First click on the block with the output port (e.g. Entity Generator), hold the CTRL key, then click on the block with the input port (e.g. Entity Terminator).
- You can zoom in and out of the model using your scroll button on the mouse
- You can automatically fit the model to the window by hitting the space bar
- You can search for blocks without using the library browser, by clicking on the canvas and typing the block name

Block Name	Library	Quantity	Parameters
Entity Generator	SimEvents	1	Default
Entity Terminator	SimEvents	1	**Statistics Tab:** Check: Number of Entities arrived, a
Scope	Simulink\Sinks	1	Default

Table 1

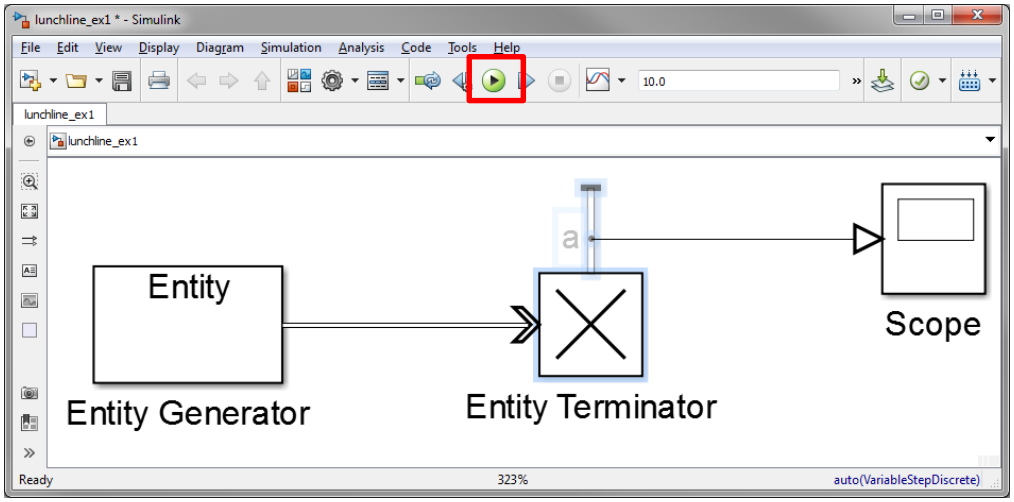

Figure 5: Exercise 1

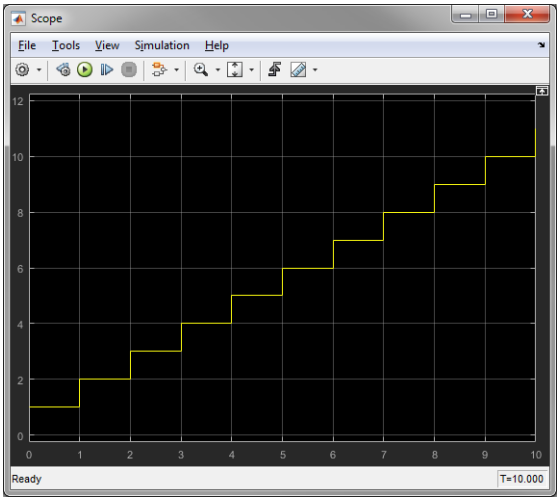

Figure 6: Exercise 1 scope display

Examining the Results:

The scope counts the number of entities that have arrived at the Entity Terminator block. Notice that entities (customers) are generated periodically from the start of the simulation. This is not very realistic for a lunch line. You would expect customers to arrive non-deterministically, but with some stochastic distribution. You will alter the model in the next exercise to model this behaviour.

Now look at the wires that connect the blocks. Notice that there are two different types of wires. The double lined arrow represents the movement of an entity (customers). The single lined arrow with empty arrow head represents an event based signal. The value of an event based signal is updated based on an events, e.g. increments when an entity arrives at the block. This is different to a time based signal whose value is updated based on time, e.g. y = sin(t). Time based signals in Simulink are denoted by a solid arrow head.

Exercise 2: Including randomness in the model

TIP: You can open lunchline_ex1.slx if you wish to start from this exercise.

In this exercise, you will modify the block parameters in the previous exercise to model the random arrival of customers in our lunch line.

Instructions:

1. Double click on the Entity Generator block. This opens the dialog to edit the block parameters. For more information, click on the 'Help' button to open the documentation for this block. The block is currently setup to generate entities with a constant intergeneration time of period = 1. This explains the scope display from the previous exercise.
2. Uncheck the 'Generate entity at simulation start' box. This will prevent the simulation from generating a customer at the very start of the simulation (at time = 0).
3. Change the 'Time Source' parameter to 'MATLAB Action' by using the dropdown menu. This allows you to write MATLAB code to control the entity intergeneration time used by this block. This intergeneration time must be assigned to a variable called 'dt'. By default, the block assigns 'dt' from a uniform random distribution.
4. In the "Intergeneration time action" box, change the MATLAB action to define 'dt' from an exponential distribution with mean of 0.5, as shown in Figure 7.
5. Click OK to accept the change and close the Block Parameters dialog box.
6. Click Run to start the simulation and you should see the scope as shown in Figure 8.

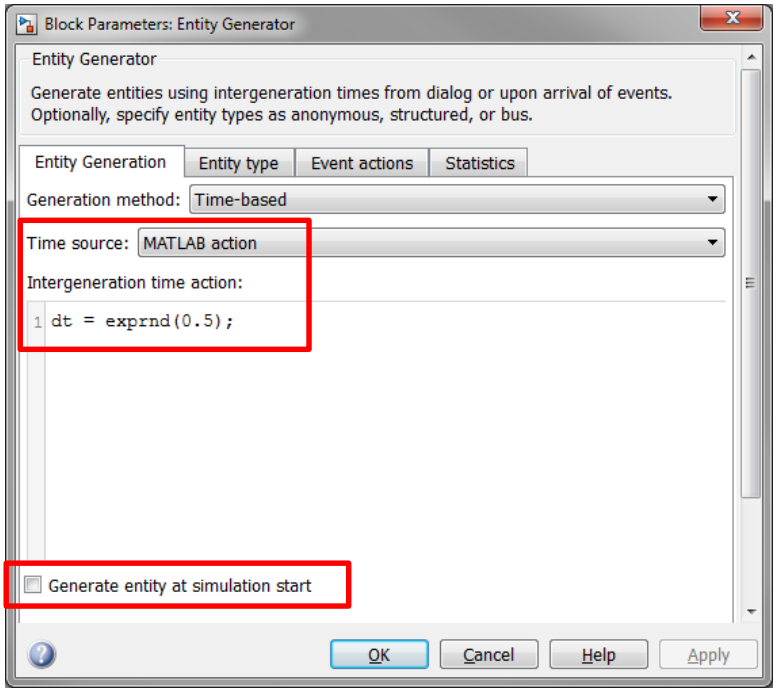

Figure 7: Non-deterministic Intergeneration Time

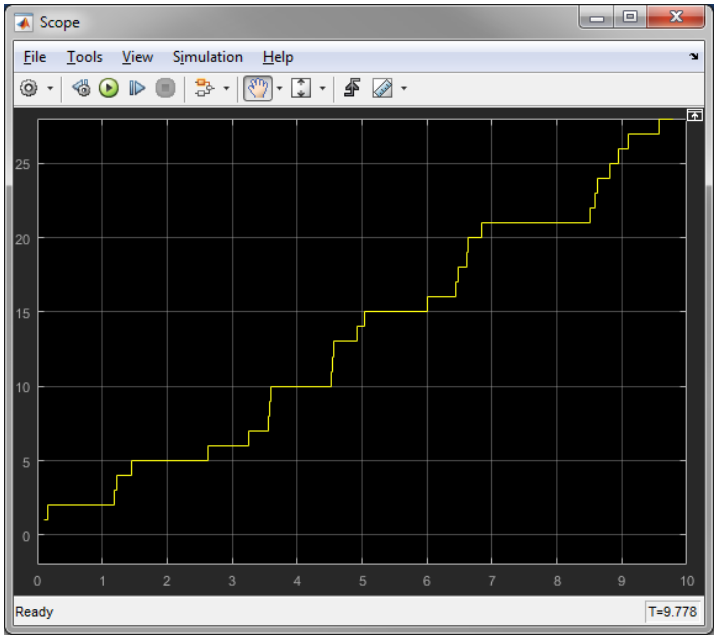

Figure 8: Exercise 2 scope display

Examining the Results:

The scope now shows randomness with respect to the arrival of customers at the entity terminator. You may also have noticed that the display lasts for 10 time units. This is determined by the simulation stop time, which is defined in the Simulink Editor (right of the run button). So what are the units of time? The units are what you define the time to be, but the key is to be consistent. If you want the units to be in minutes, then you must ensure that you define all the parameters in units of minutes.

For example, when you defined the mean of the random number generator to be 0.5, this is 0.5 minutes and the simulation stop time is 10 minutes. If you wanted the same simulation results with the units in seconds, you would have to change the mean parameter to 0.5x60 = 30. But you would also have to change the simulation stop time to 10x60 = 600. We will assume a time unit of minutes for the remainder of this workshop.

Other Resources (for additional information):

Open the MATLAB documentation (exceute >>doc) and click on:

1) SimEvents -> Examples -> Tutorials: Generating and Initializing Entities
2) SimEvents -> Modeling -> Entity Generation

Exercise 3: Modelling the lunch line server

TIP: You can open lunchline_ex2.slx if you wish to start from this exercise.

In this exercise you will include the lunch line server. If you consider this to be a queue to order fresh sandwiches, think of this as the sandwich maker. It basically adds a delay into the system. For this example you will model 2 sandwich makers.

Instructions:

1. Uncheck the "Number of entities arrived" box from the Statistics tab in the Entity Terminator block
2. Delete the red unconnected wire from the Scope block (click on the wire and hit delete).
3. Delete the entity wire that connects the Entity Generator to the Scope.
4. Add and configure the blocks from Table 2, then connect as shown in Figure 9.
5. Change the simulation stop time to 90.
6. Run the simulation and you should see the scope as shown in Figure 10. This scope displays the length of the queue during the simulation.

Block Name	Library	Quantity	Parameters
Entity Queue	SimEvents	1	**Statistics Tab:** Check: Number of entities in block, n.
Entity Server	SimEvents	1	**Main Tab:** Capacity: 2 Other: Default

Table 2

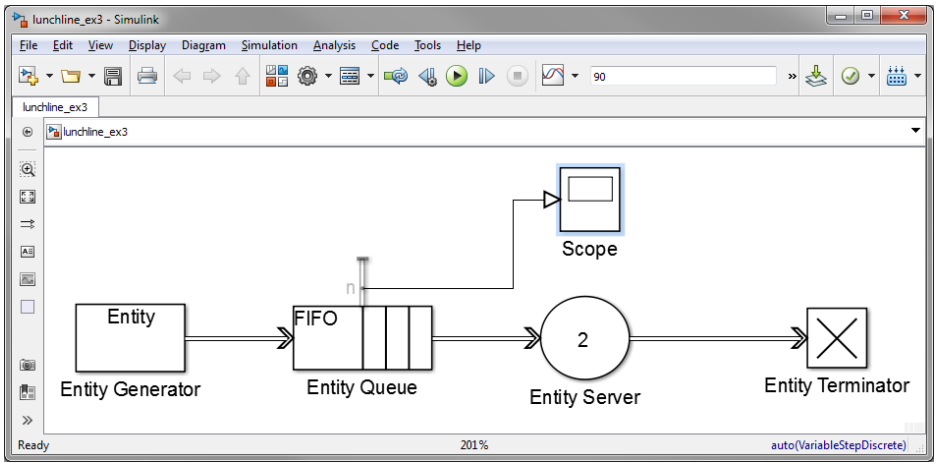

Figure 9: Exercise 3

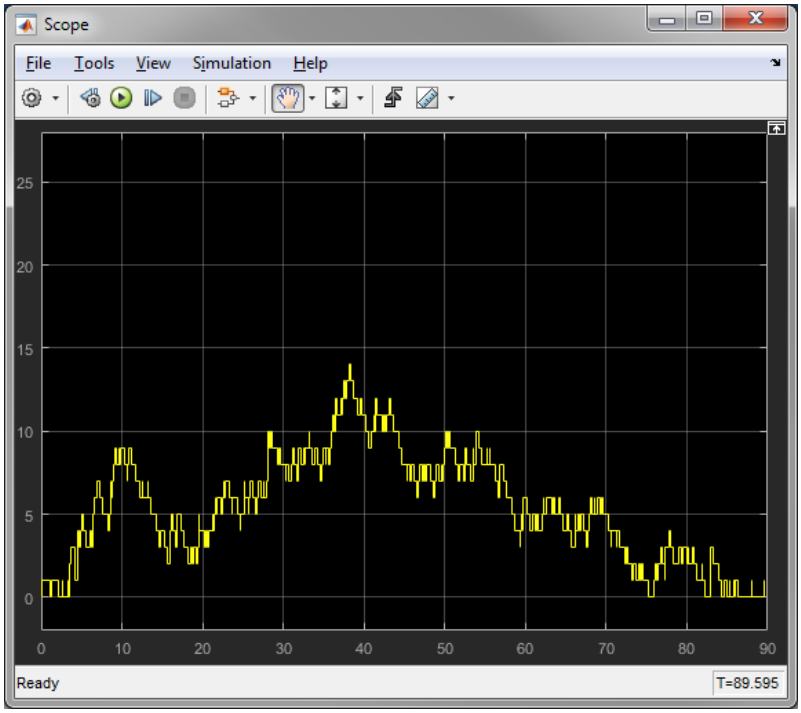

Figure 10: Exercise 3 queue length

Examining the Results:

The scope now shows the length of the queue at any time during the simulation. This was enabled within the statistics tab from the Entity Queue block, much like we did earlier with the Entity Terminator block. Many SimEvents blocks have a statistics tab that allows you to output different statistics from the different blocks. This can be very useful to analyse your simulation results.

You may have also noticed that this simulation is currently setup to model a constant service time of 1 minute. Again, this is not very realistic. You would expect a non-deterministic time to serve each customer. You will modify this part of the model in the next exercise.

Other Resources (for additional information):

Open the MATLAB documentation (execute >>doc) and click on:

1) SimEvents -> Examples -> Tutorials: Comparing Queuing Strategies
2) SimEvents -> Modeling -> Queuing
3) SimEvents -> Modeling -> Resources

Exercise 4: Attributes and Blocking

TIP: You can open lunchline_ex3.slx if you wish to start from this exercise.

In this exercise you will define the non-deterministic length of time each entity spends in the server block, i.e., the length of time it takes to make the sandwich. It is possible to assign the time directly within the Entity Server block by changing the "Service time source" to "Signal port" or "MATLAB Action" from the drop down menu. In this exercise, however, you will define the time using an attribute. An "attribute" is simply numeric data that is attached to, and travels with, the entity.

Instructions:

1. Double click on the Entity Generator block and select the "Entity Type" tab. Make the changes as shown in Figure 11.
2. Now select the "Event actions" tab. Here you can define actions immediately after entities are generated, or immediately before entities exit the block. We want to define an action to generate a ServiceTime value after each entity is generated, so ensure that "Generate" is highlighted in the "Event Actions" window.
3. The action is expressed using MATLAB code. Attribute values can be accessed using dot notation. With this in mind, make the changes as shown in Figure 12.
4. Since we are defining the service time as an attribute, we now need to let the Entity Server block know to use this attribute. Double click on the Entity Server block, make the changes as shown in Figure 13
5. Run the simulation and observe the results as shown in Figure 14.

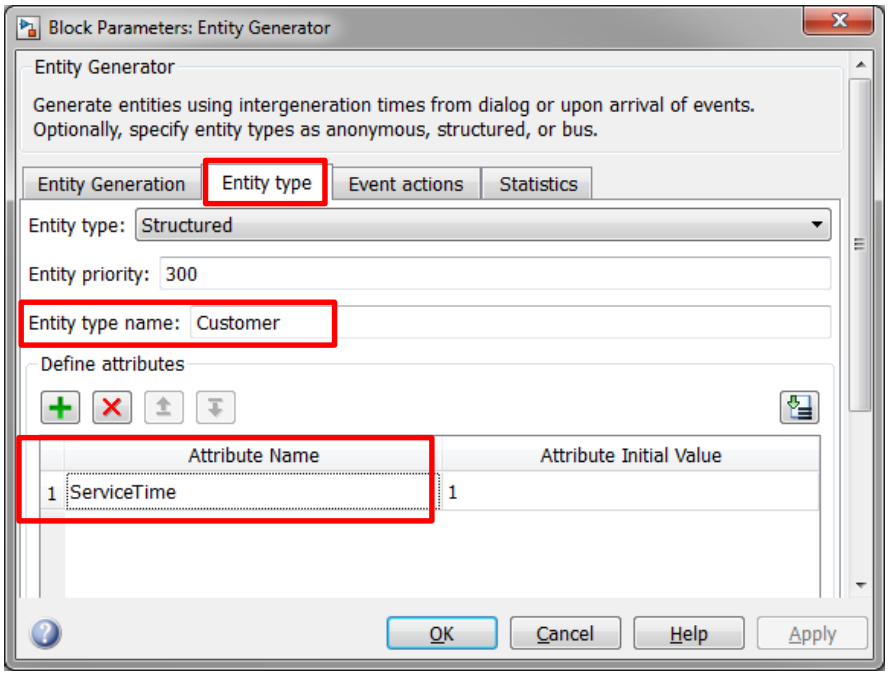

Figure 11: Define the Entity Type and Attributes

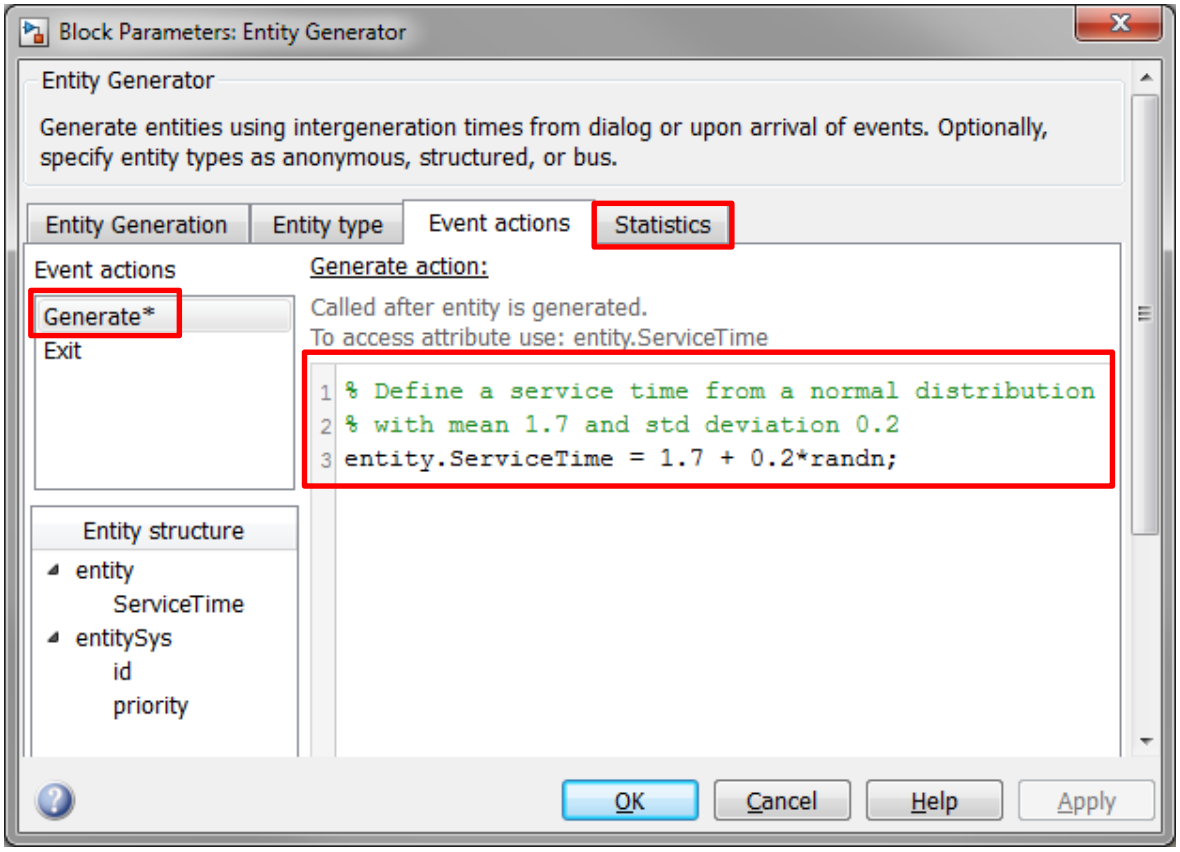

Figure 12: Define attribute values after entity generation

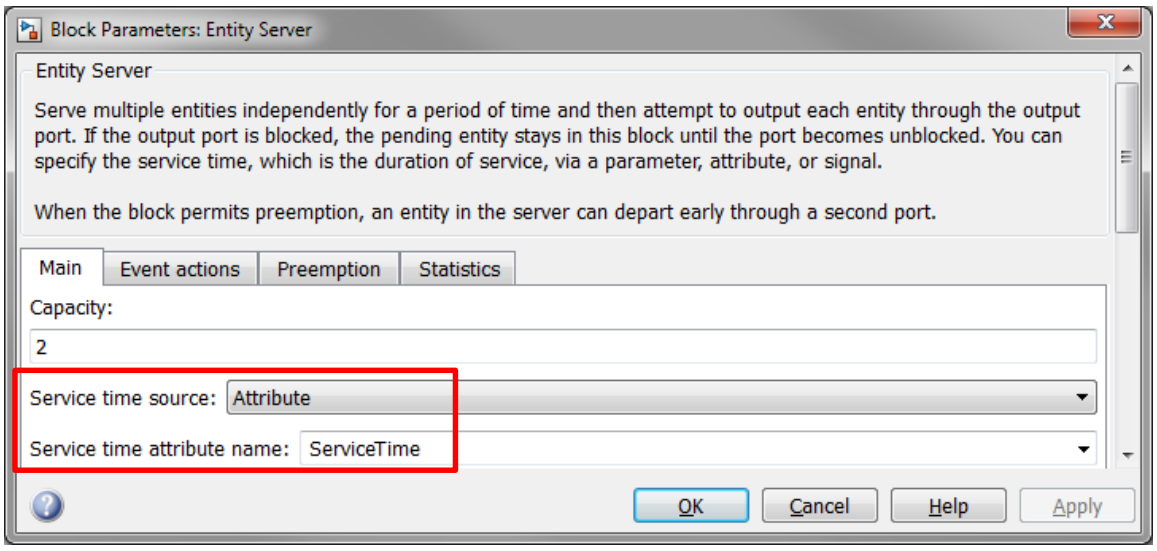

Figure 13: Setup the server block to use an attribute for service time

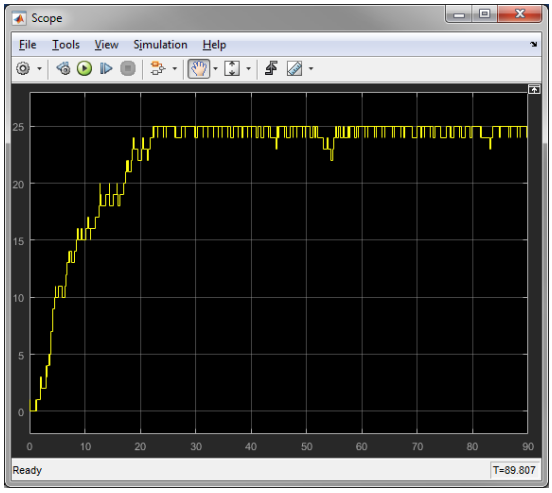

Figure 14: Exercise 4 queue length

Examining the Results:

The server now introduces a non-deterministic delay (making the sandwich) based on the stochastic distribution you used when assigning the value of the ServiceTime attribute in the Entity Generator block. Notice that the queue looks like it is saturating at 25. This is because the Entity Queue block has a default capacity of 25 (double click on the Entity Queue to see this parameter).

When the queue is full, this path in the simulation becomes "blocked". This has implications upstream in the simulation, specifically at the Entity Generator. When the path becomes blocked, the Entity Generator pauses, stores the pending entity, then immediately restarts when the path is no longer blocked. You can see view the pending entity by checking the "Pending entity present in block, pe" option as show in Figure 15.

Since the queue grows too quickly, try increasing the number of servers to handle the number of customers. Increase the number of servers to 3, and then rerun the simulation. The queue is still quite large. Try increase the number of servers to 4 instead. The queue now looks manageable.

Also notice on the **{…}** graphic on the lower left corner of Entity Generator block. This lets you know that an event action has been defined within the block. Hover your mouse over this graphic to see a preview of the action.

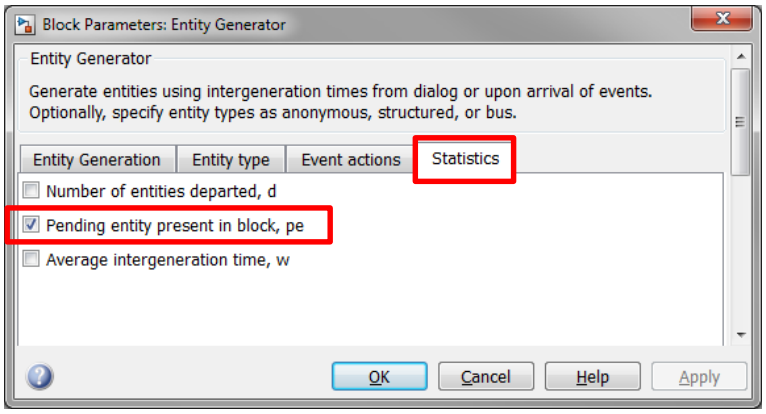

Figure 15: Output signal to indicate a pending entity in the Entity Generator block

Exercise 5: Multiple Paths

TIP: You can open lunchline_ex4.slx if you wish to start from this exercise.

In this exercise you will extend the lunch line model to have 2 separate lunch stations, one for sandwiches and one for hot food. Each customer, therefore, must have a lunch preference in order to join the appropriate queue. We will assume that the probability of choosing sandwiches or hot food is 50/50 (equiprobable). You will also learn about a modelling concept called subsystems. Subsystems can help "clean up" a model aesthetically by adding hierarchy to the model.

Instructions:

1. Build the model shown in Figure 16 by adding the extra block from Table 3. Since the hot food station is a replica of the blocks for the sandwich station, a simple copy and paste can be performed. The top most path will be the sandwich station.
2. Rename the top scope to be "Sandwich Queue". This can be achieved by clicking on the "Scope" text and editing appropriately. Do the same for the lower "Scope1" but rename it "Hot Food Queue".
3. Other modifications required:
    a. Ensure the capacity in each of the Entity Server blocks is set to '2'.
    b. Add an extra attribute in the "Entity Type" tab of the Entity Generator block by clicking on the green '+' button. Set this attribute name as "Lunch" as shown in Figure 17.
    c. Define the value of the new "Lunch" attribute inside the "Event actions" tab of the Entity Generator block as shown in Figure 18. A value of 1 will represent sandwiches, while a value of 2 will represent hot food (to be explained later).
4. Add some hierarchy to the model by creating separate subsystems for the sandwich and hot food stations (see Figure 19). To do this,
    a. Select the Entity Queue, the Entity Server and Scope blocks for the sandwich station (top path) by clicking and dragging a window to highlight these blocks.
    b. Right click on one of the blocks
    c. Select "Create subsystem from Selection".
    d. Resize the block appropriately and change the name of the subsystem to 'Sandwiches' (Click on the text to edit)
    e. Repeat the above steps for the hot food queue and name it 'Hot Food'.
5. Run the simulation and observe the results as shown in Figure 20.

Block Name	Library	Quantity	Parameters
Entity Output Switch	Simevents	1	Number of output ports: 2 Switching criterion: From attribute Switch attribute name :Lunch

Table 3

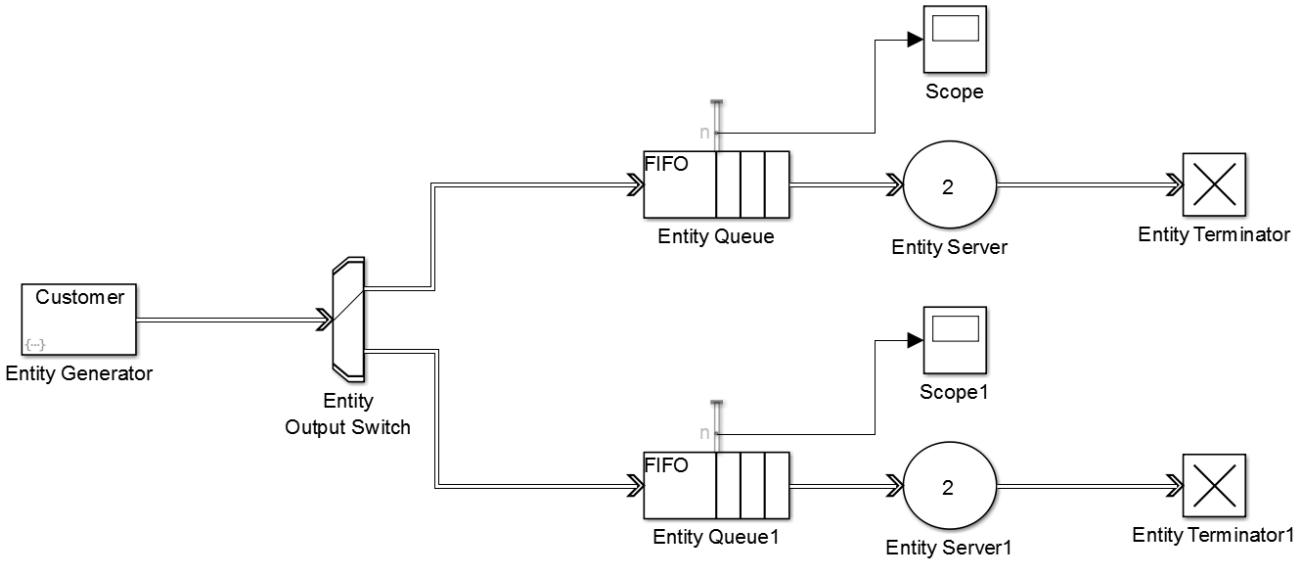

Figure 16: Exercise 5

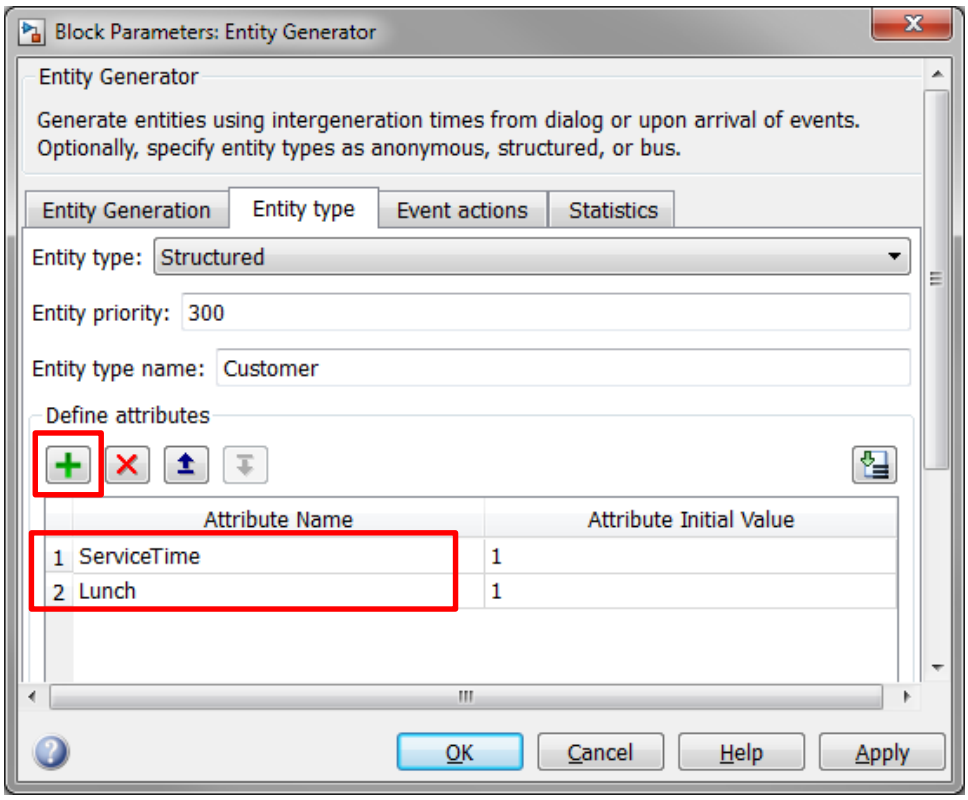

Figure 17: Add an attribute named “Lunch”

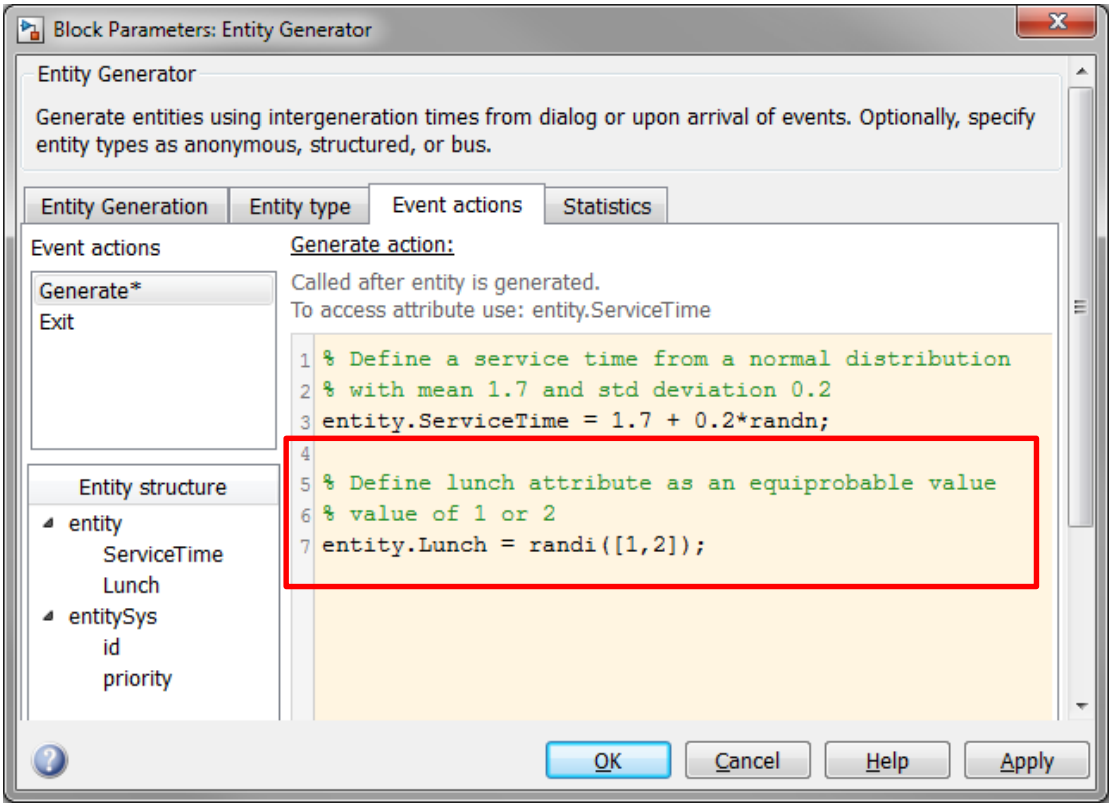

Figure 18: Define the value of the "Lunch" attribute

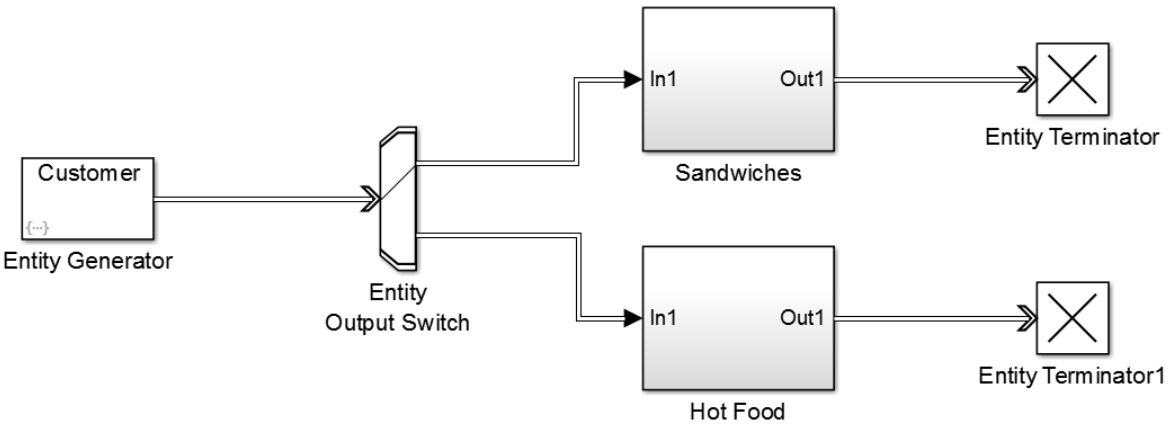

Figure 19: Exercise 5 with subsystems

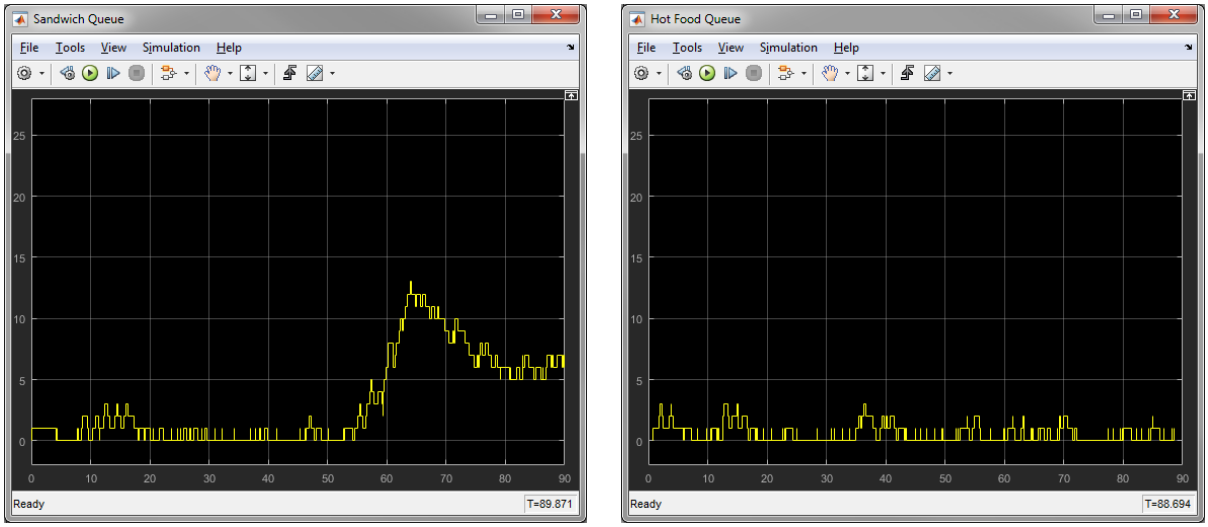

Figure 20: Exercise 5 queue lengths

Examining the Results:

To control the routing of the customers, an attribute ('Lunch') was attached to each entity with a value of 1 or 2. The values are important, because the value dictates the path the customer takes when it reaches the Entity Output Switch block. A value of 1 sends the customer through the first output port and along the top path which is the sandwich station. A value of 2 sends the customer through the second output port and along the bottom path which is the hot food station.

In the results, you now see two separate scopes showing the queue length for the sandwich and hot food stations. The name of the scope (specified in step 2) appears in the title bar of the scope window. This helps you differentiate between the two queues. In this simulation, the sandwich line seems to be the busiest (based on the random values generated for Lunch and ServiceTime).

Tips:

- To look under a subsystem, double click on the subsystem.
- To navigate through the hierarchy of subsystems, or to move back and forward like a web browser, use the blue arrow buttons in the Simulink Editor toolbar.
- You can also open the Model Browser by selecting, **View->Model Browser->Show Model Browser**. This provides a tree view of the hierarchy.

Other Resources (for additional information):

Open the MATLAB documentation (execute >>doc) and click on:

1) SimEvents -> Modeling -> Routing
2) Simulink -> Component-Based Modeling -> (Frequently Viewed Topics) Create a Subsystem
3) SimEvents -> Modeling -> SubSystems

Exercise 6: Routing based on logic

TIP: You can open lunchline_ex5.slx if you wish to start from this exercise.

In this exercise you will add some further complexity to the model. We will make the customers choose the shortest lunch line queue, unless they are equal in length, in which case they will randomly choose a queue. To achieve this, you will need to make use of the signals in the simulation (queue lengths) to calculate the shortest length queue.

This example will introduce the Simulink Function block, which is very useful for performing these event based calculations that are functions of signals in the simulation. It will also introduce the "Goto" and "From" routing blocks to help minimise signal-wire clutter in the model.

Instructions:

We will use the blocks in Table 4 to modify our existing model. The modifications in this exercise are a little more complex than previous exercises, so please follow them carefully step by step.

1. Start with a Goto block and update the Sandwich subsystem as shown in Figure 21. The "Goto Tag" should be set to "q1". Repeat for the Hot Food subsystem, but with the "Goto Tag set to "q2". The "Tag visibility" for both Goto blocks should be set to "global". The Goto block allows you to route the signal to a different part of the model, without having to physically connect a wire.
2. Now add the two From blocks and the Simulink Function block to the top level of the model as shown in Figure 22 (use the blue arrow buttons to navigate to the top level). The From blocks should be configured to use "Goto tag" q1 and q2. The From blocks retrieve the signals from the Goto blocks without having to physically connect them.
3. Edit the prototype function on the Simulink block from **y=f(u)** to be **y = calcShortestQueue()**. Simply click on the green text to edit, and you should end up with a block that looks like Figure 23. You may have to resize the block for aesthetics.
4. Double-click on the Simulink Function block to edit the contents. Add the two In blocks as well as the MATLAB Function block as shown in Figure 24.
5. Double click on the MATLAB Function block and enter the code as shown in Figure 25. When you have finished, click on the "Go to Diagram" button (Figure 26) to update the MATLAB Function block ports and return to the Simulink model.
6. Connect the blocks as shown in Figure 27, and move/resize the blocks as you like.
7. Navigate back up to the top level and rewire the blocks as shown in Figure 28.
8. The final step is to call this Simulink Function from the event action that assigns the value of the Lunch attribute. To do this, double click on the Entity Generator block, and modify the "Generate" action in the Event actions tab as shown in Figure 29.
9. Run the simulation and observe the queue lengths as shown in Figure 30.

Block Name	Library	Quantity	Parameters
Goto	Simulink\Signal Routing	2	Goto Tag: q1 (and q2 for the second block) Tag visibility: global (for both blocks)
From	Simulink\Signal Routing	2	Goto tag: q1 (and q2 for the second block)
Simulink Function	Simulink\User-Defined Functions	1	Default
In1	Simulink\Sources	2	Default
MATLAB Function	Simulink\User-Defined Functions	1	Default

Table 4

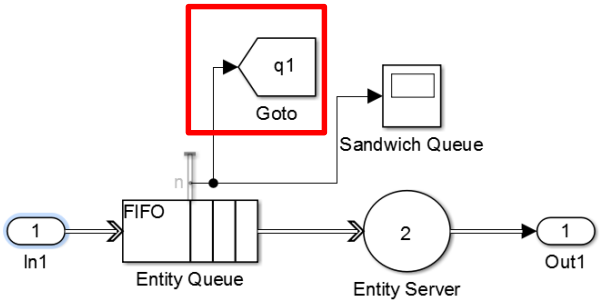

Figure 21: Add a Goto block to the Sandwich subsystem

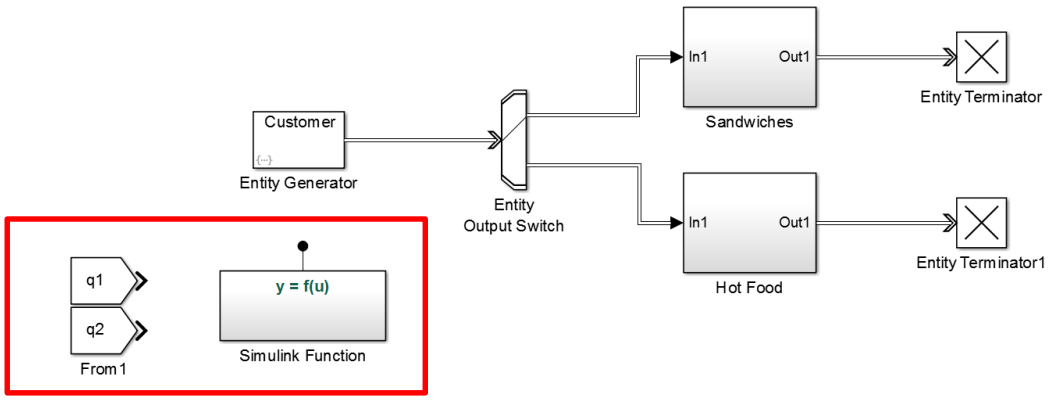

Figure 22: Add the From blocks and Simulink Function block

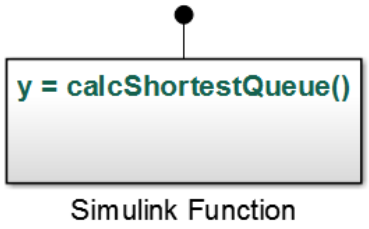

Figure 23: Edit the prototype function in the Simulink Function block

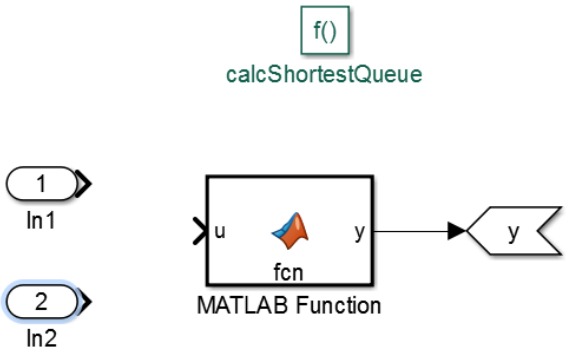

Figure 24: Contents of the Simulink Function block

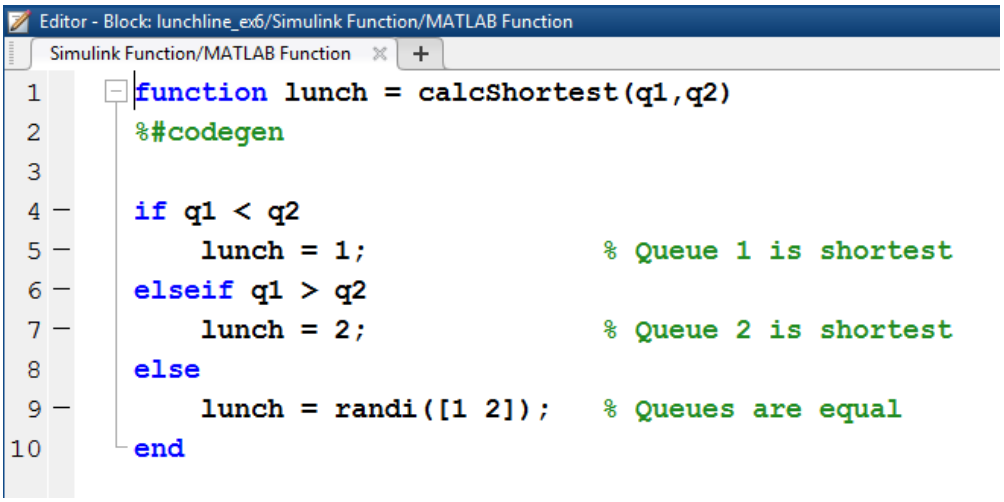

Figure 25: Contents of MATLAB Function block

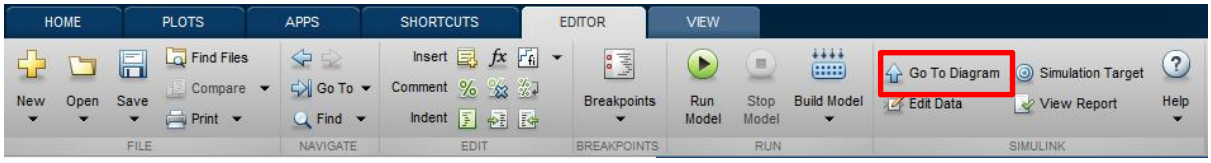

Figure 26: Click Go To Diagram to update MATLAB Function block and return to model

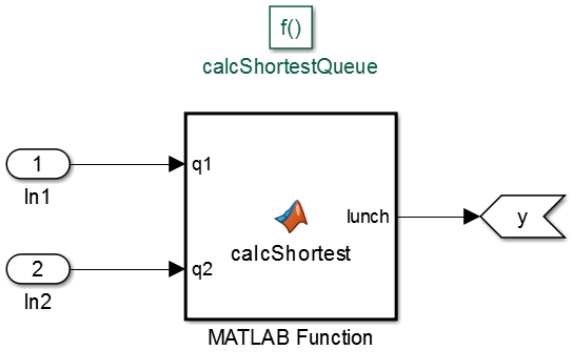

Figure 27: Final contents of Simulink Function block

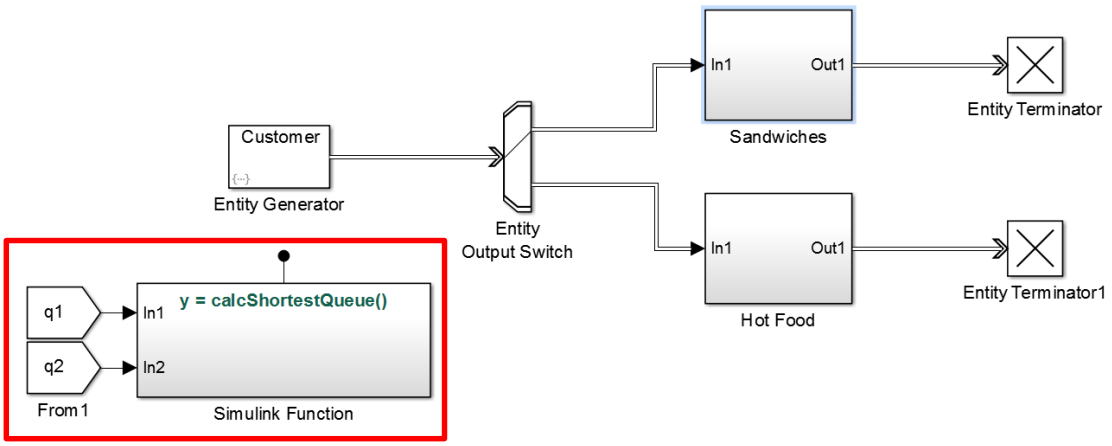

Figure 28: Final look of top level model for Exercise 6

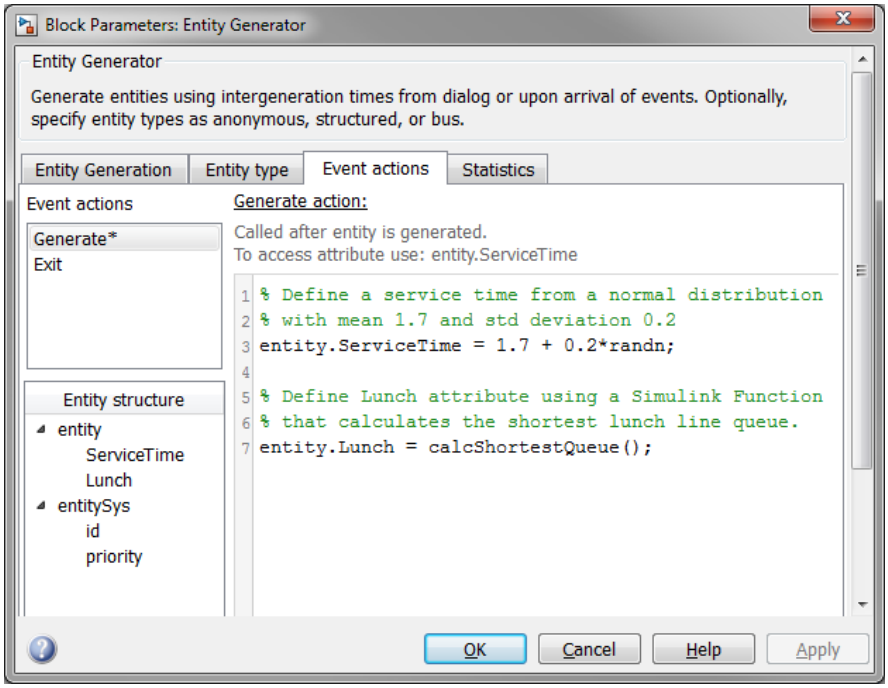

Figure 29: Update the event action that assigns the Lunch value attribute

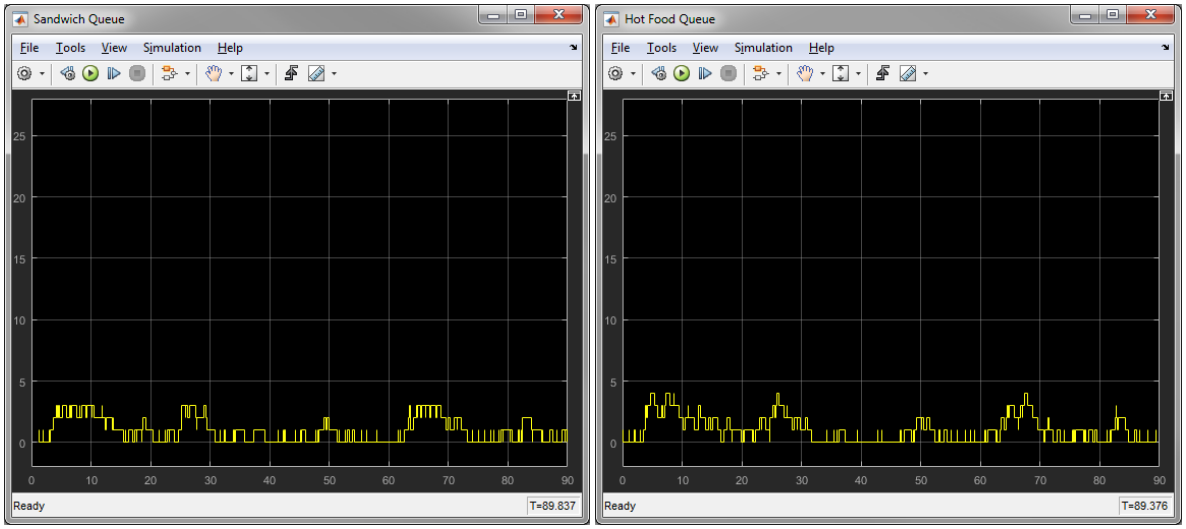

Figure 30: Exercise 6 queue lengths

Examining the Results:

Applying this simple logic of choosing the shortest queue, results in lunch lines that are similar in length. The logic is entirely encapsulated within the Simulink Function.

When an entity (customer) is generated, the Simulink Function is called as part of the event action. The output of the Simulink function is returned as the Lunch attribute value for the current entity.

When the Simulink Function is called, the current length of each queue is passed to the function as inputs (using the From blocks). The MATLAB Function block then assigns a value to its output based on the shortest queue, using a simple if-else statement. This output value becomes the output of the Simulink Function block, and hence, the value of the Lunch attribute. As in the previous exercise, the value of the Lunch attribute is used to determine the output port of the Entity Output Switch.

Tips:

- Click on **Display->Function Connectors** to highlight a link between the Simulink Function block, and the block that calls the Simulink Function.

Other Resources (for additional information):

Open the MATLAB documentation (execute >>doc) and click on:

1) SimEvents -> Modeling -> Events -> Events and Event Actions
2) SimEvents -> Examples -> Tutorials -> Comparing Queuing Strategies

Exercise 7: Examining events with the Message Viewer

TIP: You can open lunchline_ex6.slx if you wish to start from this exercise.

In this exercise we will investigate the Message Viewer. The Message Viewer window acts like a sequence diagram to help visualise the movement of entities between blocks. Understanding how the entities move within the simulation is helpful when analysing and debugging your model.

Instructions:

1. Add the Message Viewer block (located in the SimEvents library) to the top level of your model as shown in Figure 31.
2. Run the simulation.
3. Double click the Message Viewer block to open the Message Viewer Window.
4. Rearrange the blocks at the top of the window into the order as shown in Figure 32 (or any order you wish). You can expand the Sandwiches and Hot Food subsystem blocks by clicking on the + sign.
5. Navigate to the first event in the timeline by clicking the "Go to First Event" button, which is highlighted in red in Figure 32.
6. Examine the output of this window and compare to the scopes displaying the Sandwich and Hot Food queue lengths. Pay particular attention to the times to verify that the events are identical.

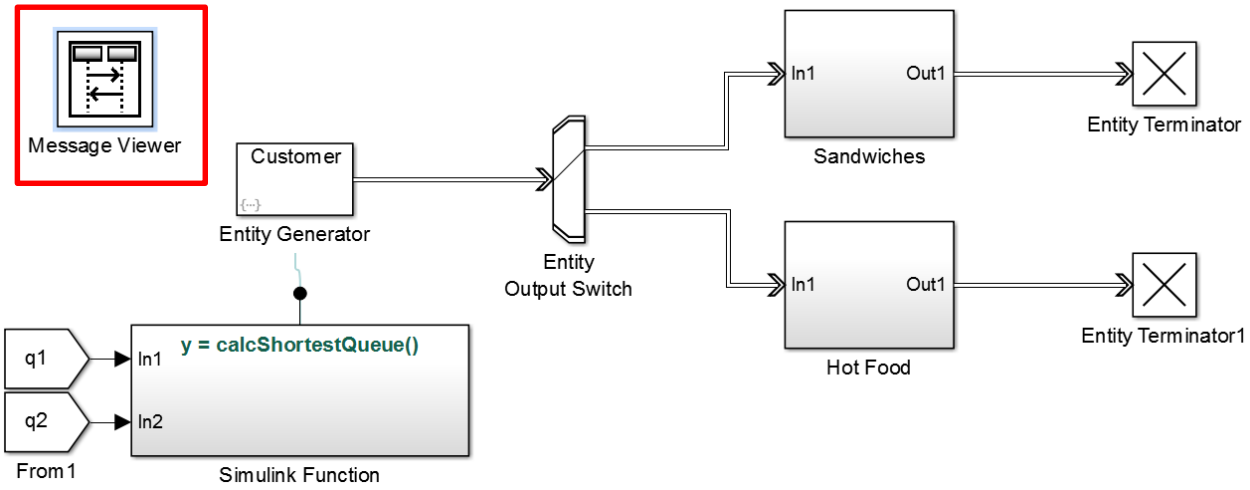

Figure 31: Add the Message Viewer block

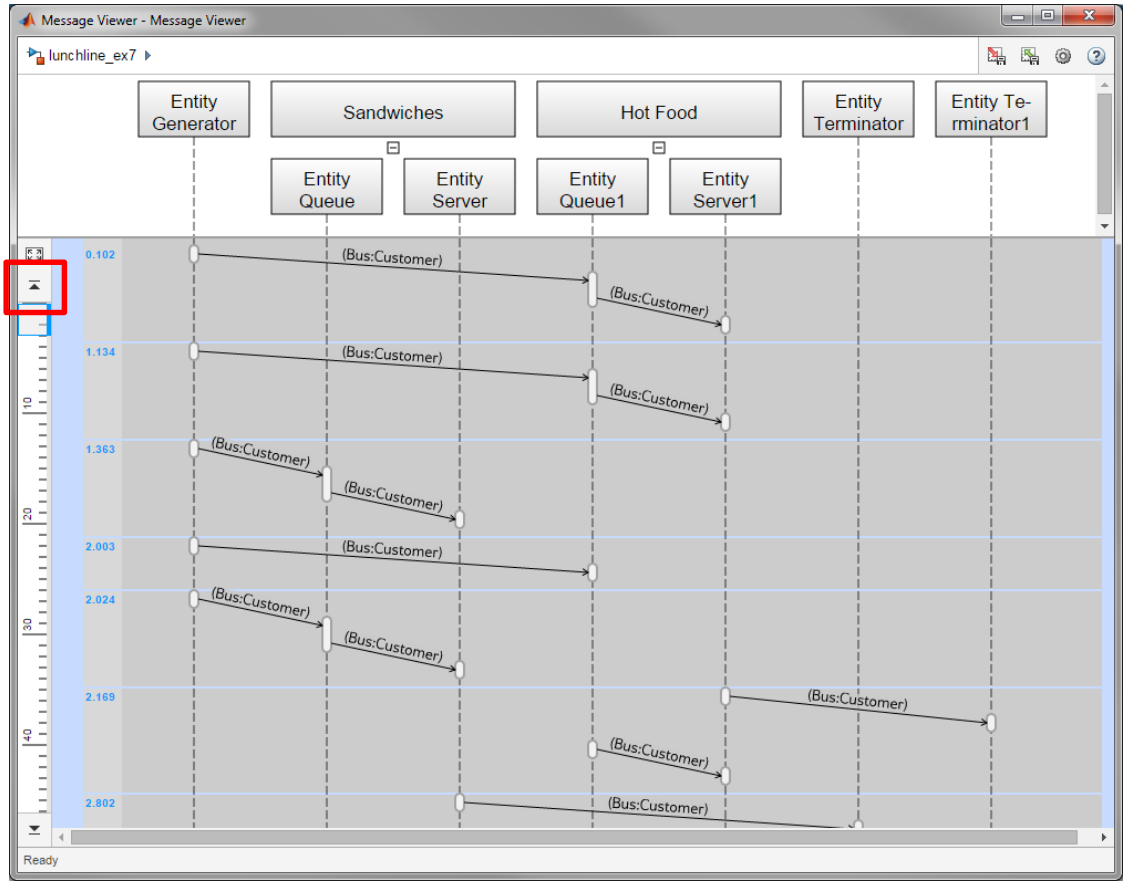

Figure 32: The Message Viewer window

Examining the Results:

The vertical lines in the Message Viewer window are known as lifelines. All SimEvents blocks that can store entities appear as lifelines on the Message Viewer. Entities moving between these blocks appear as lines with arrows indicating the direction of movement.

The first event shows that a customer is generated at t = 0.102. The customer moves to the Hot food queue, and then the server instantaneously (also at t = 0.102). The next customer is generated at t = 1.134, and also moves through the Hot Food queue to the server instantaneously. Both these times agree with first two spikes visible in the Hot Food Queue scope display. The spike indicates that a customer enters the queue and leaves instantaneously (the server is available so there is no need to wait).

You can also verify the 3rd event, which is a customer being generated at t = 1.363 and moving through the Sandwich queue to the server. This is indicated by the first spike in the Sandwich Queue scope display.

Exercise 8: Close the Hot Food Queue after 60 minutes

TIP: You can open lunchline_ex6.slx if you wish to start from this exercise. If you are continuing from the previous exercise, simply delete the Message Viewer block as we will not be using this.

In this exercise you will close the hot food queue after 60 minutes. All remaining customers in the queue will be served, but no further customers can join the queue and will be forced into the sandwich queue. We will construct this concept using the simulation clock and an "enable gate", together with the appropriate routing.

Instructions:

1. Build the model shown in Figure 33 by adding the extra blocks from Table 5. Note that the Entity Generator and Simulink Function blocks have been omitted from Figure 33 but should remain in your model.
2. Run the simulation and observe the queue lengths as shown in Figure 34.

Block Name	Library	Quantity	Parameters
Clock	Simulink\Sources	1	Default
Compare to Constant	Simulink\Logic and Bit Operations	1	Operator: <= Constant Value: 60 Other: Default
Message Send	SimEvents	1	Default
Entity Gate	SimEvents	1	Operating mode: Enable Gate
Entity Input Switch	SimEvents	1	Number of input ports: 2 Active port selection: All
Entity Output Switch	Simevents	1	Number of entity output ports: 2 Switching criterion: First port that is not blocked

Table 5

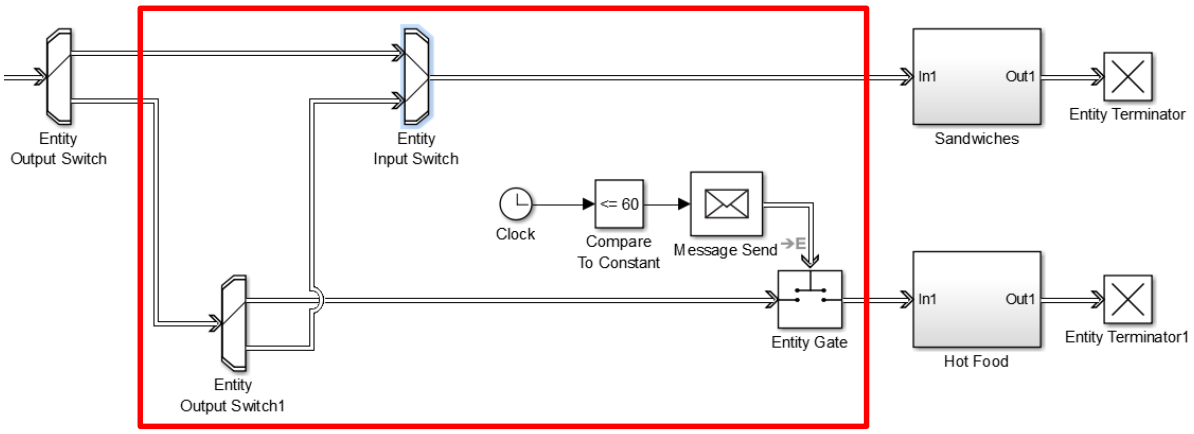

Figure 33: Exercise 8

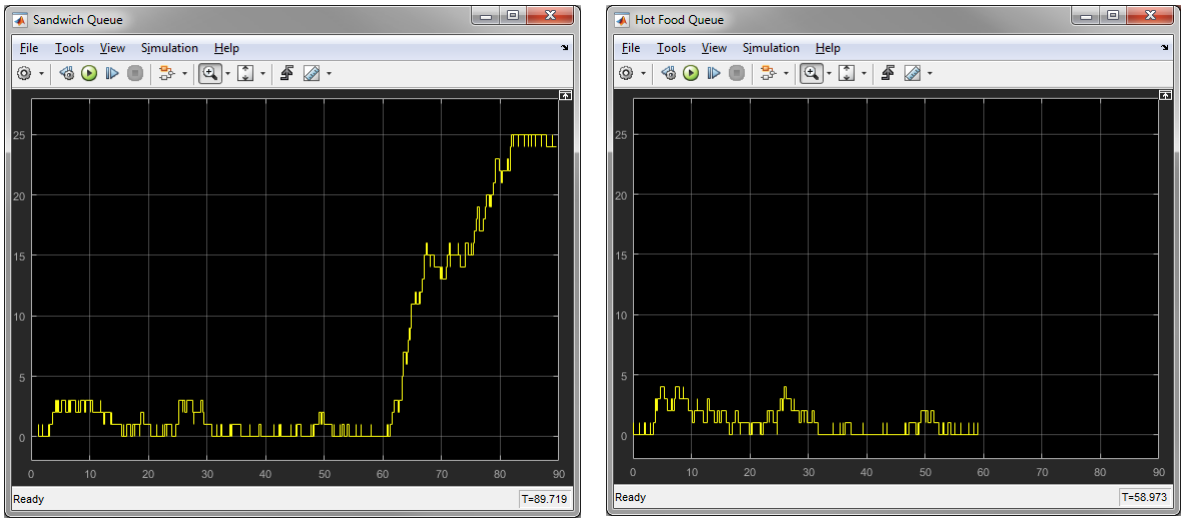

Figure 34: Exercise 8 queue lengths

Examining the Results:

Notice that the Hot Food queue ceases to display any activity around the sixty minute mark. The last event is when a customer passes through the queue at t = 58.973. The Sandwich queue, on the other hand, rapidly increases after 60 minutes due to the influx of customers forced into the sandwich line.

The rerouting of hot food customers into the sandwich line is achieved by blocking with the enable gate. Using the simulation clock and logical operator, we generate a value of one up until t = 60, and zero after. The Message Send block sends this value as a message to the Entity Gate, which remains open when the value is positive, and closes when the value is otherwise.

When the gate closes, it effectively "blocks" output port 1 of the preceding Entity Output Switch block. Since this Entity Output Switch block is configured to switch to the "first port that is not blocked", it re-routes all customers through output port 2 after time reaches 60 minutes.

Output port 2 of the Entity Output Switch block is connected to the Entity Input Switch block. The Entity Input Switch block effectively merges all the input customers into the sandwich queue. As a result, the sandwich queue is stressed since the number of servers does not increase to handle the additional customers.

Advanced Topic:

The "clock" block is a time driven block and outputs the simulation time at the sample rate of the model. This means that the downstream blocks (Compare to Constant and Message Send) are also executing every sample instant. You can connect a "To Workspace" (Simulink->Sinks) block to the clock output signal to investigate these sample times (logged in a workspace variable) if you like. In this model it is not a big deal, but in a more complex model, these unnecessary computations may have a noticeable effect on the simulation speed.

An alternative to the "Clock" block is the "Digital Clock" block also in the Simulink->Sources library. The "Digital Clock" block allows you to specify the "Sample Time" of the output. So you can set the sample time to be a large integer divisor of the desired clock time (e.g. 30 or 60) to limit the number of times these blocks are executed during the simulation. This can result in a faster simulation. You need to be careful, however, that the sample time is selected to allow the "Compare to Constant" block to execute at the desired time. For example, a sample time of 40 in this model, would result in the gate closing at t = 80 and not t=60.

Tips:

To enable Simulink wires to "hop" when overlapping

1) Click on **File->Simulink Preferences** from your Simulink model.
2) Select "Editor Preferences" from the left window
3) Select "Line Hop" (or whatever you prefer) from the right window

Other Resources (for additional information):

Open the MATLAB documentation (execute >>doc) and click on:

1) SimEvents -> Modeling -> Entity Management
2) SimEvents -> Examples -> Tutorials -> Using Entity Priority to Sequence Departures

Exercise 9: Parameterising the model

TIP: You can open lunchline_ex8.slx if you wish to start from this exercise.

Up until now, you have been entering simulation parameter values directly into the blocks. This is useful when initially building a model, however, it is also useful to define the parameters using a MATLAB script. Defining all parameters in a central location can make future modifications of the model much simpler since it is easier to keep track of the values.

Instructions:

1. Double click on the Compare to Constant block that defines the time when we close the hot food line. Redefine the "Constant value" as 'CloseHotFoodLine' as shown in Figure 35.
2. Create a MATLAB script called 'init_parameters.m' and save it to the same directory as the model. Also ensure that the current working directory of MATLAB is within this same folder (check your "Current Folder" window in the desktop), otherwise Simulink will not be able to find this script. The script should contain the following line of code (Figure 36):

   CloseHotFoodLine = 60;

3. Open the model properties dialog from the model window as shown in Figure 37 (by selecting **File->Model Properties -> Model Properties**).
4. Click on the 'Callbacks' tab, highlight 'InitFcn', then enter 'init_parameters;' in the right window as shown in Figure 38.
5. Run the simulation.

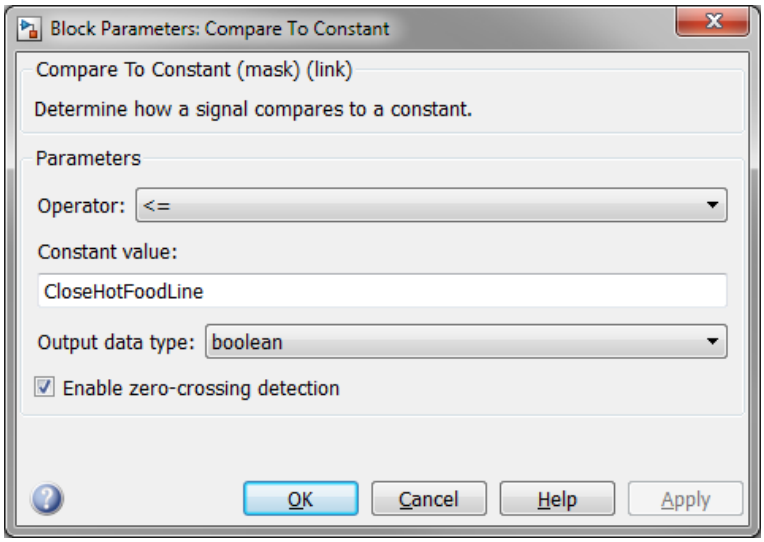

Figure 35: Parameterise the time that we close the hot food line

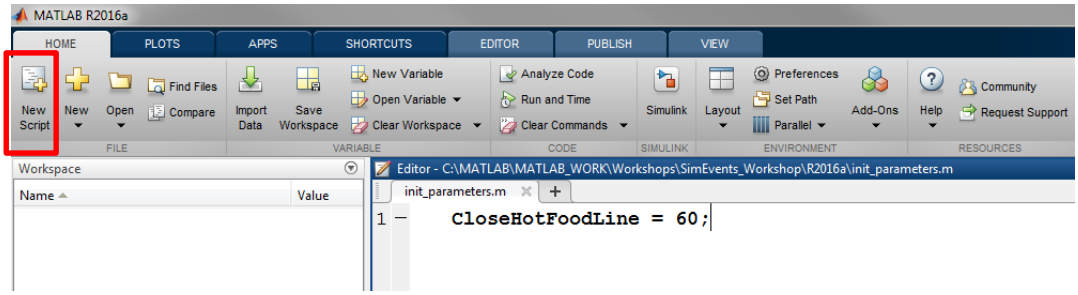

Figure 36: MATLAB script for init_parameters.m

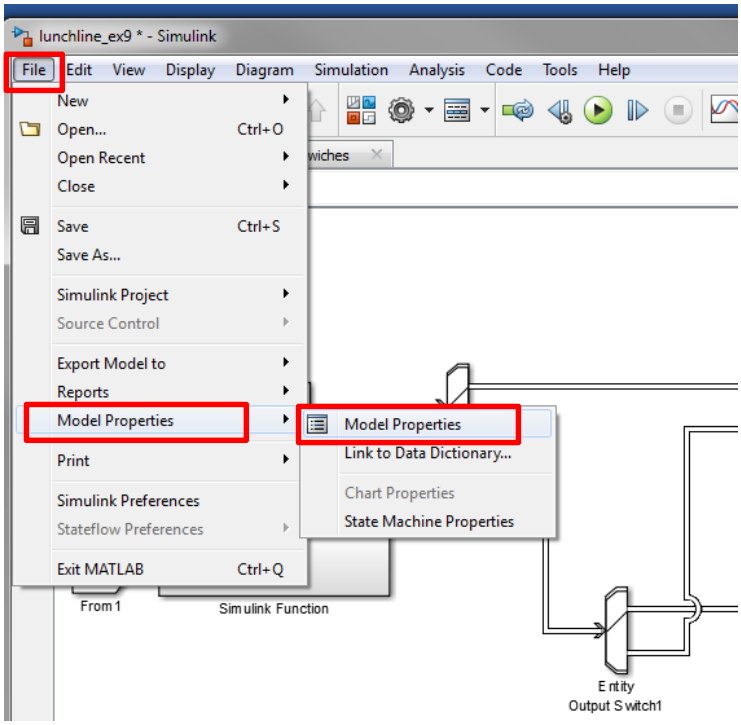

Figure 37: Model Properties

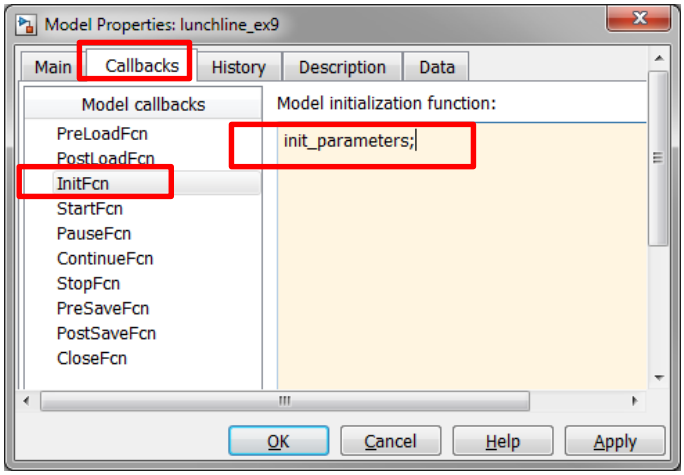

Figure 38: Model Callbacks (InitFcn)

Examining the Results:

The results of this simulation are exactly the same as the previous exercise. The difference is that you parameterised one of the blocks, and defined the value within a central MATLAB script. This could be repeated for all the blocks if you like. The idea is to use a callback function from within the Simulink model. Before a Simulink model begins simulating, it first goes through an initialisation stage. By specifying an 'InitFcn' callback, you are telling Simulink to execute the 'init_parameters.m' file during the initialisation stage. Upon execution, the parameters in the script are loaded into the MATLAB workspace and are visible to Simulink at the start of the simulation. Similarly, you can specify callback functions at other stages of the simulation, for example, when the simulation starts, pauses, stops, etc.

Note: (The below limitation regarding event actions was removed in R2016b)

Using a MATLAB script to define parameters is possible, because the Simulink block parameter dialogs can access variables in the MATLAB workspace. An exception to this rule is that MATLAB event actions cannot access the MATLAB workspace. MATLAB event actions behave like a function, and have their own workspace. You can, however, call MATLAB scripts/functions from within the MATLAB event action to define these parameters within the event action workspace. Rather than define these parameters every time the event action is called, however, a more efficient method could be to

1) Have init_parameters.m define all the parameters within fields of a single structure variable.
2) Within the MATLAB event actions, define a persistent variable to hold this structure.
3) Use "if isempty(parameters_structure), parameters_structure = init_parameters; end" within the MATLAB event action to define the persistent parameters structure. This will execute only on the first call to the action, but the parameters structure will then persist within the action for future calls.
4) Access the parameters within the MATLAB event action by indexing the relevant fields of the structure.

Other things to try:

1. Appropriately annotate your model
    - Rename blocks, scopes, ports, etc., to have more descriptive names. Note that no two blocks can share the same name if they are at the same level of hierarchy.
    - Add annotations to the model by double clicking anywhere on the blank portions of the canvas.
    - Format the text, add background colour and annotation borders by right clicking on an annotation or block.
    - Take a look at 'lunchline_annotations.slx' for an example of annotations.
2. Peruse the other SimEvents examples in the documentation.
3. Watch the first 6 videos in the "Getting Started" section of these Simulink Videos http://au.mathworks.com/products/simulink/videos.html

<END OF WORKSHOP>