A Simple DES Model with Potential for DIS

# Scenario: Multiple Server Queue

Customers periodically arrive to a service facility where they require service from one of k identical servers. An arriving customer finding all servers busy with other customers waits in a first in-first out queue. When a server completes service on a customer, the server starts serving the next customer in the queue. If the queue is empty, however, the server become idle. An arriving customer finding a server is available may begin processing immediately.

# Event Graph Model

There are several different ways to model a multiple server queue. The one chosen here is oriented towards separating the customer arrivals from the service facility. Recall that an Event Graph component is defined by its parameters (quantities that do not change during a given replication), state variables (quantities that do change during a given replication), events (defining the instantaneous state transitions), and scheduling relationships between events. The latter two can be represented as a directed graph, with the nodes as events and the directed arcs depicting the scheduling relationships between events. Part of the definition of a state variable is its initial value, depicted here in parentheses after the definition.

## Customer Arrivals (Arrival Process Component)

### Parameters

* {tA} = interarrival times

Here the interarrival times are a sequence of values. These are typically generated from a probability distribution, although they could be a set of deterministic values.

### State Variable

* N = # arrivals (0)

### Event Graph



Figure . Arrival Process Event Graph

The Run event in Figure 1 initializes the number of arrivals, N, to 0 and schedules the first Arrival event with a delay of tA. When that Arrival event occurs, in increments N and schedules the next Arrival event with a delay of the next value of tA.

## Service Facility (Simple Server Component)

### Parameters

* k = number of servers
* {tS} = sequence of service times

As with the interarrival times, the service times are a sequence of numbers, typically generated from a probability distribution.

### State Variables

* Q = number of customers in the queue (0)
* S = number of available servers (k)
* N = number customers served (0)

This simple model does not explicitly represent each customer, but simply counts the number in the queue as one state variable. Note that the total number in the service facility is given by the expression Q + k – S.

Note also that the N in this component is a very different state variable than the N in the arrival component.

### Event Graph



Figure . Simple Server Event Graph

The Simple Server Event Graph in Figure 2 is not a complete model, but a component. The Run event initializes the state variables but does not schedule any events. In order for the Event Graph to function, another component (such as an Arrival Process defined in 2.1) must first schedule Arrival events. When each of those Arrival events occur, they must be “heard” by the simple server component. In stand-alone models, this is done by SimEvent Listening.

# SimEvent Listening

SimEvent Listening is when one Event Graph component “listens” to the events of another Event Graph component. Whenever an event in the source component is executed, it is “heard” by the listener. If the listener has an identical event, it executes it. If it has no matching event, it is simply ignored. The one exception is the Run event. Since each component has its own Run event that initializes its own state variables, Run events do not get passed to SimEvent Listeners.

A listening relationship is depicted in Figure 3. The connector can be thought of as like a stethoscope listening to the events of the source component.



Figure . Sim Event Listening Diagram

The diagram in Figure 3 depicts the complete model of the customer arrivals plus the service facility. The Arrival Process component will dutifully schedule a sequence of Arrival events. These in turn will be heard by the Simple Server component, thereby triggering its own Arrival event and subsequent events and state transitions.

All of this can easily be implemented in Simkit.

# Connecting Components with DIS

An alternative to the SimEvent Listening approach in Section 3 is housing each of the two components on separate computers and having the Arrival Process component announce each arrival by sending a DIS PDU. This would be received via the network by the computer housing the Simple Server component, which would process the PDU so as to trigger the Arrival event.

A simple way to implement the source component is a small Event Graph that listens for Arrival events and dispatches a PDU. This is shown in Figure 4.



Figure . PDU Dispatched for Arrival Events

The arrow from Arrival to the PDU transmitter is not scheduling an event but simply instantiating and dispatching a PDU corresponding to the Arrival event.

The receiving end is similar (Figure 5).



Figure . Receiving PDU for Arrival Events

Here, the PDU receiver needs to only schedule Arrival events for which an Arrival PDU has been received.

Note that the model can be extended simply by creating multiple Arrival Process clients sending Arrival events to the service facility. This can model situations in which there are multiple streams of customers to the facility.

# Measures of Interest

Two primary measures of interest can be obtained from this simple model:

1. Average number in queue
2. Average utilization of the servers

One observation for each of these is obtained from one replication of the simulation. The average number in queue over the replication is given by the time average of the state trajectory for Q. Similarly, the average utilization is given by where is the time-average of the state trajectory for S.