Problem:

A process must do ServerA, then Servers 1, 2, and 3 in any order but must do all three, then finish with ServerC. Illustrates how a simple modification can be made to an object to permit more intelligent use, in this case allowing selection of only objects that still need the service it provides.

Categories:

Node Lists, Custom Object

Key concepts:

Entity Destination Type, Routing Logic, State Variable, Selection Goal, Selection Condition, Custom Object, Candidate

Assumption:

The three pooled servers have the same processing time and each has a single buffer position. The entities can process through the 3 servers in any order, but must process at each of the 3 servers exactly once before moving on to Server C. The server with the smallest overload is selected. If no server is available that is still needed, the entity will wait in the Dispatching area (a node).

General Approach:

There are two key requirements:

1) Each entity must keep track of which of the three servers it needs to visit and which have already been visited. This is done by adding three states to each ModelEntity.
2) The easiest way to select between servers is to use the TransferNode's built-in Entity Destination Type and in particular it's Selection Condition. In order to take advantage of this, we created a custom MyServer by subclassing from Server. The only change on MyServer is to add a HasEntityVisited property into which can be passed the appropriate entity state corresponding to that server.

Embellishments:

We added 3 status labels (one for each custom state) on each entity picture so you can see which stations have been visited.

We added 3 pictures to the entity that gets successfully darker as more stations are visited so you can tell its status at a glance.