Discrete Event Simulator: User Manual

Author: Morgan McLaughlin

Created for use in CIS 2460 - Modelling of Computer Systems describing the Discrete Event Simulator written by Dr. Mark Wineberg.

Introduction

This user manual will explain how to use the discrete event simulator (DES) to create your own custom simulator. While the manual will briefly describe how the simulator works at some point that is not the purpose of this document. The goal of this user manual is to help you with your solution for assignment 3. For details on how the simulator works please view the class slides for an overview of how the DES works and the code itself for a detailed look on how it was implemented.

File List

This section will briefly overview each file's role in the simulation.

- **Internal DES code**
  - DESystemFunctions.R – Core functions providing basic simulation functionality
  - discreteEventSimulator.R – This is the main DES driver, it contains the functions which ties together the entire simulator.
  - futureEventScheduler.R – Internal functions for running the future event scheduler
  - priorityQueue.R – A generic priorityQueue, used by the future events list
  - schedulerPrintFunctions.R – Used to output data when the scheduler runs, mainly used for debugging
  - state_stats_Functions.R – Internal function for holding the system state and its stats

- **DES Interface**
  - DES_Summary long.R – Interface to the simulator for Event Update Routines
  - DES_Summary short.R – Basic interface functions available to be used by the simulation user

- **Custom Simulation**
  - singleServerSystem.R – Sample simulator for a single server system using the DES

- **Other**
  - loadSimulator.R – Loads all of the simulator functions. Makes using the simulator more efficient.
  - runSingleServerSystem.R – A file used with singleServerSystem to run a variety of simulations

Running the Discrete Event Simulator

Instructions for running a simulation using the DES.

1. Load the DES files

The files can be easily loaded using loadSimulator.R. This file will need to be edited to change the ‘basePath’ variable to where your files are saved. You could also just set the ‘basePath’ variable to a blank string and make sure your workspace is set to where your files are saved.

If you wish you could just source each file manually, the loadSimulator file is only meant to make things easier.

2. Load your custom simulation file

Load your simulation as you would any other code file using source(). The load simulator only loads the DES functions, you need to load your own code.

3. Run your simulation

Use the command: discreteEventSimulator(simSystem)
simSystem: the custom system you’ve created
endTime: The time your simulation will end (default of 100)

Optional variables:
startTime: The time your simulation starts (default of 0)
verbose: When true you will see more output from your simulation

4. Viewing stats

The simulator returns an environment variable which contains the stats information. In order to view them you can use the function:

```
print.stats(results)
```

1. All-in-one: runSingleServerSystem.R

This file has been created to allow the user to more easily run the simulation and run your simulation with different parameters. This file contains the loadSimulator functions which you’d use in the same way as discussed in “Loading the DES files” (Step 1). It also provides a function for loading the singleServerSystem (Step 2). Then you can use the commands listed there to run your code with different parameters and compare the stats returned (Step 3-4).

Singe Server System Example

This is a sample simulation running on the DES contained in the file singleServerSystem.R. This is the example you will have access to as a sample for how to write a simulations for the DES. This is the file that you will be editing/replacing with your own simulation.

Basic Overview

This is only an overview of the singleServerSystem file, for an in-depth understanding you will need to view the code (and its many comments) for yourself.

- The first section provides the basic functionality of our server, it includes:
    - Random number generators which are used to generate our client arrival times and their process times. These could be set to whatever distribution you wish to use for your simulation
    - A set of states the system could be in
    - A scheduler which is used to handle events being put into the Future Event Log
    - The even update routines which are called whenever an event is popped from our FEL, these will then change the state of our system depending on what event was popped and what our current state is
- The second section is our stats section which is used to collect the statistics information which can be viewed later by the user
- The third section is a section used for printing out data from the DES. It contains the function to handle printing out our stats collected by the stats section. It can also be used to print out data at different stages of our program to give us a better look at what's going on and help debug code
- The fourth and final section is the function we use to create our system. This is the function passed into the DES

Running the Simulation

The file can be run using this command (assuming we have already loaded the DES):

```
discreteEventSimulator(singleServerSystem, endTime = 100 , startTime = 0, verbose = TRUE)
```

A sample run (using a shorter end time for space considerations):

```
results <-discreteEventSimulator(singleServerSystem, endTime = 10 , startTime
= 0, verbose = TRUE)
Clock = 0 -> 6  (timeDiff = 6)
     Event = A  (Next = A)
     Future events = (A, 6)
     state(LS = 1, LQ = 67)
     stats(N = 67, NW = 67, WS = 316, WQ = 10197, T0 = 0, Tmax = 316)

Clock = 6 -> 10  (timeDiff = 4)
     Event = A  (Next = E)
     Future events = (A, 13)
     state(LS = 1, LQ = 68)
     stats(N = 68, NW = 68, WS = 320, WQ = 10469, T0 = 0, Tmax = 320)

Clock = 10 -> 13  (timeDiff = 3)
```

```
     Event = E  (Next = A)
     Future events = no new events generated
     state(LS = 1, LQ = 68)
     stats(N = 68, NW = 68, WS = 320, WQ = 10469, TO = 0, Tmax = 320)
```

Then we can view the stats:

```
print.stats(results)
stats(N = 68, NW = 68, WS = 320, WQ = 10469, TO = 0, Tmax = 320)
```

Hints for Creating Your Own Simulation

You have now seen an example of a simulation using DES. These are some general hints for how you might want to go about writing your own simulation.

- You could either start from scratch and just use the given simulation, or begin editing the example to suit your needs. Editing the file is the recommended approach.
- Make sure the singleServerSystem example works for you before you start doing anything, if you can't get the (working) provided example running then you won't get anywhere with your own
- Whenever you edit a function, if it is an isolated change make sure your simulation still runs, don't wait until you've made numerous changes and can't find what change caused the error
- Many of the functions in the example should only need a few minor updates (if any at all) for your assignment
- You will also need to add a few functions, states, stats, etc as your assigned simulation is slightly more complex, but the basic skeleton for your simulation has been given to you; it is advised to follow the same style (it also makes it easier to mark!)