A Student-Designed Interactive Simulator for the Study of Queuing Theory and Other Applications[i]

Jamielynn Savino, William Stefanko, Gordon Silverman
Electrical and Computer Engineering
Manhattan College
Riverdale, New York, 10471

Abstract

Queuing theory relates to the study of process congestion. Where engineering programs do not have provision for a focused study of such problems, an interactive simulator can convey the basic concepts without extraordinary or extended explanations. A "customer-centered" queuing simulator can be used in a range of engineering courses and a variety of engineering disciplines. A simple study is provided as an example.

Introduction

Commonly accepted definitions of queuing theory relate to the theoretical study of waiting lines, expressed in mathematical terms. (Examples of such studies come from transportation and/or infrastructure (e.g., client/server traffic for an internet service provider).) In short, queuing arises from mathematical analysis to describe "production" processes along with statistical/probabilistic techniques to account for varying dynamic patterns in a productive process. The problems encountered – those that occasioned the development of such theory – are simply referred to as "congestion" problems or what happens when a system does not operate smoothly or efficiently. Queuing analysis is directly applicable to network telecommunications, server queuing, mainframe computer queuing of telecommunications terminals, and advanced telecommunication systems. There are numerous software simulation tools for discrete event simulation. (1) These have origins in FORTRAN or PASCAL subroutine libraries (1980s) but newer tools include GPSS, SimScript, and CSL, to name a few. (2)

Undergraduate engineering programs normally do not have provision for queuing theory as a separate course. This is true for several reasons:

- It requires a specialized mathematical curriculum development;
- Students are required to spend time learning the commercial software tools before they can obtain meaningful results which limits appreciation of the underlying problems;
- It may not contribute to program objectives (or outcomes) in a direct and obvious manner.

However, queuing theory can readily be introduced into a variety of courses and convey the basic concepts without extraordinary or extended explanations. In such courses, queuing theory serves as an "example" and may be applied to such curriculum-related topics as undergraduate programming, simulation, and probability.

Technology

Traditional Queuing Theory may be characterized by several alternatives:

- Single Channel Queuing: one queue and a single server; (customer) arrivals are unlimited.
- Multiple-Channel Queuing: a 2^{nd} queue is added to the model as well as a second server.
- Constant Service Time Model: using the Single Channel model, a constraint is added whereby each customer is handled in the exact amount of time.
- Limited Population Model: a limited number of arrivals and a single server; this applies to circumstances where a single server deals with a limited number of machines (the queue).

As a direct consequence of the underlying problem, the simulation design herein has some unique characteristics:

- It is a multiple queue environment (greater than the two studied in classical theory).
- There is no limit to "arrivals".
- Most especially, the customers ("cars") may frequently switch queues, a phenomena not normally considered in traditional studies. (The queue is "customer-centered.")

Inspired by a commonly encountered highway phenomena – driving on a congested, three-lane highway – students in an Electrical and Computer Engineering program developed an interactive (software) application that permits users to appreciate queuing theory within a simple context so that principles could be understood without extensive prerequisite information such as applications associated with telecommunication networks to cite one example. (Drivers in such situation may wonder if it is best to stay in the current lane or switch to an adjacent lane that appears to be moving faster. A "penalty" is incurred when a time delay is introduced during such a lane switch.). Using Visual Basic (VB), (3) these students provide 'front panel' animation as well as controls that permit a user to set parameters such as: average speed in each of three traffic lanes, duration of the simulation ("Distance"), lane switch latency ("Latency"), behavioral latency ("Think"). Results of the simulation are compatible with spreadsheet (e.g., Excel) analysis. Those parameters that are not obvious are briefly described as follows:

> *Lane switch latency*: a parameter that permits the user to adjust the time that it takes for a car ("customer" in queuing theory parlance) to move to a new lane (queue).
> *Behavioral latency*: a parameter that simulates behavioral hesitation when deciding when to switch lanes.

All transactions (e.g., lane switch) within the simulation are influenced by the random number generator in VB. Figure 1 depicts the user interface. The manipulanda include "slide controls" (with associated digital indicators to display the current values. Button controls permit a user to start a simulation ("Go"), interrupt the simulation ("Pause"), restart a simulation ("Refresh"), and end the simulation ("Exit"). Two additional auxiliary digital indicators report the current status (distance traveled) of the merging car as well as a car that chooses to follow a "non-weaving" strategy. When a switch (merge) occurs, the car icon will move to the new lane. Car progress indicator in the lower right hand corner tracks vehicle progress and an increasing (color-coded) "thermometer" indicator also tracks advancement.

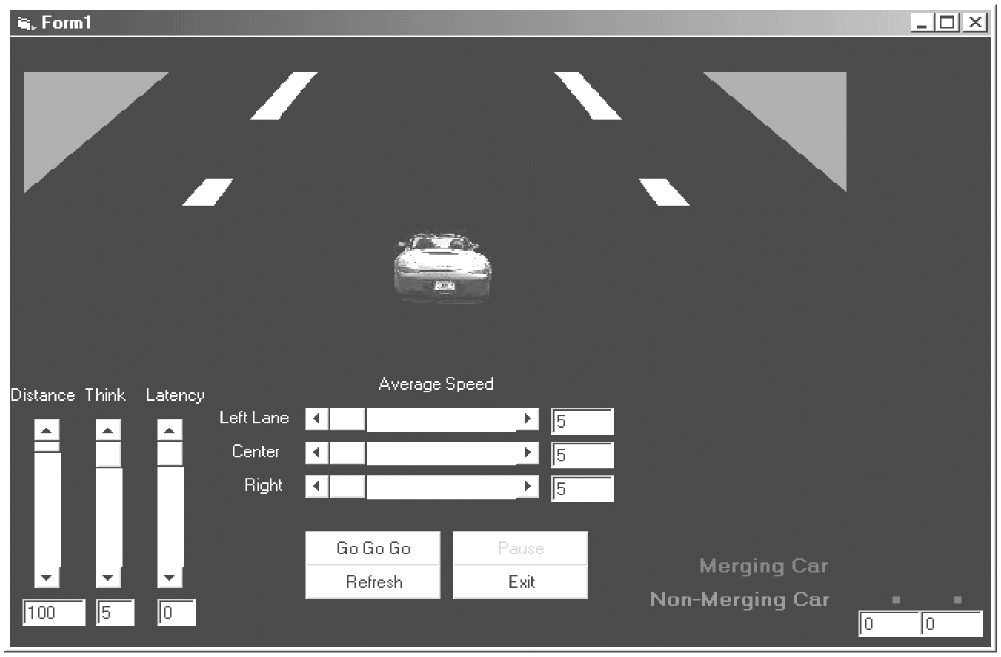

Figure 1. Interactive Simulation Panel (gray scale representation)

The simulation software produces a spreadsheet-compatible (Excel) file that traces the history of car movement within each lane as well as when lane switching occurs. A fragment from the file is shown below together with significant annotation:[1]

"Resetting..."	*Indicates simulator reset
"Left Center Right"	*The sets of 3 numbers listed below
	*are in the order of Left, then center, then right
1,6,24	*In this time interval, the left lane
	*moved one unit, the center 6 units, etc
20,4,7	
1,24,26	
30,4,14	
1,21,18	
30,17,22	*The same is true for the these other entries
4,21,10	

[1] The font has been changed from "Courier New" to "Times New Roman" to conform to publications requirements. The simulator produces the output file in the Courier New font and annotations are omitted.

```
10,26,9
11,28,16
11,14,5
2,20,21
"Time to merge-> Totals for each lane:"
                                *There is a configurable parameter in the code
                                *that sets the number of intervals (iterations)
                                *between merger decisions.  In this case, after 11
                                *sampling intervals it is time to merge
                                *into the fastest lane.  We use the total
                                *number of units moved by each car in
                                *each lane over the past 11 sampling
                                *intervals to decide which lane is best
121,185,172                     *The total units moved by left lane is 121,
                                *center lane is 185, and right lane is 172.
                                *(See numbers to the left.)
                                *A random number in conjunction with the
                                *average speed parameter determines how
                                *much a car in each lane will move in a
                                *sample interval.

"Center lane is best"
"Latency is 3"                  *The driver will take 3 intervals to merge lanes.
"Best Lane is Current Lane, no Latency"
                                *However, our driver started in the center lane,
                                *therefore, the driver does not merge (change).
"Left Lane Total during Latency: 0"
                                *If the driver had merged, the simulator
                                *keeps track of the total units moved during
                                *the latency period.
"Right Lane Total during Latency: 0”
"Center Lane Total during Latency: 0"
"At this point, our car has gone 185"
"A non-merging car would have gone 185"
                                *If our driver never merged, this (185) would
                                * have been his/her progress
"End Latency"
 15,20,27
18,12,11
9,28,25
30,21,26
26,30,25
27,4,4
22,7,2
30,15,2
30,9,19
```

```
27,22,23
27,22,23
"Time to merge-> Totals for each lane:"
261,190,187
"Left lane is best"              *Our driver needs to change lanes
"Latency is 3"                   *This will take 3 intervals
24,25,5                          *The simulator tracks the progress of these lanes
3,11,30                          *during the latency period.
21,2,28
"Left Lane Total during Latency: 48"
                                 *Total units moved by each lane during driver's *merge
"Center Lane Total during Latency: 38"
"Right Lane Total during Latency: 63"
"End Latency"
"At this point, our car has gone 423"
                                 *Same as before, except notice that the
                                 * the driver has gained 10 units by merging.
"A non-merging car would have gone 413"
```

The file continues with this format until the entire epoch (determined by the "distance" parameter (control)) is complete.

Application

A simple study is performed in which strategy is tracked as a function of the behavioral parameter ("Think"). The parameters for the study were set as follows:

Average speed, left lane, middle lane, right lane	15
Distance:	1000
Latency:	4

The "Think" parameter was varied in steps: 5, 10, 15, and 20. At each step, the simulator was repeated a total of 20 times. The results of the study are summarized in Table 1.

Table 1
Results of the Simulation

Think parameter	Merge	Non-Merge	Average Advantage for Merge (%)	Average Advantage for Non-merge (%)
5	7	13	5.1	6.5
10	11	9	6.1	3.1
15	8	12	7.2	3.9
20	9	11	5.2	6.1
Overall avg. (%)			5.9	4.9

To interpret the results, consider the following. When the Think parameter was set to 5 (see Table 1), seven of the twenty epochs – wherein an epoch is a completion of an arbitrary "distance" of 1000 traveled – suggested that merging was a better alternative in seven instances while non-merging produced better results in thirteen epochs. Where merging was a better alternative, the "gain" (advantage) in distance was 5.1%, while in those cases where non-merging produced better results, the average advantage was 6.5%.

In this particular study, overall results suggest that non-merging is a better strategy than merging ("weaving"). (Except for the case where the Think parameter was 10, better performance was achieved with non-merging.) However, it should be noted that for those outcomes (epochs) where merging was the better strategy, the average improvement (distance advantage) was 5.9%, while non-merging strategies produced an average advantage of only 4.9%.

Summary

A "customer-centered" queuing simulator can be used in a range of engineering courses and a variety of engineering disciplines. (Traditional queuing theory concentrates on server performance.) The simulator can be used in courses where topical material includes simulation, probability or programming. (It may even have application in such non-engineering courses as Psychology.) Within such courses, students could be asked to modify (reprogram) the simulator to account for more "realistic" life experiences. For example, when the car does merge (switch lanes), the speed within the new lane might be reduced because of the presence of this new congestion. As currently designed, the car may move from the extreme left lane to the extreme right lane within a single latency period. While some drivers might attempt such a dangerous maneuver, it is more realistic to require a second latency ("penalty") when such a change is accomplished. When used in a course in probability, studies could be performed to develop density (or distribution) functions as a function of one of the parameters based on the uniform random number generator found in VB. More sophisticated studies might be addressed by modifying the simulation so that the random number generator has some other shape (e.g., normal). Such a simulation may even be used in a Psychology course where the effects of psychological delay (as noted in the study) impact a driving strategy.

Bibliography

(1) Wolff, R.W., *Stochastic Modeling & the Theory of Queues*, Prentice-Hall, 1997
(2) Graduate Education & Research Services (GEARS), www.gears.aset.psu.edu/hpc/software/queue.
(3) Schneider, D.I., *Introduction to Programming Using Visual Basic 6.0*, Prentice-Hall, 1999

Jamielynn Savino is currently completing a Master of Science Degree in Computer Engineering at Manhattan College. She is a staff engineer at ITT Laboratories.
William Stefanko holds an M.S. in Computer Engineering from Manhattan College and is a staff engineer at ITT Laboratories.
Gordon Silverman is a Professor of Electrical and Computer Engineering at Manhattan College and is currently Chair of the department.

[i] The simulator is available and will be transmitted upon request. E-mail such requests to gordon.silverman@manhattan.edu. The file occupies approximately 3Megs and will be sent using a "zip" format. All other correspondence should be addressed to Dr. Gordon Silverman, Manhattan College, Riverdale, NY 10471