1 Introduction to Discrete-Event Simulation

Here we will consider simulating a stochastic process, $\{X(t) : t \geq 0\}$, with state space $\mathcal{S}$ forwards in time, for purposes of computing or estimating various quantities of interest. The basic idea is to sequentially keep moving in time to the next *event*, which in general refers to a change of state; such as an arrival or departure from a queueing system. Typically, there are several events, $e_1, e_2, \ldots, e_k$, scheduled and competing to be the next one. Once we determine which one is next, we move forward to that time, update some system variables of interest, generate some new random variable(s) (so as to schedule another such event) and continue. In many cases, we have a pre-specified termination time t^* at which time we plan to stop the simulation. In some applications, t^* is a random variable called a *stopping time*; such as "the first time that the queue has 10 people in it". It is best to illustrate through examples, which we do next.

1.1 FIFO GI/GI/1 Queue

Recall that for the FIFO GI/GI/1 queue, we can use the recursion for customer delay:

$$D_{n+1} = (D_n + S_n - T_n)^+, \ n \geq 0. \tag{1}$$

Here $\{S_n\}$ are iid service times distributed as $G(x) = P(S \leq x), \ x \geq 0$, and $\{T_n\}$ are iid interarrival times distributed as $A(x) = P(T \leq x), \ x \geq 0$. The two sequences are assumed independent. This does not involve a discrete-event simulation.

But there are other quantities of interest besides delay, some of which are continuous-time quantities, and here we will consider $L(t)$ = the total number of customers in the system at time t; where for simplicity of exposition we will assume that $L(0) = 0$; the system is initially empty. Thus our stochastic process is $\{L(t) : t \geq 0\}$ and the state space is the non-negative integers $\mathcal{S} = \{0, 1, 2, \ldots\}$.

We shall assume here that we are interested in computing $L(t^*)$ and $\frac{1}{t^*}\int_0^{t^*} L(s)ds$ for a pre-specified value of time t^*.

1.1.1 Events

There are two (2) events that occur over time : e_1 = an arrival occurs, e_2 = a departure occurs. Let t_n denote the arrival time of C_n (the n^{th} customer), $n \geq 1$, and let $t_n^{(d)}$ denote the departure time of C_n. At any given time t (current time), we let $t_A > t$ denote the time in the future at which the next arrival will occur, and t_D denote the time at which the next departure will occur. As we move forward in time, the values of t_A and t_D need to be updated. Initially at time $t = 0$, we generate the first interarrival time $T \sim A$ (such as via inverse-transform $T = A^{-1}(U)$) and set $t_A = T$. No one is in service yet now at time 0, so for mathematical convenience (trickery) we set $t_D = \infty$, by which we mean that we set t_D = a very large number. The point is that then we know that $t_A = \min\{t_A, t_D\}$, and hence an arrival will occur next as should be. We thus move to time t_A, by re-setting $t = t_A$. We also increase the number in system (denoted by n) by 1: re-set $n = n + 1$. ($n = 0$ initially.) Now we need to generate a new $T \sim A$, but also a service time $S \sim G$ for our first customer. Then we re-schedule the two events: Re-set $t_A = t + T$ and $t_D = t + S$.

We also must check to see wether we have passed by our desired time t^*. For example, if our first arrival time $t_A = T > t^*$, then we would stop and conclude that $L(t^*) = \frac{1}{t^*}\int_0^{t^*} L(s)ds = 0$.

If our first arrival time $T_A < t^*$, then we can begin to update both $L(t^*)$ and $\int_0^{t^*} L(s)ds$ as we move along.

1.1.2 Three Cases for the Simulation

In general, as we move forward in time, we check to see which of the two events is next, and wether or not we have exceeded our termination time t^*. Let $INT = \int_0^t L(s)ds$, the area under the curve $\{L(s) : s \geq 0\}$ up to t. Initially (at time $t = 0$), $INT = 0$. Noting that this area is simply the sum of the area of rectangles, we can easily update INT over time by adding on the next rectangle area. Initially, $n = 0 = L(0)$.

1. *ARRIVAL IS NEXT ($t_A \leq t_D$ and $t_A \leq t^*$):*
   (a) *Update the area INT:* Re-set $INT = INT + n(t_A - t)$. (The new rectangle has length $t_A - t$ and height n, hence area $n(t_A - t)$.)
   (b) *Update the number in system n:* Re-set $n = n + 1$.
   (c) *Update time t:* Re-set $t = t_A$.
   (d) *Schedule next arrival and possibly next departure:* Generate $T \sim A$ and re-set $t_A = t + T$. If $n - 1 = 0$ (recall (b) above), then this arrival found the system empty and will enter service immediately, so generate $S \sim G$ and re-set $t_D = t + S$. (Otherwise, someone is still in service, so no need to schedule a new departure time.)
2. *DEPARTURE IS NEXT ($t_D < t_A$ and $t_D \leq t^*$):*
   (a) *Update the area INT:* Re-set $INT = INT + n(t_D - t)$. (The new rectangle has length $t_D - t$ and height n, hence area $n(t_D - t)$.)
   (b) *Update the number in system n:* Re-set $n = n - 1$.
   (c) *Update time t:* Re-set $t = t_D$.
   (d) *Schedule next departure:* If $n \geq 1$, then generate $S \sim G$ and re-set $t_D = t + S$ (the next person in line entered service with this new service time S.). Otherwise $n = 0$ (the line is empty, hence no one is present to enter service), re-set $t_D = \infty$.
3. *NEXT EVENT OCCURS AFTER TIME t^* ($\min\{t_D, t_A\} > t^*$):*
   (a) *Update the area INT:* Re-set $INT = INT + n(t^* - t)$. (The new rectangle (cut off at time t^*) has length $t^* - t$ and height n, hence area $n(t^* - t)$.)
   (b) Stop. Output n and INT/t^* .

1.1.3 Estimating expected values $E(L(t^*))$, and $E\left[\frac{1}{t^*}\int_0^{t^*} L(s)ds\right]$.

By running the simulation independently m times (m large) to obtain m iid copies of $L(t^*)$ (denoted by $L_1, \ldots, L_m$) and $\frac{1}{t^*}\int_0^{t^*} L(s)ds$ (denoted by $Y_1, \ldots, Y_m$), one can then utilize Monte Carlo simulation to estimate the expected values:

$$E(L(t^*)) \approx \frac{1}{m}\sum_{i=1}^{m} L_i,$$

$$E\left[\frac{1}{t^*}\int_0^{t^*} L(s)ds\right] \approx \frac{1}{m}\sum_{i=1}^{m} Y_i.$$

1.1.4 Collecting Additional Data During the Simulation for Output

It might be desirable to also collect along the way in the discrete-event simulation, further information to output at the end. Examples might include: The number of arrivals that occurred during $(0, t^*]$, the number of departures that occurred during $(0, t^*]$, and the list of corresponding arrival and departure times.

For example, by letting N_A denote the current total number of arrivals so far, and N_D the current total number of departures so far, we would, up front in CASE 1, add in a line $N_A = N_A + 1$, and up front in CASE 2 add in a line $N_D = N_D + 1$. By also creating vectors $a[i]$ = arrival time of C_i, and $d[i]$ = departure time of C_i, $i \geq 1$, we also would attach the new arrival and departure times when they occur:
CASE 1: $N_A = N_A + 1$ and $a[N_A] = t_A$
CASE 2: $N_D = N_D + 1$ and $d[N_D] = t_D$

Then at the end of the simulation we would also output
$N_A,\ N_D,\ \{a[i] : 1 \leq i \leq N_A\},\ \{d[i] : 1 \leq i \leq N_D\}$. Thus we would be outputting the arrival times $(t_1, \ldots, t_{N_A})$, and departure times $(t_1^{(d)}, \ldots, t_{N_A}^{(d)})$.

1.2 Other Service Disciplines such as LIFO (Last-In-First-Out of line)

When a server chooses (or is sent) its next customer differently from FIFO, then the delay sequence recursion 1.1 no longer holds; hence another reason why a discrete-event simulation might be necessary/useful. As an example, let us consider LIFO. In this case, a new arrival joins the front of the line instead of its end. We are also assuming here *non-preemption*, meaning that once a customer enters service they are not bumped out by a new arrival; they complete their service.

With a little thought, it is immediate that the stochastic process $\{L(t) : t \geq 0\}$ is the same for FIFO and LIFO. That is because the iid service times are generated only when they are needed, and the arrival process is the same; thus the jump times causing ± 1 changes of state for $L(t)$ are the same across both models. But the delays of individual customers are different. For example, consider D_2, the delay of C_2. Under LIFO If C_2 arrives finding C_1 still in service, then C_2 must wait in the line. But furthermore, unlike FIFO, if while C_2 is waiting, C_3 arrives, then C_2 must also wait until C_3 is served and so on. Under FIFO, however, C_2 only has to wait for C_1 to complete service. We will explain how to simulate LIFO D_2 momentarily, but first note that since $\{L(t) : t \geq 0\}$ is the same, we can obtain the same output for LIFO as we did for FIFO; but we must be careful to re-interpret the departure times $(t_1^{(d)}, \ldots, t_{N_A}^{(d)})$. Under LIFO, $d[i]$ = the time of the i^{th} departure, which is not necessarily $t_i^{(d)}$, the time that C_i departed. For example $d[2]$ might end up being the time at which C_3 departed, $t_3^{(d)}$. Under FIFO, customers depart in the same order that they arrived so indeed $d[i] = t_i^{(d)}$ is the departure time of C_i.

To obtain LIFO delays, we need to *tag* each customer so as to keep track of who they are and collect the times at which they *enter* service. Let us illustrate by simulating a copy of D_2. Simulate $\{L(t)\}$ until C_2 arrives at time t_2. At this point: If $n = 0$, then (since C_1 already departed) set $D_2 = 0$. Otherwise, if $n = 1$ (C_1 is still in service), then continue to simulate $\{L(t)\}$ under LIFO until finally C_2 enters service, at a time denoted by $t_2^{(s)}$. Then $D_2 = t_2^{(s)} - t_2$. The way to do this easily (when $n = 1$) is to imagine that C_2 must wait is a separate chair, and watch the system (including any new arrivals) until it empties. In other words, at time t_2 if $n = 1$, then we do not change the system state to $n = 2$; it remains at $n = 1$ and we simulate the system $\{L(t)\}$ under FIFO (not LIFO) but without C_2 participating, until for the first time in the future $n = 0$. We do not know apriori when this will occur, our termination time $t^* = t_2^{(s)}$ is now a random stopping time.

Algorithm for simulating D_2 under LIFO

Simulate $\{L(t) : t \geq 0\}$ using FIFO until time $t = t_2$. If $n = 0$, set $D_2 = 0$; otherwise, if $n = 1$ continue simulating under FIFO but do not allow C_2 to enter the line nor allow C_2 to increase n to $n + 1 = 2$; keep $n = 1$. Stop the simulation at time $t^* = \min\{t > t_2 : L(t) = 0\}$. Define $t_2^{(s)} = t^*$, define $D_2 = t-_2^{(s)}$. Stop. Output D_2.

1.2.1 Random Selection

Another discipline which can be handled similarly is RS (Random selection). Under RS, when a service completion occurs, the server randomly (equally likely) chooses a customer from the line to be served next. Once again, $\{L(t) : t \geq 0\}$ remains the same as for FIFO, but individual delays differ. In this case we would need to keep a record of the position in the line of each customer. If there are $k \geq 1$ in line right after a service completion, then the server will next choose the customer in position i with probability $1/k,\ 1 \leq i \leq k$. In this case, we label the positions in the line from 1 (head of the queue) downwards. When a customer in position i is chosen and enters service, we must move all customers below position i up by one position.

1.2.2 Processor Sharing (PS) GI/GI/1 queue

In this discipline, all customers enter service upon arrival (there is no line), but the server simultaneously processes all of them proportional to how many are in service: If there are $n \geq 1$ in service, then each one has its service time processed at rate $1/n$. Thus, workload, $\{V(t)\}$, is the same as for FIFO; the total rate at which work is processed is still 1 if $L(t) > 0$. Clearly, however, now both $\{L(t)\}$ and delays and such are completely different from FIFO.

PS is used as an approximation to the way that (say) web servers share processing among more than one job. A discrete-event simulation under PS involves keeping track over time of not only $n = L(t)$, but also the remaining service times (all in service) of these n customers. Given n at a given time t and the remaining service times, denoted by (say) $Y_1, \ldots, Y_n$, we know that Y_i will be completed at time $t + nY_i$ in the absence of any new arrivals or any departures. So each time an arrival or departure occurs, we need to update/re-schedule the service completion times of those remaining in service. In addition to simulating $\{L(t)\}$, it is of interest to simulate customer sojourn times. The sojourn time $W_j \overset{\text{def}}{=} t_j^{(d)} - t_j$, is the total amount of time that C_j spends in the system from arrival to departure. Even obtaining a copy of W_1 requires some non-trivial simulating: We place C_1 in service alone and them keep simulating under PS until C_1 finally departs. We will simulate this model in detail later.