Train Unloading System Simulation

MD. Irfanur Rahman Rafio
Department of Computer Science and Engineering
Islamic University of Technology
Dhaka, Bangladesh
ID: 190041125
irfanurrahmanrafio@gmail.com

***Abstract*—Train Unloading System simulation is a variation of the Single Server Queuing System where the practical scenario of unloading a train is simulated. Its properties, although quite similar to a regular SSQS, are a bit more complex. In this experiment, we simulate a Train Unloading System of our own and analyze its results.**

***Index Terms*—Train Unloading System, Hog-out**

I. Introduction

A Single Server Queuing System is a basic simulation model that can be used to predict the behavior of a system with a single server serving trains in a single queue. The only difference between a regular SSQS and the Train Unloading System is that in the Train Unloading System, the train is unloaded by a crew instead of a fixed server. The difference between the behavior of a crew and a server is that after a certain period, a crew leaves the system which can make the system idle despite having one or more trains waiting for service. This scenario is referred as the crew 'hogging out' which makes the Train Unloading System a bit more complex than the regular Single Server Queuing System by increasing the service time by a random amount.

II. System Description

A. Problem Statement

Coal trains arrive to an unloading facility with inter-arrival times that are independent and identically distributed (IID) random variables. If a train arrives and finds the system idle, the train starts unloading immediately. Unloading times for the trains are also IID random variables. If a train arrives to a busy system, it joins a FIFO queue.
However, the situation is complicated by what the railroad calls 'hogging out'. In particular, a train crew can work for a certain period of time, and a train cannot be unloaded without a crew present. When a crew's time expires, it leaves immediately and a replacement crew is called. The amount of time between when a replacement crew is called and when it actually arrives is independent and identically distributed as well.
If a train is being unloaded when its crew hogs out, unloading is suspended until a replacement crew arrives. If a train is in queue when its crew hogs out, the train cannot leave the queue until its replacement crew arrives. Thus, the unloading equipment can be idle with one or more trains in queue.

B. Input Data

In our simulation, it is given that the inter-arrival times are distributed exponentially having mean 10 hours. Unloading times for the trains are independent and distributed uniformly between 3.5 and 4.5 hours. When a train arrives, the remaining crew time (out of 12 hours) is independent and distributed uniformly between 6 and 11 hours. The amount of time between when a replacement crew is called and when it actually arrives is independent and distributed uniformly between 2.5 and 3.5 hours.
The simulation program will run for 30 days (720 hours).

C. State Variables

The state of the system in a certain point of time t is represented by two variables: server status x(t) and queue length q(t).

$$x(t) = \begin{cases} 0, & \text{if server is idle at time t} \\ 1, & \text{if server is busy at time t} \end{cases}$$

$$q(t) = \text{queue length at time t}$$

D. State Space

Unlike the regular SSQS, here the system status of the server can be idle despite having one or more trains waiting for service. Because of this, the state space can be the cross product of the set of possible system status and the set of possible queue lengths.

$$\text{State Space, } X = \{0, 1\} \times (\{0\} \cup \mathbb{N})$$

E. Event Set

The system has the two events from the Single Server Queuing System: Arrival(a) and Departure(d). Arrival either changes the server status from idle to busy, or increases the queue length. Departure either changes the server status from busy to idle, or decreases the queue length.
An extra event Termination(t) is required to end the simulation since the duration of the simulation is given a terminating condition instead of the number of trains.

$$\text{Event Set, } E = \{a, d, t\}$$

F. Feasible Event Set

When the system is idle, Departure event is not possible. Otherwise, any event is feasible in any point of time.

$$\text{Feasible Event Set, } F(t) = \begin{cases} \{a, t\}, & \text{if } x(t) = 0 \\ \{a, d, t\}, & \text{if } x(t) = 1 \end{cases}$$

G. State Equations

The state equations of the system:

$$x(t^+) = \begin{cases} x(t) == 0 ? \, 1 : x(t), & \text{arrival at time t} \\ q(t) == 0 ? \, 0 : x(t), & \text{departure at time t} \\ 0, & \text{termination at time t} \\ x(t), & \text{otherwise} \end{cases}$$

$$q(t^+) = \begin{cases} x(t) == 0 ? \, 0 : q(t) + 1, & \text{arrival at time t} \\ q(t) == 0 ? \, 0 : q(t) - 1, & \text{departure at time t} \\ 0, & \text{termination at time t} \\ q(t), & \text{otherwise} \end{cases}$$

H. Statistical and Output Variables

1) Variables associated with train i:

$$\text{arrival time} = a_i \geq 0$$
$$\text{service start time} = b_i \geq a_i$$
$$\text{queue delay} = d_i = b_i - a_i$$
$$\text{hog-out delay} = h_i \geq 0$$
$$\text{service time} = s_i \geq h_i \geq 0$$
$$\text{system delay} = w_i = d_i + s_i$$

2) Variables associated with system:

$$\text{total run-time} = T$$
$$\text{traffic intensity} = \rho = \lambda/\mu$$
$$\text{server utilization} = \bar{u}$$

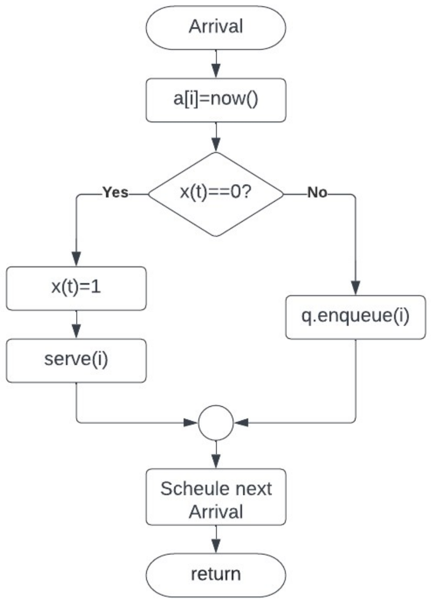
I. Output Equations

Output equations for measuring the performance of the system:

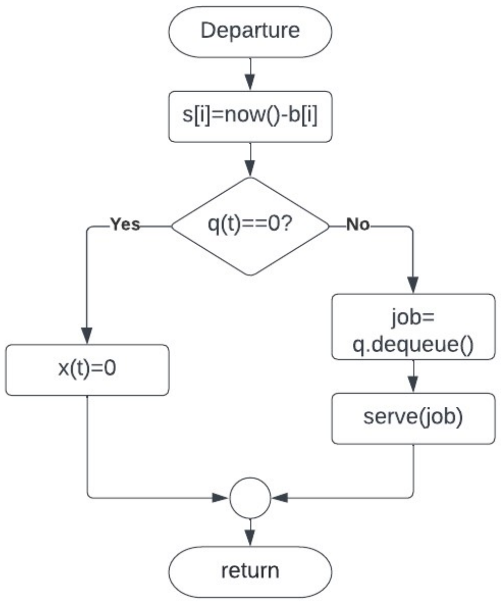
$$\text{maximum system delay, } w_{max} = \max_{i=1}^{n}(w_i)$$
$$\text{average system delay, } \bar{w} = \frac{\sum_{i=1}^{n}(w_i)}{n}$$
$$\text{maximum queue length, } q_{max} = \max_{i=1}^{n}(q_i)$$
$$\text{average queue length, } \bar{q} = \frac{\sum_{i=1}^{n}(q_i)}{n}$$
$$\text{busy ratio, } \bar{u} = \int_0^T x(t)dt$$
$$\text{idle ratio, } 1 - \bar{u} = \int_0^T (1 - x(t))dt$$
$$\text{hogged out ratio, } \bar{h} = \frac{\sum_{i=1}^{n}(h_i)}{T}$$

J. Event Routines

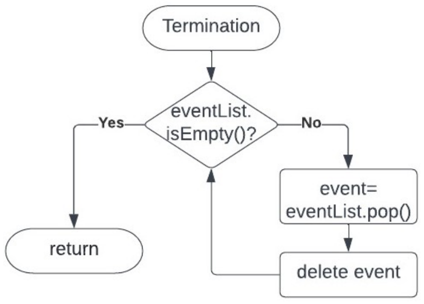
1) Arrival Event: When a new train arrives in the system, we save its arrival time. Then we check if the server is idle. If it is, we change its status to busy and we immediately serve the train. If not, we add the train to the queue. After that we schedule the arrival of the next train. For determining its time, we apply the inverse transform method using the inter-arrival rate as a parameter.



2) Departure Event: After the service of a train ends, we save its service time. Then we check if the queue is empty. If the queue is found empty, there is no train to unload at the moment. So, we change the server status to idle. Otherwise, we pick the first train from the queue and start serving it.

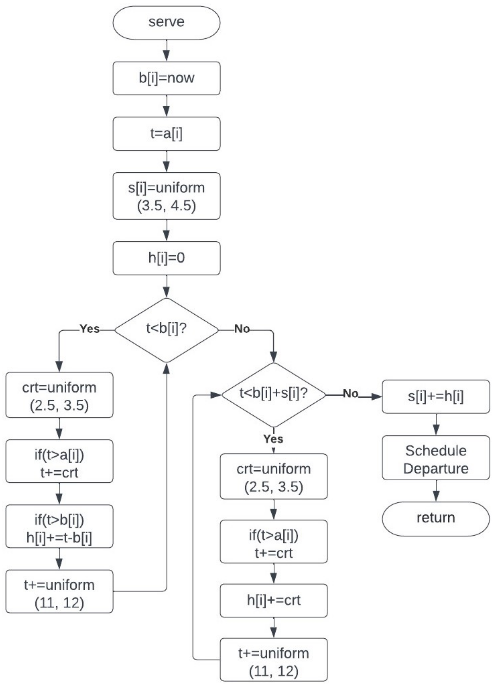


3) Termination Event: When triggered, this event deletes all the other events from the simulation without giving them any chance to call the handle function.

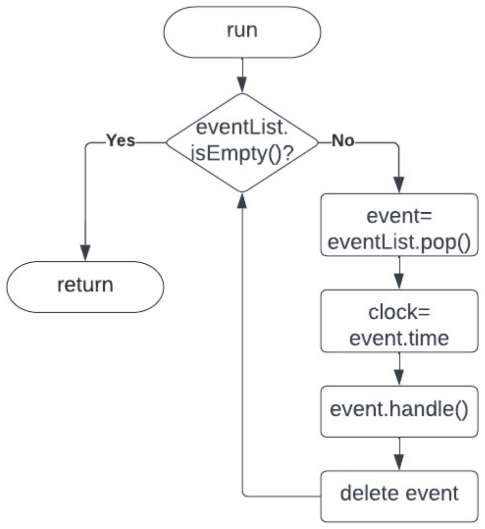


III. Simulation Program Description

Just like the Single Server Queuing System program, this program consists of three major classes: Simulator, Server and Event. Minor classes include the Train class, a FIFO Queue and a min Heap.
The event class is abstract. Its child classes represent the arrival, departure and termination event. Each of those classes have a handler function which manipulates the state of the Server.
The server class consists of the system variables, the queue, and the statistical variables. Handling an event involves updating the statistical variables and creating changes to the system variables. There is a utility function in the Server class that performs the task of serving the trains. This serve function represents the main difference between our system and a regular SSQS.



Finally, we have the Simulator class which mainly comprises of the event list and system clock. The event list is a min heap where the upcoming events are stored. The Simulator has a run function which pops an event from the event list and handles it. In every occurrence of a new event, the clock is updated.



The full source code of the simulation program can be found here: https://github.com/rafio-iut/Simulation-and-Modeling-Lab/tree/main/Train%20Unloading%20System

IV. Result and Analysis

TABLE I
Simulation Results: Hog-out phenomena Enabled

Max Delay	18.5791 hours
Average Delay	5.91063 hours
Max Queue Length	2.96667
Average Queue Length	0.387054
Busy Ratio	0.393138
Idle Ratio	0.606862
Hogged out Ratio	0.18047

The simulation results are taken by running the simulation 30 times. For analysis, we simulate the result for a system without the hog-out phenomenon. By comparing the two tables, we can understand that by increasing the service time of the trains, the hog-out phenomenon increase the queue lengths and delays drastically.

TABLE II
Simulation Results: Hog-out phenomena Disabled

Max Delay	11.2543 hours
Average Delay	3.95656 hours
Max Queue Length	1.83333
Average Queue Length	0.120232
Busy Ratio	0.386336
Idle Ratio	0.613664
Hogged out Ratio	0

V. Conclusion

The Train Unloading System shows that the results of a practical scenario can deviate from an ideal one because of a few small adjustments. It is a good step for us to move forward from simple systems like the Single Server Queuing System to more complex ones.