MIDTERM #1

Single Server Queue

Charles Acevedo Díaz
System Engineering
Universidad Tecnológica de Bolívar
Cartagena, Colombia
chad9591@gmail.com

I. ABSTRACT

In this paper the main topic will be a single server queue and an algorithm will be used to calculate the delay time of a given set of jobs. Then the results will be analyzed using some statistics metrics that will allow us to make decisions in order to improve the functionality of the system.

keywords: single-server, queue, FIFO, simulation, discrete, events

II. INTRODUCTION

The simulation of a system is a powerful tool that allow us to check how a system would behave without affecting it, as it is replicated on a model, and based on the simulation we may have an idea of its behavior and how to manage in the correct way the system

This assignment intends to implement in a practical way an algorithm to calculate de delay time of a set of jobs that are given in a single file, the main idea is to obtain the necessary information to calculate some statistics metrics to check if the server is enough for the amount of jobs that arrives.

The file contain 1000 registers of jobs and each register has the *arrive time* and the *service time.* Just enough to calculate information like delay time, departure time, and more, so we can get the statistics.

III. THEORY

Single-server node

By definition, a single server node is a system that only has a server and its queue. The jobs arrive at certain time and, depending on the queue and its behavior they are attended; in this assignment we assume that there are not preference and it behaves as a FIFO queue.

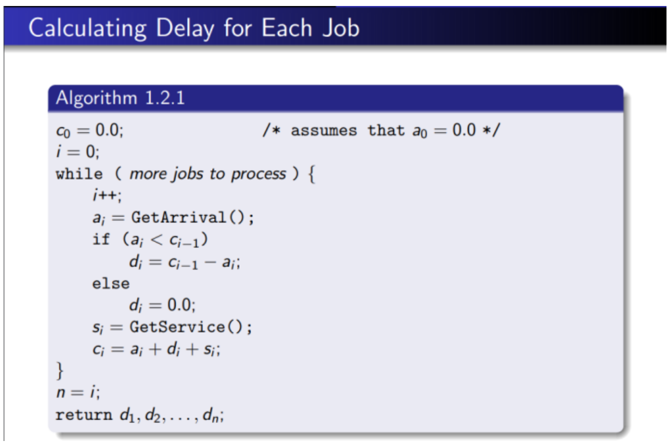
The variables that are involved in this system are:

- Arrival time (a)
- Delay time (d)
- Begin time (b)
- Service time (s)
- Wait time (w)
- Departure time (c)

Besides, the statistics metrics are:

- Average interarrival time
- Average service time
- Average wait time
- Average delay time

IV. ALGORITHM



Calculating Delay for Each Job

Algorithm 1.2.1

```
$c_0 = 0.0$;                /* assumes that $a_0 = 0.0$ */
$i = 0$;
while ( more jobs to process ) {
    $i$++;
    $a_i$ = GetArrival();
    if ($a_i < c_{i-1}$)
        $d_i = c_{i-1} - a_i$;
    else
        $d_i = 0.0$;
    $s_i$ = GetService();
    $c_i = a_i + d_i + s_i$;
}
$n = i$;
return $d_1, d_2, \ldots, d_n$;
```

In the above image we can appreciate how the algorithm will work in a general way, and it will iterate over the number of jobs and will calculate the departure time based on the arrival, delay and service time. As the delay time is not given in the file, it is calculated using the previous departure time (which starts at zero for i = 0) and the arrival time, then it returns the delay time for each job as an array.

V. RESULT

A. *Job-averaged statistics*

Metric	Value
Interarrival Time	4909.7305 unit of time per job
Arrival rate	0.00020 jobs per unit time
Service time	7.124851 unit time per job
Service rate	0.14035 jobs per unit time
Delay time	18.58574 unit time per job
Wait time	25.71059 unit time per job

B. *Time-averaged statistics*

Metric	**Value**
Time used in service node	25710.594 unit time
Time used in queue	18585.743 unit time
Time used in service	7124.85 unit time

VI. CONCLUSSION

Using the statistics metrics, we are going to make some conclusions so we can know whether if the system is working *ok* or if it needs to be improved.
Based on the arrival time, we can see that there are not too many jobs arriving per unit of time, and we could easily say that the system is working fine, but if we consider the delay time and the wait time we see that the wait time is way too high in comparison with service time. We can see that the wait average time is around 26 unit time per job, while the service time is 7 unit of time per job. This could be due to the average delay time that is around 19 units of time which means that jobs expend more time in queue than being served.
Also, if we take a look at the time that was used in service node and each part of it, we see that most of its time is in queue, which means, the jobs has to wait a lot before being served. So, it may be necessary to implement a second server to the system.

VII. REFERENCES

[1] Lawrence Leemis, “DISCRETE-EVENT SIMULATION: A First Course” / december 1994.

[2] Class notes.