Computer System Performance Analysis: Simulation and Modeling

Taleb A. S. Obaid

Department of Computer Science, University of Basra, Basra, IRAQ.

e-mail: tasobaid@yahoo.com

Abstract

A process is a program in execution. As processes enter the system, they are put into a job queue, which consists of all processes in the system. The operating system must select processes from those queues in some fashion called scheduling. There are many queuing algorithms to select the next process. A process terminates when it finishes executing its final statement.

In this study, we implemented a discrete-event simulation techniques and applied it to three different queuing disciplines. Computational simulation provides a decisions-maker with some information alternatives. According to the statistical criteria we could identify the best discipline that fits to the problem. We got some simulation results which are quite interesting and help understand the nature of the scheduling algorithms of the operating systems. The selection of the appropriate discipline among others is quite complex and it is dependent on the nature of the environment of the problem and on the problem's data.

INTRODUCTION

Queuing analysis is one of the most important tools for those involved in computer and network analysis. The number of questions that can be addressed with a queuing analysis is endless and touches on virtually every area in computer science. The ability to make such an analysis is an essential tool for those involved in this field.

Although the theory of queuing is mathematically complex, the application of queuing theory to the analysis of performance is, in many cases, remarkably straightforward. Knowledge of elementary statistical concepts (means and standard deviations) and a basic understanding of the applicability of queuing theory is all that is required. Armed with these, the analyst can often make a queuing analysis on the back of an envelope using readily available queuing tables, or with the use of simple computer programs that occupy only a few lines of code.

Time-sharing and multiprogramming require several jobs to be kept simultaneously in memory. Since in general main memory is too small to accommodate all jobs, the jobs are kept initially on the disk in the job pool. This pool consists of all processes residing on disk awaiting allocation of main memory. If several jobs are ready to bring into memory, and if there is not enough room for all of them, then the system must choose among them. Making this decision is job scheduling, (Silberschatz, 2005).

In a single-processor system, only one process can run at a time; any others must wait until the CPU is free and can be rescheduled. We assume that no delay is involved from the time a server becomes available and the time service is started on an entity that was waiting in the queue[*]. When an entity waits at a queue, it is stored in a file which maintains the entity's attributes and the relative position of the entity with respect to other entities waiting at the same queue. The order in which the entities wait in the queue is specified outside the network on a priority statement which defines the ranking rule for the file associated with the queue (Alan, 1999). Files can be ranked on: First-Come First Served (FCFS); Last-Come First Served (LCFS); Shortest Job First (SJF); High-Value First (HVF); Round-Robin Scheduling (RRS); and Priority Scheduling (PS). FCFS is a default scheduling discipline, for details see (Stalling, 2000), and (Taha 2002).

In discrete simulation, the state of the system can change only at event times. Since the state of the system remains constant between events times a complete dynamic portrayal of the state of the system can be obtained by advancing simulated time from one event to the next. This timing mechanism is referred to as the next event approach and is used in most discrete simulation languages.

A discrete simulation model can be formulated by: 1) Defining the changes in state that occur at each event time; 2) Describing the activities in which the entities in the system engage; or 3) Describing the process through which the entities in the system flow. An event takes place at an isolated point in time at which decisions are made to start or end activities. A process is a time-ordered sequence of events and may encompass several activities.

1- The Single-Server Queue

The simplest queuing system is depicted in Figure 1. The central element of the system is a server, which provides some service to items. Items from some population of items arrive at the system to be served. If the server is idle, an item is served immediately. Otherwise, an arriving item joins a waiting line**. When the server has completed serving an item, the item departs. If there are items waiting in the queue, one is immediately dispatched to the server. The server in this model can represent anything that performs some function or service for a collection of items. Examples: a processor provides service to processes; a transmission line provides a transmission service to packets or frames of data; an I/O device provides a read or write service for I/O requests, (Stallings, 2000).

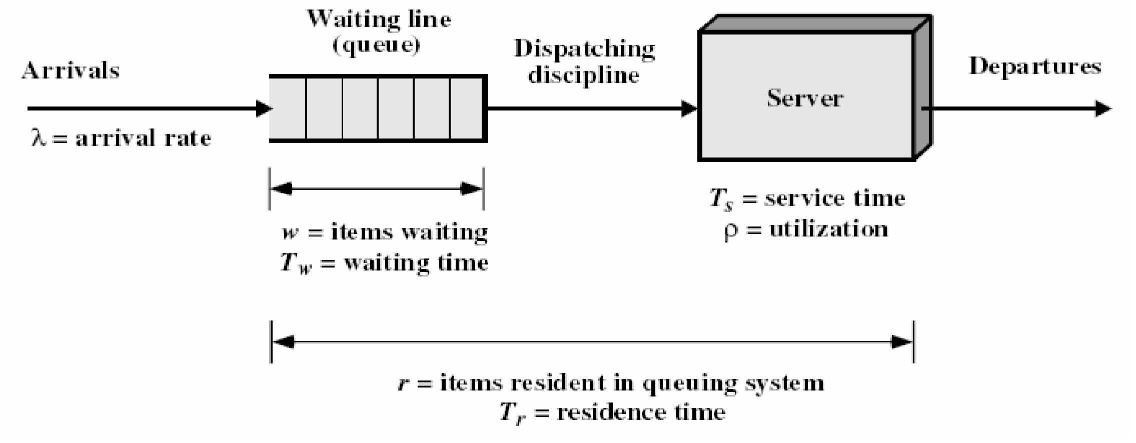

Figure 1, Queuing System Structure and Parameters for Single-Server Queue

2- Queue Parameters

Figure 1, also illustrates some important parameters associated with a queuing model. Items arrive at the facility at some average rate (items arriving per second) λ. At any given time, a certain number of items will be waiting in the queue (zero or more); the average number waiting is *w*, and the mean time that an item must wait is T_w. T_w is averaged over all incoming items, including those that do not wait at all. The server handles incoming items with an average service time T_s; this is the time interval between the dispatching of an item to the server and the departure of that item from the server. Utilization, ρ, is the fraction of time that the server is busy, measured over some interval of time. Finally, two parameters apply to the system as a whole. The average number of items resident in the system, including the item being served (if there is any) and the items waiting (if there is any), is *r*; and the average time that an item spends in the system, waiting and being served, is T_r; we refer to this as the mean residence time, for more information see (Stalling 2000).

If we assume that the capacity of the queue is infinite, then no items are ever lost from the system; they are just delayed until they can be served. Under these circumstances, the departure rate equals the arrival rate. As the arrival rate, this is the rate of traffic passing through the system, increases, the utilization increases with it, congestion. The queue becomes longer, increasing waiting time. At ρ = 1, the server becomes saturated, working 100% of the time. Thus, the theoretical maximum input rate that can be handled by the system is:

$$\lambda_{max} = 1 / T_s$$

for more details, see (Stalling, 2000).

However, queues become very large near system saturation, growing without bound when ρ = 1. Practical considerations, such as response time requirements or buffer sizes, usually limit the input rate for a single server to 70-90% of the theoretical maximum. To proceed, we need to make some assumptions about this model:

Item population: Typically, we assume an infinite population. This means that the arrival rate is not altered by the loss of population. If the population is finite, then the population available for arrival is reduced by the number of items currently in the system; this would typically reduce the arrival rate proportionally.

Queue size: Typically, we assume an infinite queue size. Thus, the waiting line can grow without bound. With a finite queue, it is possible for items to be lost from the system. In practice, any queue is finite. In many cases, this will make no substantive difference to the analysis. This issue is addressed briefly below.

Dispatching discipline: When the server becomes free, and if there is more than one item waiting, a decision must be made as to which item to dispatch next. The simplest approach is first-in, first-out; this discipline is what is normally implied when the term queue is used. Another possibility is last-in, first-out. One that you might encounter in practice is a dispatching discipline based on service time. For example, a packet-switching node may choose to dispatch packets on the basis of shortest first (to generate the most outgoing packets) or longest first (to minimize processing time relative to transmission time). Unfortunately, a discipline based on service time is very difficult to model analytically, (Tanenbaum, 2006).

Process Concept

A process is a program in execution. As a process executes, it changes state (process's current activity). Each state may be in one of the following states:

- New: The process is being created.
- Running: Instructions are being executed.
- Waiting: The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
- Ready: The process is waiting to be assigned to a processor.
- Terminated: The process has finished execution.

When more than one process is in the ready state and there is only one CPU available, the operating system must decide which process to run first. The part of the operating system that makes the choice is called the scheduler; the algorithm it uses is called the scheduling algorithm. When a process is not executing, it is placed in some waiting queue, for more details see (Silberschatz, 2005).

Scheduling Algorithms

As processes enter the system, they are put into a job queue, which consists of all processes in the systems. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called the ready queue. The operating system must select processes from these queues in some fashion. The selection process is carried out by the appropriate scheduler. CPU scheduling deals with the problem of deciding which of the processes in the ready queue are allocated the CPU. There are many different CPU scheduling algorithms available, see (Silberschatz, 2005). Some of these algorithms, which are implemented in this study, are described below.

First-Come First-Served (FCFS) Scheduling

The simplest of all scheduling algorithms is first-come, first-served. With this scheme, the process that requests the CPU first is allocated the CPU first. When a process enters the ready queue, its PCB (Process Control Block) is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue. The great strength of this algorithm is that it is easy to understand and easy to program. The average waiting time under the FCFS policy is often quite long, (Taha, 2002)

Shortest Job-First (SJF) Scheduling

This algorithm associates with each process the length of the process's next CPU burst. When the CPU is available, it is assigned the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie. The SJF scheduling algorithm is provable optimal, in that it gives the minimum average waiting time for a given set of processes. Moving a short process before long one decrease the waiting time of the short process more than it increases the waiting time of the long process. Consequently, the average waiting time deceases.

Priority Scheduling

Shortest-Job First algorithm makes the implicit assumption that all processes are equally important. Frequently, the people who own and operate multiuser computers have different ideas on that subject. At university, for example, the pecking order may be deans' first, then professor, secretaries, janitors, and finally students. The need to take external factors into account leads to priority scheduling. The basic idea is straightforward; each process is assigned a priority, and the runnable process with highest priority is allowed to run.

A priority is associated with each process, and the CPU is allocated to the process with the highest priority. Equal-priority processes are scheduled in FCFS order. Priorities are generally indicated by some fixed range of number, such as 0 to 5. However, there is no general agreement on whether 0 is the highest or lowest priority. Some systems use low numbers to represent low priority; others use low numbers for high priority. In this study, we assume that low numbers represent high priority.

Computer Simulation

Simulation has several different meanings. For some people, simulation is the greatest modeling and analysis procedure available for problem solving. For others, simulation connotes a sense of unreality, that is, a fake. However, simulation involves abstracting a real system into something that is not real; this transformation can be accomplished logically, efficiently and effectively.

In its broadest sense, computer simulation is the process of designing a mathematical-logical model of a real system and experimenting with this model on a computer. Thus simulation encompasses a model building process as well as the design and implementation of an appropriate experiment involving that model. These experiments, or simulations, permit inferences to be drawn about systems without building them, without disturbing them, and without destroying them. So, in this way, simulation models can be used for design, procedural analysis and performance assessment, for more details about simulation see (Alan, 1999).

In general, simulation involves the generation of an artificial history of a system and observation of that artificial history to draw inferences concerning the operating characteristics of the real system. The simulation model usually takes the form of a set of assumptions concerning the operation of the system. A validated model can be used to investigate a wide variety of "what if" questions about the real-world system. In some instances, a model can be solved by mathematical methods. Many real-world systems are so complex that models are virtually impossible to solve mathematically. In these instances, computer-based simulation can be used to imitate the behavior of the system over time. This simulation-generated data is used to estimate the measures of performance of the system, see (Bank, 2005).

Queuing models provide a powerful tool for designing and evaluating the performance of queuing systems. Typical measures of system performance include server utilization, length of waiting lines, and delays of jobs. Queuing theory and simulation analysis are used to predict these measures of system performance as a function of the input parameters. The input parameters include the arrival rate of jobs, the service demands of jobs, the rate at which a server works, for more details see (Obaid, 2004), (Obaid, 2005), and (Bank, 2005).

Measures Performance

The objective is to develop a simulation model that can be used to analyze the single runway situation in term of the following statistics relationships, see Banks [4].

$$\text{Average Utilization} = \frac{\Sigma \text{ Service Time}}{\text{Length of Simulation}} \qquad \ldots\ldots\ldots\ldots \quad (1)$$

$$\text{Average Service Time per Airplane} = \frac{\Sigma \text{ Service Time}}{\text{No. of Airplanes Served}} \qquad \ldots\ldots\ldots\ldots \quad (2)$$

$$\text{Average Waiting Time per Airplane} = \frac{\Sigma \text{ Waiting Time}}{\text{No. of Airplane served}} \qquad \ldots\ldots\ldots\ldots \quad (3)$$

$$\text{Average Content of Waiting Time} = \frac{\Sigma \text{ Total Waiting Time}}{\text{Length of Simulation}} \qquad \ldots\ldots\ldots\ldots \quad (4)$$

$$\text{Percentage of Time Runway (Server) is idle} = \frac{\Sigma \text{ Total Idle Time}}{\text{Length of Simulation}} \qquad \ldots\ldots\ldots \quad (5)$$

$$\text{Average number of Airplanes in queue} \quad L_q = \frac{\Sigma m_i t_i}{T} \qquad \ldots\ldots\ldots\ldots \quad (6)$$

m_i is a Number of Customer in queue of interval t_i , T is the total simulation time.

$$\text{Average number of Airplanes in System} \quad L_s = \frac{\Sigma n_i t_i}{T} \qquad \ldots\ldots\ldots\ldots \quad (7)$$

n_i is a Number of Customer in system of interval t_i

Results and Discussion

The main objectives of implementing simulation technique are to provide a decision maker with some alternative information of the problem under study. The decision maker would be interested in some statistical performances measurement such as average utilization, total average service time, average waiting time , average content of waiting time, percentage of processor idle time, and average number of jobs in queue, for details see (Bank, 2005) and (Obaid, 2005). The focus in this study is on the three different scheduling disciplines (First-Come First-Served, Shortest Job-First, and Priority Scheduling).

In order to illustrate the manner in which the simulation calculations are carried out, we specified the job arrival time and service time of the first ten jobs randomly, as shown in the second and third columns in Table 1, Table 2, and Table 3. The comparison performance of those three scheduling disciplines will be shown in Table 4. The three tables show the simulation computation information of the: Begin-Time, Finish-Time, and Waiting-Time for each job besides the Arrival-Time and Service-Time.

Table 1 shows the simulation computations of the "First-Come First-Served (FCFS)" schedule discipline. The Waiting-Time for each job in the facility is calculated as the difference of its arrival time (Arrival-Time) and its starting services time (Begin-Time). The statistical performance measurement of FCFS will be shown in Table 4 together with other disciplines.

The simulation computation of the second schedule discipline "Shortest-Job First" is shown in Table 2. In this discipline the concern is with the service times of all the jobs first before selecting the next job. Since the facility is free to start the simulation, so, we don't care about the time needed for the first job. That is, the first job is served immediately once it arrive the facility. To process other jobs we have to scan the service time of all the jobs that arrived the facility before the first job is completed and select the job with minimum service time. According to our assumed data entry, the first job is being served at time 5 and finished at time 25. The next selected job will be number 2, since this job has a minimum service time among the remaining jobs in the facility arrived so far. However, job number 2 is selected and being processed at time 25 and finished at time 26.

We may notice that job number 7 is served before other jobs that already arrived to the facility, as shown in the Table 2. Thus, job number 7 should be served before jobs number 3, 4, 5, and 6 because its service time is the lowest among them. So, job number 7 is served at time 26 and finished at time 27. Next, we have to select job with the lowest service time among other jobs that arrived the facility. Job number 3 has been selected and started at 27 and then finished at 29. According to the same fundaments, job number 5 started before job number 4. Job 5 is started at time 29 and finished at time 31. Processing job number 4 is postponed because its needed the highest service time among the other jobs in the facility. Thus, it will be processed at the end. Follow the same principle to select the remaining other jobs until the simulation session will be over and all job got the service, as shown in the Table 2.

Table 3 shows the simulation computations of the "Priority" discipline. The priority of each job has been assigned randomly and appears in the fourth column of Table 3. We started our simulation computations by selecting the first arrival job once it arrived the facility. Job number 1 is started at time 5 and finished at time 25. To choose the job with highest priority, we have to scan the entire arrived job's priority. According to the data given in the third column we select job number 7, because it has the highest priority "4" among other jobs that arrived the facility. So, job 7 has been processed immediately after job number 1 is completed, i.e., started at time 25 and finished at time 26. Again job 4 has been selected once job number 7 is completed, i.e., started at time 26 and finished at time 36. Process other jobs according to their priorities shown in the Table 3.

Now we are interested to investigate the performance of those three different queuing disciplines. Our investigations focus on some statistical performances such as the Total-Waiting time of the facility, Total-Idle time that the facility is free, etc. The computational results of the "First-Come First-Served" discipline, "Shortest Job-First", and "Priority Scheduling" are given in the first, second, and third columns of Table 4, respectively. The Total-Waiting time is the most significant criterion among others of the three disciplines. The best discipline is the one with smallest Total-Waiting time. As we expected the "Shortest-Job First" discipline leads to dramatic reduction in waiting time as long as some little jobs come behind the big jobs. Waiting-Time reduces from 142 to 88, as shown in the Table 4 and Figure 1.

Figure 2, compares the waiting time for both FCFS and SJF. The figure shows the beneficial of using SJF because of the reduction in the waiting time. But, of course, when all the jobs have almost the same service times, then the SJF has no advantages on the FSFC and the performance are the same.

Our objective in the third discipline "Priority Schedule" discipline is to reduce the waiting time for all the critical (the most important) jobs that have higher priorities and execute them without further delay. The less important jobs will be postponed until the facility processes all important jobs. Figure 3 shows the comparison of

FCFS and Priority approach. Job number 7 has the highest priority job (4) so, has zero waiting time. The other jobs executed according to their priorities. Again when all the jobs have the same priorities, then this discipline is treated as the "First-Come First-Served" discipline.

Figure 4 shows the waiting time of the SJF. We could easily notice the waiting time of the short jobs (needs a little service time) is very small compared with other jobs. Figure 5 shows the waiting time of the priority approach. We notice the jobs with higher priority executed first. Job number 7 is executed first and has no waiting time.

Finally, the selection of the appropriate discipline among others is quite complex and it is dependent on the nature of the environment of the problem and on the problem's data. Simulation calculation provides the decision-makers with wide alternative information for each situation to select the best discipline that is convenient to the problem under study.

References:

Alan, A. "*Simulation with Visula SLAM and AweSim*". John Wiley & Dons, 1999.

Bank, J., Carson, J., Nelson, B., and Nicol D. "*Discrete-Event System Simulation*", Prentice Hall, 2005.

Law, A. M. & Kelton, W. D. "*Simulation Modeling and Analysis*", New York, McGraw-Hill, 2000.

Obaid T. "Computer Simulation for Decision Making using Aspiration Cost Model" Abhath Al-Yarmouk Journal, Pure Science and Engineering Series, Yarmouk University, Irbid-Jordan, Vol. 14, No.2, 2005, pp289-297

Obaid T. "Computer Simulation of a Heat Transfer Using Mone Carlo Technique", J. of Science Faculty AlManufia University, 14 (2004) 123-139.

Silberschatz A., Gragne G., and Galvin P., "*Operating System Concepts*". 7th ed. John Wiley & Sons, 2005.

Stalling, W. "Queuind Analysis", williamStalling.com/studentsuport.html. 2000.

Taha, Hamdy A., "*Operations Research* Prentice-Hall", 7th edition 2002.

Tanenbaum A. and Woobaum A., "*Operating Systems Design and*

Implementation", 3rd ed. Prentice Hall, 2006.

Table 1, "First-Come First-Served" Simulation Discipline

No	Arrival Time	Service Time	Begin Time	Finished Time	Waiting Time
1	5	20	5	25	0
2	6	1	25	26	19
3	10	2	26	28	16
4	11	10	28	38	17
5	15	2	38	40	23
6	24	2	40	42	16
7	25	1	42	43	17
8	31	2	43	45	12
9	34	1	45	46	11
10	35	1	46	47	11

Table 2, "Shortest Job First" Simulation Discipline

No	Arrival Time	Service Time	Begin Time	Finished Time	Waiting Time
1	5	20	5	25	0
2	6	1	25	26	19
3	10	2	27	29	17
4	11	10	37	47	26
5	15	2	29	31	14
6	24	2	31	33	7
7	25	1	26	27	1
8	31	2	33	35	2

9	34	1	35	36	1
10	35	1	36	37	1

Table 3, Priority Simulation Discipline

No	Arrival Time	Service Time	Job Priority	Begin Time	Finished Time	Waiting Time
1	5	20	1	5	25	0
2	6	1	1	36	37	30
3	10	2	1	37	39	27
4	11	10	3	26	36	15
5	15	2	1	39	41	17
6	24	2	1	41	43	17
7	25	1	4	25	26	0
8	31	2	1	43	45	14
9	34	1	1	45	46	12
10	35	1	1	46	47	11

Table 4, Comparison Performance of three Scheduling Disciplines

	First-Come First-Served Discipline	Shortest-Job Discipline	Priority Discipline
Total Service Time	42	42	42
Total Simulation Time	47	47	47
Total Idle Time	10.6 %	10.6 %	10.6 %
Total Busy Time	89.4 %	89.4 %	89.4 %
Total Waiting Time	**142**	**88**	**143**

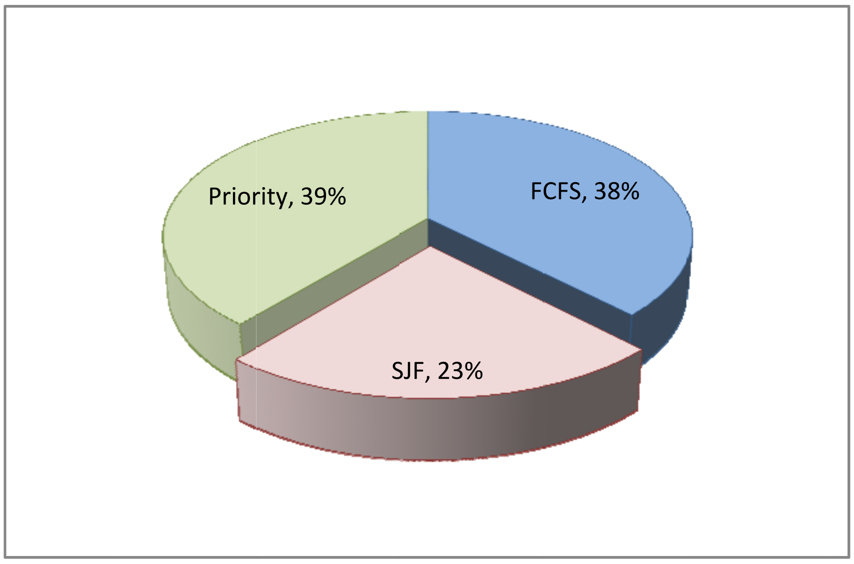

Figure 1, the waiting time of the FCFS, SJF, and Priority approaches.

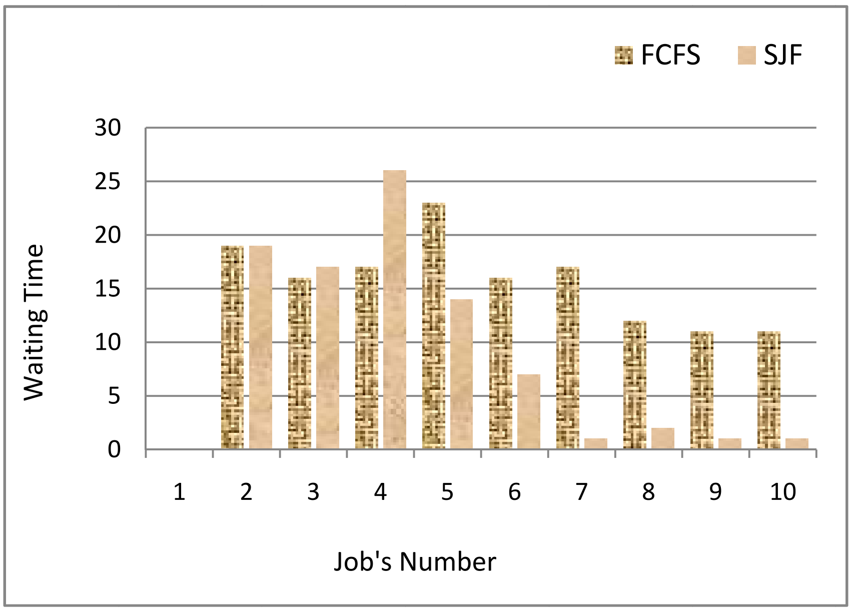

Figure 2, the waiting time for FCFS and SJF

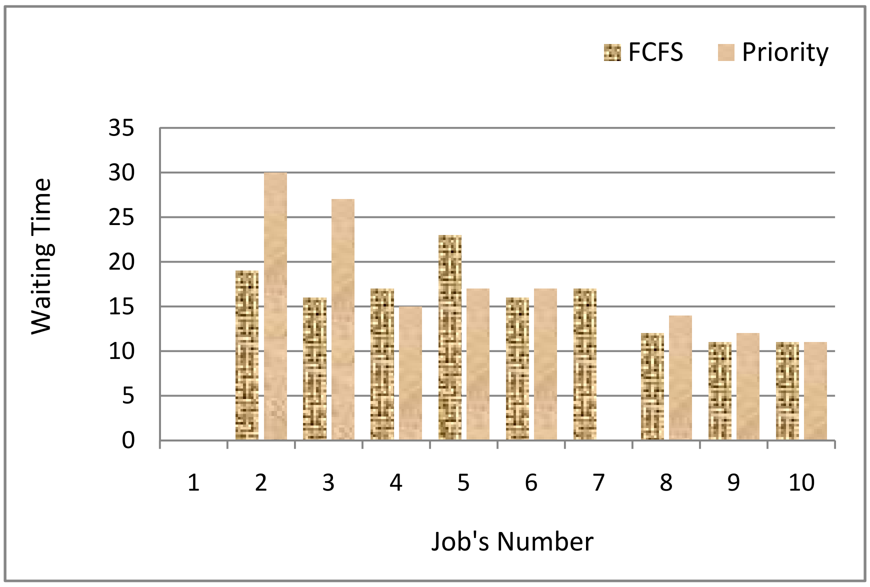

Figure 3, the waiting time of FCFS and Priority Approaches

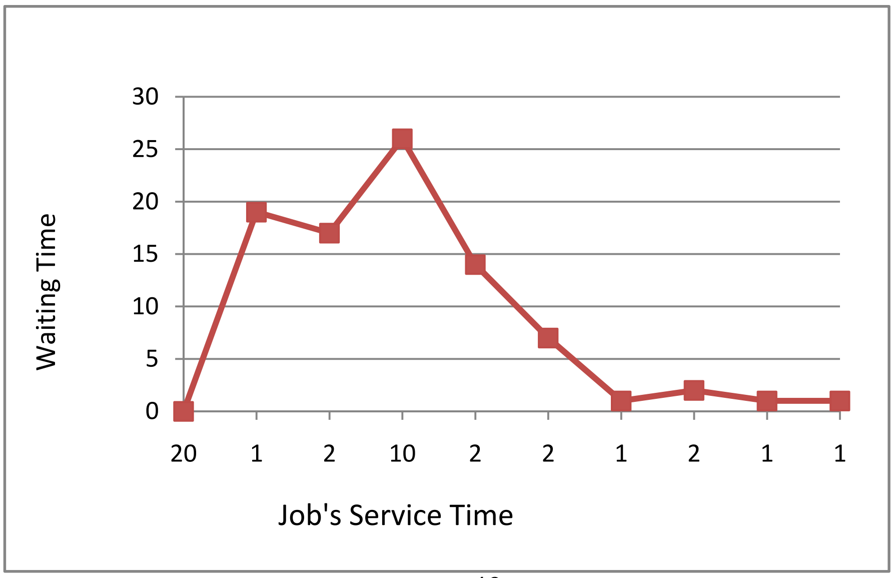

Figure 4, the waiting time of the SJF approach

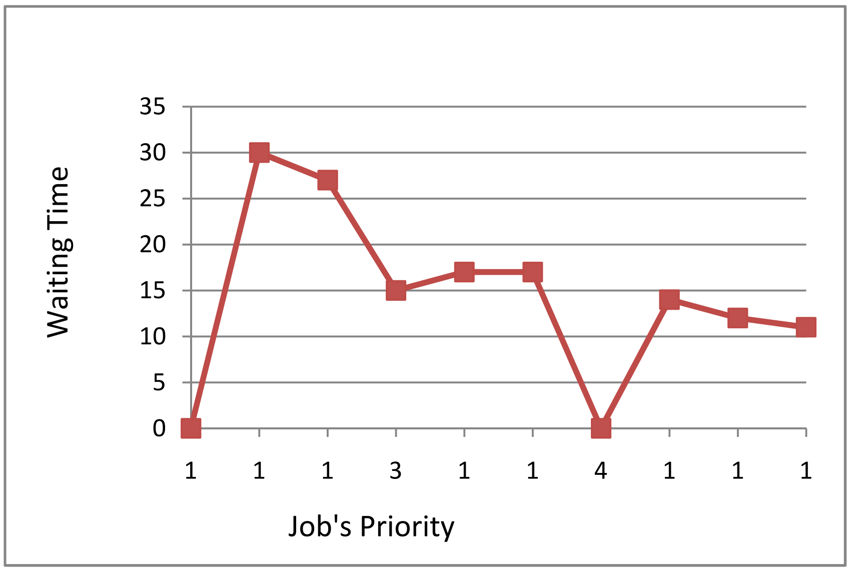

Figure 5, the waiting time of the Priority approach

تحليل أداء نظام الحاسب: نمذجة و محاكاة

طالب عبيد

قسم علوم الحاسبات- جامعة البصرة - محافظة البصرة

جمهورية العراق

e-mail: tasobaid@yahoo.com

المستخلص:

تعرف العملية (process) في نظام الحاسب بأنها برنامج في حالة التنفيذ. في نظام مفرد المعالج تصطف العمليات التي تصل النظام لطلب الخدمة في طابور بانتظار تقديم الخدمة لها. نظام التشغيل في الحاسب يختار احد تلك العمليات من طابور الانتظار ضمن أسلوب معين يسمى الجدولة. توفر لنا نظرية الطوابير جملة التقنيات (الخوارزميات) لأسلوب اختيار العملية اللاحقة. العملية تغادر الطابور و تنتهي حال حصولها على الخدمة المطلوبة.

تم في هذه الدراسة استعمال تقنية النمذجة والمحاكاة و تطبيقها على ثلاثة نظم طوابير مختلفة. توفر حسابات النمذجة والمحاكاة بدائل متعددة لمساعدة متخذي القرارات في تبني أفضل البدائل المتوفرة. تم تقييم نتائج كل من نظم الطوابير المستخدمة على أسس إحصائية ضمن المعطيات المتوفرة. أهم تلك المعايير الإحصائية هو زمن الانتظار الكلي لجميع تلك العمليات. لقد حصلنا على نتائج إحصائية مهمة لفهم طبيعة عملية المعالجة و اختيار نظام الطوابير الأفضل. أن اختيار أفضل البدائل للمسألة ذات البحث لا يكون في الغالب يسيرا لأنه يعتمد على طبيعة المسألة ويعتمد على البيانات المتوفرة.