Optimizing Time Quantum Of Round Robin Scheduling Using Fuzzy Logic And Age Based Allocation

Dissertation submitted in
partial fulfilment of the requirement
for the award of the degree of

Master of Technology

in

Computer Science and Engineering

by

RAJENDRA SINGH NIKHURPA

University Roll No. 2K12/CSE/14

Under the Esteemed Guidance of

Mr. VINOD KUMAR
Associate Professor, Computer Engineering Department, DTU

2012-2014

COMPUTER ENGINEERING DEPARTMENT

DELHI TECHNOLOGICAL UNIVERSITY

DELHI – 110042, INDIA

Computer Engineering Department
Delhi Technological University
Delhi-110042
www.dce.edu

CERTIFICATE

This is to certify that the dissertation titled "**Optimizing Time Quantum Of Round Robin Scheduling Using Fuzzy Logic And Age Based Allocation**" is a bonafide record of work done by **Rajendra Singh Nikhurpa, Roll No. 2K12/CSE/14** at **Delhi Technological University** for partial fulfilment of the requirements for the degree of Master of Technology in Computer Science & Engineering. This project was carried out under my supervision and has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma to the best of my knowledge and belief.

Mr. Vinod Kumar
Associate Professor
Department of Computer Engineering
Delhi Technological University

Date: _____

ACKNOWLEDGEMENT

First of all, I would like to express my deep sense of respect and gratitude to my project supervisor Mr. Vinod Kumar for providing the opportunity of carrying out this project and being the guiding force behind this work. I am deeply indebted to him for the support, advice and encouragement he provided without which the project could not have been a success.

Secondly, I am grateful to Dr. O.P.Verma, HOD, Computer Engineering Department, DTU for his immense support. I would also like to acknowledge Delhi Technological University for providing the right academic resources and environment for this work to be carried out.

Last but not the least I would like to express sincere gratitude to my parents for constantly encouraging me during the completion of work.

Rajendra Singh Nikhurpa
University Roll no: 2K12/CSE/14
M.Tech (Computer Science & Engineering)
Department of Computer Engineering
Delhi Technological University
Delhi – 110042

Date……

ABSTRACT

Now in the modern days computing requires effective use of resources and high work rate. The operating system is software that manages the resources for the user and provides high resource utilization. It helps to use the available resources in a effective way.

Processor or CPU is one of the most important resources in the machine. It executes the processes and communicates for data. The modern day systems use multi-programmed environment so as to keep more programs in the memory so as to effectively use the processor. The major task now is to schedule the processes so as to give them processor efficiently.

The scheduling of processor is done by program called scheduling algorithm. The function of scheduling algorithm is to efficiently share the processor among the processes. The scheduling algorithms used commonly are First come First Serve, Shortest job first, Round robin scheduling, Priority scheduling. Among these the designer has to generally use one according to the need of the system. Round robin is widely used because the task of finding the next process to schedule is the one that is next in the queue. However the efficiency of round robin algorithm is highly dependent on the time quantum chosen. Therefore, value of time quantum must be chosen appropriately. However, optimal value is different with different processes. Therefore, we propose a system to find the efficient value of time quantum. The algorithm takes into consideration the quantity of processes. Also the average burst time of the processes is seen. Depending on these values a efficient value of time quantum is is decided. The decision is taken by the fuzzy decision system. To give more time to the older processes the aging is also introduced in the model. The process with the more age gets more proportion of the time quantum.

The model was implemented in simulation and the results were collected.

Key Words : Operating System, CPU Scheduling, Round Robin, Fuzzy Logic, Aging, Time quantum.

Table of Contents

Chapter 2

Chapter 3

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

An operating System acts as an interface between users and hardware. It enables users so that they can execute programs more conveniently and efficiently. Technically operating system is a software managing available hardware. It is a program which is running all the times and rest of the programs being executed as application programs. It acts as an manager which allocates resources and services such as memory, processors, devices and information.

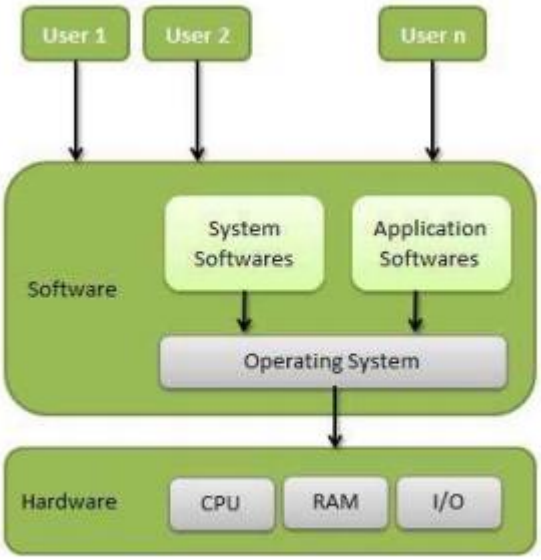


Figure 1: Operating System diagram

To allocate resources and information, it has other programs which help OS in managing the resources and information e.g. traffic controller, scheduler, memory manager, I/O programs, and a file system.

1.2.Functions of Operating System

Operating system has following sets of key roles:

- Memory Management
- Processor Management
- Device Management
- File Management
- Security

1.2.1 Memory Management

Memory management manages the status of each and every allocated or free memory location. It decides the memory allocation like how much memory a process should get, when a process should receive it and which all processes will get the memory, among processes which are competing against each other for resources. After memory allocation it decides the memory locations for the processes. It also tracks when a memory location is getting freed up and then updates the status.

1.2.2 Processor Management

Processor management in operating system decides which process gets the processor and the time for which it is allowed to use the processor which in common terms is known as scheduling and the program to ensure this is known as “Scheduler”. Scheduler ensures that each process and application gets enough amount of processor’s time for proper functioning. It also ensures that maximum processor utilization.

1.2.3 Device management

Device manager allows users to view and control the hardware attached to the computer. Whenever a piece of hardware is not working properly, it is notified to the user so that the same can be replaced by the user.

1.2.4 File management

File management unit controls how the data can be stored and its retrieval. Without it the information placed in the storage is of no use as it also helps in determining where the information is starting from and where it is ending. It separates data into small pieces and gives each piece a name for easy identification. Each piece of information is known as “file”. The rules and structure which are used to manage this information group is called “file system”.

1.2.5 Security

Operating system provides access of the resources available to the processes which are running. It decides which requests should be allowed to be processed and which are not allowed to be processed. It achieves the same by identifying the requestor identity by user name. It uses authentication process to establish the identity. Usually user name is associated with a password for authentication but other methods like biometric can also be used for authentication.

1.3 Types of Operating System

1.3.1 Batch operating system

In batch operating system users do not interact directly with the computer. Each user prepares his job and submits it to the operator using an off-line device like punch cards. Operator creates the batch of the programs with similar needs and run as a group to speed up processing. Batch operating system lacks interaction between the user and the job. The CPU utilization is very less as the I/O devices which are mechanical in nature are slower than CPU. It is difficult to provide desired priority for jobs.

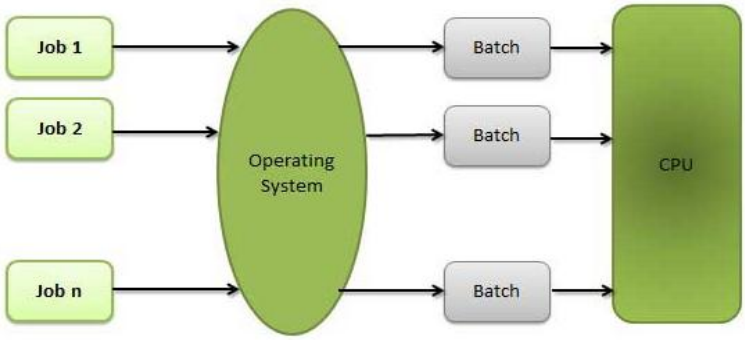


Figure 2: Batch operating system

1.3.2 Time-sharing operating systems

Time sharing operating system enables multiple users who are located at different terminals to use a particular system at the same time. The processor is shared between multiple users simultaneously. The objective of time sharing operating system is to minimize the response time. Multiple jobs are executed by switching them very frequently between the processor which in turn results in immediate response. These type of operating system uses scheduler and multiprogramming to give each user a quantum of processor's time. The advantages of Timesharing operating systems are that they provide faster response and reduce processor idle time. The disadvantages include reliability, security and integrity of user programs and data.

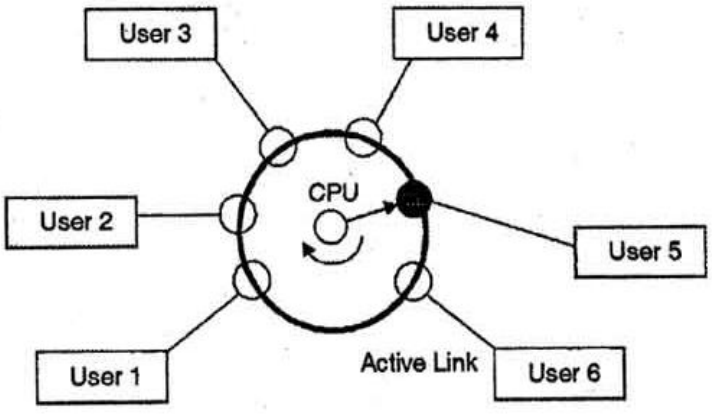


Figure 3:Time sharing operating system

1.3.3 Distributed operating System

This type of operating system uses multiple processors to server multiple processes and users. The jobs are distributed among the processors to the ones which can perform the job more effectively. The processors communicate with one another through various mediums like telephone lines. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers and so on. As these systems share resources which each other, it enables the user who is present at one site to use the resources available at another site. They can speed up the exchange of data with one another using electronic mail. It provides better service to customers as, if one site fails then the other sites can continue to operate. Distributed operating systems also help in reducing the load on the host computer and in turn reduce delay in data processing.

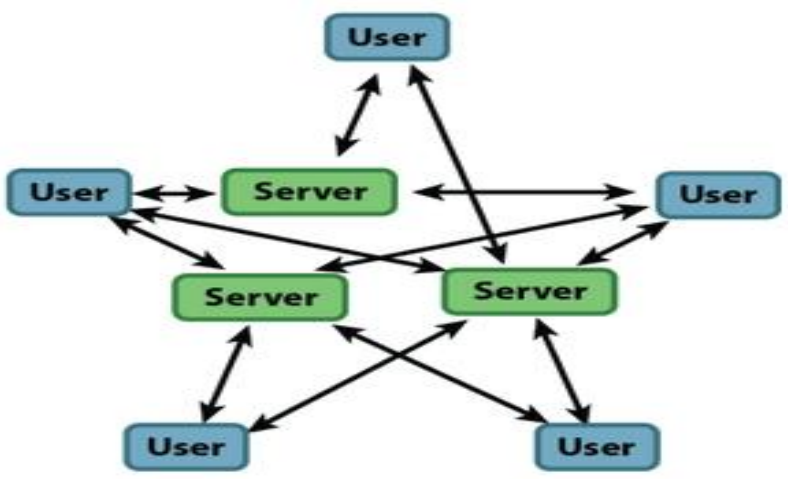


Figure 4: Distributed operating system

1.3.4 Network operating System

These type of operating system runs on a server and equip server with the capability to manage data, users, application and other networking functions. They allow shared file and printer access to multiple computers in a network i.e. LAN or private network. Examples of network operating systems are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD. These operating system runs on centralized servers which are highly stable and securely managed. They are easily upgradable to new technologies and hardware. They provide remote access to servers from different locations. However the cost of buying and running a server is very high and they also comes up with dependency on a central location for most of the operations. Maintenance and updates are required on a very regular interval to run these operating systems.

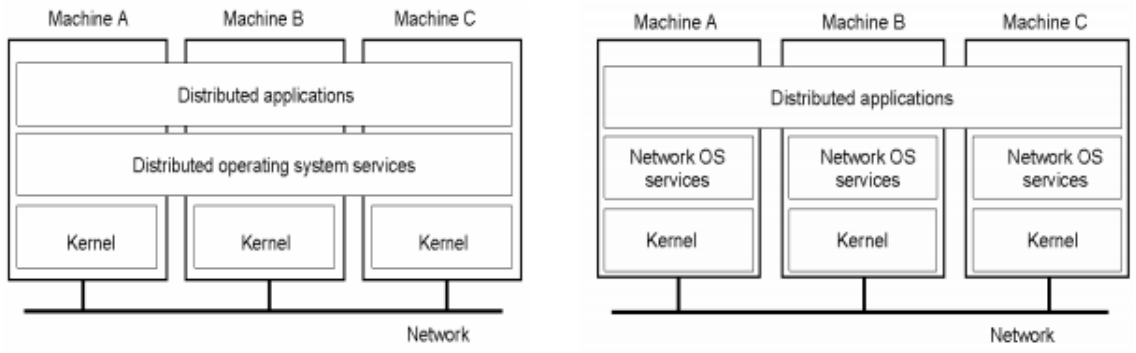


Figure 5: Network Operating System

1.3.5 Real Time operating System

This type of operating system controls the environment as the time interval required to process and respond to process the inputs is very small. Real time processing is always on line whereas on line system need not be real time. Response time is the time taken by the system to respond to an input and display of required updated information. So in this operating system response time is very less as compared to the online processing. Real-time systems are used when the requirement on the operation of processor or the flow of information are very rigid e.g. Scientific experiments, medical imaging systems, home-appliance controllers, Air traffic control system etc.

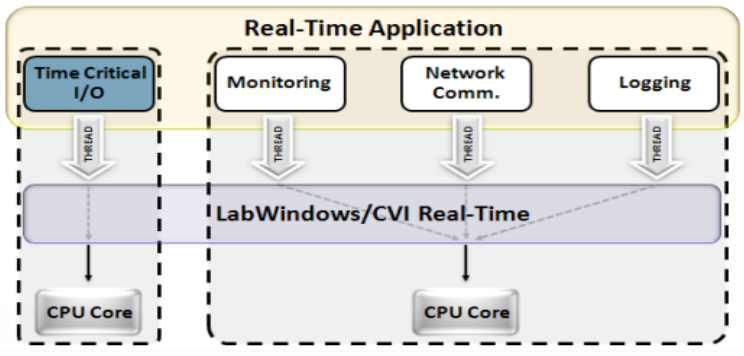


Figure 6: Real-Time operating system

Real time operating systems are categorized in two types:

i). Hard-real Time Systems

These systems guarantee that critical tasks complete on time. Secondary storage is limited or missing in these systems and are usually stored in ROM. Virtual memory is never found in these systems.

ii). Soft-real Time Systems

Critical real-time task in these systems gets priority over other tasks and retains the priority until it completes. They have limited utility than hard real-time systems e.g. Multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers etc.

In Multiprocessing system, the multiprogramming is one of the most critical and important aspects of operating systems. There are several processes which are to be kept simultaneously in memory. The aim of which is to maximize the CPU utilization. If these several processes which are in the memory and ready to run at the same time, then operating system must choose which one among them to run first and it is responsibility of operating system to make this happen very efficiently. To make this call we have several CPU scheduling algorithms like first come first serve, shortest job first ,round robin etc. CPU scheduling is the basis of multiprogramming systems. Hence operating system uses which algorithm is going to be very suitable for current situation. It is made by part of operating system called the scheduler, using a CPU scheduling algorithm.

Round Robin algorithm will allow the first process in the ready queue to run until its time quantum expires, and then run the next process in the ready queue. In a situation where the process needs more time, the process runs for the full length of the time quantum and then it is preempted and then added to the tail of the queue. In Round Robin we faces large number of context switches and there will be less throughput. Hence we worked on this problem by optimizing time quantum of round robin and there is age based allocation of CPU among processes.

.

CHAPTER 2

BACKGROUND

2.1 Process

A process is sequential program in execution. A process defines the fundamental unit of computation for the computer. Components of process are :

- Object Program
- Data
- Resources
- Status of the process execution.
- Object program i.e. code to be executed. Data is used for executing the program. While executing program, it may requires some resources. Last component is used for verifying the status of process execution. A process can run to complete execution only when all of requested resources have been allocated to the process. Two or more processes may be executing same program, each using their data and resources.

2.2 Processes and Program

Process is a dynamic entity, that is a program in execution. A process is a sequence of information executions. Process exists in a limited span of time. Two or more processes could be executing the same program, each using their own data and resources. Program is a static entity made up of program statement. Program contains the instructions. A program exists at single place in space and continues to exist. A program does not perform the action by itself.

When a process executes, it changes its state. Process state is defined as the current activity of the process. Fig. 2 shows the general form of the process state transition diagram. Process state contains five states. Each process is in one of the states.

The states are listed below.

1. *New* : A process that just been created.

2. *Ready* : Ready processes are the processes which waiting to get CPU.

3. *Running* : The process that is currently being executed. A running process possesses all the resources needed for its execution, including the processor.

4. *Waiting* : A process that can not execute until some event occurs such as

completion of I/O operation. The running process goes to suspended by invoking an I/O module.

5. *Terminated* : A process that has been released from the pool of executable

processes by the operating system.

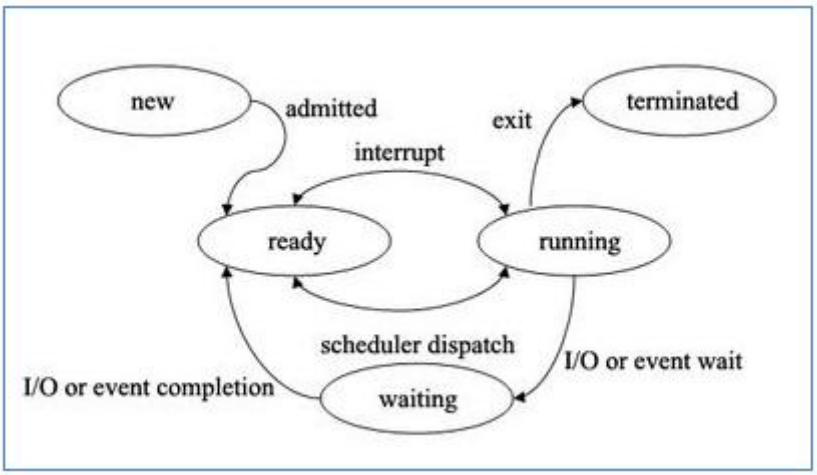


Figure 7: Process State Diagram

2.3 Process Management or Process Scheduling:

In a multi-programmed system multiple processes competing for the CPU at the same time. 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 operating system that makes the choice is called short term scheduler or CPU scheduler and scheduling algorithm is used to schedule these processes in a efficient way so that CPU utilization is maximum and to schedule we have several scheduling algorithms. Different scheduling algorithms have different properties and the choice of a particular algorithm may favour one class of processes over another and we can say that these are condition dependent i.e. we can't say in particular that such algorithm is best algorithm in any condition. Many criteria have been suggested for comparing CPU scheduling algorithms and deciding which one is the best algorithm[1].

A process in an operating system is represented by a data structure known as a process control block (PCB) or process descriptor. The PCB contains important information about the *specific process including:*

Process state: The state may be new, ready, running, waiting, halted, and so on. Program counter. The counter indicates the address of the next instruction to be executed for this process.

CPU registers: The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information.

CPU scheduling information: This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.

Memory management information: This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the OS.

Accounting information: This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.

I/O status information: This information includes the list of I/O devices allocated to the process, a list of open files, and so on.

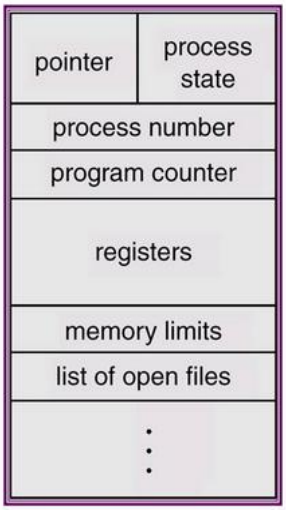


Figure 8: process control block

2.4 Types of Processor Scheduling

The aim of processor scheduling is to assign processes to be executed by the processor in a way that meets system objectives, such as response time, throughput, and processor efficiency. In many systems, this scheduling activity is broken into three separate functions

(i) Long term scheduling

(ii) Medium term scheduling

(iii) Short term scheduling

2.4.1 Long Term Scheduling

The long term scheduler determines which programs are admitted to the systems for processing, thus, it controls the degree of multiprogramming [1]. Once admitted, a job or user program becomes a process and is added to the queue for the short term scheduler. In a batch system, long term scheduler creates processes and forms the queue wherever it is possible. The more processes are created; the smaller is the percentage of time for which each process can be executed. Long term schedulers may limit the degree of multiprogramming to provide satisfactory service to the current set of processes.

2.4.2 Midterm Scheduling

The scheduling of processes is mainly based on the requirement of the resources. It is essentially concern with memory management and often design as a memory management subsystem of an operating system [2,3]. It temporarily removes a process from the main memory which is of low priority or has been inactive for a long time. This is known as "swamping out" of a process. The scheduler may decide to swamp out the process which is frequently page-faulting or a process which is taking large amount of memory. Its efficient interaction with the short term scheduler is very essential for the performance of the systems with virtual memory.

2.4.3 Short Term Scheduling

In terms of frequency of execution, the long term scheduler executes relatively infrequently and makes the coarse grained decision of whether or not to take on a new process and which one to take. Short term scheduler is invoked whenever an event occurs that may lead to the blocking of the current process that may provide an opportunity to preempt a currently running process in favor of another. CPU scheduling decisions can occur on the given conditions:

(a) either the running process changes from running to waiting state or when the running process terminates.

(b) The waiting process becomes ready.

(c) The current process switches from running to ready state.

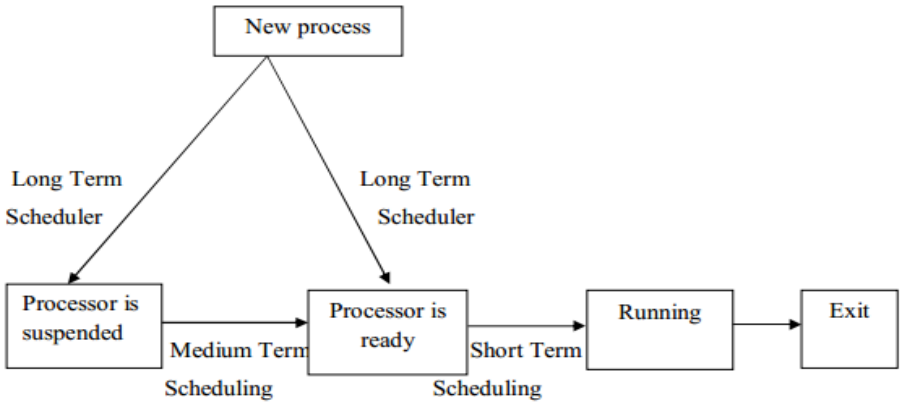


Figure 9 : Types of scheduler

2.5 Scheduling Algorithm goals for Different Systems:

There are some goals that must be achieved in order to perfectly schedule the task on the processor. Some of these goals are mentioned below:

(i) Fairness : Fairness is important under all circumstances. A scheduler makes sure that each process gets its fair share of the CPU and no process suffer indefinite postponement. Note that giving equivalent or equal time is not fair [3].

(ii) Policy Enforcement : The scheduler has to make sure that system's policy is enforced. For example, if the local policy is safety then the safety control processes must be able to run whenever they want to, even if it means delay in payroll processes.

(iii) Efficiency: Scheduler should keep the system busy cent percent of the time when possible. If the CPU and the entire input/output device run for entire time, more work gets done per second [3].

(iv) Meeting deadlines: Scheduler should finish all its processes before the deadline of process otherwise catastrophic results can occur [4].

The CPU scheduling can be defined as the art of determining which processes run on the CPU when there are multiple running processes. Also, it is the problem of deciding which computer process in the ready queue (in other words, which particular programs need some

processing and are ready and waiting for it) is to be allocated to the CPU for processing.CPU scheduling is one of several system components that make a whole multimedia system. Scheduling problems of resources in general has significance in understanding CPU scheduling. Scheduling implies multiplexing a resource among several tasks to ensure all throughput requirements are met. For batch or interactive processes, secondary storage or CPU cycles can be scheduling using a round robin or first-come first-serve policy.

- **CPU Utilization**: Keep CPU utilization as high as possible.
- **Throughput**: number of processes completed per unit time.
- **Turnaround Time**: mean time from submission to completion of process.
- **Waiting Time**: Amount of time spent ready to run but not running.
- **Response Time**: Time between submission of requests and first response to the request.

2.6 CPU Scheduling Criteria

CPU scheduling criteria are the basis on which the performance of CPU scheduling algorithms is evaluated. There are many possible criteria:

- **CPU Utilization** : This is a measure of how busy the CPU is. Usually, the goal is to maximize the CPU utilization.
- **Throughput :** This is the number of processes completed per unit time. Usually, the goal is to maximize the throughput.
- **Turnaround Time :** This is the amount of time from submission to completion of process. Usually, the goal is to minimize the turnaround time.
- **Waiting Time :** This is the amount of time spent ready to run but not running. It is the difference in start time and ready time. Usually, the goal is to minimize the waiting time.
- **Response Time :** This is the amount of time between submission of requests and first response to the request. Usually, the goal is to minimize the response time.

2.7 CPU Scheduling Algorithms

Four algorithms commonly used in CPU scheduling are discussed below :

- **First-Come First-Served (FCFS) :** In this algorithm CPU is allocated to the process which requests the CPU first. This algorithm is easily implemented with a FIFO queue. New process which enters into the queue joins the tail of the queue and leaves from the head of the queue (when the process is allocated to the CPU). The processes are allocated to the CPU as they arrive in queue. Once the CPU is allocated to the process, it is then removed from the queue.
- **Shortest Job First (SJF) :** The SJF algorithm completely depends on the length of the next CPU burst with each process in a manner that the processes that have the smallest CPU burst and they have been allocated the CPU first. If there is a conflict i.e. if two or more processes have same burst time then they will be processed as they arrived i.e. according to their arrival time. The SJF algorithm can be further subdivided into two categories i.e. as either preemptive or non-preemptive algorithms. When currently running process is interrupted in order to give the CPU to a new process with a shorter next CPU burst, it is known as preemptive SJF. On the other hand, the non-preemptive SJF will allow that the currently running process to finish its CPU burst before a new process is allocated to the CPU.
- **Priority Scheduling (PS) :** The Priority Scheduling algorithm associates with each process a priority and the CPU is allocated to the process based on their priorities. Usually, lower numbers are used to represent higher priorities. The process with the highest priority is allocated first. If there are multiple processes with same priority, typically the FCFS is used to break tie.

- **Round Robin (RR) :** The RR algorithm is designed especially for time-sharing systems and is similar to the FCFS algorithm. Here, a small unit of time (called time quantum or time slice) is defined. A time quantum is generally from 10-100 milliseconds. So, the RR algorithm will allow the first process in the queue to run until it expires its quantum (i.e. runs for as long as the time quantum), then run the next process in the queue for the duration of the same time quantum. The RR keeps the ready processes as a FIFO queue. So, new processes are added to the tail of the queue. Depending on the time quantum and the CPU burst requirement of each process, a process may need less than or more than a time quantum to execute on the CPU. In a situation where the process need more than a

time quantum, the process runs for the full length of the time quantum and then it is pre-empted. The pre-empted process is then added to the tail of the queue again but with its CPU burst now a time quantum less than its previous CPU burst. This continues until the execution of the process is completed). The RR algorithm is naturally pre-emptive.

2.8. Fuzzy Logic

Fuzzy Logic was introduced in 1965 [5], [6], [7], by Lotfi A. Zadeh , professor for computer science at the University of California in Berkeley. Basically, Fuzzy Logic (FL) is a multi-valued logic, that allows linguistic values to be clear between conventional evaluations like true or false, yes or no, high or low, etc i.e. only partial truth. Phrase like rather tall or very fast may be formulated mathematically and processed by computers, in order to relate a more human like manner of thinking in the programming of computers [8]. Fuzzy systems is an traditional notions alternative for set membership and logic that has origins in ancient Greek philosophy. Precision of mathematics owe its success in greater part to efforts of Aristotle and philosophers who preceded him. In their efforts to plan a concise theory of logic, and later mathematics the so called ”Laws of Thought” were proposed [9]. One of these is ”Law of the Excluded Middle,” which states that every proposition must either be True or False. and then when Parminedes proposed his first version of this law (around 400 B.C.). There were strong and gradual objections: for example, Heraclitus projected that things could be simultaneously True and not the whole Truth. It was Plato who laid the foundation for what would become the fuzzy logic, indicating that there is a third region (beyond True and False) where the opposites ”tumbled about.” Other, more modern philosophers echo his sentiments, particularly Hegel, Marx, and Engels. But it was Lukasiewicz who was first to proposed a systematic alternative to the bi valued logic of Aristotle [10]. Even in present time some Greeks are still terrific examples for fussiness. Fuzzy Logic has emerge as a gainful tool for controlling and steering of system and complex industrial processes, as well as for household and entertainment electronics, as well as for extra expert systems and applications like the classification of data.

2.8.1 Fuzzy Sets and Crisp Sets

The very basic notation of fuzzy systems is a fuzzy sub set. In classical mathematics we are familiar with what we call this as crisp sets. For example, the possible interfere metric coherence g values are set of X of all real numbers between 0 and 1. From this set X a subset A which can be defined, (e.g. all values $0 \leq g \leq 0.2$). The characteristic function of A, (i.e. this function assigns a number 1 or 0 to every element in X, depending upon whether the element is in the subset A or not) is shown in Fig.12. The elements which have been given the number 1 can be interpreted as elements which are in the set A and elements which have assigned the number 0 as the elements that are not in the set A.

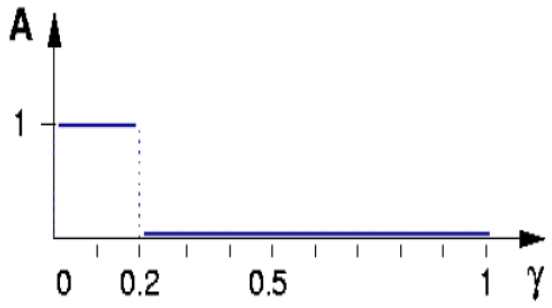


Fig. 10:Characteristic Function of a Crisp set

This concept is sufficient for many areas of applications, but it can be seen, which is lacks in flexibility for some applications similar to classification of remotely sensed data analysis. For example it is well known that water shows low inter ferometric coherence g in SAR images. Since g starts at 0, the lower range of this set ought to be clear. The upper range, on the other hand, is rather hard to define. As a first attempt, we set the upper range to 0.2. Therefore we get B as a crisp interval B=[0,0.2]. But by means of that a g value of 0.20 is low but a g value of 0.21 not. Obviously, this was a structural problem, for if we moved towards the upper boundary of the range from g =0.20 to an arbitrary point we can pose the same question. A more natural way to construct the set B would be to relax the strict separation between low and not low. This can be done by allowing not only the (crisp) decision Yes/No, but more flexible rules like ” fairly low”. A fuzzy set allows us to define such a notion. The aim is to

use fuzzy sets in order to make computers more ’intelligent’, therefore, the idea above has to be coded more formally. In the example, all the elements were coded with 0 or 1. A straight way to generalize this concept, is to allow more values between 0 and 1. In fact, infinitely many alternatives can be allowed between the boundaries 0 and 1, namely the unit interval I = [0, 1]. The interpretation of the numbers, now assigned to all elements is much more difficult. Of course, again the number 1 assigned to an element means, that the element is in the set B and 0 means that the element is definitely not in the set B. All other values mean a gradual membership to the set B. This is shown in Fig. 2. The membership function is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs that are processed, define functional overlap between inputs, and ultimately determines an output response. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion.

The membership function, operating in this case on the fuzzy set of interferometric coherence g, returns a value between 0.0 and 1.0. For example, an interferometric coherence g of 0.3 has a membership of 0.5 to the set low coherence (see Fig. 2). It is important to point out the distinction between fuzzy logic and probability. Both operate over the same numeric range, and have similar values: 0.0 representing False (or non-membership), and 1.0 representing True (or full-membership). However, there is a distinction to be made between the two statements: The probabilistic approach yields the natural-language statement, ”There is an 50% chance that g is low,” while the fuzzy terminology corresponds to ”g’s degree of membership within the set of low interferometric coherence is 0.50.” The semantic difference is significant: the first view supposes that g is or is not low; it is just that we only have an 50% chance of knowing which set it is in. By contrast, fuzzy terminology supposes that g is ”more or less” low, or in some other term corresponding to the value of 0.50.

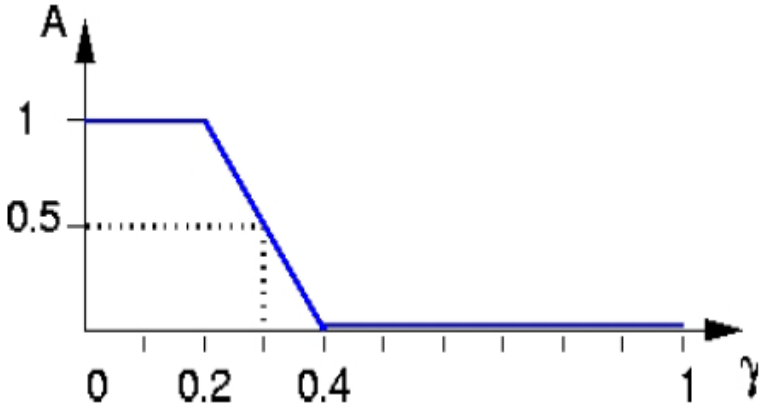


Figure 11: Characteristic Function of Fuzzy Set

2.8.2 Membership Functions

For representation of the membership functions, we can use the following functions:

• **Triangular Membership Functions**

A triangular MF, as shown in Figure 2.7 (a), is a function with 3 parameters defined by

$$\text{triangle}(x\ ;\ a\ ,b\ ,c) = \max(\min(\frac{x-a}{b-a}, \frac{c-x}{c-b}\ ),0)$$

• **Trapezoidal Membership Functions**

A Trapezoidal MF, as shown in Figure 2.7 (b), is a function with 4 parameters defined by

$$\text{trapezoid}(x\ ;\ a\ ,b\ ,c\ ,d) = \max(\min(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\ ),0)$$

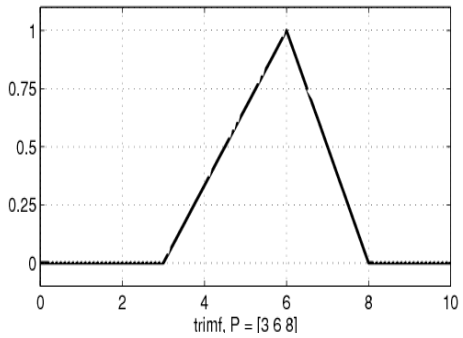


Figure 12: Triangular membership function

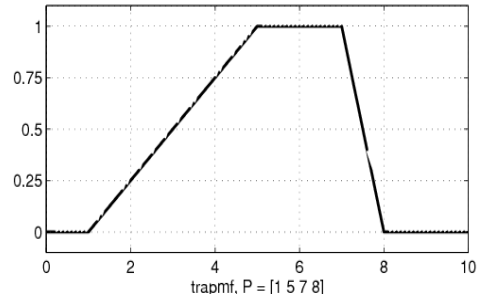


Figure13: Trapezoidal membership function

• Gaussian Membership Functions

A Gaussian MF is a function with two parameters defined by

$$\text{gaussian}(x\ ;\ \sigma\ ,c) = e^{\frac{-(x-c)^2}{\sigma^3}}$$

where c is the center and σ is the width of membership function

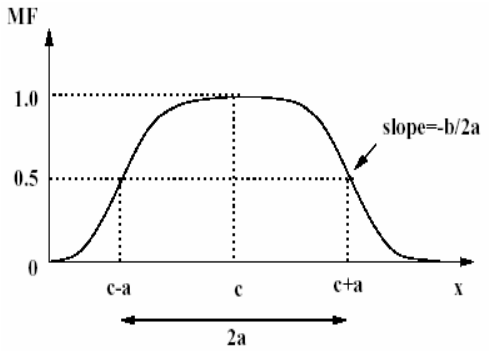


Figure14: Bell membership function

• Bell Membership Functions

A bell MF, as shown in Figure 2.8, is a function with two parameters defined by

$$\text{bell}(x\ ;\ a\ ,b\ ,c) = \frac{1}{1+|\frac{(x-c)^{2b}}{a}|}$$

• Sigmoidal Membership Function

A Sigmoid MF is a function with two parameters defined by

$$\text{sigmoid}(x;\ k\ ,c) = \frac{1}{1+ e^{-k(x-c)}}$$

where parameter k influences sharpness of function in the point where a = c. If k >0 the function is open on right site, on the other hand, if k<0 the function is open on left site and therefore this function can be use for describing conceptions like “very big” or “very small”. Sigmoid function is very often used in Neural Networks like activation function.

2.8.3 Fuzzy Rules and Fuzzy Reasoning

Fuzzy rules and fuzzy reasoning are the backbone of fuzzy inference systems, which are the most important modeling tool based on fuzzy set theory. They have been applied to a wide range of real-world problems, such as expert systems, pattern recognition, and data classification. A detailed discussion about fuzzy inference systems is provided in [11].

2.8.3.1 Fuzzy If-Then Rules

Fuzzy if-then rules (also known as fuzzy conditional statements) are expressions of the form

If x is A , then y is B

where A and B are linguistic labels defined by fuzzy sets on universe of discourse X and Y, respectively. Often "x is A" is called the antecedent or premise, while "y is B" is called the consequence or conclusion. Due to their concise form, fuzzy if-then rules are often used to capture the imprecise modes of reasoning and play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision. Fuzzy if-then rules have been used extensively in both modelling and control. From another angle, due to the qualifiers on the premise parts, each fuzzy if-then rule can be viewed as a local description of the system under consideration.

2.8.3.2 Fuzzy Reasoning

Fuzzy reasoning, also known as approximate reasoning, is an inference procedure that derives conclusions from a set of fuzzy if-then rules and known facts.

CHAPTER 3

RELATED WORK

Round-robin (RR) is one of the scheduling algorithms to schedule processes in a system. In RR time slices i.e. time quantum is assigned to each process in equally and in circular order, handling all processes without precedence. Round-robin scheduling is simple and easy to implement and also processes doesn't suffer from starvation.

The RR Scheduling has certain disadvantages which are longer average waiting time, higher context switches, higher turnaround time and low throughput. In Round Robin Scheduling the time quantum play a very important role for scheduling, because if time quantum is very large then Round Robin Scheduling Algorithm is same as the FCFS Scheduling. If the time quantum is extremely too small then Round Robin Scheduling is called as Processor Sharing Algorithm and number of context switches is very high. So there are many researches have been done related to RR improvement and they have shown that RR works very efficiently in certain circumstances.

In these past years many researchers have done lots of works and have presented their ideas to reduce the context switching and to improve the time quantum of Round Robin scheduling algorithm.

3.1 Computation of time quantum using FIS

They have proposed an algorithm to improve the time quantum of round robin scheduling algorithm using fuzzy inference system. They have chosen number of processes and average burst time as input and on that basis they have generated the time quantum which is good enough to reduce context switching between the processes and hence efficient throughput.

An Fuzzy inference engine consists of three stages commonly known as input , processing, and output stages. The first stage i.e. The input stage maps into the inputs such as deadline, execution time, and average waiting time etc to the corresponding membership function and their truth values. The second stage i.e. processing stage which invokes each relative rule and thus generates a result for each. It then combines the results of the rules. Last stage i.e. output

stage converts the all results into a specific output value [12]. As processing stage, which is known as the inference engine is based on a collection of logic rules in the form of if-then statements where if part is called antecedent and the then part is called the consequent. Typical fuzzy inference systems includes dozens of rules. These rules are stored in a knowledgebase. An example of fuzzy if-then rules: IF number of users is high then time quantum is low, in which number of user and time quantum are linguistics variables and high and low are linguistics terms. The five steps toward a fuzzy inference are as follows:

- Raking fuzzying inputs
- Apply fuzzy operators
- Apply implication methods
- Aggregate outputs
- Defuzzyfying results

3.1.1. FIS Algorithm

On the basis of fuzzy logic they have proposed the following algorithm [13]

1. Find ABT, the average burst time of the processes.
2. Give N, the number of users and ABT to the FIS designed above.
3. Take output of FIS as the time quantum.
4. Invoke Round Robin Scheduling Algorithm.

3.2 Minimization Of Context Switching Using Simplex Algorithm

Mahesh Kumar M R , Renuka Rajendra B , Sreenatha M , Niranjan C K[14] proposed the simplex algorithm to reduce context switching between processes and their algorith as follows:

1. They have converted given problem into Linear Programming Models to form objective function which is containing process burst time.
2. Constraints consist of waiting time and turnaround time of each process and average waiting time and average turnaround time as right side of the constraints.
3.Introduce slack variables to convert Linear Programming Models into standard form

$$Z=10X_1+1X_2+2X_3+1X_4+5X+0S_1+0S_2$$

Subject to constraints

$$9X_1+1X_2+5X_3+3X_4+9X_5+1S_1=27$$

$$9X_1+2X_2+7X_3+4X_4+14X_5+1S_2=46$$

4. Obtain starting basic feasible solution to create simplex table.
5. Compute net value of $Z_j - C_j$ to identify entering and leaving variable in simplex table.
6. Repeat above steps until we get all values of $Z_j - C_j$ are positive.
7. Once we get all values are positive now check the values of Z_j this will gives us a new quantum time is 2 for our problem.

$$Z=10X_1+1X_2+2X_3+1X_4+5X+0S_1+0S_2$$

Subject to constraints

$$9X_1+1X_2+5X_3+3X_4+9X_5+1S_1=27$$

$$19X_1+2X_2+7X_3+4X_4+14X_5+1S_2=46$$

8. Calculate waiting time and turnaround time for each process using new quantum size.

3.3 Selection Of Smart Time Slice Using Shortest Burst Time Approach

Saroj Hiranwal, Dr. K.C.Roy[15], they have introduced the Intelligent Time Slice for Adaptive Round Robin. And proposed algorithm which eliminates the defects of implementing simple Round Robin scheduling algorithm in operating system by developing a concept called smart time slicing which depends on priority, average CPU burst or mid process CPU burst, and context switch avoidance time. The proposed algorithm allows the user to issue priority to the system based on execution time or burst time. They have evaluated smart time slice that will be based on all CPU burst of currently new running processes. The smart time slice evaluated according to the processes burst time; if the number of processes are placed into the ready queue are odd in number then the smart time slice will be the medium process burst time else the number of processes are even in number then in ready queue the smart time slice is average of all processes burst time is given to the processes. The new proposed algorithm called Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice[13]. It is a Priority Driven Scheduling algorithm based upon burst time of processes. First of all we may arrange the processes according to the execution time or burst time in increasing order that is smallest the burst time have higher priority of the running process. The next idea of this approach is to choose the smart time slice (STS) which is mainly depends upon number of processes. The smart time slice is equal to the medium process burst time of all CPU burst time which when the number of process given odd. If number of process given even then we choose the time quantum which is according to the average CPU burst of all running processes. Based on the experiments and calculations the proposed algorithm radically evaluated the fixed time quantum problem which is considered as a challenge for Round Robin Scheduling Algorithm. The use of scheduling algorithm is to increased the performance as well as stability of the operating system and supporting the building of an self-adaptation operating system, which means that the system that will adapt itself as per the requirements of the user and not vice versa.

The Adaptive Round Robin Scheduling Algorithm[13] focuses on the drawbacks of simple Round Robin Algorithm which will give equal portion of time to all the processes (processes are scheduled in first come first serve manner) because of all the drawbacks in Round Robin Algorithm which is not efficient for processes having smaller CPU burst. This will lead to the increase in waiting time and response time of processes which decreases the system

throughput. The proposed algorithm eliminates the drawbacks of implementing a simple round robin algorithm in by scheduling of processes based on the CPU execution time. The allocated processor used to reduce the burden of the main processor which is given to processes according to the priority, the smaller CPU burst of the process, higher the priority. The proposed algorithm solves the problem of higher average waiting time, turnaround time, response time and more context switches thereby improving the system performance.

Smart Time Slice = Mid Process Burst Time (If number of processes are odd)

or

Smart Time Slice = Average Burst Time (If number of processes are even)

Then processes are executing according to the smart time slice and give superior result comparison to existing simple Round Robin Scheduling Algorithm and can be implemented in operating system.

3.3.1 Adaptive Round Robin Pseudo Code

1. First of all check ready queue is empty
2. When ready queue is empty then all the processes are assigned into the ready queue.
3. All the processes are rearranged in increasing order that means smaller burst time process get higher priority and larger burst time process get lower priority.
4. While (ready queue != NULL)
5. Calculate smart Time Slice:

If (Number of process%2= = 0)

STS = average CPU burst time of all processes

Else

STS = mid process burst time

6. Assign smart time slice to the i^{th} process:

P_i <- STS

7. If (i < Number of process) then go to step 6.
8. If a new process is arrived update the ready queue, go to step 2.
9. End of While

10. Calculate average waiting time, turnaround time, context switches and throughput.

11. End

3.4 SRBRR Model

P.Surendra Varma ,Vijayawada, have proposed an algorithm which applicable in uni-processor and all processes are independent of each other and they have also worked on Shortest Remaining Burst Round Robin in order to give better turnaround time, average waiting time and minimizes context switch and there proposed algorithm works as follow:

1. All processes present in ready queue are sorted in increasing order.
2. While (ready queue! = NULL)

    TQ = Ceil (sqrt (median * highest Burst time)).
3. Assign TQ to process P_i ->TQ.
4. If (i < n) then go to step 3.
5. If a new process is arrived, Update the counter n and go to step1 End of while.
6. Average waiting time, average turnaround time and Number of context switches are calculated.
7. End.

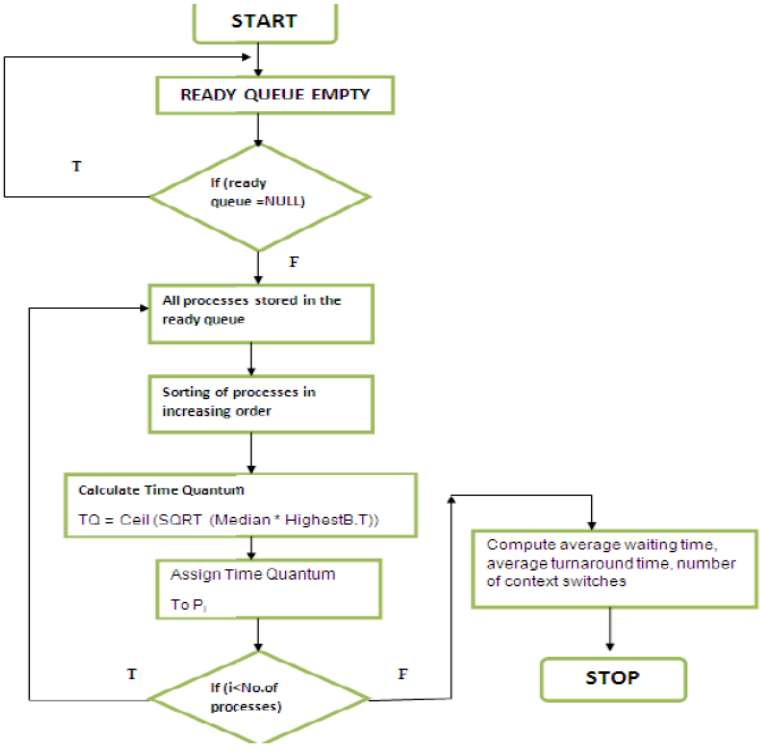


Figure 15: Flow Chart Of Algorithm

3.5 NIRR Scheduling

Abdulrazaq Abdulrahim, Saleh E Abdullahi , Junaidu B. Sahalu[16] they have proposed an CPU scheduling algorithm is a modification of the algorithm. It assumes another queue called the ARRIVE queue which holds processes according to their arrival times while there are other processes in the ready queue (say REQUEST) waiting for CPU allocation.

Algorithm takes to the REQUEST queue, in the first step process (i.e.pr[1]) that enters in the ARRIVE queue, and which allocates the CPU to it for the period of burst time (i.e.bt[1]). Processes that arrives while the CPU is executing this process will be added in the end of the ARRIVE queue as they their arrival time. After execution of the process, all the processes which are in the ARRIVE queue so they will be moved to the REQUEST queue and arranged in increasing order of burst times. The algorithm take ceiling of average burst time of all the processes in the REQUEST queue as the time quantum and allocates the CPU to first process in REQUEST queue for time quantum. When the time quantum for the process finished, the algorithm checks on the remaining CPU burst time of the currently running process. If the left over CPU burst time is less than or equal to half of the time quantum, then CPU will again be allocated to currently running process for the remaining CPU burst time. In this case, this process will finish its execution and will be removed from the REQUEST queue. Otherwise, if the left behind CPU burst time of the currently running process is longer than half of time quantum, the process will be moved to ARRIVE queue. CPU scheduler will then proceed to next process in REQUEST queue. During execution of processes in REQUEST queue, any process that arrives the system will be placed in the ARRIVE queue. These activities continue until no process is available in the REQUEST queue.

After execution of the processes in REQUEST queue, transferred processes from REQUEST queue to ARRIVE queue in previous execution cycle and recently arrived processes in ARRIVE queue will be queued to REQUEST queue in increasing order of burst times and a new time quantum will be calculated (i.e. the ceiling of average of burst times of the processes). CPU will be allocated to processes in REQUEST queue as usual using the recently determined time quantum. These activities continue until no process is available in the REQUEST and ARRIVE queues.

The algorithm as follows:

1.Start

2.Create ARRIVE queue, where processes will be placed when they arrive the system before they are moved to the ready queue.

3. Create a ready queue, REQUEST

4. Do

5: If (procees_index= 1)

{

time_quantum =burst_time[1]

Move the first process (pr [1]) to REQUEST queue

}

Else

{

Move all processes in ARRIVE queue to REQUEST queue in ascending burst time order

$$time\_quantum = \left\lceil \frac{\sum_{i=1}^{n} burst\_time[i]}{n} \right\rceil$$

}

6. Do

7. Allocate the CPU to the first process in REQUEST queue for a period of 1 time quantum.

8.If the remaining CPU burst time of the currently running process is less than or equal to half time quantum then allocate the CPU again to the currently running process for remaining

CPU burst time. After completion of execution, remove the process from the ready queue and go to step 7.

9. If the remaining CPU burst time of the currently running process is longer than half time quantum, remove the process from the REQUEST queue and put it in the ARRIVE queue and go to step 7.

10. If a new process arrives the system, it is placed in the ARRIVE queue.

11.WHILE queue REQUEST is not empty.

12.WHILE queue ARRIVE is not empty.

13. Calculate AWT, ATAT, ART and NCS.

14. END

CHAPTER 4

PROPOSED MODEL

4.1 Problem Statement And Proposed Model

As seen from the various scheduling algorithms the criteria for scheduling determine the applicability of the algorithm. However, it was seen that the Round Robin scheduling algorithm is good for variety of situations. Its have features like lesser starvation, upper bound on waiting time, high CPU utilization and constant time for scheduling to do task. However, the performance of this algorithm is highly dependent on time quantum we choose. By choosing large time quantum leads to less context switches but have large waiting time and if the time quantum is less then there will be more context switches and which will lead to decrease the performance i.e. throughput will be less.

Now what we have to do is to choose such a time quantum which is neither too large nor too small and as we know that the time quantum is completely depend on which type of processes are there in queue or which will be added next as they arrive.

Now we look at the various considerations that are to be responsible for choosing time quantum. Now what we do we look into ready queue for processes which are there. We can judge the later jobs or processes on the basis of first few jobs in ready queue. The reason for this is locality of reference, we know that the same kind of processes coming from same program with almost similar burst time. This will lead us to the time quantum which will be appropriate for most of the processes hence, we propose that by factor of first few process to find efficient time quantum. Therefore, we decided to choose time quantum based on first 10% of the processes in ready queue.

The factors to be considered as

1.Burst Time

2.Number of processes

1.Burst time

As we see that if the burst time is too small then there will be more context switches which affect the performance of system. So as a result we consider that the efficient time quantum should be proportional to average burst time of the processes.

2.Number of processes

Now we also consider that if the number of processes are large and the time quantum is also large then this will lead to increase the waiting time for the processes. i.e. efficient time quantum is inversely proportional to number of processes.

Now we see that there are contradicting situations i.e. if the average burst time and the number of processes both are large and if the average burst time and the number of processes both are small then we are not in a situation to choose efficient time quantum and hence we stuck. Therefore, we introduces fuzzy logic here to resolve the problem probabilistically. Fuzzy logic deals with partial truth i.e. degree of truth which is not the boolean(0/1) value like crisp logic. We define three soft boundaries namely low, medium and high for both the factors i.e. the average burst time and the number of processes.

S. No.	Burst time	No. of processes	Time quantum
1	Low	Low	Low
2	Medium	Low	Medium
3	High	Low	High
4	Low	Medium	Low
5	Medium	Medium	Medium
6	High	Medium	Medium
7	Low	High	Low
8	Medium	High	Low
9	High	High	Medium

Table1:Representing time quantum

The process which have been in the system for long time should be given more priority. Therefore, we have done here is that we assign CPU to processes for the portion of time quantum i.e. directly proportional to the time it has been in system we called this as "aging

factor" and we also divide this into three categories same as previous ones i.e. low, medium, high.

4.2 Proposed Algorithm

1. Get no. of processes and age boundary values.

2. Generate randomly arrival time and burst time and sort them according to arrival time and resolve clashes by burst time.

3. Get the value of time_quantum.

4. Set currnet_time = first_arrival.

5.while(flag==1)

{

i Add unexecuted processes with arrival_time upto current_time to the queue.

ii.while(queue!=empty)

. {

a. process_p=remove first element from queue.

b. find age and get the value of allocated time quantum.

c. if(burst_time[process_p]>allocated time quantum)

{

A. burst_time[process_p]= (burst_time[process_p]- allocated time quantum)

B.current_time=current_time+allocated time quantum

C.add process_p to queue.

}

d.else

{

A.current_time=(current_time+allocated time quantum-burst_time[process_p])

B. burst_time[process_p].

C. process_count++.

}

e. Add unexecuted processes with arrival_time upto current_time to the queue.

}

iii.if (process_count==processes)

{

a.(flag=-1)

}

iv.else

{

a. currrent_time=arrival time of next process

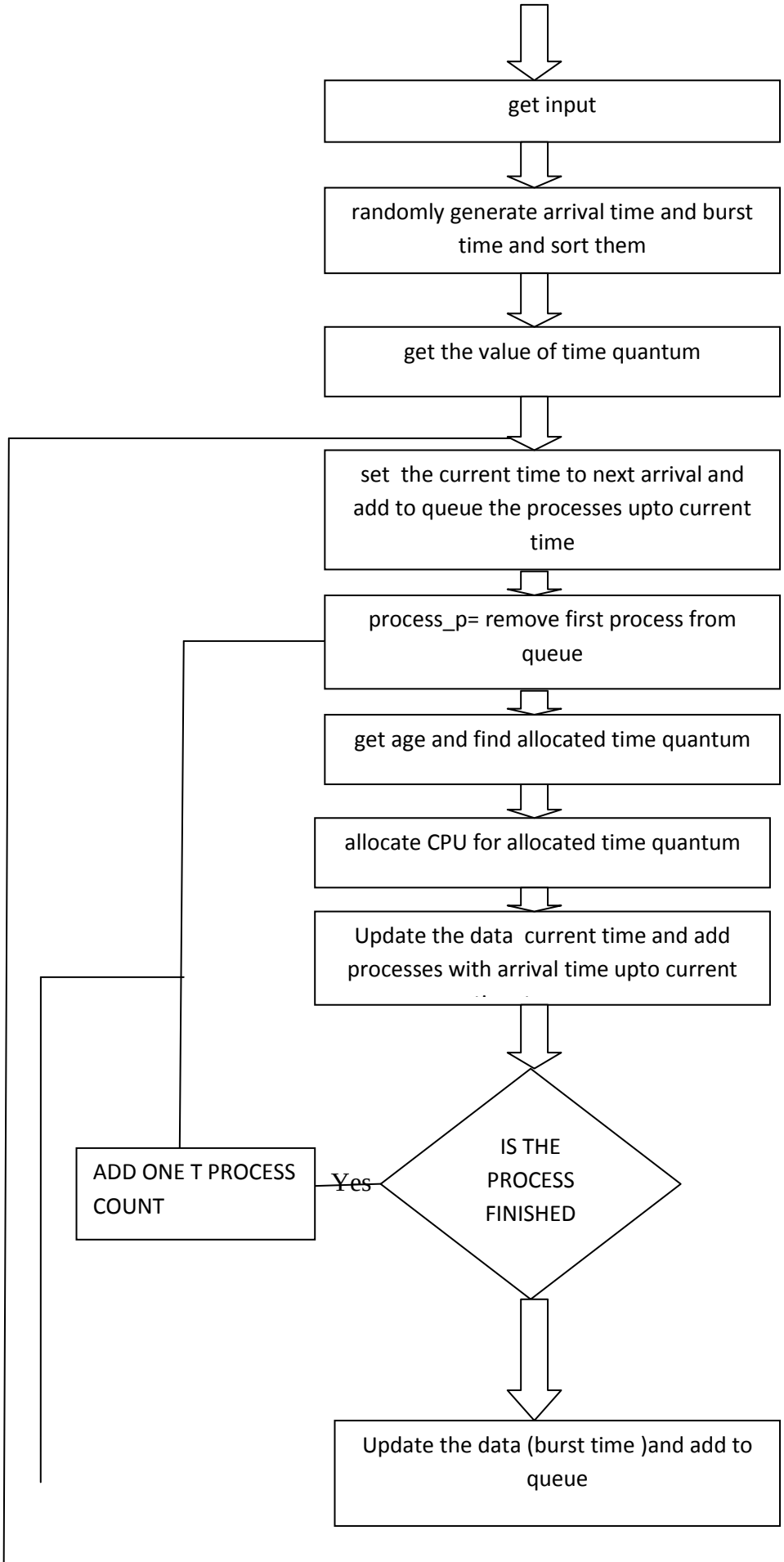
b. Add unexecuted processes with arrival_time upto current_time to the queue.

}

}

6.Get statistics.

4.3 Flow Chart



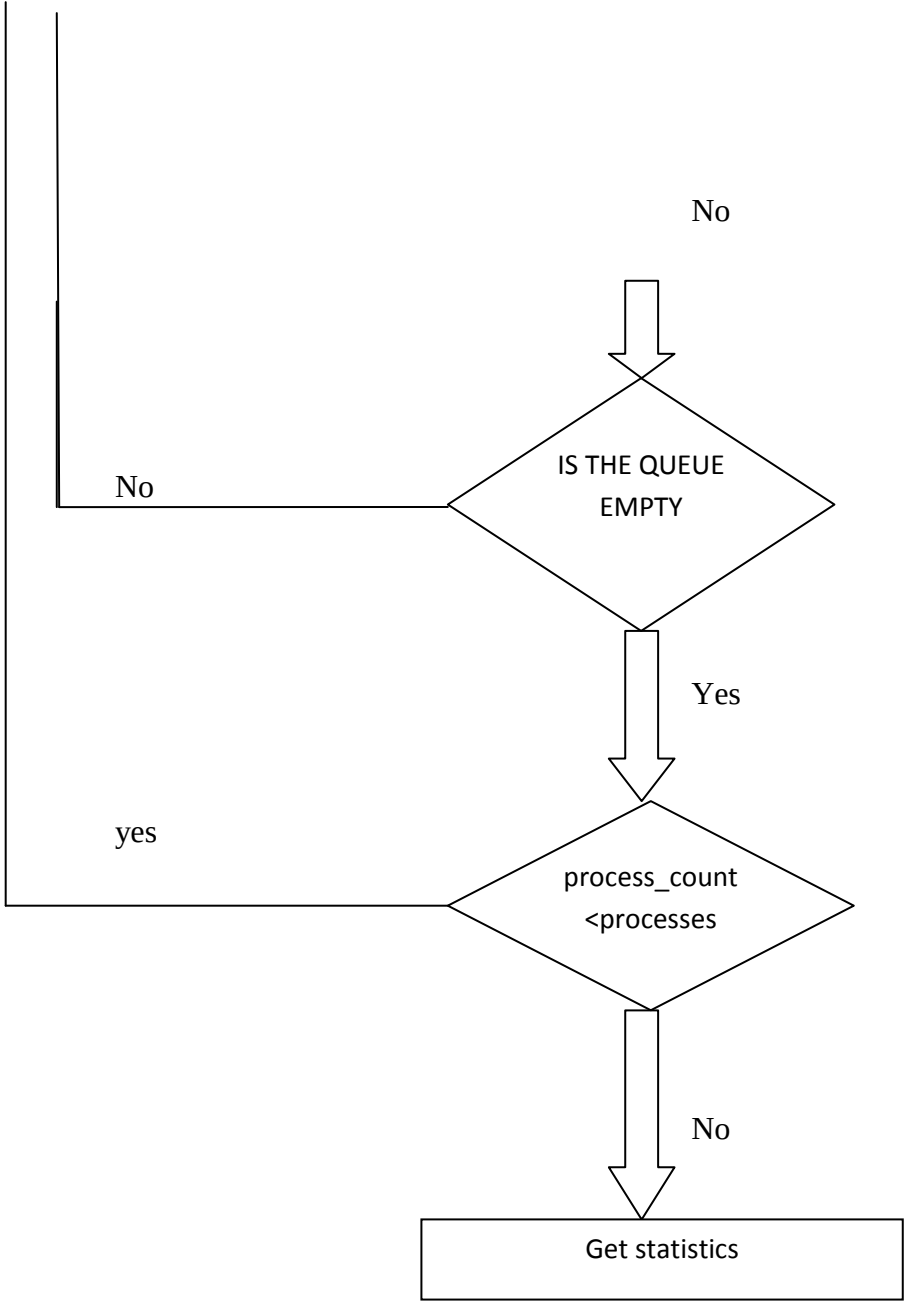


Figure 16:Flow chart of proposed model

CHAPTER 5

SIMULATION AND RESULTS

Now we look at the implementation of the model and test in on a variety of inputs.

5.1 System Architecture

The proposed model was implemented on Java Platform. The input to the system were

I. Number Of Processes

Ii. Burst Time Of Each Process

Iii. Arrival Time Of Each Process.

Iv. Boundary Of The Ages Ranges.

The number of processes are to entered and the boundary for the age are also entered then the system generates the arrival time and burst time of the process randomly using the function available in the Java. The values being random are essential so as to completely test the model against all the situations. Also the random values are important for comparing the model with the existing systems and getting results in a normal way.

After generating the random inputs the system is setup to simulate the model on these inputs as per the model. The important steps in doing so are

5.1.1 Prepare a table for the processes – This process table are prepared as storing table for the given input. This table will be used to schedule the processes. The values used in the table are

- Process id - Generated linearly according to the arrival of the process in the system.
- Arrival time – Generated Randomly.
- Burst time – Generated Randomly.
- Beparture time – To be calculated after the system is implemented.

Process Id	**Arrival Time**	**Departure time**	**Burst Time**
.....	.....	.....	.....

Process Table

5.1.2 Set up the counting time – This is the time which is used to keep track of the time and is used to properly run the simulation. It is initialized to the first event in the system.

5.2 Finding The Optimal Time Quantum For The Processes

In this step we find optimal time quantum for the processes. We look at the first 10% of the processes and calculate the factors. This is actually an indicator of the processes to come. The factors used for the calculation of the time quantum are

- **Number of the processes** – This factor is the number of processes on which the system has to function on.

- **Average burst time** – This is the average burst time of the processes under the consideration and this is very important for the system.

The variables are classified in three categories and the time quantum is decided upon categories of the variables

- **Average Burst Time**

    1. Low 2. Medium 3. Large

- **Number Of Processes**

    1. Low 2. Medium 3. Large

The value for the time quantum is decided by the following values

5.3 Setting Up Age Boundary To Find The Age Of The Process

In this step we decide the age boundaries to decide the age section of the process. The boundaries are entered by the user. The age classified are

i). Low – Below the first parameter.

2). Medium – Between first and second parameter.

3). Large – Above the second parameter.

5.4 Finding Proportion Of Time Quantum To Be Given To Each Age Boundaries

According to the of the process the process will get a proportion of the time quantum. The decided time quantum for each age is

i) Low – 0.6

ii) Medium - 0.8

iii) Large – 1

5.4.1 Procedure

The system now sets the working conditions to simulate the model. Now the algorithm is ran on the input. The algorithm will now act according to the input and the conditions. The output is then collected and statistics are now calculated.

5.4.2 Comparison

The algorithm is compared against the standard algorithms

- First, we compare it with round robin scheduling of a particular quantum.
- Secondly, we compare it with round robin scheduling with another time quantum.
- Thirdly, we compare it with the First come First serve Algorithm.

5.5 Finding The Optimal Fuzzy Values For The Input Categories

Now we find the appropriate values for the fuzzy classes for the variables

i) Processes – Number of processes

ii) Average burst time – the average burst time of the processes

iii) Time quantum – the time quantum to be calculated

Now we find the values for the fuzzy classes LOW, MEDIUM, LARGE for every of the inputs. The classes will then be used to classify the values into the numbers .The following test cases were run to find the optimal values.

	LOW	**MEDIUM**	**LARGE**
No. of processes	0-50	50-80	>80
Average burst time	0-15	15-25	>25
Time quantum	15	25	30

Table 1 :First configuration

Average turnaround time = 459.54

	LOW	**MEDIUM**	**LARGE**
No. of processes	0-50	50-80	>80
Average burst time	0-15	15-25	>25
Time quantum	20	25	30

Table 2 :Second configuration

Average turnaround time = 729.53

	LOW	MEDIUM	LARGE
No. of processes	0-50	50-80	>80
Average burst time	0-15	15-25	>25
Time quantum	10	20	30

Table 3 :Third configuration

Average turnaround time =337.36

	LOW	MEDIUM	LARGE
No. of processes	0-100	100-160	>160
Average burst time	0-25	25-50	>50
Time quantum	10	20	30

Table 4 : Fourth configuration

Average turnaround time = 1428.19

	LOW	MEDIUM	LARGE
No. of processes	0-10	10-15	>15
Average burst time	0-25	25-50	>50
Time quantum	10	20	30

Table 5 : Fifth configuration

Average turnaround time = 1624.83

	LOW	MEDIUM	LARGE
No. of processes	0-5	5-8	>8
Average burst time	0-5	5-10	>10
Time quantum	5	10	15

Table 6 : Sixth configuration

Average turnaround time = 308.33

The above results were obtained by running the configuration multiple times with different aging boundaries.

As found out the optimal value for variables in the general case is the last one i.e. the sixth configuration. As we take 10% of values the range specified is according to the requirement of the system model these are good values.

Also the average burst time is optimal because it once processes have average burst time greater than a value they get a good size of time quantum.

5.6 Results Analysis

Now we compare the results of the algorithm with the results of the existing algorithms.

	Proposed algorithm	**Round Robin(5)**	**First Come First Serve**	**Round Robin(10)**
Number of processes	10	10	10	10
Average Turnaround time	10.1	15.7	10.5	20.9
Average waiting time	5.1	10.7	5.5	15.9

Table7 : Simulation-1

	Proposed algorithm	**Round Robin(5)**	**First Come First Serve**	**Round Robin(10)**
Number of processes	20	20	20	20
Average Turnaround time	41.55	43.05	44.35	46.05
Average waiting time	36.94	38.44	39.75	41.44

Table 8 : Simulation -2

	Proposed algorithm	**Round Robin(5)**	**First Come First Serve**	**Round Robin(10)**
Number of processes	30	30	30	30
Average Turnaround time	66.63	69.46	67.86	70.43
Average waiting time	61.46	64.3	62.69	65.26

Table 9 : Simulation -3

	Proposed algorithm	Round Robin(5)	First Come First Serve	Round Robin(10)
Number of processes	50	50	50	50
Average Turnaround time	91.76	97.44	139.68	134.54
Average waiting time	85.9	91.58	133.82	128.67

Table 10 : Simulation -4

	Proposed algorithm	Round Robin(5)	First Come First Serve	Round Robin(10)
Number of processes	70	70	70	70
Average Turnaround time	144.21	160.32	176.84	149.91
Average waiting time	139.0	155.111	171.62	144.7

Table 11 : Simulation -5

	Proposed algorithm	Round Robin(5)	First Come First Serve	Round Robin(10)
Number of processes	100	100	100	100
Average Turnaround time	190.08	193.5	292.96	315.46
Average waiting time	184.18	187.6	287.06	309.56

Table 12 : Simulation -6

As we can see our algorithm is better than the others in most cases. This is because it judges the requirement of the processes in advance and the fuzzy logic provides a good value judging by few of the starting processes. Also the system model can be implemented in modern system without much of extra computation. Also the priority is defined by the process of aging. However, the aging parameter have not yet been completely defined. The system would give better results if these parameters are defined.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this research work we try to propose a new scheduling algorithm. The algorithm used round robin scheduling and was deciding the time quantum based on number of processes and their burst time using fuzzy computation. Also to prioritize older processes, aging was used.

The system was tested by implementing it as a simulation model. The results were compared against some standard algorithms like round robin with fixed time quantum and first come first serve and the results are gathered and the average turnaround time and average waiting time were compared. The model was found out to be better in most of the cases. This was due the fact that the time quantum was chosen according to processes. Also aging helped in better scheduling.

In future a better process for calculating the aging factor, with respect to the processes will be worked on. This will improve the performance of the system further by facilitating older processes. Also there is scope of improvement in fuzzy rule.

References

[1] Franco Callari, "Types of Scheduling - Long Term and Medium Term Scheduling".

[2] Silberschatz, A., Peterson, J. L., and Galvin, P.B., Operating System Concepts, Addison Wesley, 8th Edition.

[3] Daniel P. Bovet and Marco Cesati, "Understanding the Linux Kernel", O'Reilly Online Catalogue, 2000.

[4] Giorgio C. Buttazzo, "Hard Real Time Computing Systems: Predictable Scheduling Algorithms and Applications", Springer, Third edition.

[5] S. Vishwakarma and A. Agrawal, "A survey on activity recognition and behavior understanding in video surveillance," The Visual Computer, 2012.

[6] S.-W. Joo and R. Chellappa, "Attribute Grammar-Based Event Recognition and Anomaly Detection," in Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on, pp. 107–107.

[7] J. C. San Miguel and J. M. Martinez, "Robust unattended and stolen object detection by fusing simple algorithms," in Advanced Video and Signal Based Surveillance, 2008. AVSS'08.IEEE Fifth International Conference on, pp. 18–25, IEEE, 2008.

[8] J. C. SanMiguel, M. Escudero-Vinolo, J. M. Martinez, and J. Bescós, "Real-time singleview video event recognition in controlled environments," in Content-Based Multimedia Indexing (CBMI), 2011 9th International Workshop on, pp. 91–96, IEEE, 2011.

[9] O. P. Popoola and K. Wang, "Video-Based Abnormal Human Behavior Recognition— A Review," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 42, no. 6, pp. 865–878, 2012.

[10] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surv., vol. 41, pp. 1–58, July 2009.

[11] B. Zhao, L. Fei-Fei, and E. P. Xing, "Online detection of unusual events in videos via dynamic sparse coding," in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 3313–3320, 2011. 43 BIBLIOGRAPHY 44 .

[12]Wang Lie-Xin, A course in fuzzy systems and control, Prentice Hall, August 1996.

[13]Bashir Alam, 1M.N. Doja, R. Biswas,Finding Time Quantum of Round Robin CPU Scheduling Algorithm Using Fuzzy Logic ,2008.

[15] Saroj Hiranwal, Dr. K.C.Roy, "Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice," International Journal of Data Engineering (IJDE), Volume 2, Issue 3.

[16] Abdulrazaq Abdulrahim, Saleh E Abdullahi , Junaidu B. Sahalu ,A New Improved Round Robin (NIRR) CPU Scheduling Algorithm,2014.