Volume 2, No. 4, April 2011
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH ARTICLE
Available Online at www.jgrcs.info

PERFORMANCE EVALUATION OF A NEW PROPOSED SHOTREST EXECUTION FIRST DYNAMIC ROUND ROBIN (SEFDRR) SCHEDULING ALGORITHM FOR REAL TIME SYSTEMS

Prof. Rakesh Mohanty[1], Debapriya Maharana[2], Swarnaprava Tripathy[3]

Department of Computer Science and Engineering, Veer Surendra Sai University of Technology, Burla, Odisha, India
rakesh.iitmphd@gmail.com[1]
debbandana41@gmail.com[2]
Swarnapravatripathy7@gmail.com[3]

Abstract: Round Robin (RR) scheduling algorithm is not suitable for real time operating system because of high context switch rate, larger waiting time, and larger response time. In this paper, we have proposed a novel improved algorithm which is a variant of RR. Our proposed Shortest execution First Dynamic Round Robin (SEFDRR) algorithm calculates individual time slice for each task in each round. Our Experimental results show that SEFDRR algorithm performs better than Priority Based Simple Round Robin Algorithm (PBSRR) by decreasing the number of context switches, average waiting time, and average turnaround time.

Keywords: Operating System; Real Time System; Scheduling; Round Robin, Time slice; Priority.

INTRODUCTION

Operating system is a program that acts as an intermediary between the user and the computer hardware [8]. The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner. Operating System is responsible for managing the hardware resources of a computer and hosting applications that run on the computer. A *Real-Time Operating System* (RTOS) is an operating system intended to serve real-time application requests and guarantees a certain capability within a specified time constraint. RTOS is specially designed to run applications with very precise timing and a high degree of reliability. Space research, audio conferencing, video conferencing, money withdrawal from ATM are some of the important applications of real time systems.

Scheduling is an essential task for operating system in which the processes are assigned to the CPU for execution in multitasking environment. Mmultitasking is a method where multiple tasks, also known as processes, share common processing resources such as CPU . In the case of a computer with a single CPU, only one task is said to be *running* at any point in time, meaning that the CPU is actively executing instructions for that task. Multitasking solves the problem by scheduling tasks and determines which task may be the one running at any given time, and when another waiting task gets a turn. In multiprogramming systems, the running task keeps running until it performs an operation that requires waiting for an external event or until the computer's scheduler forcibly swaps the running task out of the CPU. Multiprogramming systems are designed to maximize CPU usage Multitasking is a logical extension of multiprogramming. Real Time Systems always have time constraints on computation. Real-time schedulers can schedule individual tasks for execution either offline (prior to the system entering its running state) or online (while the system is in an active, running state). Scheduler deals with many time related parameters that a task can complete on or before its deadline. Scheduling Algorithms are divided into sub-classes such as fixed-priority and dynamic-priority. If priority of the tasks does not change during running time, then it is called fixed priority. In dynamic priority, priority of the tasks change during the running time

RTOS Scheduling Algorithm

RTOS scheduling algorithms are divided into two types such as static scheduling and dynamic scheduling. Static scheduling algorithm is mainly done offline. Dynamic scheduling is an online scheduling algorithm. Static scheduling can be Rate Monotonic and Deadline Monotonic. Rate monotonic is an optimal fixed-priority policy where the higher the frequency (1/period) of a task, the higher is its priority. In deadline monotonic, tasks are assigned priorities according to their deadlines; the task with the shortest deadline being assigned the highest priority. Dynamic scheduling algorithms can be Earliest Deadline First (EDF). EDF is a dynamic pre-emptive scheduling, in which, the task with closest deadline is executed earlier.

RELATED WORK

Baskiyar and et al. have made a extensive survey on memory management and scheduling in RTOS[1]. A worst case response time analysis of real time tasks under hierarchical fixed priority pre-emptive scheduling is done by Bril and Cuijpers[2]. Yaasuwanth and et. al. have developed an modified RR algorithm for scheduling in real time systems[3]. Recently, a number of CPU scheduling algorithms have been developed for predictable allocation of processor. Self-Adjustment Time Quantum Round Robin Algorithm [5] is based on a new approach called dynamic time quantum in which, time quantum is repeatedly adjusted according to the burst time of the running processes. Dynamic Quantum with

Readjusted Round Robin Scheduling Algorithm [4] uses the job mix order for the algorithm in [5]. According to [4], from a list of N processes, the process which needs minimum CPU time is assigned the time quantum first and then highest from the list and so on till the Nth process. Again in the 2nd round, the time quantum is calculated from the remaining CPU burst time of the processes and is assigned to the processes and so on. Algorithms proposed in both [4] and [5] are better than RR scheduling and overcomes the limitations of RR scheduling. Recently improved variants of round robin algorithms SRBRR[7] and PBDRR[8] have been developed. A New Dynamic Round Robin and SRTN Algorithm with Variable Original Time Slice and Intelligent Time Slice has been proposed in [9]. In this paper, the time quantum that is repeatedly adjusted on a run-time basis according to the burst time of the running processes are considered to improve the waiting time, turn-around time and number of context switches.

Our contribution

The limitation of RR is the allocation of static time slice to the processes in every round of execution. In this paper, we have proposed a new variant of RR algorithm. In our proposed algorithm, we have assigned time slice to each process such that it changes with each round of execution dynamically. The overall performance of Shortest Execution First Dynamic RR(SEFDRR) is observed to be improved by using dynamic time quantum over Priority Based Static RR(PBSRR) for real time systems as per our experimental results

Organisation of our paper

Section II shows the background and preliminaries and the pseudo code and illustration of our proposed algorithm. In section III, experimental results are presented. Conclusion and future work is presented in section IV.

BACKGROUND PRELIMINARIES

Terminologies

A *process* is a program in execution. *Ready queue* holds the processes waiting to be executed or to be assigned to the processer. *Burst time (b_t)* is the time, for which a process requires the CPU for execution. The time at which a process arrives is called its *arrival time(a_t). Time quantum(t_q) or time slice* is the period of time CPU is assigned to each process. *Turn around time (t_{at})* is the time gap between the instant of process arrival and the instant of its completion. The number of times CPU switching from one process to another is called the *context switches (c_s). Range* is the average of maximum and minimum burst time.

Uniqueness of our Approach

The shorter processes which have less assumed CPU burst time than the processes are removed early form the ready queue and get better turnaround time and waiting time. So in our proposed algorithm, the shorter processes are given more time slice and early finish their execution. Round Robin algorithm upon size of time slice. If time quantum is very small, it cause too many context switches. If time quantum is very large, then the algorithm becomes FCFS. So our algorithm solves this problem by making choice of dynamic time slice appropriately, where the time slice are adjusted in every cycle according dynamic priority.

PSEUDO CODE OF OUR PROPOSED ALGORITHM

Here R =Range
N = No. of processes
P = No. of priorities

Input : No of processes(P_1, P_2, ……., P_n),
Burst time of processes (Bt_1, Bt_2,….Bt_n),
Priority of processes (Pr_1, Pr_2……Pr_m).

Output : Tav = Average turnaround time,
Wav = Average waiting time,
Ncs = Number of context switches.

Method:

1. Calculate Range as follows.
Range =(Bt_{max}+ Bt_{min}) / 2

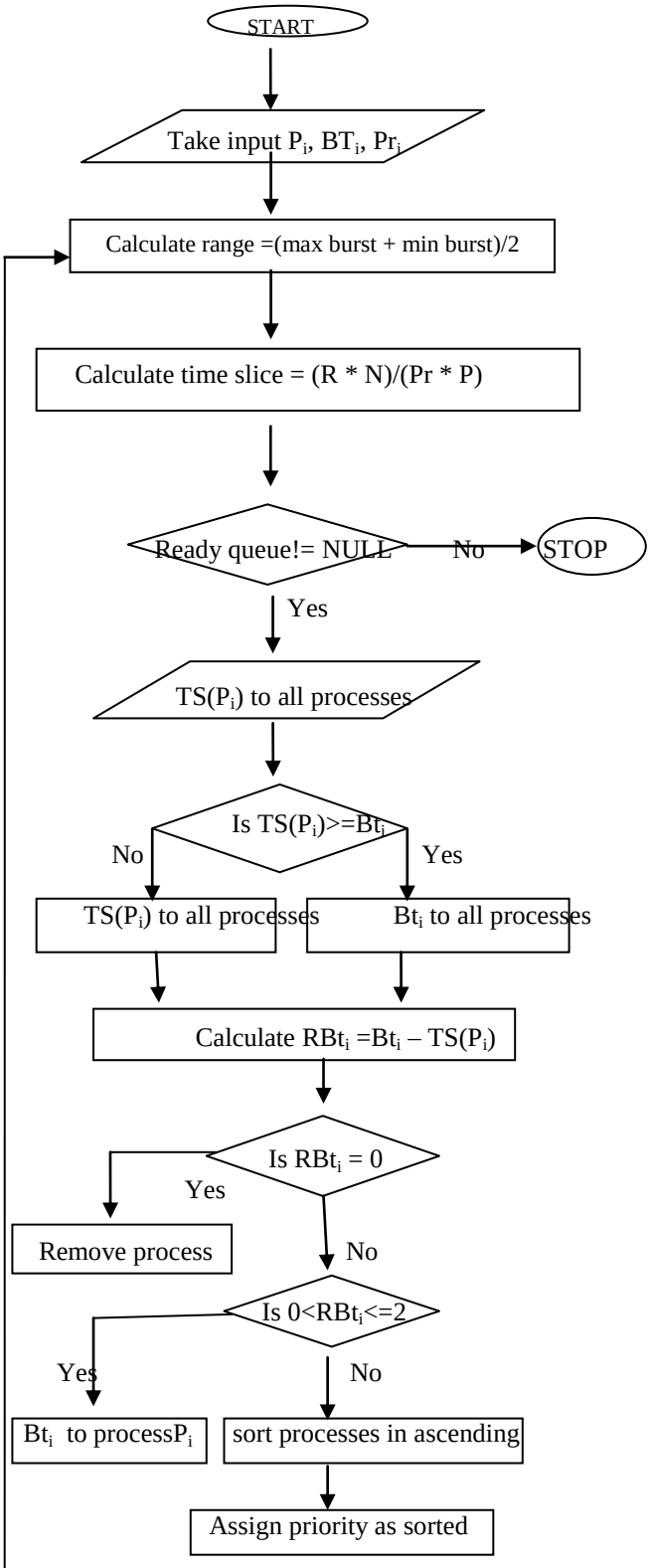
2. For i=1,2,…..n, Calculate time slice for each processes Pi as follows.
TS(Pi)=(R * N) / (Pr * P)

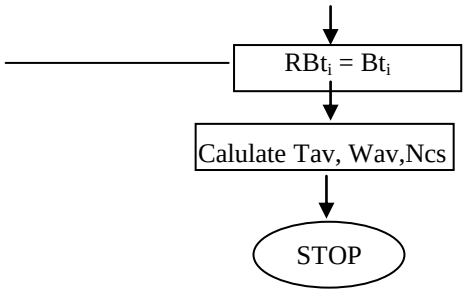
3. While (Ready queue != NULL)
{
Assign TS(P_i) to each process Pi for i=1,2,…..n.
for (i=1,2,……n)
{
If (TS(P_i) >= Bt_i)
Assign Bt_i to process P_i .
Else if (TS(P_i) < Bt_i)
Assign TS(P_i) to process P_i.
Else
RBt_i =Bt_i –TS(P_i).
}
For (i=1,2,……n)
{
If (RBt_i = = 0)
Remove Pi from the ready queue
Else if (0< RBt_i < 2)
Assign Bt_i to process P_i
}
}
4. Sort the processes present in the ready queue in Ascending order of RBt_i.
For all sorted processes P_i assign priority 1,2,….i
Assign RBt_i =Bt_i and go to step 1

5. Calculate Tav, Wav and Ncs.

End.

Flowchart of our Proposed Algorithm





.

Illustration of our Proposed Algorithm

To illustrate our proposed algorithm we have considered the following example. Let burst time of five processes are given as - 9 42 23 35 15 with used priority 3 2 1 4 5 respectively. Range is calculated by using equation I and is found as 26. We calculate the time slice by using equation II and time slice are found to be 9 13 23 7 6 respectively for the first round. Remaining burst time are found to be 0 29 0 28 9 after round 1. Processes having 0 remaining bust time are removed from the ready queue. The processes remaining in the ready queue are sorted in ascending order of remaining burst time. After that we take the priority as sorted and priority of burst time are 3 2 1. Now we again calculate just like above calculation. Here we found time quantum for the second round are 7 10 9. After completion of second round the remaining burst time are 22 18. And are given priority 2 1. After calculation there time slices are 10 18 respectively for the next round. This is the third round time quantum. And time slice of last round is 12. Now the time quantum are available for each round execution of processes. Then make the Gantt chart and calculate Tav, Wav, and no of cs.

EXEPERIMENTAL RESULTS

Assumptions

The environment where all the experiments are performed is a single processor environment and all the processes are considered to be independent. Time quantum is assumed to be not more than the maximum burst time. All the attributes like burst time, number of processes, the time slice of all the processes are known before submitting the processes to the processor. All the processes are assumed CPU bound.

Experimental framework

Our experiments consist of several input and output parameters. The input parameters consist of no of processes, burst time, and priorities. The output parameters consist of average waiting time, average turnaround time and number of context switches. We consider four different cases for our experiments.

CASE 1:

We Assume five processes with increasing burst time (P1 = 5, P2 = 12, P3 = 16, P4 = 21, p5= 23) and priority (p1=2, p2=3, p3=1, p4=4, p5=5) as shown in Table-1. The Table-2 and Table-3 show the output using PBSRR algorithm and our new proposed SEFDRR algorithm respectively. Fig-4.1 shows Gantt chart for both the algorithms respectively.

Process	Burst time	priority
1	5	2
2	12	3
3	16	1
4	21	4
5	23	5

Table1: Input data for CASE1

Pi	Bt	Pr	R	N	P	TS
1	5	2	14	5	5	7
2	12	3	14	5	5	5
3	16	1	14	5	5	14
4	21	4	14	5	5	4
5	23	5	14	5	5	3

Table 2: Time slice for PBSRR (CASE 1)

process	Burst time	Priority(Pr)	Rounds			
			1ST	2ND	3RD	4TH
1	5	2	5	0	0	0
2	12	3	5	7	0	0
3	16	1	16	0	0	0
4	21	4	4	4	5	8
5	23	5	3	3	3	14

Table 3: Dynamic time slice for SEFDRR (CASE 1)

P1	P2	P3	P4	P5	P2	P4	P5	P4	P5	P4	P5

0 5 10 26 30 33 40 44 47 52 55 63 77

Fig1: Gantt chart for SEFDRR (CASE 1)

ALGORITHM	TAV	WAV	NCS
PBSRR	47.2	31.8	19
SEFDRR	42.2	26.8	11

Table 4: Comparison of PBSRR and SEFDRR(CASE 1)

Case 2: We Assume five processes with decreasing burst time (P1 = 63, P2 = 54, P3 = 30, P4 = 12, p5= 5) and priority (p1=3, p2=2, p3=4, p4=1, p5=5) as shown in Table-5. The Table-6 and Table-7 show the output using PBSRR algorithm proposed in paper and our new proposed algorithm respectively.

PROCESS	BURST TIME	PRIORITY
1	63	3
2	54	2
3	30	4
4	12	1
5	5	5

Table 5: Input data for CASE 2

Pi	Bt	Pr	R	N	P	TS
1	63	3	34	5	5	12
2	54	2	34	5	5	20
3	30	4	34	5	5	9
4	12	1	34	5	5	12
5	5	5	34	5	5	5

Table 6: Time slice for PBSRR (CASE 2)

Process	Burst time	Priority(Pr)	ROUNDS			
			1ST	2ND	3RD	4TH
1	63	3	12	12	14	25
2	54	2	20	18	16	0
3	30	4	9	21	0	0
4	12	1	12	0	0	0
5	5	5	5	0	0	0

Table 7: Dynamic time slice of SEFDRR (CASE 2)

P1	P2	P3	P4	P5	P1	P2	P3

0 12 32 41 53 58 70 88 109

….	P1	P2	P1

109 123 139 164

Fig 2: Gantt chart for SEFDRR (CASE 2)

ALGORITHM	TAV	WAV	NCS
PBSRR	109.8	77	14
SEFDRR	106.4	73.6	10

Table 8: Comparison of PBSRR and SEFDRR (CASE 2)

Case 3: We Assume five processes with random burst time (P1 = 30, P2 = 8, P3 = 24, P4 = 19, p5= 46) and priority (p1=5, p2=3, p3=2, p4=1, p5=5) as shown in Table-9. The Table-10 and Table-11 show the output using PBSRR algorithm and SEFDRR algorithm respectively.

Process	Burst time	priority
1	30	5
2	8	3

3	24	2
4	19	1
5	46	4

Table 9: Input data (CASE3)

1	10	2
2	23	4
3	15	1
4	34	3
5	15	5

TABLE 13: Input data (CASE 4)

Pi	Bt	Pr	R	N	P	TS
1	30	5	27	5	5	6
2	8	3	27	5	5	8
3	24	2	27	5	5	14
4	19	1	27	5	5	19
5	46	4	27	5	5	7

TABLE 10: Time Slice of PBSRR (CASE 3)

Pi	Bt	Pr	R	N	P	TS
1	10	2	22	5	5	10
2	23	4	22	5	5	6
3	15	1	22	5	5	15
4	34	3	22	5	5	8
5	15	5	22	5	5	5

TABLE 14: Time Slice of PBSRR (CASE 4)

process	Burst time	Priority(Pr)	Rounds			
			1ST	2ND	3RD	4TH
1	30	5	6	13	11	0
2	8	3	8	0	0	0
3	24	2	14	10	0	0
4	19	1	19	0	0	0
5	46	4	7	9	11	19

TABLE 11: Dynamic time slice for SEFDRR (CASE3)

process	Burst time	Priority(Pr)	Rounds			
			1ST	2ND	3RD	4TH
1	10	2	10	0	0	0
2	23	4	6	9	8	0
3	15	1	15	0	0	0
4	34	3	8	6	7	13
5	15	5	5	10	0	0

TABLE15: Dynamic time slice of SEFDRR (CASE 4)

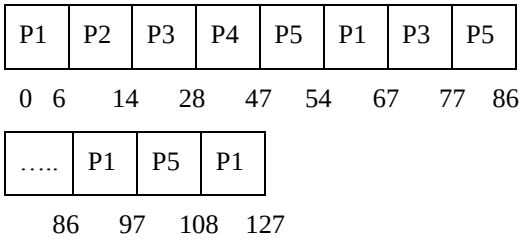


Fig 3: Gantt chart of SEFDRR (CASE 3)

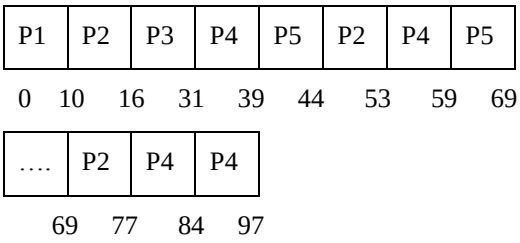


Fig 4: Gantt chart of SEFDRR (CASE 4)

ALGORITHM	TAV	WAV	NCS
PBSRR	74.4	48	15
SEFDRR	73.4	47	10

TABLE12: Comparison of PBSRR and SEFDRR (CASE3)

TYPE OF TS	TAV	WAV	NCS
PBSRR	61.6	41.8	13
SEFDRR	56.8	36.8	10

TABLE16: Comparison of PBSRR and SEFDRR (CASE 4)

Case 4: We Assume five processes with same burst time (P1 = 10, P2 = 23, P3 = 15, P4 = 34, p5= 15) and distinct priority (p1=2, p2=4, p3=1, p4=3, p5=5) as shown in Table-13. The Table-14 and Table-15 show the output using PBSRR algorithm and our new proposed SEFDRR algorithm respectively.

Process	Burst time	Priority

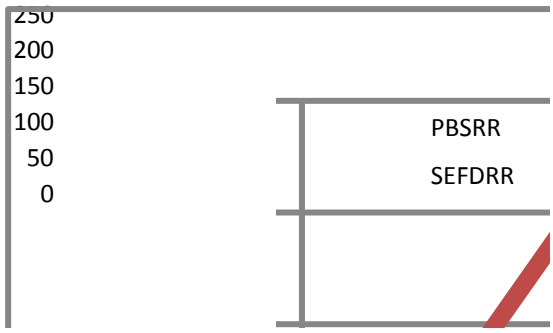


Fig 5: Comparison of Performances of SEFDRR and PBSRR based on Average Turn Around Time

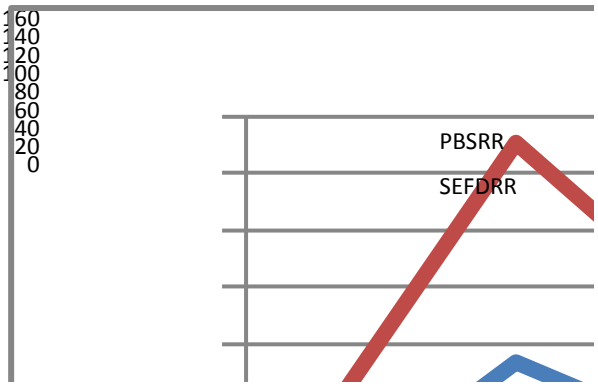


Fig 6: Comparison of Performances of SEFDRR and PBSRR based on Average Waiting Time

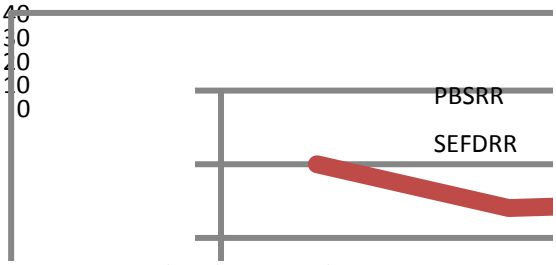


Fig 6: Comparison of Performances of SEFDRR and PBSRR based on number of context switches

CONCLUSION

Our experimental results show that our new proposed SEFDRR algorithm is performing better than the PBSRR algorithm in terms of average waiting time, average turnaround time and number of context switches. Deadline can be considered as another input parameter along with priority in our proposed algorithm to develop new variant algorithm suitable for hard real time systems.

REFERENCES

[1] S. Baskiyar and N. Meghanathan *: A Survey On Real Time Systems*, Informatica (29), 233-240, 2005

[2] Reinder J. Bril and Pieter J. L. Cuijpers: *Analytical of hierarchical fixed priority pre-emptive scheduling revised*, TU/e, CS-Report, 06-36, December, 2006.

[3] C. Yaashuwanth and R. Ramesh: Design of real time scheduler simulator and Development of Modified Round Robin Architecture, 2010

[4] H.S. Behera, R. Mohanty, Debashree Nayak "*A New Proposed Dynamic Quantum with Readjusted Round Robin Scheduling Algorithm and its Performance Analysis*", International Journal of Computer Applications(0975-8887) Volume 5- No.5, August 2010.

[5] Rami J. Matarneh. "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of Now Running Processes", American J. of Applied Sciences 6(10):1831-1837, 2009.

[6] A. Silberschatz, P.B. Galvin, G. Gange, "*Operating Systems Concepts*, 7th Edn, John Wiley and Sons, USA, ISBN:13:978-0471694663, 2004.

[7] Rakesh Mohanty, H.S. Behera and et. al, *Design and Performance Evaluation of a new proposed Shortest Remaining Burst Round Robin(SRBRR) scheduling algorithm*, Proceedings of the International Symposium on Computer Engineering and Technology(ISCET), March, 2010.

[8] Rakesh Mohanty, H.S. Behera and et. al., *Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems*, International Journal of Advanced Computer Science and Applications(IJACSA), Vol 2 No. 2, pp 46-50, February 2011.

[9] H. S. Behera and et. al. A New Dynamic Round Robin and SRTN Algorithm with Variable Original Time Slice and Intelligent Time Slice for Soft Real Time Systems. International Journal of Computer Applications 16(1):54–60, February 2011.

SHORT BIODATA OF ALL THE AUTHORS

Prof. Rakesh Mohanty is currently working as a Lecturer in Dept. of Computer Science and Engineering, VSS University of Technology, Burla, Orissa, India. His research areas of interest include Operating Systems, Data Structures and Algorithms.

Debapriya Maharana is a 4th year B. Tech. student in Dept. of Comp. Science and Engineering, VSS University of Technology, Burla, Orissa, India. Her area of interest is operating systems.

Swarnaprava Tripathy is a 4th year B. Tech. student in Dept. of Computer Science and Engineering, VSS University of Technology, Burla, Orissa, India. Her areas of

interest are operating systems and Data structures.

