

Assessment of Response Time for New Multi Level Feedback Queue Scheduler

M.V. Panduranga Rao¹, K.C. Shet²

¹Research Scholar, NITK, Surathkal, Mangalore, India

²Professor, NITK, Surathkal, Mangalore, India

¹raomvp@yahoo.com

²kcschet@rediffmail.com

Abstract— Response time is one of the characteristics of scheduler, happens to be a prominent attribute of any CPU scheduling algorithm. The proposed New Multi Level Feedback Queue [NMLFQ] Scheduler is compared with dynamic, real time, Dependent Activity Scheduling Algorithm (DASA) and Locke's Best Effort Scheduling Algorithm (LBESA). We abbreviated beneficial result of NMLFQ scheduler in comparison with dynamic best effort schedulers with respect to response time.

Keywords— NMLFQ, scheduler, process, queue, deadline, preemption, priority, Multi Level, DASA, LBESA, Dynamic, Best Effort and algorithm.

I. INTRODUCTION

In a multiprocessing environment of computer, the operating system's theme is to assign the execution time for various processes. The main component of kernel that creates the "selection" is called the scheduler [1][4][31][44][51]. The representation of the policy of scheduler employed is called the scheduling algorithm. Numerous different scheduling algorithms have been deduced over the years.

II. REVIEW OF DYNAMIC BEST EFFORT REAL-TIME SCHEDULING ALGORITHMS

At least two flavors of best-effort real-time scheduling algorithms exists. This comprises of the Dependent Activity Scheduling Algorithm (DASA) and Locke's Best Effort Scheduling Algorithm (LBESA) [2][6][38][42][49].

In fact, the DASA and LBESA scheduling algorithms are comparable with the canonic Earliest Deadline First (EDF) algorithm during under-loaded conditions. In this case, the EDF promises to meet entire deadlines and is always optimal. However, in the case of an over-loaded circumstance, DASA and LBESA attempt to increase the total task benefit [12][17][34].

The author R.K. Clark et al. [43], demonstrates that DASA more often, overtakes LBESA for the duration of over-loaded circumstances. Peng Li and Ravindran [41], confirms that DASA overtakes the Robust Earliest Deadline First (RED) scheduling algorithm [15][19][35].

Burns A et al. [10], asserts that alternative features of DASA and LBESA are utilized for the development of MK7.3

kernel. R.K. Clark et al. [43], emphasizes that wide-ranging characteristics of DASA and LBESA are exploited for the growth of Alpha real-time operating system [13][16].

According to R.K. Clark et al. [43], DASA and LBESA are the decent, benefit accrual scheduling algorithms. Both of these algorithms are employed to utmost extent for the growth of mission critical systems [25][26].

The characteristic and performance measurement parameters of each of the scheduler are known from previous work of others. The literature survey provided several outstanding noticeable features [9][14][27][32].

After studying policy mechanisms of different available schedulers, a New Multilevel Feedback Queue (NMLFQ) scheduling algorithm is proposed. NMLFQ includes all necessary modules to compete as a real time scheduler applied in embedded system domain [18][22].

The MLFQ principle of operation is used in NMLFQ scheduling algorithm in such a way that the response time is reduced and the functionality of the scheduling is improved [21][23][33]. The maximum number of queues and the quantum burst time for each queue are found using dynamic method. In NMLFQ scheduling, the operating system can modify the number of queues and the quantum of each queue according to the obtainable processes, deadline and as well as urgency consideration of the process (Garcia P. et al. [16]).

In *Earliest-Deadline-First* scheduling (EDF) priority level for every task is neglected. It primarily concentrates on completion of each task. It focuses on selection of task with nearest deadline for execution. Respectable CPU utilization will be gained, only when the chosen task is having precise deadline [3][20][36][42]. The canonic algorithm for real-time operating systems is EDF. It practices dynamic scheduling principle. The ready process from pool of priority queue is to be selected. The chosen process from the queue will be searched, for the process nearest to its deadline. The selected process is then scheduled to begin execution. The main drawback of the algorithm is, as and when the system is overloaded, the set of processes will miss deadlines and the system leads to deadlock.

Fig. 1, demonstrates the problem of missing of tasks in response to a deadline. The X axis corresponds to the values of time in milliseconds. Y axis corresponds to light weight processes or small threads. In the example of a load sharing of several time restraint jobs, response is expected at accurate time [5][7][39][47][53]. As the fig. 1, illustrates, request is made at 3 milliseconds. The response is expected within deadline period. Here a dispatch latency of 1 millisecond is suffered during the operation. Henceforth the response is sent after the deadline period. The response is sent at 4 milliseconds missing the deadline, in turn results in invalid response. The data object is passing some vital data to another object involving many processes and threads. We expect to have a response to a request within three milliseconds. The requirement will be perfect if the system reacts within two milliseconds. If the response comes after three milliseconds, the job execution crashes. There exists a sleek time of 1 millisecond from 3 to 4 milliseconds. So here the task missing the deadline results in invalid response.

In general a scheduler is written precisely to respect application priorities. It is necessary for real time applications need to be developed with limited dispatch latency [8][10][28][40][45]. The term *dispatch latency* designates the amount of time a system takes to respond to a request for a process to begin operation. The complete application response time comprises of the interrupt response time, the dispatch latency and the application's response time. The system detects that a process with higher priority than the interrupted process is now ready to dispatch and dispatches the process. The time to switch context from a lower priority process to a higher priority process is comprised in the dispatch latency time.

The shorter version of the pseudo-code of NMLFQ scheduler is described in the following section. This includes dynamic priority of arriving processes. Construction of queue is performed by loading with, certain number of processes to carry out small jobs [11][20][29][37].

Depending on the load of jobs for the scheduler, the numbers of processes are dynamically varied in ready queue. We can designate several levels of queues, depending on waiting condition of processes and urgency conditions of tasks. At any instant of time and with any number of processes and jobs to be completed, scheduler must guarantee meeting the deadlines of the hard real-time tasks.

```
PUBLIC CLASS BEGIN : DynamicPriority
    VARIABLE : Vector<Process> readyQueue =new Vector();
    VARIABLE : Hashtable<Integer, Process> pQueue = new
    Hashtable<Integer, Process>();
    VARIABLE : public Process p=null;
    VARIABLE : public Process removedProcess=null;
    CONSTRUCTOR BEGIN : DynamicPriority()
        //Initially loading Queue with some number of
        processes.
```

```
MONITOR BEGIN :
    FOR BEGIN :(int i=1;i<=pQueue.size();i++)
        //Creating process
        p=Runtime.getRuntime().exec("ps -ef");

        insertIntoPQueue(i*10,p);
    FOR END
MONITOR END
CATCH BEGIN (Exception ex)
    ex.printStackTrace();
CATCH END
CONSTRUCTOR END
CONSTRUCTOR BEGIN : public DynamicPriority(int i)
    MONITOR BEGIN :
        //Removing the requested number of processes from
Queue
        OUTER IF BEGIN (i<10)
        FOR BEGIN (int j=0;j<i;j++)
            removedProcess=removeAtEnd(j);
            INNER IF BEGIN (removedProcess != null)
                System.out.println("process is removed from
queue");
            INNER IF END
            INNER ELSE BEGIN
                System.out.println("process removed from the
queue is null");
            INNER ELSE END
        FOR END
        OUTER IF END
        OUTER ELSE BEGIN
            addAtFront(i, Runtime.getRuntime().exec("ps -
ef"));
        OUTER ELSE END
    MONITOR END
    CATCH BEGIN :(Exception ex)
        ex.printStackTrace();
    CATCH END
CONSTRUCTOR END
```

/ This method add the process to the readyQueue based on priority. It will check the priority and moves the other processes in readyQueue accordingly. */*

```
METHOD BEGIN : public void addAtFront(int priority,
Process temp)
    Enumeration enumitr = pQueue.keys();
    WHILE BEGIN :(enumitr.hasMoreElements())
        IF BEGIN : (!(priority
<(Integer)enumitr.nextElement()))
            pQueue.put(priority, temp);
            readyQueue.add(temp);
        IF END
    WHILE END
METHOD END
```

/ This method will insert some processes initially into Queue and also readyQueue(implementation details) */*

```

METHOD BEGIN : public void insertIntoPQueue(int priority,
Process process)
    //Adding process to the Queue and also into
readyQueue
    IF BEGIN :(process != null)
        pQueue.put(priority, process);

        readyQueue.addAll(pQueue.values());
        System.out.println("process is added into Ready
Queue");
    IF END
    ELSE BEGIN
        System.out.println("process is null");
    ELSE END
METHOD END

```

```

/* This method removes the process from the Queue. */
METHOD BEGIN : public Process removeAtEnd(int index)
    //Removing the process from the index specified
from both readyQueue and pQueue
    readyQueue.remove(index);
    RETURN : return(pQueue.remove(index));
METHOD END

```

/* Algorithms guarantee that if a task is accepted for execution, the task and all previous tasks accepted by the algorithm will meet their time constraints [22][26][30][48]. The planning based algorithms attempt to improve the response and performance of a system to aperiodic and soft real-time tasks while continuing to guarantee meeting the deadlines of the hard real-time tasks. */

```

METHOD BEGIN : public void planningBased()
    IF BEGIN (true)
        //Task is accepted for execution
        //Resource are allocated.
        //Responds in time
    IF END
METHOD END

```

```

METHOD BEGIN : public void bestEffortBased()
    IF BEGIN : (true)
    MONITOR BEGIN
        Enumeration enumitr = pQueue.keys();
        Vector sorted = new Vector();

        WHILE BEGIN : (enumitr.hasMoreElements())
            int dynArr =(Integer)enumitr.nextElement();
            sorted.add(dynArr);
        WHILE END
            Collections.sort(sorted);
        FOR BEGIN (int i=0;i<sorted.size();i++)
            //Before calling shortest job first set the priority for
the job.
            ExecutingProcess((Process)sorted.get(i++));
        FOR END
    MONITOR END

```

```

CATCH BEGIN (Exception ex)
    ex.printStackTrace();
CATCH END
    IF END
METHOD END

/* Actual execution started. */
METHOD BEGIN : public void ExecutingProcess(Process
process)
    //Actual Execution happens
    System.out.println("Execution started...");
METHOD END

```

```

/* We are creating pQueue as a placeholder for processes
which are ready to execute. As it is DynamicPriorityQueue,
we have to use HashTable for storing both priority and
processes. Processes will be added based on the priority to the
Queue and will removed from the end or again, based on
priority form [24][46][50]. */
METHOD MAIN BEGIN : public static void main(String[]
args)
    // TO-DO Auto-generated method stub
    //Creating Queue
    CALLING METHOD : new DynamicPriority();
    //Removing processes from Queue
    CALLING METHOD : new DynamicPriority(2);
    CALLING METHOD : new DynamicPriority(110);
    MONITOR BEGIN
        CALLING METHOD : new
DynamicPriority().addAtFront(2,Runtime.getRuntime()).exec(
"ps -ef");
        CALLING METHOD : new
DynamicPriority().planningBased();
        CALLING METHOD : new
DynamicPriority().bestEffortBased();
    MONITOR END
        CATCH BEGIN(IOException e)
            // TO-DO Auto-generated catch block
            e.printStackTrace();
        CATCH END
    METHOD MAIN END
CLASS END

```

A job is a function of many tasks or processes. Division of jobs into task modules is performed from analysis of jobs. All the steps of NMLFQ scheduler are summarized in fig. 2. This model depicts the coordination amongst various modules of scheduler. Jobs are divided into task modules after the analysis of the nature of jobs. The history of jobs is maintained for future use. The processes are allocated to lock the resources with the control of interrupt handler. The task controller is made to govern for time stamping of deadline. It also supervises ready, pending, blocked and sleeping tasks. The evaluation of priority levels of processes, ordering of processes and division into time slices is controlled by scheduler through the task controller. The peripherals are associated to the CPU through input output controller. This

occasionally makes use of context switching. Several queues are supervised by a substantial queue controller. As stated earlier, mapping of processes to critical resource is achieved by semaphores. Dynamic prioritization is a vital task of scheduler, which necessitate raising or scaling of priority of processes.

III. NMLFQ COMPARISON WITH BEST-EFFORT REAL-TIME SCHEDULING ALGORITHMS - DASA AND LBESA

The NMLFQ real time scheduler is compared with existing best-effort real-time scheduling algorithms. This comprise of the Dependent Activity Scheduling Algorithm (DASA) and Locke’s Best Effort Scheduling Algorithm (LBESA) [20][52][54]. The comparison of NMLFQ scheduler with DASA and LBESA is accomplished for twenty test cases. The results of NMLFQ, DASA and LBESA for twenty different set of inputs are reckoned. The results provided ameliorate characteristics for CPU utilization, overall turnaround time, average turnaround time, average waiting time and average response time of schedulers. The comparative results are analyzed for several processes correspondingly, as shown in fig. 3 and generalized graph of hundreds of processes can also be drawn. Fig. 3, exemplifies, Average Response time for NMLFQ scheduler, compared with real time DASA and LBESA scheduers for twenty testcases, proves 10 to 25% reduction of response time. In this research paper, We have depicted the performance with respect to Average Response time for NMLFQ scheduler

Fig. 3, proves as per the achieved results, average response time for NMLFQ scheduler, compared with real time DASA and LBESA scheduers for twenty testcases, illustrates 10 to 25% reduction of response time in each subsidiary stage.

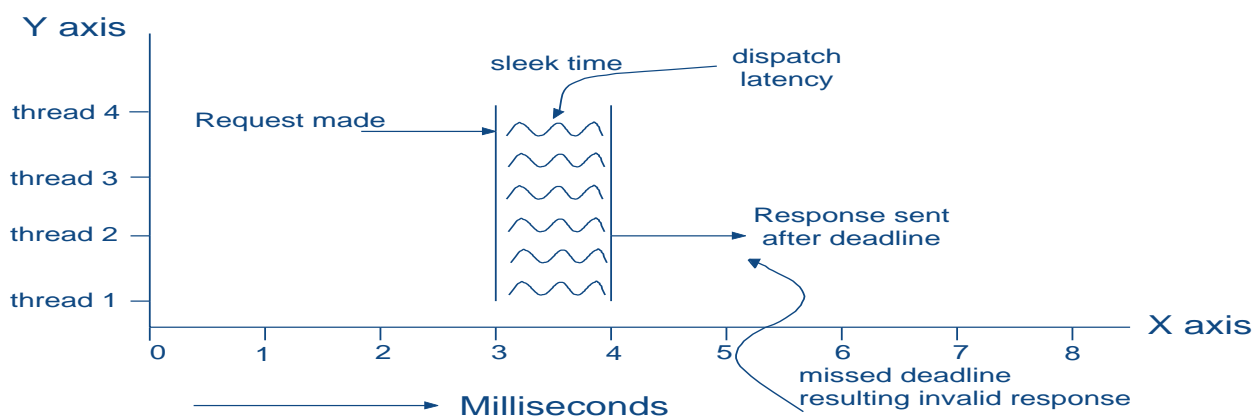


Fig. 1 Task missing the deadline results in invalid response.

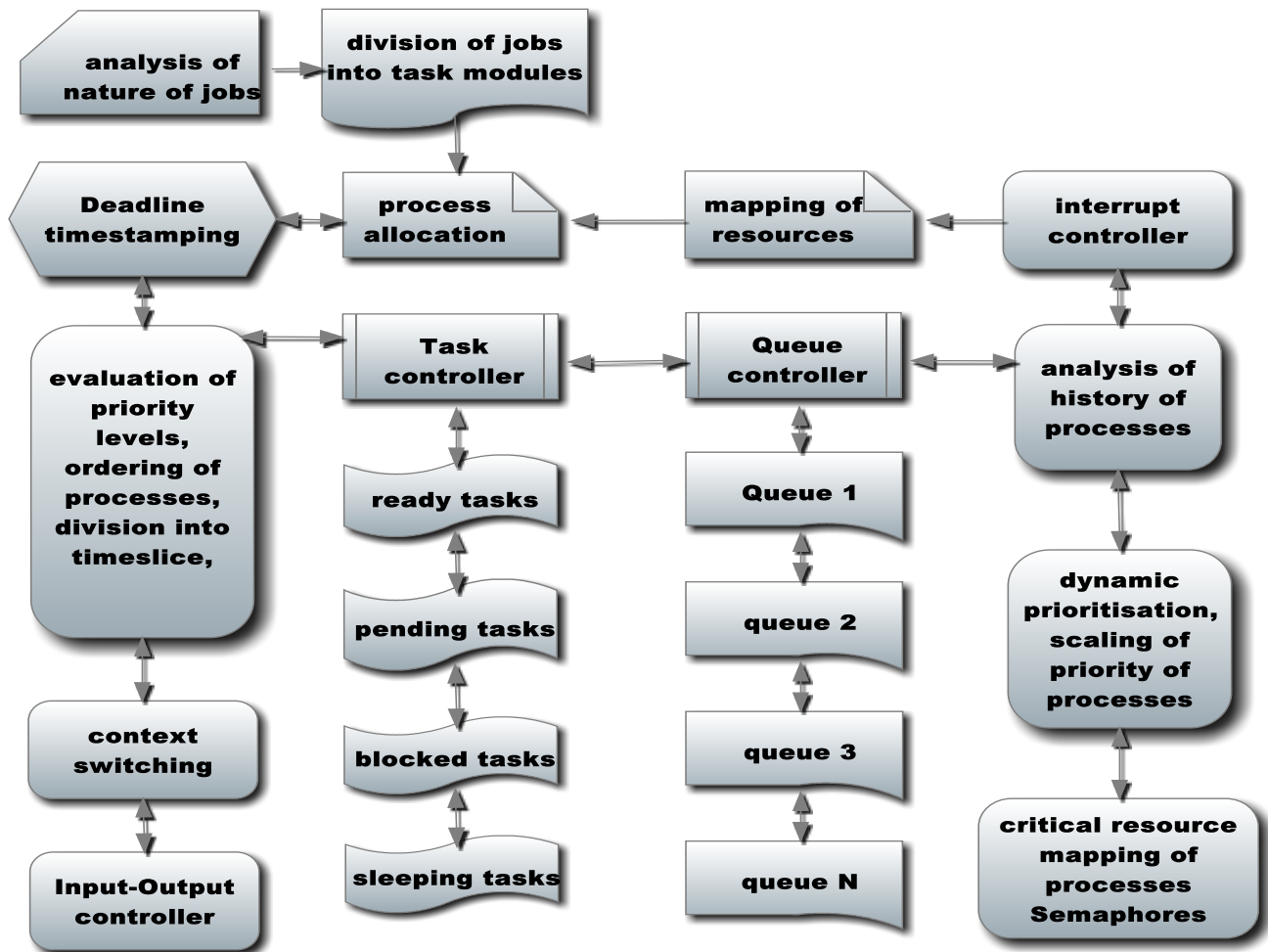


Fig. 2 Model of detailed conceptual view of NMLFQ including several modules.

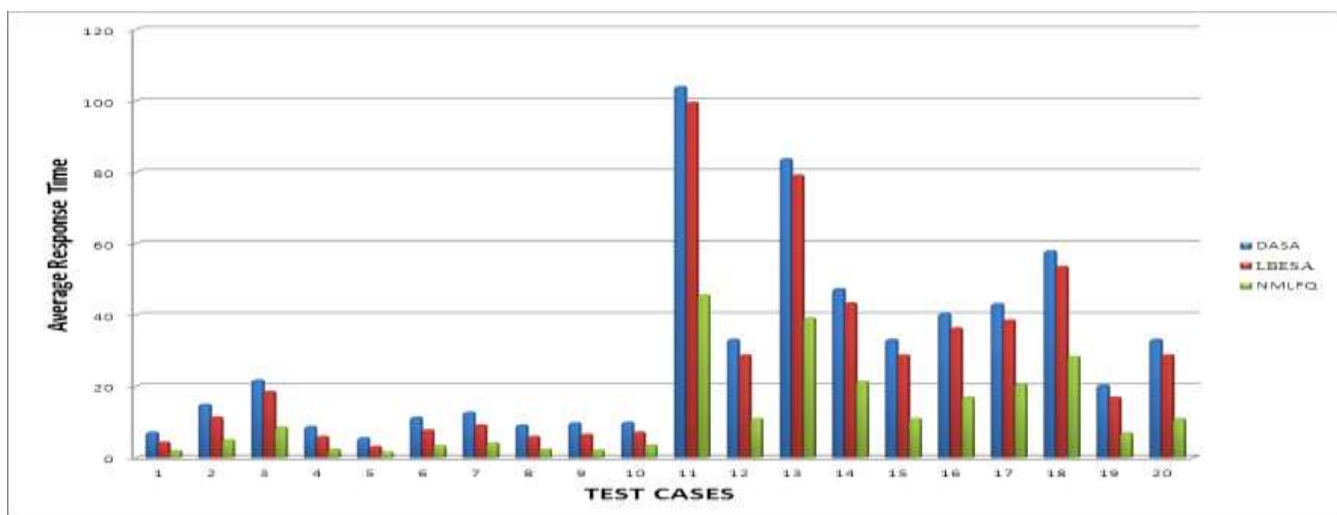


Fig. 3 Average Response time for NMLFQ scheduler, compared with real time DASA and LBESA schedulers for twenty testcases, depicts 10 to 25% reduction of response time in each subsidiary stage.

IV. CONCLUSIONS

In this research paper, we have discussed the themes associated with NMLFQ scheduler. The basic review of dynamic best effort real-time scheduling algorithms is explained. Comparison of proposed scheduler is made, with best-effort real-time scheduling algorithms - DASA and LBESA. Eventually, Model of detailed conceptual view of NMLFQ including several modules is also discussed in short.

REFERENCES

- [1] Abdelzaher T. F., Sharma V., and Lu C., "A Utilization Bound for Aperiodic Tasks and Priority Driven Scheduling", in *IEEE Transactions on Computer*, Volume 53, Number 3, Page(s): 334-350, 2004.
- [2] Albert Haque, "An Analysis and Comparison of Processor Scheduling Techniques", *University of Texas at Austin*, ahaque@cs.utexas.edu, December 7, 2012.
- [3] Arezou Mohammadi and Selim G. Akl, "Scheduling Algorithms for Real-Time Systems", *Technical Report No. 2005-499, Natural Sciences and Engineering Research Council of Canada*, School of Computing, Queen's University, July 15, 2005.
- [4] Ashiq Anjum, Richard McClatchey, Arshad Ali and Ian Willers, "Bulk Scheduling With Diana Scheduler", in *proceedings of IEEE Transactions on Nuclear Science*, ISSN 0018-9499, Volume 53, Issue 6, Page(s): 3818-3829, 2006.
- [5] Ayan Bhunia, "Enhancing the Performance of Feedback Scheduling", *International Journal of Computer Applications*, ISSN 0975 – 8887, Volume 18, Number.4, March 2011.
- [6] Baskiyar S and Meghanathan N, "A Survey of Contemporary Real-time Operating Systems", *Informatica*, Volume 29, Auburn University, Auburn, USA, Page(s): 233–240, 2005.
- [7] Becchetti, L., Leonardi, S. and Marchetti S.A., "Average-Case and Smoothed Competitive Analysis of the Multilevel Feedback Algorithm", *Mathematics of Operation Research*, Volume 31, Number 1, February, Page(s): 85–108, 2006.
- [8] Behera H. S., Rakesh Mohanty, Sabyasachi Sahu and Sourav Kumar Bhoi, "Comparative Performance Analysis of Multi-Dynamic Time Quantum Round Robin (MDTQRR) Algorithm with Arrival Time", *Indian Journal of Computer Science and Engineering*, IJCSE, ISSN: 0976-5166, Volume 2, Number 2, Page(s): 262-271, Apr-May 2011.
- [9] Brebner G and Diessel O, "Chip-based Reconfigurable Task Management", in *Field-Programmable Logic and Applications (FPL'01)*, Springer Verlag, Berlin, Germany, Page(s): 182–191, 2001.
- [10] Burns A, Tindell K and Wellings A.J. "Fixed priority scheduling with deadline prior to completion", in *proceedings of Sixth Euromicro Workshop on Real-Time systems*, Research Group Department of computer Science University of York, UK. Page(s): 138 – 142, 1994.
- [11] Cai Sufeng and Hugh Anderson, "Queueing Theory", *Operating System, Group Project Report, Group 36*, NUS, National University of Singapore, school of computing, 2007 - 2008.
- [12] David B. Stewart, Donald E. Schmitz, and Pradeep K. Khosla, "The Chimera II Real-Time Operating System for Advanced Sensor-Based Control Applications", *IEEE Transactions on Systems, Man, and Cybernetics*, Volume 22, Number 6, Page(s): 1282-1295, Nov / Dec 1992.
- [13] Dharamendra Chouhan, SM Dilip Kumar and Jerry Antony Ajay, "A MLFQ Scheduling Technique using M/M/c Queues for Grid Computing", *International Journal of Computer Science Issues*, IJCSI, ISSN: 1694-0784, Volume 10, Issue 2, Number 1, Page(s): 357-364, March 2013.
- [14] Dodd R.B, "Coloured Petri Net Modelling of a Generic Avionics Mission Computer" under "Avionics Enabling Research and Development", *Defense Science and Technology Organization, DSTO*, Australia, DSTO-TN-0692, April 2006.
- [15] Dror G. Feitelson, "Notes on Operating Systems", *School of Computer Science and Engineering, The Hebrew University of Jerusalem*, Israel, 2011.
- [16] Garcia P, Compton K, Schulte M, Blem E and Fu W., "An overview of reconfigurable hardware in embedded systems", in *EURASIP Journal on Embedded Systems*, Page(s): 1–19, 2006.
- [17] Gary J. Nutt, "Implementing Processes, Threads, and Resources" and "scheduling" along with "Basic Synchronization Principles", in *text book Operating Systems, Third Edition*, ISBN 0-201-77344-9, Addison-Wesley, 2004.
- [18] Hoganson, Kenneth, "Reducing MLFQ Scheduling Starvation with Feedback and Exponential Averaging", *Consortium for Computing Sciences in Colleges, Southeastern Conference*, Georgia, 2009.
- [19] Jean-François Hermant and Gérard Le Lann, "Fast asynchronous uniform consensus in real-time distributed systems", in *IEEE Transactions on Computers*, Volume 51, Number 8, Page(s): 931–944, 2002.
- [20] krithi Ramamritham and John A. Stankovic, "Scheduling Algorithms and Operating Systems Support for Real-Time Systems", *Proceedings of the IEEE*, Volume 82, Issue 1, Page(s): 55-67, Jan 1994.
- [21] Kruk L, Lehoczyk J. P, Shreve S. E and Yeung S.N, "Earliest deadline first service in heavy traffic acyclic networks", in *Annals of Applied Probability*, Volume 14, Number 3, Page(s): 1306-1352, 2004.
- [22] Lauzac S and Melhem R, "An Improved Rate-Monotonic Admission Control and Its Applications", in *IEEE Transactions on Computers*, Volume 52, Number 3, Page(s): 337-350, 2003.
- [23] Li Lo and Liang-Teh Lee, "An Interactive Oriented Fair Scheduler with Bounded Starvation for Desktop Time-Sharing Systems", *Thesis for Master of Science, Department of Computer Science and Engineering*, Tatung University, January 2008.
- [24] Maricruz Valiente, Gonzalo Genova and Jesus Carretero, "UML 2.0 Notation for Modeling Real Time Task Scheduling", *proceedings in Journal of Object Technology*, Volume 5, Number 4, Page(s): 91-105, May-June 2006.
- [25] Michael B. Jones et al, "CPU Reservations and Time Constraints: Efficient, Predictable Scheduling of Independent Activities", in *Proceedings of the sixteenth ACM Symposium on Operating Systems Principles*, ACM SOSP '97, ISBN:0-89791-916-5, Page(s): 198-211, 1997.
- [26] Micro digital, "Simple Multitasking Executive", in *proceedings of online document SMX RTOS*, [smxinfo](http://smxinfo.com), Embedded Software Outfitters, SMX, Symmetry Innovations, Australia, December 2004.
- [27] Mohammad Reza Effat Parvar, Karim Faez, Mehdi EffatParvar, Mehdi Zarei, Saeed Safari, "An Intelligent MLFQ Scheduling Algorithm (IMLFQ) with Fault Tolerant Mechanism", in *proceedings of Sixth International Conference on Intelligent Systems Design and Applications*, ISDA'06, Volume 3, Page(s): 80-85, Oct 2006.
- [28] Mohammad Reza Effat Parvar, Mehdi Effat Parvar, Abolfazl Toroghi Haghighat, Reza Mahini, Mehdi Zarei, "An Intelligent MLFQ Scheduling Algorithm (IMLFQ)", in *proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications*, PDPTA, Las Vegas, Nevada, USA, Page(s): 1033-1036, June 2006.
- [29] Mohammad Reza Effat Parvar, Akbar Bemana, Mehdi EffatParvar, "IMLFQ Scheduling Algorithm with Combinational Fault Tolerant Method", in *proceedings of International Conference on Enformatika Systems Sciences and Engineering*, Prague, Czech Republic, Jan 2006.
- [30] Mohammad Reza Effat Parvar, Mehdi Effat Parvar, Saeed Safari, "A Starvation Free IMLFQ Scheduling Algorithm Based on Neural Network", *International Journal of Computational Intelligence Research*, ISSN 0973-1873, Volume.4, Number.1, Page(s): 27–36, 2008.
- [31] Panduranga Rao M.V. and Shet K.C, "Study and Development of a New Multi Level Feedback Queue Scheduler for Embedded Processor", *International Journal of Computer Sciences and Engineering Systems*, IJCSES, ISSN 0973-4406. Volume 4, Number 3, Page(s): 203-213, July 2010.
- [32] Panduranga Rao M.V. and Shet K.C, "A Simplified Study of Processor Scheduler for Real Time Operating System", *International Journal of Computational Cognition*, IJCC, ISSN 1542-5908 (online); ISSN 1542-8060 (print). Volume 8, Number 3, Page(s): 5-16, September 2010.
- [33] Panduranga Rao M.V. and Shet K.C, "A Research in Real Time Scheduling Policy for Embedded System Domain", *CLEI Electronic Journal*, ISSN 0717- 5000. Volume 12, Number 2, Paper 4, August 2009.

- [34] Panduranga Rao M.V. and Shet K.C, "Analysis and Refining of Scheduler for Real Time Operating System", *ICFAI University Journal of Computer Sciences*. ISSN: 0973-9904. Volume. 3, Number 4, Page(s): 7-22, October 2009.
- [35] Panduranga Rao M.V. and Shet K.C, "New Approaches to Improve CPU Process Scheduler in the Embedded System Domain", *i-manager's Journal on Future Engineering and Technology, JFET*, ISSN: 0973- 2632. Volume 4, Number 1, Page(s): 72-84, July – September 2009.
- [36] Panduranga Rao M.V. and Shet K.C, "A Simplistic Study of Scheduler for Real Time and Embedded System Domain", *International Journal of Computer Science and Applications, IJCSA*, ISSN: 0974-1003. Volume 2, Number 2, Page(s): 99-108, November 2009.
- [37] Panduranga Rao M.V. and Shet K.C, R. Balakrishna and K. Roopa, "Development of Scheduler for Real Time and Embedded System Domain", *22nd IEEE International Conference on Advanced Information Networking and Applications - Workshops, WAINA '08, 2008*. FINA 2008, Fourth International Symposium on Frontiers in Networking with Applications, Gino-wan, Okinawa, JAPAN. IEEE Computer Society 2008. DOI 10.1109/WAINA.2008.33, Page(s): 1 – 6, 25 to 28th March 2008.
- [38] Panduranga Rao M.V., Shet K.C and K. Roopa. "Efficient and predictable process scheduling", In proceedings of *International Conference on Contemporary Computing IC3- MACMILLAN JOURNAL*, Noida, New Delhi, India, University of FLORIDA (UFL & JIITU) and Jaypee Institute of Information Technology University, page(s): 131-140, August 7-9 2008.
- [39] Panduranga Rao M.V., Shet K.C, K. Roopa and K.J. Sri Prajna, "Implementation of a simple co-routine based scheduler", In proceedings of *Knowledge based computing systems & Frontier Technologies NCKBFT, MIT Manipal, Karnataka, INDIA*, Page(s): 161-164, 19 to 20th Feb 2007.
- [40] Paolo Di Francesco, "Design and implementation of a MLFQ scheduler for the Bacula backup software", *Master thesis in Global Software Engineering*, Universita degli Studi dell'Aquila, Italy, 2011 / 2012.
- [41] Peng Li and Binoy Ravindran, "Fast, Best-Effort Real-Time Scheduling Algorithms", *IEEE Transactions On Computers*, Volume 53, Issue 9, Page(s):: 1159-1175, Sept 2004.
- [42] Raymond Keith Clark, "Scheduling Dependent Real-Time Activities", *PhD Thesis, School of Computer Science, Carnegie Mellon University, CMU-CS-90-155*, August 1990.
- [43] Raymond Keith Clark, Jensen E. D and Rouquette N. F. "Software organization to facilitate dynamic processor scheduling", in *IEEE Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, Page(s): 122b, 2004.
- [44] Sami Khuri, Hsiu-Chin Hsu, "Visualizing the CPU Scheduler and Page Replacement Algorithms", *Proceedings of the 30th ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE '99*, New Orleans, Louisiana, USA, ACM Press, New York, Page(s):: 227-231, March 1999.
- [45] Shahrooz Feizabadi et al., "Utility Accrual Scheduling With Real-Time Java", *On The Move to Meaningful Internet Systems, OTM Workshops, ISBN 978-3-540-39962-9*, Page(s):: 550-563, 2003.
- [46] Shahrooz S. Feizabadi, "Garbage Collection Scheduling for Utility Accrual Real-Time Systems", *PhD Thesis, Virginia Polytechnic Institute & State University, Blacksburg, Virginia, December 7, 2006*.
- [47] Sifat Islam et al., "Concurrency Compliant Embedded System Modeling Methodology", in *2nd Annual IEEE International Systems Conference, SysCon 2008, ISBN: 978-1-4244-2149-7*, Montreal, Canada, Page(s):: 1-8, April 2008 .
- [48] Silberschatz, A., P.B. Galvin and G. Gagne, "Operating Systems Concepts", *7th Edn., John Wiley and Sons, USA., ISBN: 13: 978-0471694663*, Page(s):: 944, 2004.
- [49] Sinnen O and Sousa L, "On Task Scheduling Accuracy: Evaluation Methodology and Results", in *The Journal of Supercomputing*, Volume 27, Number 2, Page(s): 177-194, 2004.
- [50] Sukanya Suranauwarat, "A CPU Scheduling Algorithm Simulator", *Session F2H, 37th ASEE / IEEE Frontiers in Education Conference, Milwaukee, WI*, Page(s):: F2H-19 - F2H-24, October 10-13, 2007.
- [51] Tanenbaum A.S., "Modern Operating Systems", *3rd Edn., Prentice Hall, ISBN: 13: 9780136006633*, Page(s):: 1104, 2008.
- [52] Torrey, L.A., Coleman, J and Miller, B.P., "A Comparison of the Interactivity in the Linux 2.6 Scheduler and an MLFQ Scheduler", *Software Practice and Experience, John Wiley & Sons, Ltd*, Volume 37, Number 4, Page(s):: 347-364, 2007.
- [53] Yaashuwanth C and Ramesh R, A New Scheduling Algorithms for Real Time Tasks, *International Journal of Computer Science and Information Security, IJCSIS, ISSN 1947-5500*, Volume 6, Number 2, Page(s):: 61-66, 2009.
- [54] Zhi Quan and Jong-Moon Chang, "A Statistical Framework for EDF Scheduling", in *Journal on IEEE Communication Letters*, Volume 7, Number 10, Page(s): 493-495, 2003.

V. AUTHOR BIOGRAPHIES



Prof. M.V. Panduranga Rao is a research scholar at National Institute of Technology Karnataka, Mangalore, India. He has completed Master of Technology in computer science and Bachelor of Engineering in electronics and communication.

His research interests are in the field of Real-Time and Embedded Systems on Linux platform. He has published various research papers in journal and conferences across India, Also in the IEEE international conference in Okinawa, Japan. He has authored two reference books on Linux Internals. He is the Life member of Indian Society for Technical Education and IAENG.

His webpage can be found via
<http://www.pandurangarao.i8.com/> .



Dr. K.C. Shet obtained his PhD degree from Indian Institute of Technology, Bombay, Mumbai, India, in 1989. He has been working as a Professor in the Department of Computer Engineering, National Institute of Technology, Surathkal, Karnataka, India, since 1980.

He has published over 200 papers in the area of Electronics, Communication, & computers. He is a member of Computer Society of India, Mumbai, India, and Indian Society for Technical Education, New Delhi, India.

His webpage can be found via
<http://www.nitk.ac.in/~kcschet/index.html> .