Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 7, July 2014, pg.807 – 816

RESEARCH ARTICLE

Task States, Triggers and Timeline of New Multi-Level Feedback Queue Scheduler

M.V. Panduranga Rao[1], K.C. Shet[2]

[1]Research Scholar, NITK, Surathkal, Mangalore, India

[2]Professor, NITK, Surathkal, Mangalore, India

[1] raomvp@yahoo.com; [2] kcshet@rediffmail.com

***Abstract* — We are articulating the task states of New Multi Level Feedback Queue [NMLFQ] Scheduler in this research paper. The contingent of task transitions with triggers which leads to a change of state is depicted. In real time scenario, the literal time line of real time process, with time instants and intervals are elucidated.**

***Keywords* — NMLFQ, scheduler, states, triggers, process, time line, queue, deadline, preemption, priority, Multi Level, transition diagram and execution time**

I. The State Transition Diagram of Process

All implementations in NMLFQ are achieved using tasks or threads. In the transition each task can have one of the seven different states: new, ready, running, wait, blocked, zombie and succeeded state [2][5][10][14]. When the tasks are created, its state is said to be in new state. Tasks which are obtainable for running are said to be in ready state. Existing job is started to execute in running state [1][32][43][49]. Waiting tasks are the one waiting for a resource to be accessible. If a task is sleeping for certain predefined amount of time then it is said to be in blocked state. The zombie process is one, which is unable to fulfill its job, as it is killed due to uncertain event. The fairly completed task reaches the succeeded state of process [7][13][21][54].

The state transition diagram of process is described with the possible states in fig. 1. The transition of process must be accounted for all the states of a real time application [27][33][44][51].

The literal pseudo code of NMLFQ task state is provided as below.

```
METHOD BEGIN void processing()
        MONITOR BEGIN
                int i=0;
                WHILE BEGIN (pQueue.isEmpty())
                        ExecutingProcess(pQueue.get(i++),"wait");
                WHILE END
        MONITOR END
        CATCH BEGIN(Exception ex)
```

```
                    ex.printStackTrace();
          CATCH END
METHOD END

METHOD BEGIN void ExecutingProcess(Process process,String state)
          //Actual Execution happens
          IF BEGIN (state.equalsIgnoreCase("wait") || state.equalsIgnoreCase("Ready") ||

                                        state.equalsIgnoreCase("DuringEvent"))
                    System.out.println("Execution started...");
          IF END
          ELSE BEGIN
                    System.out.println("Cannot execute process as its not in proper state  to
                                        execute");
          ELSE END
METHOD END
```

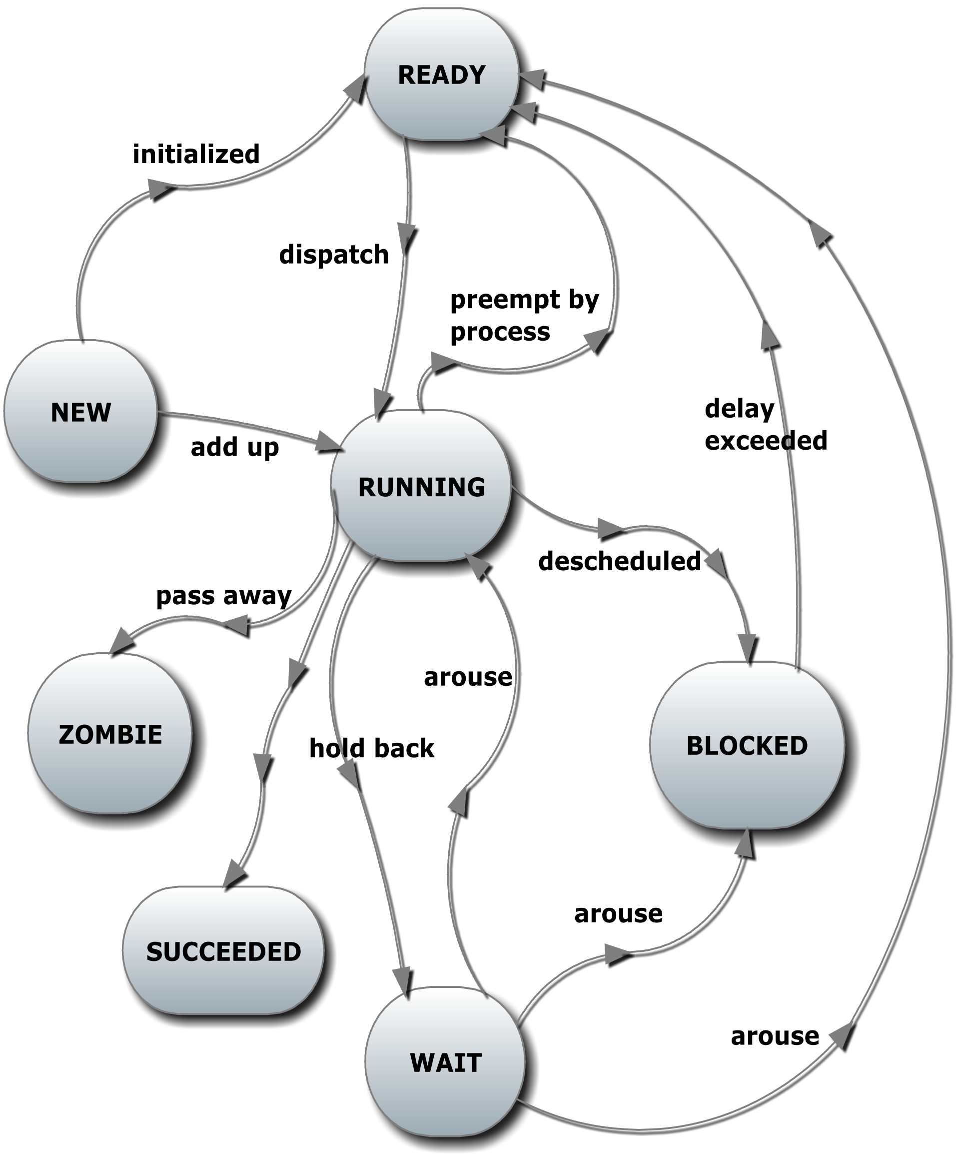

Fig. 1 The state transition diagram of process, representing the execution of an entire task, to be invoked through several states at dissimilar stages.

The object oriented programming concept is adopted in the developmental approach of New Multi Level Feedback Queue Scheduler [28][31][37][45][52]. The object oriented way of pseudo-code, which is almost similar to actual code, is explained in brief as follows.

```
public class TaskStates {
        Vector<Process> readyQueue =new Vector();
        Hashtable<Integer, Process> pQueue = new Hashtable<Integer, Process>();
        public Process p=null;
        public Process removedProcess=null;
        public TaskStates(){
                // initially loading Queue with few number of processes.
                try{
                        for(int i=1;i<=pQueue.size();i++){
                                // Creating process
                                p=Runtime.getRuntime().exec("ps -ef");
                                insertIntoPQueue(i,p);
                        }
                }catch(Exception ex){
                        ex.printStackTrace();
                }
        }

public TaskStates(int i){
                try{
                        // removing the requested number of processes from queue.
                        if(i<10){
                                for(int j=0;j<i;j++){
                                        removedProcess=removeAtEnd(j);
                                        if(removedProcess != null){
                                          System.out.println("process is removed from queue");
                                        }else{
                                          System.out.println("process removed from the queue is null");
                                        }
                                }
                        }else{
                                addAtFront(i, Runtime.getRuntime().exec("ps -ef"));
                        }
                }catch(Exception ex){
                        ex.printStackTrace();
                }
        }

  /* The following method is used to add the process to the ready-Queue based on Task-States.
    * It will check the Task-States and moves the other processes in ready-Queue accordingly.  */

public void addAtFront(int priority, Process temp){
                Enumeration enumitr = pQueue.keys();
                while(enumitr.hasMoreElements()){
                        if(!(priority <(Integer)enumitr.nextElement())){
                                pQueue.put(priority, temp);
                                readyQueue.add(temp);
                        }
                }
        }

  /* The following method, will insert some processes initially into queue and also for the ready-Queue */

public void insertIntoPQueue(int priority, Process process){
                // adding process to the Queue and also into ready-Queue
```

```
            if(process != null){
                pQueue.put(priority, process);
                readyQueue.addAll(pQueue.values());
                System.out.println("process is added into Ready Queue");
            }else{
                System.out.println("process is null");
            }
    }
```

/* The following method, removes the process from the queue. */

```
public Process removeAtEnd(int index){
            // removing the process from the index specified from both ready-Queue and pQueue
            readyQueue.remove(index);
            return(pQueue.remove(index));

    }
```

/* The following method, will allocate the CPU based on the priority of the processes [4][42][46][50]. */

```
public void processing(){
            try{
            int i=0;
            while(pQueue.isEmpty()){
                ExecutingProcess(pQueue.get(i++),"wait");
            }
            }catch(Exception ex){
                ex.printStackTrace();
            }
    }
```

/* Now the actual execution will start. */

```
public void ExecutingProcess(Process process,String state){
            //Actual Execution happens
        if(state.equalsIgnoreCase("wait")        ||        state.equalsIgnoreCase("Ready")        ||
state.equalsIgnoreCase("DuringEvent"))
             System.out.println("Execution started...");
          else
             System.out.println("Cannot execute process as it's not in proper state  to execute");
    }
```

/* Here, we are creating pQueue as a placeholder for processes which are ready to execute [9][12][17][22][48]. However, as it is Priority-Queue based on task-states, 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 based on priority. Correspondingly the task states are restored [16][18][20][24][41]. */

```
public static void main(String[] args) {
            // TODO Auto-generated method stub
            // creation of Queue
            new TaskStates();
            //Removing processes from Queue
            new TaskStates(2);
            new TaskStates(110);
            try {
                new TaskStates().addAtFront(2,Runtime.getRuntime().exec("ps -ef"));
                new TaskStates().processing();
            } catch (IOException e) {
                // TODO Auto-generated catch block
```

```
                        e.printStackTrace();
                }
        }
}
```

II. Transitions with Triggers which Leads to a Change of State

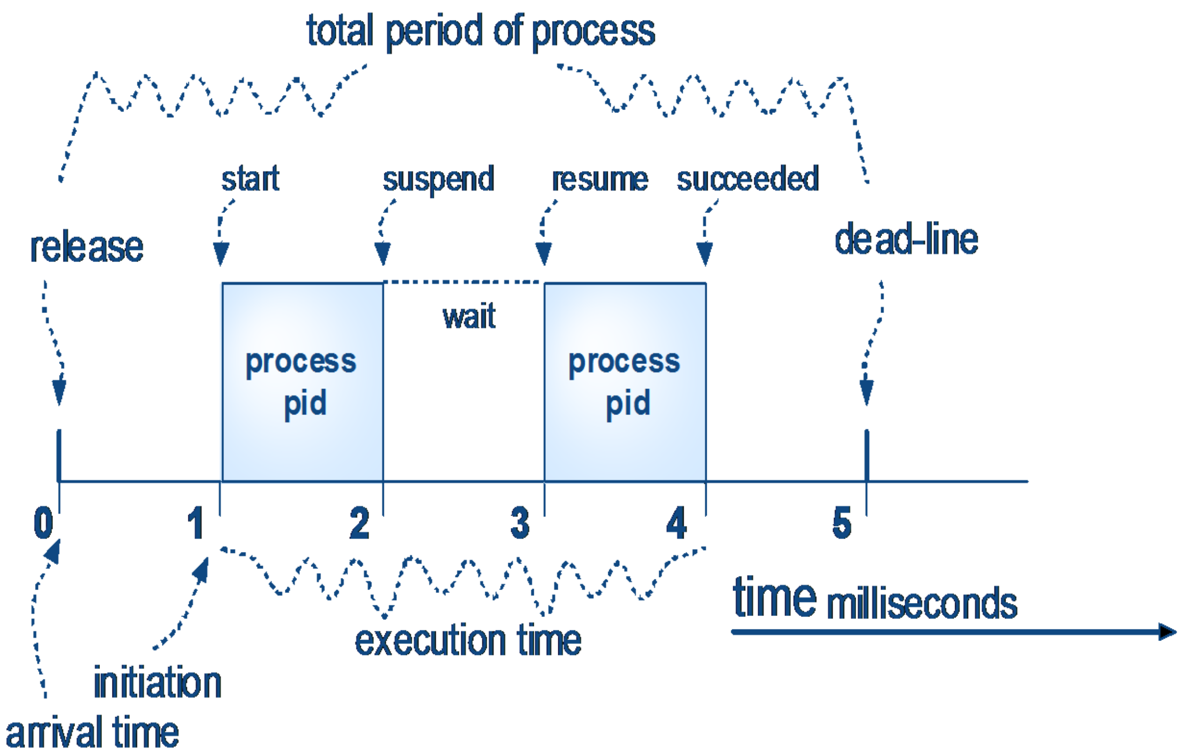

Fig. 2 Task comprising of total period and execution time of a process. The state transitions along with the triggers for each of the state are represented.

The total period of process corresponds to, utilization time of CPU within a fixed amount of time. Sometimes it gets succeeded within deadline, consuming less execution time [6][11][18][23][40]. The earlier completion of task guarantees, to yield the output precisely, at the expected response time. Fig. 2 shows, the task consisting of total period comprising of execution and deadline time. Process arrives at t=0, this is also called the release time of process. The process starts waiting in the ready queue [36][39][53]. At t=1 millisecond, process gets admitted. The process elected from ready pool of processes. The process with its unique process identification number, [pid] starts to run at t=1 msec. On the occurrence of any event, such as Input/ Output, [I/O] process is suspended to wait or blocked in anticipation of free resource. It sometimes sleeps to extend a delay time. The process resumes at t=3 msecs and succeeds at t=4 msecs, before the fixed deadline of process at t=5 msecs. The illustrations of start, suspend, wait, resume and succeeded are designated in fig. 2. The states of process for the execution of whole process, exploring through every state is shown in fig. 1.

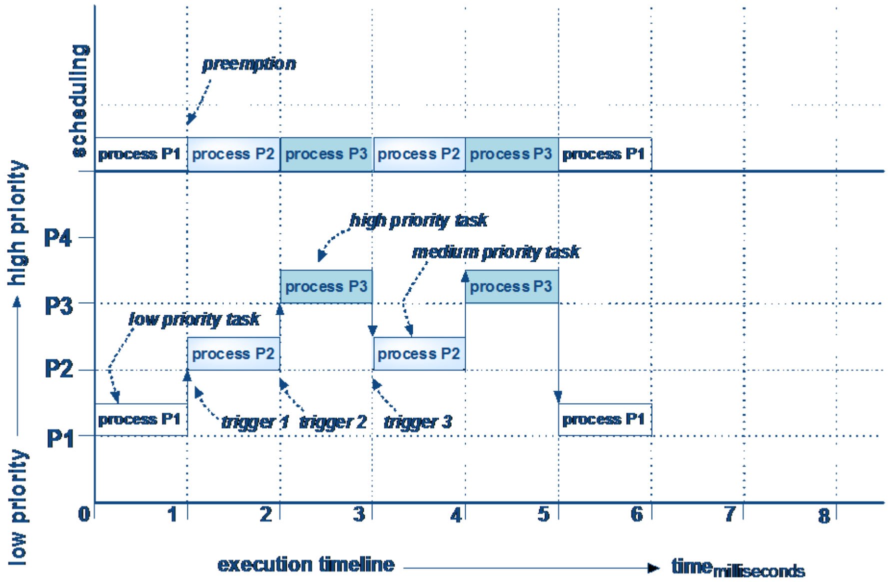

Fig. 3 Preemption of a low priority process by the high priority process occurred at each trigger in a real time kernel, depending on the decision of scheduler.

Preemption of a low priority process by the high priority process occurred at each trigger in a real time kernel, depending on the decision of scheduler is shown in fig. 3. Initially process P1 is started [5][8][26][30][35]. It is suspended by kernel a t=1 msec in order to run medium priority process. Now the suspended process P1 waits till t=5 msecs. The medium priority process continues execution till t=2 msecs. The high priority process P3 preempts the medium priority process now to lock a processor peripheral [3][29][34][38][54]. It finds the peripheral is already locked by another process. Henceforth it suspends itself to give access for medium priority process to run at t=3 msecs. The process P2 runs and unlocks the peripheral at t=4 msecs. The process P3 is started again by kernel and completes the task at t=5 msecs as the peripheral is available.

III. TIME LINE OF REAL TIME PROCESS WITH TIME INSTANTS AND INTERVALS

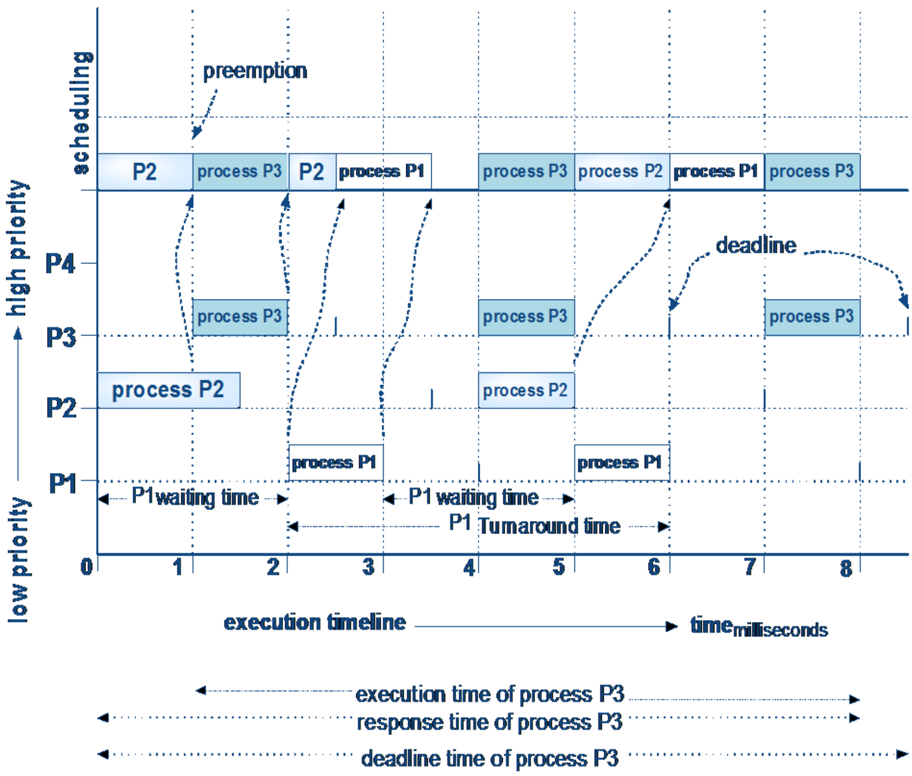

Fig. 4 Representation of time line of real time process.

Fig. 4, represents the timeline of real time process depicting the deadline, response and execution time of process [15][19][25][47]. Several processes arrive and get CPU time, in accordance to the decision of scheduler. The top row shows the schedule and completion instance of processes. The processes P1, P2 and P3, arrives with increasing priority. The high priority process P3 pre-empts at t=1msec. The process P2 waits for a duration from t=1 to t=2 milliseconds. It completes immediately after starting at t=2 msecs. The arrow marks often shows several pre-emption points. A few deadline points for each process are also depicted in fig. 4.

IV. CONCLUSIONS

In this research paper, we have discussed the issues related to task states, triggers and timeline of new multi-level feedback queue scheduler. The process will be executed through dissimilar states. In the transition each task can have one of the seven different states: new, ready, running, wait, blocked, zombie and succeeded state. pseudo code of NMLFQ with several task states is apprised. The object oriented way of pseudo-code is explained bearing respective methods. Comments are provided for the welfare of realization.

HashTable for storing both priority and processes are utilized, effectively for Priority-Queue depending on task-states. The state transitions along with the triggers for each of the state are represented. Preemption of a low priority process by the high priority process occurred at each trigger in a real time kernel, depending on the decision of scheduler is also articulated. Finally, the timeline of real time process depicting the deadline, response and execution time of process is conversed.

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, Lehoczky 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,* 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.

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/~kcshet/index.html .