An Analytical Survey of Real Time System Scheduling Techniques

Ayeni, J. A.[1], Odion A. E. [2], Ogbormor-Odikayor I.F[3]

[1]Department of Physical Sciences, Ajayi Crowther University,Oyo, Oyo State, Nigeria

[2, 3]Department of Computer Science and Mathematics, Benson Idahosa University, Benin City, Edo State, Nigeria.

Abstract: ***Real-time systems are systems that are time dependent. The time dependency nature of these systems compared to conventional operating systems such as the Unix time sharing multiuser and multitasking systems and its variants, single user and multitasking Microsoft windows systems and single user and single task palm and handheld device systems has given rise to several techniques in the management of computer resources. Scheduling is that all-important functionality of the operating system that ensures Operating system's and computer's resources are managed in such a way as to ensure none of the competing processes is starved of such resources at any given point in time. In real time system, scheduling is effected using certain criteria that ensure processes complete their various tasks at a predetermined stipulated time of completion. The scheduling techniques depend on the two basic types of the existing real time systems; Hard and Soft Real-Time System. The pre-determined time of completion of tasks must be met in Hard Real Time Systems otherwise the resultant effect can be disastrous. Tasks in Real Systems that fail to complete within the pre-determined completion time are tolerable but not often desirable. In these two known types of Real-Time System, pre-emption and multitasking techniques are often used. In this study, a critical analysis of these scheduling techniques and their performance (strengths and weaknesses) in existing systems will be carried out and highlighted. The various scheduling techniques and their associated parameters are examined and analysed. In addition, a survey of known scheduling techniques in real time systems and approaches used by the different techniques are presented.***

Keywords: scheduler, pre-emptive, tasks, hard, soft, critical

1. Introduction

A real-time system is one whose correctness is based on both the correctness of the outputs and their timeliness [1], as a computation output of a real time system after the allocated time of execution (deadline) is of no value and could be catastrophic. Scheduling could be defined as the process of assigning computer and operating systems' resources to a set of tasks or processes on a given system in accordance to predetermined rules on a given system. In the conventional operating systems environment where tasks are not being subjected to a pre-determined (deadlines) completion time, scheduling is less complex. In real-time systems, scheduling is the most important task and the selection of an appropriate task scheduling algorithm is central to its proper functioning [2]. The selection of an appropriate scheduling algorithm in meeting the time constraints of task is the basic technique employed by a real-time system. The objective of a real-time task scheduler is to guarantee the deadline of tasks in the system as much as possible. Mostly all the real-time systems in existence use pre-emption and multitasking [3].Generally, tasks in real-time systems are classified as real-time tasks and possess certain degree of urgency and as such mission-criticality. The urgency and mission-critical nature of real-time tasks (applications) depend on the results of their executions and could be classified into the following two classes; *hard and soft*. A hard real-time task must meet its deadline to avoid disaster or damage while a soft real-time task equally has an associated deadline that is not mandatory but essentially could be allowed to complete for a desirable result even if the deadline is missed. A real time system is expected to change its state in real time even after the controlling processor has stopped its execution [4]. A deadline is defined as the bound in which real time applications are needed to respond to the change of event in their environment.

2. Related Works

The recent advances in embedded systems the past decades have resulted in several researches in the field of real-time systems and real-time operating systems. An embedded system has been described as a typical example of real-time systems and a system within a larger system. An embedded system is a combination of hardware and software and perhaps a mechanical part to perform certain function [5]. Such system sits are being controlled by Real-Time Operating System. A Real-Time Operating system (RTOS) and the Real-Time Scheduler are the basic components of an embedded system which offers an easy design and expansion of real-time applications. Most embedded systems also have real-time requirements demanding the use of Real-Time Operating systems capable of meeting the embedded system requirements [5].

Khera and Kakkar in [4], carried out a comparative study of scheduling algorithms for real time environment. In the study, the various scheduling techniques based on different parameters were classified and the techniques used for scheduling in real time environment were analysed and comparison between different techniques also presented. Kaladevi and Sathiyabama [3] compared scheduling algorithms for real-time tasks and stated that Real-time systems have well defined, fixed time constraints i.e., processing must be completed within the defined constraints otherwise the system will fail. In their paper, real-time scheduling techniques were classified into two categories: Static and Dynamic. Static algorithms assign priorities at design time and all assigned priorities remain constant for the lifetime of a task. Dynamic algorithms assign priorities at runtime, based on execution parameters of tasks which could be either with static priority or dynamic priority. A research into priority based round robin scheduling algorithm for real time systems was carried out by Rajput and Gupta [6] with the main objective of developing a new approach for round robin CPU scheduling algorithm which improves the performance of CPU in Real Time Operating System. They also presented a comparative analysis of the proposed algorithm with existing round robin scheduling algorithms on the basis of varying time quantum, average waiting time, average turnaround time and number of context switches and concluded that the proposed algorithm is more efficient than the simple round robin in the context of the stated metrics [6]. Larsson and .Hargglund [7] researched into two RTOS, namely RTLinux and Chimera. The comparative study of scheduling techniques of these two systems with emphasis on their scheduling methods was carried out. It was concluded that the two real time operating systems "are very different in many regards, mainly flexibility; the modularity and customizability of RTLinux, together with its ability to cooperate with regular Linux, makes it a generalist system that can be used in almost any situation whereas Chimera is more limited but probably also more efficient" [7]. Nandanwar and U. Shrawanka [8] proposed an adaptive algorithm for scheduling of hard real time system with single processor and pre-emptive task sets and introduced the concept of EDF, ACO and FPZL and combined this approach to get the algorithm. The advantage of the proposed algorithm is it will automatically switch between the algorithm and overcome the limitation of existing algorithms. The adaptive algorithm is very useful when future workload of the system is unpredictable [8]. In all the above cited works, the researchers laid emphasis on the techniques of the various existing real time schedulers. There are some advantages of these techniques and performance gain of some real-time applications that make use of them and depend on the urgency and mission critical nature of the applications.

3. Real Time Systems

Real-Time systems have been defined in several literatures in different ways and the core objective of these definitions present a real-time system as a system with timeliness with correctness of logical results in/of the execution or its tasks; have deadlines for completion. According to Jane [9], real–time systems are those systems in which the correctness of the system does not depend only on the logical results of computations but also on the time at which the results are produced. Laplante in [10] defined a real-time system as one whose correctness involves both the logical correctness of the outputs and their timeliness. Barr[11] also defined a real-time system as computer system that has timing constraints and is partly specified in terms of its ability to make certain calculations or decisions in a timely manner. There are two basic types of real-time system and its classification depends on their timeliness and capacity of producing outputs before their deadlines; **hard** and **soft** real-time systems. In hard real-time systems, deadline must be met otherwise a system failure that may lead to system degradation or catastrophe. On the other hand, a soft real-time system has a non-mandatory associated deadline that could be missed and such tasks are allowed to complete as outputs could still be useful. Tasks in real-time systems could be *aperiodic* or *periodic*. *Aperiodic* tasks are those that have time constraints (start/Stop) or both (start and stop). *Periodic* tasks are stated to execute within a time period (i.e. every *T secs*). The main operating system for real-time systems (such as embedded systems) is the Real-Time Operating System (RTOS). An RTOS differs from common Operating Systems (i.e. *Single-user and single task*, *single-user and Multitasking and Multi-user*), in that the user when using the former has the ability to directly access the microprocessor and peripherals and such ability of RTOS helps to meet deadlines [12]. The *Kernel* is the core component of all operating systems and provides task scheduling, dispatching and inter-process communication. However, the *Kernel* perform differently in these operating systems as different techniques and criteria are used in the selection of tasks to run, dispatch and perform error recovery functions. Desirable features of a real-time operating system are [12]: ability to schedule tasks and meet deadlines, ease of incorporating external hardware, recovery from errors, fast switching among tasks, small size and small overheads. The short-term scheduler is the core of a real-time system, fairness and minimising average response time are not the only paramount characteristics of real-time systems but equally important is the fact that all hard real-time tasks complete (or start) by their deadline and that as many as possible soft real-time tasks also complete (or start) by their deadlines [13].

3.1 Real-Time Scheduling

There are two approaches to scheduling techniques in real-time systems; Static and dynamic algorithm approaches. The static approach assigns priority to tasks at design time and requires pre-knowledge of the characteristics of the tasks while the dynamic algorithm approach assigns priority at run-time with a greater run-time cost compared with the static approach [14]. The appropriateness or suitability of the algorithm for a real time system is a matter of choice for the developers and equally depends on the type of real-time system. Certainly in safety critical systems it is reasonable to argue that no event should be unpredicted and that schedulability should be guaranteed before execution and this implies the use of a static scheduling algorithm while dynamic approaches are particularly appropriate to soft systems; could form part of an error recovery procedure for missed hard deadlines and could be used if the application's requirements fail to provide a worst case upper limit (for example the number of planes in an air traffic control area) [15]. Mohammadi and Akl [16] classified scheduling techniques in uniprocessor real-time systems into two subsets; *offline* scheduling algorithms *and online* scheduling algorithms. A scheduler is static and *offline* if all scheduling decisions are made prior to the running of the system. A table is generated that contains all the scheduling decisions for use during run-time and relies completely upon *a priori* knowledge of process behaviour[15]. *Off-line scheduling algorithms* (Pre-run-time scheduling) generate scheduling information prior to system execution and the information is then utilized by the system during runtime [16]. Fohler, Lenvall and Buttazzo [17] provided examples of an Earliest Deadline First (EDF) and the off-line algorithm in their research into real-time scheduling techniques. *Online* *scheduling algorithm* makes scheduling decisions at run-time state of the system and can be either static or dynamic which is based on both process characteristics and the current state of the system in runtime [16].

3.1.1 Static Table-Driven Approach

A Static Table Driven scheduling technique takes all scheduling decisions before the running of the system, generating a table for use during run-time with a total reliance on its pre-knowledge of the process behaviour. A typical pattern in scheduling algorithms is to determine whether a schedule is produced by a schedulability analysis that may result into tasks being dispatched at run-time. A proven example of an offline scheduling that produces a schedule is the *table-driven* approach. The priority-based approach is also an example of offline scheduling where no explicit schedule is constructed; at run time, tasks are executed in a highest-priority-first basis and are much more flexible and accommodating than table-driven approaches [18].The Static Table-Driven scheduler is often used to implement periodic tasks with requirements such as; periodic arrival time, finishing deadline, execution time and the priority for each task. The scheduler attempts to develop a schedule that enables it to meet the requirements of all the periodic tasks [13]. Although a table-driven scheduler is inflexible as any change in a periodic task's requirements would necessitate the entire schedule process to be redone, it is a predictable approach that guarantees system performance. A typical example of such techniques exist for tasks that have simple characteristics and the Earliest-Deadline-First (EDF) or the Shortest-Period-First (SPF) technique are usually used to construct such static tables with required parameters known a priori.

3.1.2 Static Priority- Preemptive Approach

As earlier stated in section 3.1, scheduling decisions by online schedulers are taken at run-time and can either be static or dynamic and are determined not only by the characteristics of the tasks but also by the current state of the system. The preemptive scheduler can arbitrarily suspend the execution of a currently running process and restart it after the completion of the preempting process as a result of the latter's higher priority without altering the behaviour of the preempted process. Preemption normally occurs when a higher priority process becomes runnable and its major effects are increasing the elapsed time of the preempted process and that such a process may be suspended involuntarily [15]. Unlike the conventional non-real time systems that implement a priority driven preemptive scheduling method of assigning priority to processes based on whether the process is processor bound or Input/Output bound, assignment of priority in a real-time system is related to the time constraints associated with each task.Rate Monotonic scheduling Algorithm is an example of a priority driven algorithm with static priority assignment, in the sense that the priorities of all requests are known before their arrival, the priorities for each task being the same and known a-priori (they are determined only by the period of task) [19].In the RMA, each task is assigned a (unique) priority based on its period (completion time); the shorter the period, the higher the priority. The RMA assignment is optimal under preemptive priority-based scheduling. An *optimal scheduler* is able to produce a feasible schedule for all feasible process sets conforming to a given precondition [15].Liu and Layland [21] also analysed Earliest-Deadline-First (EDF), a dynamic priority-assignment algorithm as an example of Static Preemptive Approach for real-time schedulers: the closer a task's deadline, the higher its priority. This again is an intuitive priority assignment policy [21]. It is worthy to state that when a task has completion deadlines, a preemptive strategy will be most appropriate. The real-time scheduling algorithm is represented schematically in figure 1.0 below. The real-time scheduling algorithm is represented schematically in figure 1.0 below.

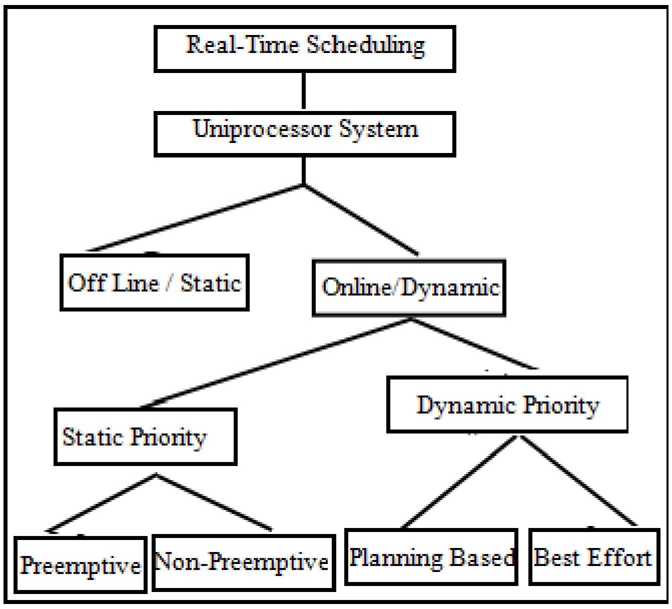

Figure 1: Schematic Representation of Scheduling Algorithm (Adapted from [17])

3.1.3 Static Priority- Non-Preemptive Approach

Unlike the preemptive scheduling approach, a non-preemptive scheduler allows tasks to complete execution without being suspended once execution is started. Tasks that have associated deadlines fit best into the non-preemptive scheduling approach as the responsibility of self-blocking will be that of the tasks after completing the mandatory or critical portion of its execution, allowing other real-time starting deadlines to be satisfied [13].Jeffay [20] suggested the treatment of a non-preemptable process scheme in which non-preemptable processes are shown to be scheduled by the earliest deadline heuristic if they can be scheduled by any other non-preemptive scheduler. The Earliest Deadline First (EDF) algorithm is the most widely used scheduling algorithm for real-time systems, optimal for a set of preemptive tasks (be they periodic, aperiodic, or sporadic), and will always find a schedule if a schedule is possible [21].It has been shown that the non-preemptive scheduling is more efficient than the preemptive approach especially for soft real-time applications and multithreaded system applications. The non-preemptive approach equally reduces the much needed switching overhead among processes/ threads. The Earliest Deadline First (EDF) approach is optimal for sporadic non-preemptive tasks, but EDF may not find an optimal schedule for periodic and aperiodic non-preemptive tasks; it has been shown that scheduling periodic and aperiodic non-preemptive tasks is NP-hard [22]. For a system not overloaded, EDF (non-preemptive approach) has been shown to produce optimal schedules for periodic and non-periodic (aperiodic) tasks. With an overloaded system, it has been established that EDF approach leads to dramatically poor performance [23]. Although the application of EDF to non-preemptive tasks has not been given pre-eminence by researchers, it is fast gaining research interests.

3.1.4 Dynamic Priority Planning Based Approach

The dynamic planning-based approaches provide the flexibility of dynamic approaches with some of the predictability of approaches that check for feasibility [24]. In this approach, the execution of a newly arrived task does not commence immediately, rather an attempt is made to create a schedule that contains the previously guaranteed task and the new task before execution commences. If the attempt in creating a schedule fails and made sufficiently ahead of the deadline, then there will be ample time alternative actions for such task and the revision of the schedule to accommodate the new task. In effect, an arriving task is accepted for execution if it is feasible to meet its time constraints and the result of the feasibility analysis is a schedule or plan that is used to decide when to dispatch such task [13].The Dynamic planning-based scheduling approach dynamically performs feasibility checks for tasks when selected for execution. Atask is guaranteed by constructing a plan for task execution whereby all guaranteed tasks meet their timing constraints, subject to a set of assumptions if these assumptions hold, once a task is guaranteed it *will* meet its timing requirements [24]. In general there are three steps involved in dynamic planning approach and are stated as follows; feasibility or schedulability analysis, schedule creation and dispatching. These steps could be implemented separated or jointly depending on the requirements of the system. In some cases there are no distinguishable delineation between the three steps. Feasibility or schedulability analysis determines whether the timing requirements (constraints) of a set of jobs at run-time can be satisfied usually under a given set of resource requirements and precedence constraints [25] Schedule construction is the process of ordering the jobs to be executed and storing this in a form that can be used by the dispatching step, and this approach is a direct consequence of the feasibility analysis [25].

3.1.5 Dynamic Best Effort Approach

In this approach, the scheduler does not carry out any feasibility or schedulability check on new tasks that arrive in the system; by implication all tasks are admitted into the system upon their arrival and the scheduler tries its best to ensure execution and therefore, there's no guarantee of task's execution. The dynamic best effort approach uses the deadlines associated with tasks to set their priorities and a task could be preempted anytime during execution and therefore, it is until the deadline arrives or the task finishes execution one can know whether actually a timing constraint had been met or not [26].The implementation technique of the dynamic best effort approach looks similar to those of priority based schedulers in non-real time systems except in the method used in assigning priorities. In dynamic best effort approaches two queues are maintained: a *ready* queue and a *wait* queue [27]. While the ready queue is sorted in priority order, tasks waiting for non-processor resources are placed in the wait queue [27]. After the completion of a task, the ready queue is re-adjusted (re-computation) based on the arrival of another task from the wait queue depending on its priority level. A currently executing task could be preempted if the task from the wait queue has a higher priority than the currently running task. Thus, the task priorities must be re-computed each time a new task enters the ready queue and the ready queue must be re-ordered based on the newly computed priorities [27].

4. Conclusion

In this paper, we present the various existing approaches to scheduling in real-time systems and real-time applications. The static table-driven approach, static priority-driven preemptive approach, static priority non-preemptive approach, dynamic planning based approach and the dynamic best-effort approach are presented. Examples of existing schedulers based on these approaches have been provided and elucidated. Generally, real-time systems have found tremendous usage in systems such as command and control systems, industrial process control systems, Space Shuttle Avionics, flight control systems and some home based systems such as micro waves and robotics. Knowledge of these systems' behaviour in real-time are presumed to be available a priori and as such their inflexibility and high cost because they are based on the static technique in their development. It is proposed that future systems should take into cognizance the importance of developing systems that are dynamically based, more predictable, flexible and more independent.

References

[1] F.,Schlindwein,(2002). EG7017 – Real-time DSP. [Online], Available at: <http://www.le.ac.uk/eg/fss1/real%20time.htm>

[2] Rajib Mall (2009)Real-Time Systems: Theory and Practice, Pearson Education, India. May 1, 2009.

[3] M.Kaladevi, and .S.Sathiyabama (2010). A Comparative Study of Scheduling Algorithms for Real Time Task. International Journal of Advances in Science and Technology, Vol. 1, No. 4, 2010

[4] I., Khera and A. Kakkar. Comparative Study of Scheduling Algorithms for Real Time Environment. *International Journal of Computer Applications* 44(2):5-8, April 2012. Published by Foundation of Computer Science, New York, USA.

[5] P.M. Sagar and V. Agarval, (2002). Embedded Operating Systems for Real-Time Applications, M.Tech Credit Seminar report, Electronic Systems Group, EE IIT, Bombay.

[6] I.S.Rajput and D. Gupta (2012), A priority based round robin scheduling algorithm for real time systems. International Journal of Innovations in Engineering and Technology, Vol 1 Issue 3, 2 Oct. 2012.

[7] J. Larsson and J. Hargglund (nd), RTLinux and Chimera: Comparative Study in Scheduling Techniques, Technical Report – at LinkopingsUniversitet

[8] J. Nandanwar and U. Shrawankar, (2012)An Adaptive Real Time Task Scheduler International Journal of Computer Science Issues, Vol. 9, Issue 6, No 1,

November 2012 ISSN (Online): 1694-0814 www.IJCSI.org.

[9] J. W.S. Liu,(2001);Real-Time Systems, Pearson Education, India, pp. 121 & 26, 2001.

[10] P.A.Laplante (1997), Real-Time Systems Design and Analysis: An Engineer's Handbook. Second Edition. IEEE Press. 1997.

[11] M. Barr. (1999). Programming Embedded Systems in C and C++ ,O'Reilly and Associates, Inc. U.S.A

[12] Yan Meng, (nd), A Survey of Real-time Operating Systems, Technical Report. [Online], Available at: <http://www.ece.stevens-tech.edu/~ymeng/courses/CPE555/papers/rtos_paper.pdf >

[13] W. Stallings (2008) Operating Systems: Internals and Design Principles, Prentice Hall, 2008, pp 453-459

[14] H. Kopetz (2011), Real-Time Systems: Design Principles for Distributed Embedded Applications, Springer Publishing. Coy. 2011. Pp. 240.

[15] N. Audsley and A. Burns (1990) Real-Time System Scheduling. [Online]. Available at: <*beru.univ-brest.fr/~singhoff/cheddar/publications/audsley95.pdf*> , and on the ESPRIT BRA Project (3092), Predicatably Dependable Computer Systems, Volume 2, Chapter 2, Part II.

[16] A. Mohammadi and S.G. Akl (2005), Scheduling Algorithms for Real-Time Systems, [Online] , Available at :<http://beru.univ-brest.fr/~singhoff/cheddar/publications/audsley95.pdf> , Technical Report No. 2005-499, School of Computing, Queen's University, Kingston, Ontario.

[17] G. Fohler, T. Lenvall and L. G. Buttazzo (2001), Improved Handling of Soft Aperiodic Tasks in Offline Scheduled Real-Time Systems using Total Bandwidth Server, [Online], Available at: <http://www.mrtc.mdh.se/publications/0316.pdf>, Accessed on 14 October, 2013, Published in Emerging Technologies and Factory Automation, 2001. Proceedings: 2001 8th IEEE International Conference on 15-18 Oct. 2001

[18] E.,Michta (2005). 174: Scheduling Systems. University of Zielona Gora, Zielona Gora, Poland. Reproduced from the Handbook of Measuring System Design. John Wiley & Sons, Ltd, 2005. Available at: <http//www.wiley.com/legacy/wileychi/hbmsd/pdfs/mm823.pdf >

[19] D. Zmaranda, G. Gabor, D.E. Popescu, C. Vancea and F. Vancea (2011). Using Fixed Priority Pre-emptive Scheduling in Real-Time Systems, Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844, Vol. VI (2011), No. 1 (March), pp. 187-195.

[20] K. Jeffay (1988), 'On Optimal, Non-Preemptive Scheduling of Periodic Tasks, Technical Report 88-10-03, University of Washington, Department of Computer Science (October 1988).

[21] C. L. Liu and J. W. Layland, (1973) "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", [Online]. Journal of the ACM, Vol. 20, No. 1, pp. 46-61.Available at: <http://www.csie.ntu.edu.tw/~ktw/rts/ch-short-course-uni.pdf>

[22] Wenming Li, Krishna Kavi and Robert Akl (nd) An Efficient Non-Preemptive Real-Time Scheduling, [Online]. Available at: http://www.cse.unt.edu/~rakl/LKA05.pdf , Department of Computer Science and Engineering, University of North Texas, U.S.A.

[23] C. D. Locke (1986), Best-effort Decision Making for Real-Time Scheduling, CMU-CS-86-134 (PhD Thesis), Computer Science Department, Carnegie-Mellon University, 1986.

[24] K., Ramamritham and J., Stankovic (1994), Scheduling Algorithms and Operating Systems Support for Real-Time Systems, [Online], Proceedings of the IEEE, vol. ***82.*** No. I, January 1994 *5*..Available at: http://www.dca.ufrn.br/~affonso/DCA_STR/aulas/stankovic2.pdf

[25] John, A. Stankovic (1998) .Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms, Springer Publisher. Pg. 98 – 100

[26] Robert Oshana (2006). .DSP Software Development Techniques for Embedded and Real-Time Systems, Newnes Publishers, Jan 9, 2006, pg. 299

[27] C. Siva Ram Murthy and G. McNamara (2001). Resource Management in Real-time Systems and Networks, MIT Press. Pg. 53-55

Author Profile

Ayeni Joshua received the BSc. and MSc degrees in Computer Science from the Universite de Paris VIII at St. Denis, (Paris suburb) in 1987. He worked briefly as a programmer, then later system analyst and finally as Chief Project Engineer with France Organisation and Commercial. He has just concluded his Ph.D programme and currently a Senior. Lecturer at Ajayi Crowther University, Oyo, Nigeria. His research interests include Distributed systems and Realtime computing.

Andrew Ebhomien Odion obtained B.Sc (Ed) Chemistry from University of Lagos, Nigeria in 1988, PGD Computer Science 1993, MBA (Information Management Technology) from Federal University of Technology Owerri, Nigeria 2004 and M.Sc (Computer Science) 2012 from University of Benin City Nigeria. He is currently running M.Phil (Computer Science), University of Benin, Benin City, and teaches Computer Science in Benson Idahosa University, Benin City Nigeria. His area of interest is Programming Languages and Web Application.

Ogbomo-Odikayor. I.F. obtained B.Sc in Computer Science from University of Benin, Benin City 1999.PG (Ed) in 2007 and Master of Science in Information Technology in 2012 and currently working on her PhD Thesis.