# Resource Allocation Technique for Distributed System using Prioritized Class-Based Algorithm

**Roshan Kavuri, Niraj Upadhayaya, A. Govardhan**

***Abstract: Distributed Systems (DS) are the systems in which the components are distributed at different locations in a network and these components interact with each other by passing messages. In DS each system has a memory and a processor pair. DS is cost-effective, more reliable and faster when compared to the conventional computer systems. But there are some issues in the allocation of resources in such systems as the resources are shared over a number of systems apart from its own individual tasks. The current literature developed various techniques to allocate the resources efficiently by using basic algorithms based on local and global messages and tradeoffs. However, these methods may not be effective as they require special storage and maintenance and constraints the resource availability. In this paper, an efficient resource allocation technique is developed using prioritization and class-based algorithm. The tasks are first given priority and then resources are allocated by fixing a certain percentage of bandwidth for each priority group per execution cycle. The simulation results show that the proposed algorithm is efficient in resource allocation when compared to the conventional techniques.***

***Index Terms: Keywords: Distributed system, Resource allocation, priority, class-based algorithm, Aging.***

## I. INTRODUCTION

Distributed systems are set of heterogeneous computer systems connected over a network using different topologies. These systems share hardware and software of another system located in the different area by passing messages in the form of request and response. The system which requests any service is called a client system and the system which will service the request and create a response is called a server system. These systems do not share memory and each system has its own memory unit, unlike parallel systems. The expense of accessing the local resource is less when compared to the accessing of resources of other systems in the network. Distributed systems are reliable and work with high speed when compared to the single system as the tasks are carried out concurrently by sharing of resources located at different systems. But resource allocation is critical in such systems and requires more concentration as the tasks are processed depending on the availability of the resources present in distributed systems.

The present literature addresses this issue by using various algorithms. These algorithms are based on local/global decision making, considering the tradeoffs of the system and resource parameters to efficiently allocate the resources. However, these techniques may require update and maintenance on regular basis and limit the resource availability period. In the presented approach the prioritized class based algorithm is designed to allocate the resources efficiently. The tasks are first classified based on their priority and then class-based scheduling is done by allocating a certain percentage of bandwidth to each of the priority group. As each group has its share in one execution cycle there is no scope of starvation of less prioritized tasks. In addition, a monitoring and aging block is added to the system to mitigate the starvation problem. The proposed approach can be stepwise discussed as

i. Prioritize the incoming tasks.
ii. Making tasks of comparatively of equal size
iii. Monitoring the tasks present in the queues of the respective priority group and applying the aging technique
iv. Scheduling the tasks in a round-robin manner by the allocation of a certain bandwidth percentage.

The rest of the paper is organized as follows in section II presents a related work, the proposed approach is discussed in section III. In section IV simulation results are presented and the paper is concluded in section V.

## II. RELATED WORK

There are many works carried out by the research community to allocate the resources efficiently and avoid starvation of the tasks. Pathak et al. [1] discussed a framework for dynamic and static resource allocation and for resource management which can adjust dynamically in grid computing environment. Hussain et al. [2] presented different methods for allocating resources in a distributed system. The authors also discussed the classification of the systems and the similarities of the then existing algorithms. The authors in [3] designed two methods for resource allocation based on the wait time polynomials considering global queuing and local messages. However, the polynomials used may create complexities. Bandaru [4] presented resource allocation method in distributed systems based on game theory using non-cooperative strategy. The approach considers the resource's present state instead of cost needed to improve the fairness of the system. Kandan and Manimegalai [5] proposed a method for resource management and its allocation in distributed system. The resource utilization in the system is handled by global and local operators. But this approach may not be efficient as it has to update the information on regular basis and request handling from the tasks may be complex. Di and Baochun [6] discussed an iterative asynchronous approach for resource allocation in distributed environment to overcome the complexities due to passing of messages for system interactions. However, this method may not be appropriate as the gradient technique used is highly sensitive.

Zoghdy et al. [7] presented a technique in which the resource allocation is done on the basis of resource occupation and communication cost which can with stand failure at single point however it cannot handle the multi point failures in the system. Smith et al. [8] proposed a scheduling algorithm based on the resource reservation. The resources are reserved in advance by the tasks and it is assumed that the tasks can be terminated and restarted. However, this algorithm may fail if any high priority tasks arrive without reservation and the delay is also increased in the system. Wolf et al. [9] designed a scheme of scheduling optimizer for distributed application (SODA) to overcome complexities of resource allocation in distributed systems and proposed a solution to handle the real time problems while maintaining the scalability of the systems. The approach applicability is restricted by the trade-off between resource and task allocation. The approach discussed in [10] presents two scheduling algorithms namely heterogeneous earliest finish time (HEFT) and critical path on a process (CPOP) to minimize the complexity of passing of messages and resource allocation in distributed environment. Kaushal [11] discussed a mathematical model by using trade-off between the computer networks and data base files for resource allocation in distributed systems. A heuristic design was proposed to allocate computer, database, files and report generation at the same time. Ramirez and Martinez [12] presented two algorithms for resource allocation depending on the limitations of equality and inequality and also to minimize the cost globally. But this approach is efficiently valid only at small step size. A digraph variation was considered [13] to allocate resources which depend on local information. However the surplus vector needs regular update.

Zhuk et al. [14] presented a scheduling hierarchy of two-levels. In level one the jobs are allocated to the resources by the broker and at the second level the tasks are scheduled using strip-packing technique by the resources. But the assumptions made in this approach may not be practically true. John and Paul [15] discussed a probabilistic technique with real time response of distributive algorithm which may not be accurate as probability technique does not provide exact information. Zbigniew [16] presented a resource allocation method depending on the global optimal of utilization of utility function. However, there are limited forms of utility function. Piotr [17] discussed an approach based on game theory for resource allocation in distributed systems by using non-cooperative and cooperative property. However, a backup system needs to be maintained and this approach may not handle the process of data updating after the execution of each remote tasks. In the presented approach class based algorithm is used with prioritization to efficiently allocate the resources.

## III. PROPOSED APPROACH

In this approach Class based method is used to schedule the tasks. In this method, the tasks are classified into three groups based on priorities of the tasks. Group 1 has highest priority ‘$P_1$’, group 2 has medium priority ‘$P_2$’ group 3 has lower priority ‘$P_3$’. In class based method the tasks that are to be processed should have the same size. In one cycle of processing, certain percentage of bandwidth or time slot is allocated to each of the group. In general the tasks have different size, to make each task of comparatively equal size we use secondary memory buffer for each of the group namely $b_1$, $b_2$ and $b_3$ for $P_1$, $P_2$ and $P_3$ respectively. If the tasks are prioritized, the low priority tasks are made to starve for the bandwidth and its turn to get processed. Hence, round robin scheduling is used in the class-based technique to overcome the starvation of low priority tasks. In this type of scheduling a percentage of bandwidth is allocated to each group of task having the same task size such that each of the group will have a chance to get processed and starvation can be avoided.

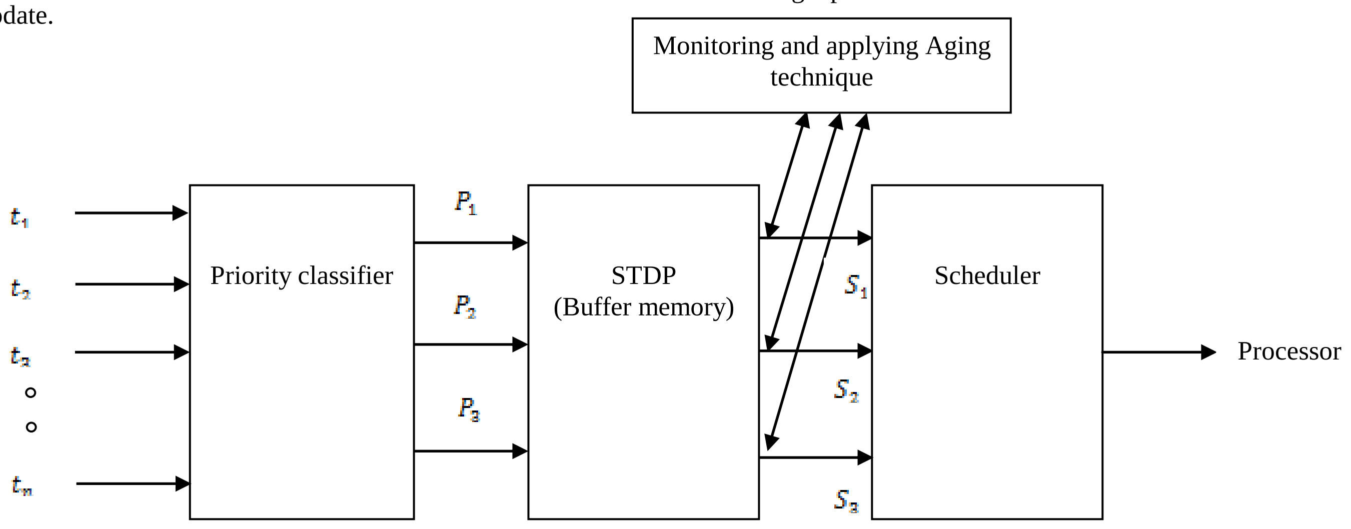


**Figure 1 Block diagram of the proposed approach**

The block diagram of the proposed approach is shown in figure 1. Buffer memory allocated to each of the groups is divided into layers of constant size '*x*'. If the size of the incoming task of each group is equal to the layer size '*x*', then the task is loaded into the memory layer. If the task size is less than the layer size '*x*', then the next task or a part of the next task is loaded into the layer along with them to make the size equal to that of '*x*'. If the size of the task is greater than '*x*', the part of the task of size '*x*' is filled into the layer and the remaining part of the task is loaded into the next layer of buffer memory. Note that the tasks can be divided into small modules. Hence it gives the flexibility to make the tasks of comparably of constant size (i.e. '*x*'). These three cases are represented as

Let the incoming task into the buffer memory be of size '*y*' such that if $y < x$ and

$$x - y = R_c$$

where $R_c$ is the remaining capacity of the buffer memory layer and the next task is loaded. This process continues till $R_c$ becomes zero and if $y > x$ then

$$y - x = E_c$$

where $E_c$ is the extra memory space required by the task apart from '*x*', hence the task is divided and $E_c$ part is loaded into the next layer of buffer memory. If the incoming task size $y = x$, then it is directly loaded into the buffer memory. Fig. 2 represents the layered structure of buffer memory of size '*x*'. After this process of resizing, the tasks are queued in $S_1$, $S_2$ and $S_3$to be scheduled and executed.

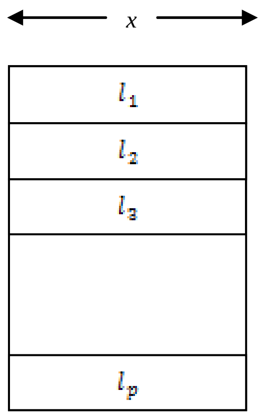


**Figure 2 Buffer memory**

where $l_k$ is the $k^{th}$ layer and $k = 1, 2, 3 \ldots\ldots p$

The tasks of comparably equal size are generated by buffer memories of each group. The tasks are scheduled in a round robin fashion by assigning the percentage of bandwidth to each group of tasks such that each group gets an opportunity to access the processor (resources). In figure 3, $P_1$ is processed twice in one processing cycle. The user can allocate the percentage of the processing cycle to each group accordingly depending on the number of tasks present in that respective group. To ensure that there is no task starving we introduce aging technique in the system such that if the task is in ready queue for a long time its priority is increased at a particular frequency such that it gets executed on gaining the priority.

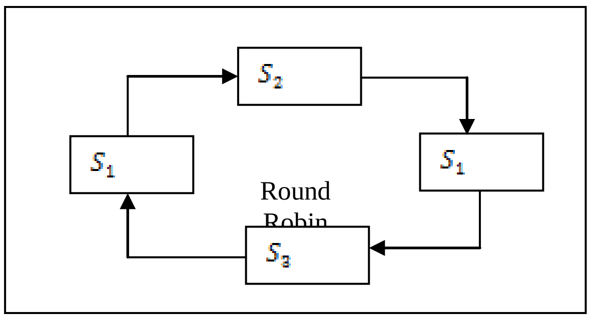


**Figure 3 Scheduler**

### 3.1. *Modeling of the Proposed Approach*

#### 3.1.1. *Prioritizing the tasks*

Let $t_1$, $t_2$, $t_3$,………….. $t_n$ are the incoming tasks these tasks are divided into three groups based on their priority. In general priority of the tasks is given from 255-0 with 255 being the highest priority and 0 being the lowest one. Hence the tasks are classified in three groups can be represented as

$$171 \le t_i \le 255 \rightarrow P_1 \quad (1)$$
$$86 \le t_i \le 170 \rightarrow P_2 \quad (2)$$
$$0 \le t_i \le 85 \rightarrow P_3 \quad (3)$$

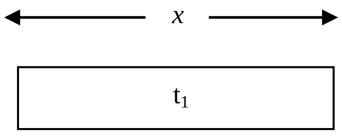
where $t_i$ is the $i^{th}$ task and i={1, 2, 3 ……n}

Note-If the tasks arrive with same priority then they are arranged in first come first serve methodology.

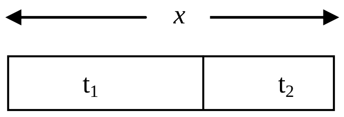
#### 3.1.2. *Creating Tasks of Equal Size*

Consider the size of the incoming task is y and each task is made to the size of x in this state three cases arise

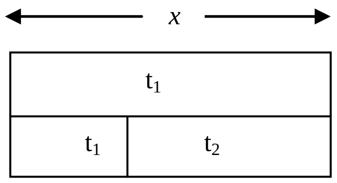
i if $y = x$ then,



ii if $y < x$ then,



iii if $y > x$ then,



Note- As the tasks can be divided into modules for concurrent processing the procedure of making the tasks of equal size may not be difficult.

### *3.1.3. Class-based scheduling of the tasks*

Buffer memories are used to make the tasks of comparably equal size. After buffer memory the equal size tasks are sent to their queues i.e. $S_1$, $S_2$ and $S_3$ having high, medium and low priority respectively. As the tasks are divided based on the priority, the traffic of the incoming tasks is reduced. In class based algorithm all queues are given equal bandwidth for execution. However, one can divide the execution cycle equally among $S_1$, $S_2$ and $S_3$. This algorithm has flexibility to consider one queue twice or more (desired by the user) in each execution cycle as depicted in fig. 3,

$$S_1 \rightarrow 50\% \text{ of bandwidth}$$
$$S_2 \rightarrow 25\% \text{ of bandwidth}$$
$$S_3 \rightarrow 25\% \text{ of bandwidth}$$

$S_1$ is repeated twice in one cycle and hence high priority tasks get executed more fast when compared to medium and lower at the same time no tasks is starving in other queues. If one queue is empty the bandwidth is allocated to the next queue without wasting the time.

### *3.1.4. Monitoring and Applying Aging Technique*

A monitoring block monitors all the three queues $S_1$, $S_2$ and $S_3$. If any tasks waits for a longer period in any of the queue (i.e. threshold time which is user define), then aging technique is applied to it and placed in the queue according to its priority value and finally gets executed. Hence, no task is subjected to starvation.

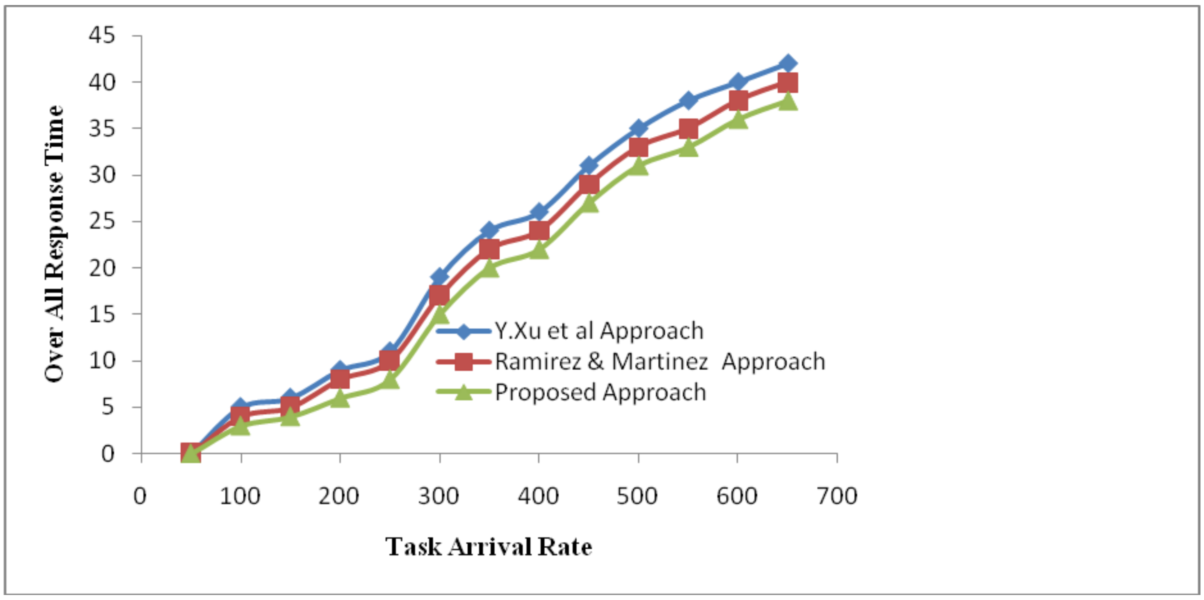


**Figure 4 Task Arrival Rate V/S Over All Response Time**

Let the threshold waiting time value be $T_h$ (user defined) above which the task is said to be starving for the resource for its execution and $W_{Ti}$ be the current waiting time of the task,

then, if

$$W_{Ti} << T_h \rightarrow \text{no action is required} \qquad (4)$$
$$W_{Ti} \approx T_h \rightarrow \text{aging technique is applied} \qquad (5)$$
$$W_{Ti} >> T_h \rightarrow \text{task is starving} \qquad (6)$$

In aging technique, the priority value of the task is increased by one for every time interval say '$v_t$' which can be decided according to the incoming traffic.

## IV. SIMULATION RESULTS

To simulate the performance of the proposed model a GridSim v 4.0 [18] heterogeneous environment was used with multiple resource specifications. All the available resources are different with respect to their operating systems, CPU speed and RAM. In this simulator all the given assignments (tasks) are represented as the objects of the GridLet. These assignments consist of the required information about the assigned tasks and the detailed information about their management system. The needed information of the available resources can be extracted from the Grid information service (GIS) entity, which monitors all the available resources in the present environment.

The simulation results were performed on a personal computer (PC) having the following specifications

Processor: Dual core

RAM: 2 GHz, 4 GHz.

Operating system: Windows 8.

The speed of the spectrum of low capacity and high capacity links is varied from 40 Mbps to 100 Mbps and the speed of low capacity links were situated to 8 Mbps (time units are considered in seconds).

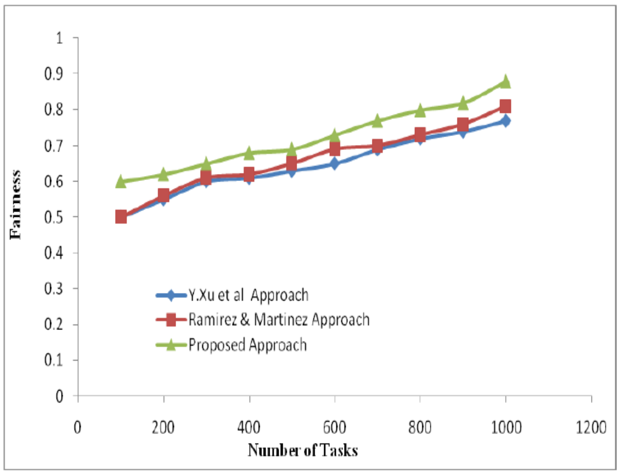


Figure 5 Number of Tasks V/S Fairness

The presented approach is correlated with the art of work (i.e. Y.Xu et al approach [13] and Ramirez & Martinez approach [12]). From figure 4 it has been observed that the response time of the proposed approach with respect to the arrival rate is improved. This is due to the usage of the additional monitoring system that continuously monitors the task parameters (namely waiting time and its estimated processing time) and accordingly processes it. Figure 5 represents the fairness of the approaches with respect to the number of tasks arrived. The proposed approach outperforms the conventional approaches as the tasks are processed before their starvation time is reached. As a result of which the presented approach improves the fairness of the system.

## V. CONCLUSION

Resource allocation is a crucial part in any system for a task to be processed. In the proposed approach an efficient resource allocation method is designed for distributed system. The designed method mitigates the task starvation in a system. Prioritized class based algorithm is presented to provide the optimal time required for the tasks based on their priority. Aging technique is also introduced to avoid the starvation of low priority tasks. The simulation results show that the

## References

1. J. Pathak, J. Treadwell, R. Kumar, P. Vitale, F. Fraticelli, and P. Alto, "A framework for dynamic resource management on the grid", HPL2005-153, August, 2005.
2. Hussain, H., et al, "A survey on resource allocation in high performance distributed computing systems". Parallel Comput. 39(11), 709–736 (2013)
3. Ivor Page, Tom Jacob, and Eric Chern, "Fast Algorithms for Distributed Resource Allocation". IEEE Transactions On Parallel And Distributed Systems, Vol. 4, No. 2, February 1993.
4. Bandaru thesis on "Resource Allocation in Physically Distributed System using Non-Cooperative Game Theory". Computer Science and Engineering National Institute of Technology Rourkela, June 2, 2013.
5. M. Kandan and R. Manimegalai "A Framework for Effective Resource Allocation in a Distributed Cloud Environment". International Journal of Applied Engineering Research, ISSN 0973-4562 Vol. 10, No.87, 2015.
6. Di Niu and Baochun Li, "An Efficient Distributed Algorithm for Resource Allocation in Large-Scale Coupled Systems [C]", *INFOCOM2013 Proceedings IEEE*, pp. 1501-1509, 2013.
7. S. F. El-Zoghdy, M. Nofal, M. A. Shohla and A. El-sawy "An Efficient Algorithm for Resource Allocation in Parallel and Distributed Computing Systems". International Journal of Advanced Computer Science and Applications, Vol. 4, No. 2, 2013.
8. Smith, W., Foster, I., & Taylor, V. (n.d.), Scheduling with advanced reservations. Proceedings 14th International Parallel and Distributed Processing Symposium, IPDPS 2000. doi:10.1109/ipdps.2000.845974
9. Wolf, J., Bansal, N., Hildrum, K., Parekh, S., Rajan, D., Wagle, R., Wu, K.L., Fleischer, L. "SODA: An optimizing scheduler for large-scale stream-based distributed computer systems". In: Issarny, V., Schantz, R. (eds.) Middleware 2008. LNCS, vol. 5346, pp. 306–325. Springer, Heidelberg 2008.
10. Haluk Topcuoglu, Salim Hariri and Min-You Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing". IEEE Transactions on Parallel and distributed systems, vol 13. no. 3, 2002.
11. Kaushal Chari "Resource Allocation and Capacity Assignment In Distributed Systems". Computers Ops Res. Vol. 23, No. 1 , pp. 1025-1041, 1996.
12. E. Ramírez-Llanos, S. Martínez, "Distributed and robust resource allocation algorithms for multi-agent systems via discrete-time iterations", *Proc. IEEE Conf. Decision Control*, pp. 1390-1396, 2015.
13. Y. Xu, T. Han, K. Cai, Z. Lin, G. Yan, M. Fu, "A distributed algorithm for resource allocation over dynamic digraphs", *IEEE Transactions on Signal Processing*, vol. 65, pp. 2600-2612, May. 2017.
14. Zhuk, S., Chernykh, A., Avetisyan, A., Gaissaryan, S., Grushin, D., Kuzjurin, N., Shokurov, A. (n.d.). Comparison of scheduling heuristics for grid resource broker. Proceedings of the Fifth Mexican International Conference in Computer Science, 2004. ENC 2004. doi:10.1109/enc.2004.1342632.
15. J. Reif and P. Spirakis, "Real-time resource allocation in a distributed system," in ACM SIGACTSIGOPS Symp. Principles of Distributed Computing, Aug. 1982, pp. 84-94.
16. Zbigniew Wesołowski, "Network Resource Allocation in Distributed Systems: A Global Optimization Framework". IEEE 2nd International Conference on Cybernetics (CYBCONF), 2015
17. Piotr Skowron, thesis on "Resource Allocation in Selfish and Cooperative Distributed Systems". University of Warsaw, Informatics and Mechanics, September 2014.
18. R. Buyya, "A grid simulation toolkit for resource modelling and application scheduling for parallel and distributed computing", www.buyya.com/gridsim/

## AUTHORS PROFILE

Roshan Kavuri has obtained his B. E Degree from Andhra University in 1992 and M. Tech (CSE) from JNT University, Kukatpally, Hyderabad. He is having nearly 25 years' experience as faculty of Computer Science and Information Technology departments and also in industry as well. He is pursuing his Ph.D. from JNTU Hyderabad.

Niraj Upadhayaya has done B.E in Electronics Engineering and M.S. in Software Systems. He did his Ph.D. in Parallel Computing from University of the West of England, Bristol, UK. He served in Indian Broadcasting services for 18 Years. He is working now as Professor and Dean (R&D) at J.B. Institute of Engineering & Technology.

A Govardhan has done B.E.(CSE) from Osmania University College of Engineering, Hyderabad in 1992, M.Tech from Jawaharlal Nehru University (JNU), New Delhi in 1994 and Ph.D. from Jawaharlal Nehru Technological University, Hyderabad in 2003. He is working as a Professor in JNTUH and has an experience of 25 years in teaching.