




Smart Priority Resource Allocation Scheduling in Distributed Systems

Saud S. Alotaibi

*Department of Information Systems, College of Computer and Information Systems,
Umm Al-Qura University, Makkah, Saudi Arabia
E-mail: ssotaibi@uqu.edu.sa*

Access this article online	
Quick Response Code:	Website: www.uqu.edu.sa/jea E-mail: jea@uqu.edu.sa Table of Contents - Current issue: https://uq.sa/CFwcAm
	
© Umm Al-Qura University Journal for E & A, Vol.10 Issue No.1, pp.21-29 September 2019 <i>Under Legal Deposit No. p- ISSN: 1658-4635 / e- ISSN: 1658-8150</i>	

Smart Priority Resource Allocation Scheduling in Distributed Systems

Saud S. Alotaibi

Department of Information Systems, College of Computer and Information Systems,
Umm Al-Qura University, Makkah, Saudi Arabia
E-mail: ssotaibi@uqu.edu.sa

الجدولة الذكية لتخصيص الموارد ذات الأولوية في الأنظمة الموزعة

سعود العتيبي

قسم علوم الحاسب الآلي، كلية الحاسب الآلي ونظم المعلومات،
جامعة أم القرى، مكة، المملكة العربية السعودية

الملخص:

تعرف الأنظمة الموزعة على أنها عبارة عن مجموعة من الأنظمة المرتبطة مع بعضها البعض من خلال طبيعة شبكية تسمح لها من تشارك الموارد فيما بينها من خلال إرسال الرسائل. حيث يعتبر حجز الموارد بتلك الأنظمة من أهم الركائز الخاصة بها، وذلك من خلال خدمة النظام لذاته أو مشاركته بعض الخدمات والطلبات من الأنظمة الأخرى المتواجدة بالشبكة المرتبطة بينهم. فمن خلال استخدام مفاهيم الذكاء الاصطناعي يمكن تقديم مفهوم ذكي لإدارة أولية حجز الموارد بهذه الأنظمة. بحيث يمكن معالجة أحد أهم القصور في جدولة الأولويات الا وهو تجويع المهام ذات الأولوية المنخفضة. لذا، تعرض هذه الورقة العلمية طريقة ديناميكية تعتمد على الذكاء الاصطناعي للتخفيف من مشكلة التجويع الحاصل لتلك المهام. حيث أظهرت نتائج المحاكاة التي تم علمها تفوق الطريقة المقترحة بهذه الورقة على الطرق التقليدية السابقة.

الكلمات المفتاحية: الأنظمة الموزعة، حجز الموارد، جدولة، النظام الذكي، الذكاء الاصطناعي.

Abstract:

A distributed system is a group of systems connected together through some network topology. The resource allocation is critical in such systems, as a single system must serve itself and the requests from other systems present in the network. The art of work presents different methods such as priority scheduling, global and local information, by passing messages and utilizing the tradeoffs of system parameters to allocate the resources in distributed systems. However, these methods may not be efficient as global and local information requires a regular update, passing messages may be complex in large networks and tradeoff consideration limits the system utilization. In this paper, a modified priority-based resource allocation technique that uses artificial intelligence is presented. The main shortcoming of priority scheduling is a low priority task starvation. In the presented approach, a dynamic artificial intelligence algorithm is used to overcome the starvation problem. Simulation results show that the proposed technique outperforms conventional techniques.

Keywords: Distributed systems, resource allocation, scheduler, smart system, artificial intelligence.

1. Introduction:

A distributed system (DS) is a network of systems connected through different topologies. These systems share resources located in different portions of the network, e.g., hardware resources such as printers and software resources such as data files. They interact with each other by passing messages. A distributed system has high speed as it can process the task (jobs) concurrently. This enhances the system's performance as traffic is shared among the systems connected to the network. DS is reliable as they can resist a single point failure, i.e., systems in the network are relatively unaffected if one system fails. Openness, transparency, and resource sharing are the primary characteristics of a distributed system. Resource allocation in such systems is difficult as data is updated frequently, and each system must serve itself as well as requests from another system.

The present literature addresses this issue with various algorithms. These algorithms are based on priority scheduling, local/global decision making and considering the tradeoffs between system parameters are used to efficiently allocate resources. However, these techniques may require regular updates, maintenance, and limit the resource availability period. In this paper, a modified priority-based scheduling method is developed to mitigate the shortcoming of priority scheduling, i.e., starvation of low priority tasks and also no tradeoff and updates are required to allocate the resources in a distributed environment. A smart scheduler with a feedback mechanism (FBM) is added to the priority scheduler to monitor the arrival times, burst time, and waiting times of various tasks in the ready queue. The time threshold value is defined in the feedback system, beyond which the task is subjected to starvation. If the waiting time for a task is equal to or greater than the FBM threshold, the task is designated to be a high priority task and is executed first, i.e., the task is not made to starve. The proposed approach is dynamic and can accurately allocate resources to various tasks. The rest of the paper is organized as follows: Related work is discussed in Section 2 and the proposed approach is presented in Section 3. Simulation results are presented in Section 4 and the paper is concluded in Section 5.

2. Related Work:

Previous research has focused on allocating resources efficiently and preventing task starvation. **Ivor et al. [1]** developed two algorithms based on local messages and global queuing that rely on polynomial wait times. However, these algorithms create complexities in passing messages due to the polynomial dependence on large networks. **The authors in [2]** discussed various resource allocation techniques for a distributed system. They also classified systems and extracted a common basis for various algorithms. **Kandan and Manimegalai [3]** designed a resource allocation and management technique for use in a distributed environment. In this approach, global and local managers are used to evaluate requests for resources. This process may be complex as evaluation and regular updating is required for resource allocation. **Zoghdy et al. [4]** discussed an algorithm in which resources are allocated based on the ratio of occupation of resources and the cost of communication. Nevertheless, this approach may fail to handle multiple points of failure.

Kaushal [5] presented a mathematical model for resource allocation in a distributed system that uses the tradeoff between file database and computer networks. The author designed a heuristic format for allocating database files and generating reports simultaneously. **Di and Baochun [6]** developed an iterative technique using an asynchronous algorithm to allocate resources and prevent complex messaging interactions among systems in the network but the application of this approach is restricted by the sensitivity of gradient method used. **Bandaru [7]** discussed a game theoretic approach of non-cooperation to allocate resources in distributed systems. The method is cost-effective as it takes into account the present state of the resource instead of the cost to improvise fairness. **Haluk et al. [8]** discussed two scheduling algorithms (heterogeneous earliest finish time (HEFT) and critical path on a process (CPOP)) to reduce complexity and effectively allocate resources.

Joel et al. [9] presented a scheduling optimizer for distributed application (SODA) technique to handle the optimization complexities in real-time and simultaneously maintain system scalability. However, there is a limit as there is a tradeoff between task allocation and resource allocation. **Piotr [10]** discussed a game theoretic approach for resource allocation in distributed systems using the cooperative and non-cooperative properties of the system. However, a backup system is needed to replicate information, and the approach may not be suitable for handling the updating process after each execution of remote tasks. **Yun et al. [11]** developed an algorithm based on digraph variation to efficiently allocate the resource based on local knowledge. However, it requires regular information regarding the surplus vector used in the algorithm. **Zbigniew [12]** proposed an approach for resource allocation in a distributed system based on a utility function for global optimization. However, the assumption that the utility function takes different forms may not be true. **Ramirez and Martinez [13]** discussed two resource allocation algorithms to globally optimize cost. However, the algorithm to the solution converges only when the steps size selected for discrete time multiple agents are small. **John and Paul [14]** presented a distributive algorithm with the real-time response using a probabilistic model for resource allocation in distributed systems. **Mohammad Asad et al. [15]** proposed a framework for resource allocation in cloud computing based on tailored active measurements. However, this method requires more effort to make the approach predictive and responsive to concurrent changes in decision values. **Arun Sukumaran Nair et al. [16]** presents a comprehensive review on multi-agent systems for resource allocation and scheduling in a smart grid, although the presented approach increased complexity by creating multi-agent systems during resource allocation, which may not provide an optimum solution for multiple tasks with a finite number of systems. **Aurélie Beynier et al. [17]** investigated sprite issues in distributed systems using dynamic and partial observations. However, owing to mathematical complexity, this approach may not provide optimal use in distributed resource allocation systems. All the above approaches are depending on global/local information, which requires a regular update of data at frequent time intervals and tradeoffs between the system parameters that create complexities and limits the resource availability time. In the presented approach, a dynamically prioritized

resource allocation method is introduced using artificial intelligence, which does not require any local/global information that needs a regular update and tradeoffs that restricts the system performance. The proposed approach mainly requires the task priority arrival time in the ready queue, waiting time in the ready queue and the task deadline to be processed. The smart system will be monitoring the tasks in each queue and is responsible to move the task from the queues to its execution if the threshold value is reached. If the threshold value does not exceed then the tasks are executed as per the priority assigned to the scheduler.

3. Proposed Approach

In the proposed approach, a modified priority scheduling mechanism is designed using artificial intelligence. The incoming tasks are given priority (i.e. high, medium, or low) based on resource requirements that depend on user considerations for example as in [18 and 19]. The arrival, burst, and waiting times of various tasks are monitored. Tasks are processed according to their assigned priority. However, if high priority tasks are present then some of the medium and low priority tasks are subjected to starvation. A feedback mechanism is designed to prevent task starvation and allocate the resources dynamically. A block diagram of the proposed mechanism is shown in **figure 1**.

Let Q_H , Q_M , and Q_L be high, medium, and low priority queues, let B_H , B_M , and B_L be the high, medium, and low priority buffers, and let W_{THi} , W_{TMi} , and W_{TLi} be the high, medium, and low priority waiting times for the i^{th} task, respectively. Let α_i , β_i , and μ_i be the threshold value of the i^{th} task with high, medium, and low priority, respectively, where $i = 1, 2, 3 \dots N$. These threshold values (i.e., α_i , β_i , and μ_i) of tasks are considered fairly less than the starvation time, where the task is in the starvation state before the application of the smart technique.

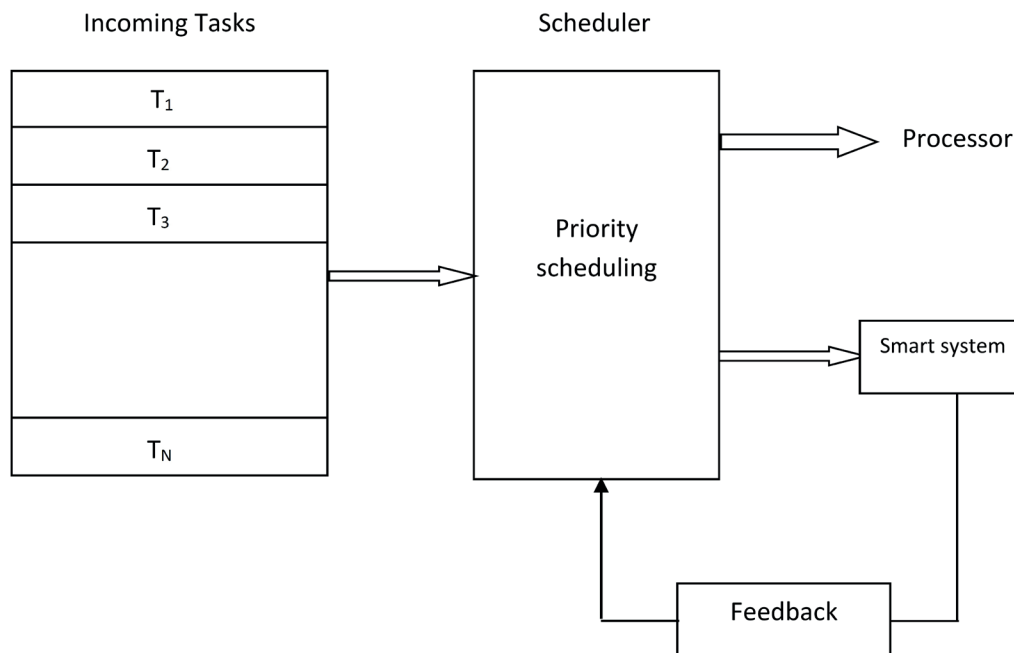


Fig. 1: Block Diagram of Proposed Approach

The anticipated starvation time depends on its waiting time in the ready queue and its deadline for being processed. These two parameters are different for different tasks. The threshold value depends on these parameters. The smart system model applied to the priority scheduler is shown in **Figure 2**.

The following are the steps of the proposed approach:

Step 1: The incoming tasks are given priority by the priority scheduler.

Step 2: The prioritized tasks are monitored and their arrival and waiting times are noted along with the deadline of each task.

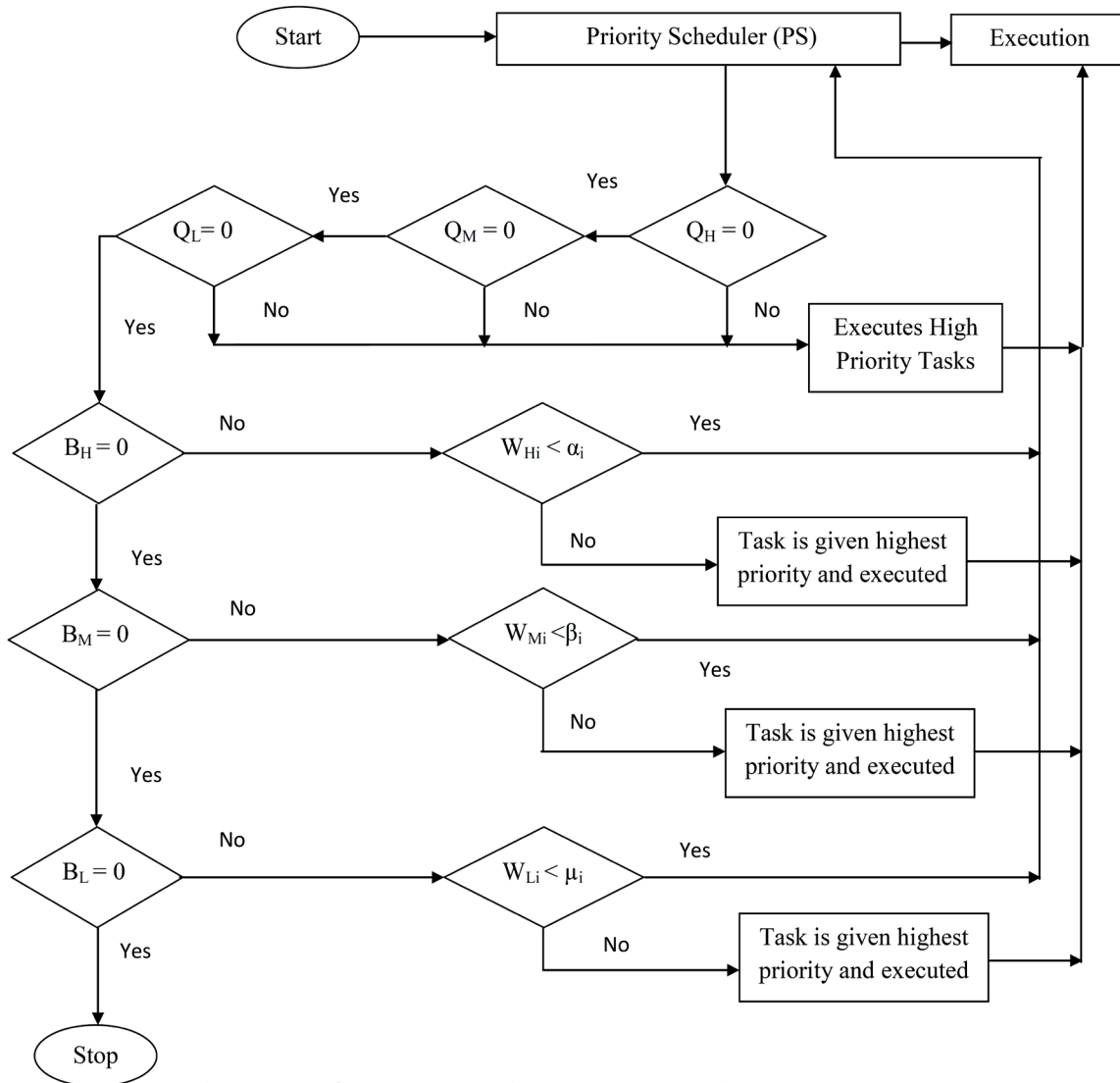


Fig. 2: Smart system Model Applied to the Priority Scheduler

Then the tasks are executed as per the priority assigned to them and if the waiting times are:

$$W_{THi}, W_{TMi}, \text{ and } W_{TLi} \geq \alpha_i, \beta_i, \text{ and } \mu_i \text{ (respectively)}$$

Then the task, which exceeds the threshold value, is given the highest priority irrespective

of the priority assigned by the scheduler and is executed first. Note that priority scheduling is pre-emptive and buffer queues are allocated to each of the primary queues (that are the queues in which tasks are placed after prioritizing) to make the migration of tasks easier and efficient from processor to the queue and vice versa.

The queues of the given priorities are checked and the highest priority tasks are given to the processor for execution. Then, each priority buffer is checked for the threshold time. If the waiting time of the task is less than the set threshold value, then that task is given to the priority scheduler for execution. Furthermore, if the waiting time of the task is equal to or greater than the threshold value, then that task is given the highest priority and is executed first. Each task has its threshold value in each of the respective priority queues according to the burst time of the task. Therefore, tasks are dynamically assigned to the CPU based on their assigned priority and threshold values.

4.Simulation Results:

The proposed approach is validated using the GridSim v4.0 simulator [20] as it is compatible with the resources of heterogeneous systems, users, and resource load balance, which are used to validate resource allocation techniques. The grid information services present in this tool kit provide information on the resources and make them available at the time of the simulation. GridLetobjects in GridSim give information regarding task management and execution. Simulations were on a PC with Windows XP, 2GB RAM, and a 3 GHz dual-core processor. The spectrum speeds of low capacity and high capacity links varied from 60Mbps to 120 Mbps, while those between low capacity links were set to 10Mbps.

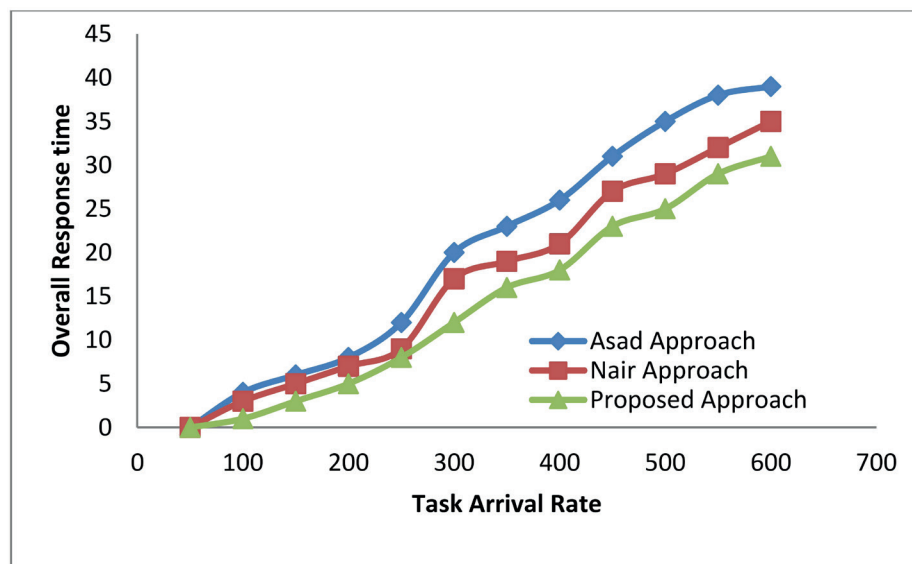


Fig. 3: Task arrival rate versus overall response time

The proposed resource allocation approach is compared with those presented by Asad [15] and Nair [16]. **Figure 3** shows the overall response time concerning the task arrival rate. One can see that the presented approach provides better response time compared to existing approaches. The presented approach runs on a modified dynamic pri-

ority mechanism, which improves the task response time and maximizes the resource allocation throughput.

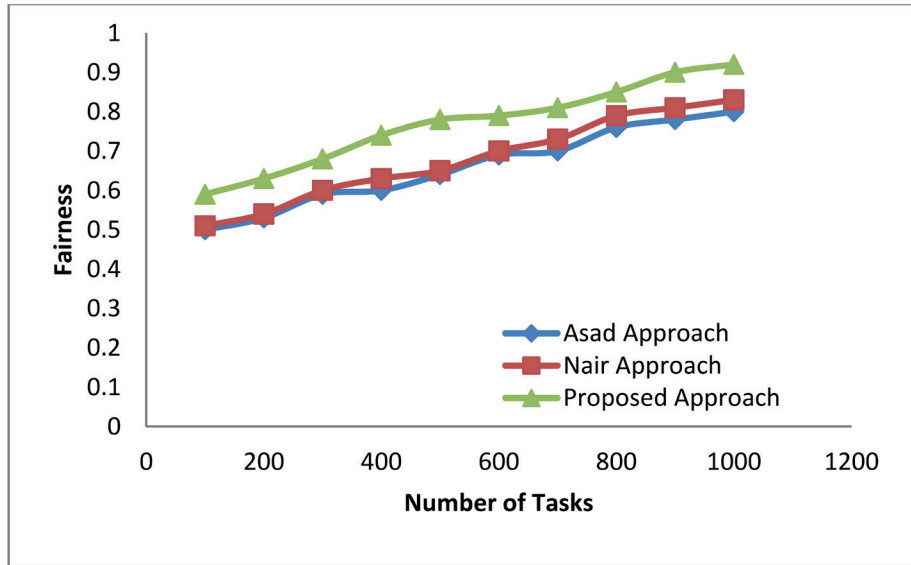


Fig. 4: Number of tasks versus fairness

Figure 4 shows the system fairness concerning the number of tasks. One can see that the proposed approach is fairer when compared to other approaches. The proposed approach is dynamic and can efficiently allocate resources with respect to time as the smart system with the feedback provides the time information required (waiting time, burst time and task deadline) for the execution of high priority task and avoiding low priority tasks starvation without any delay.

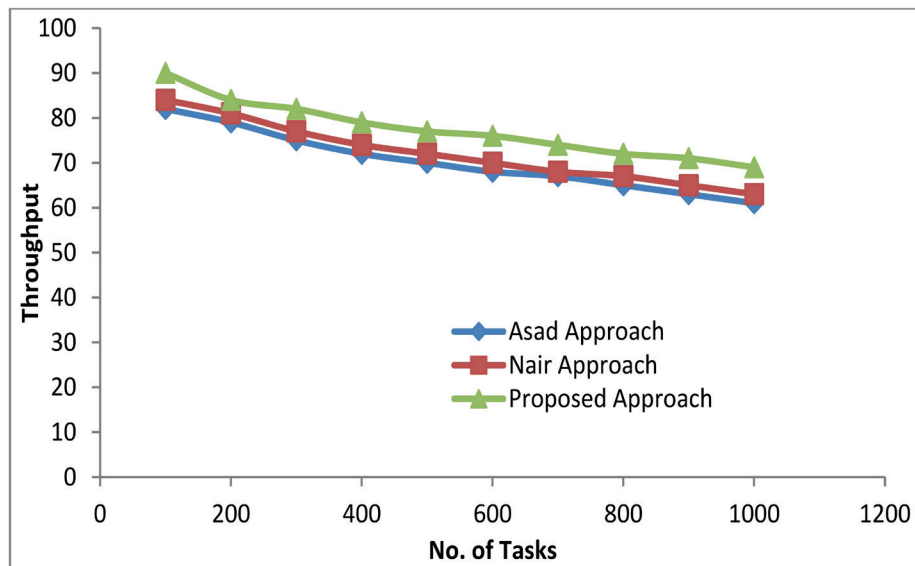


Fig. 5: Number of tasks versus throughput

Figure 5 shows the task throughput concerning the number of tasks in the distributed system. This figure shows that the throughput provided by the proposed approach is higher than that of other approaches as the tasks are given priorities based on the waiting time (in the ready queue) and the deadline of each task (refer figure 2), which reduces task

starvation, in particular, the low priority tasks as they are in general subjected to starvation.

5. Conclusion:

Resource allocation in a distributed system is crucial as these systems share either hardware or software resources with other systems present in the network. The proposed approach dynamically allocates resources based on priority and threshold time values associated with various tasks. Task starvation never occurs as the task gets executed before starvation. The presented approach is optimized and improved the results in comparison with the conventional techniques as the threshold time is set based on the waiting time of the task and its deadline to get processed. This is in contrast to previous techniques, where a fixed time interval is defined for a task to increase its priority (by 1 in each interval) and prevent task starvation. The simulation results show that the proposed technique enhances the system's performance in terms of fairness, throughput and overall response time.

References:

- Page, I., Jacob, T., and Chern, E. (1993), *Fast algorithms for distributed resource allocation*. IEEE T Parallel. Distr. 4 (2).
- Hussain, H., S.U.R. Malik, A. Hameed, S.U. Khan, G. Bickler, N. Min-Allah, M.B. Qureshi, L. Zhang, W. Yongji, N. Ghani, J. Kolodziej, A.Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J.E. Pecero, D. Kli-azovich, P. Bouvry, H. Li, L. Wang, D. Chen, and A. Rayes, (2013), *A survey on resource allocation in high performance distributed computing systems*. Parallel Comput. 39(11), 709–736.
- Kandan, M., and Manimegalai, R. (2015), A framework for effective resource allocation in a distributed cloud environment. *Int. J. Appl. Eng. Res.* 10 (87).
- El-Zoghdy, S.F., Nofal, M., Shohla, M.A., and El-sawy, A. (2013), An efficient algorithm for resource allocation in parallel and distributed computing systems. *Int. J. Adv. Comp. Sci. Appl.* 4 (2).
- Chari, K. (1996), Resource allocation and capacity assignment in distributed systems. *Comp. Ops. Res.* 23(1), 1025-1041
- Niu, D., and Li, B. (2013), An efficient distributed algorithm for resource allocation in large-scale coupled systems. *INFOCOM2013 Proc. IEEE*, 1501-1509.
- Bandaru .R.S. (2013), *Resource Allocation in Physically Distributed System using Non-Cooperative Game Theory*. Thesis, Computer Science and Engineering National Institute of Technology Rourkela.
- Topcuoglu, H., Hariri, S., and Wu, M.-Y. (2002), Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. IEEE T.Parall.Distr. 13(3).
- Wolf, J., Bansal, N., Hildrum, K., Parekh, S., Rajan, D., Wagle, R., Wu, K.L., Fleischer, L. (2008), SODA: An optimizing scheduler for large-scale stream-based distributed computer systems. In: Issarny, V., Schantz, R. (eds.) *Middleware*, pp. 306–325. Springer, Heidelberg.
- Skowron, P. (2014), *Resource Allocation in Selfish and Cooperative Distributed Systems*. Thesis, University of Warsaw, Informatics and Mechanics.

- **Xu, Y., Han, T., Cai, K., Lin, Z., Yan, G., and Fu, M. (2017)**, *A distributed algorithm for resource allocation over dynamic digraphs*. IEEE T. Sig. Proc. 65, 2600-2612.
- **Wesolowski, Z. (2015)**, *Network resource allocation in distributed systems: A global optimization framework*. IEEE 2nd International Conference on Cybernetics (CYBCONF).doi:10.1109/cybconf.2015.7175944
- **Ramírez-Llanos, E., and Martínez, S. (2015)**, *Distributed and robust resource allocation algorithms for multi-agent systems via discrete-time iterations*. Proc. IEEE Conf. Decision Control, 1390-1396.
- **Reif, J., and Spirakis, P. (1982)**, *Real-time resource allocation in a distributed system*. In: ACM SIGACTSI-GOPS Symp. Principles of Distributed Computing, pp. 84-94.
- **Asad, M.A., Pawlikowski, K., and Willing, A. (2011)**, *A framework for resource allocation strategies in cloud computing environment*. COMPSACW, 261-266.
- **Nair, A.S., Hossen, T., Campion, M., Selvaraj, D.F., Goveas, N., Kaabouch, N., Ranganathan, P. (2018)**, *Multi-agent systems for resource allocation and scheduling in a smart grid*. Technology and Economics of Smart Grids and Sustainable Energy 3(15). <https://doi.org/10.1007/s40866-018-0052-y>.
- **Beynier, A., Maudet, N., and Damamme A. (2018)**, *Fairness in multi agent resource allocation with dynamic and partial observations*. AAMAS Stockholm, Sweden, 1868-1870.
- **Huang, Y.-F., & Chao, B.-W.** *A priority-based resource allocation strategy in distributed computing networks*. *Journal of Systems and Software*, 58(3), 221–233.doi:10.1016/s0164-1212(01)00040-1
- **Mishra, K. N. (2016)**, *An efficient voting and priority based mechanism for deadlock prevention in distributed systems*. International Conference on Control, Computing, Communication and Materials (ICCCCM).doi:10.1109/iccccm.2016.7918267
- **Rajkumar Buyya and Manzur Murshed, GridSim, (Nov.-Dec.,2002)**, A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing, *The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, Volume 14, Issue 13-15, Wiley Press, www.buyya.com/gridsim/.

Received:00/00/00

Accepted:00/00/00