

A Reliability-Based Task Scheduling Method for Wireless Distributed Systems

Sonali Bodkhe[1(✉)] and Anjali Mahajan[2]

[1] Department of Computer Science, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India
sendtowiser@gmail.com

[2] Government Polytechnic, Nagpur, Maharashtra, India

Abstract. Parallel computing and distributed computing are the popular terminologies of scheduling. With advancement in technology, systems have become much more compact and fast and need of parallelization plays a major role for this compaction. Wireless computing is also a common concept associated with each new development. Scheduling of tasks has always been a challenging area and is an NP-complete problem. Moreover, when it comes to wireless distributed computing, reliable scheduling plays an important role in order to complete a task in a wireless distributed system. This work proposes an algorithm to dynamically schedule tasks on heterogeneous processors within a wireless distributed computing system. A lot of heuristics, meta-heuristics & genetics have been used earlier with scheduling strategies. However, most of them haven't taken reliability into account before scheduling. Here a heuristic that deals with reliable scheduling is considered. The scheduler also works within an environment which has dynamically changing resources and adapts itself to changing system resources. The testing was carried out with up to 200 tasks being scheduled while testing in a real time wireless distributed environment. Experiments have shown that the algorithm outperforms the other strategies and can achieve a better reliability along with no increase in make-span, in spite of wireless nodes.

Keywords: Distributed Computing · Scheduling · Reliability · Wireless Distributed Systems (WDS)

1 Introduction

One of the basic steps to exploit the capabilities of heterogeneous distributed computing systems is scheduling which is an NP-complete problem. In such cases meta-heuristic is the best choice. The use of heuristics in the initial stages of a meta-heuristic can be effective for improving the solutions obtained by meta-heuristics. Make-span is an important factor to evaluate the efficiency of scheduling algorithms. Task scheduling can be classified in to various types as shown in Fig. 1.

The process of scheduling can be carried out for wired as well as wireless systems. When a set of identical processors connected in the system have the same computational

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023, LNCS 13956, pp. 291–306, 2023.
https://doi.org/10.1007/978-3-031-36805-9_20

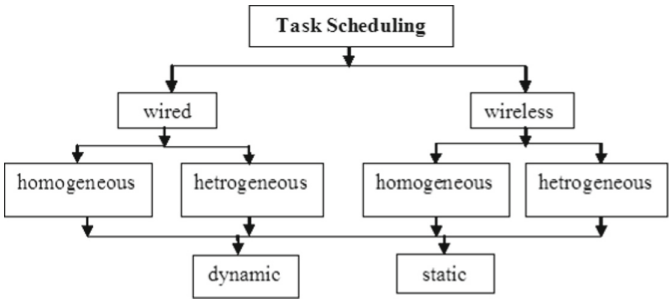

Fig. 1. Classification of task scheduling

speed and processor, the distributed systems are termed as homogeneous distributed computing systems.

When several processors with different computational speeds are interconnected through high speed links together to solve a problem with varying computational requirements, the distributed system is termed as an heterogeneous distributed computing system.

Real time scheduling in multiprocessor systems has gained increased attention over past years and is becoming common these days [1]. However, most of the work carried out targets "under loaded" systems, i.e., systems where the total task utilization demand 'U', is always less than the total available processing capacity of the system, which can be stated as 'm' for an m-processor system. The majority of research on multiprocessor systems focuses on developing scheduling algorithms by trying to understand their schedulability, utilization bounds, i.e., utilization bounds of tasks, below which all task deadlines are met. The limit of this approach is that it is possible to determine the worst-case execution-time behaviors of applications and hence determine the total task utilization demands. After the task utilization demand is known, task schedulability, i.e., the ability of processors to meet all task deadlines can be ensured. For some applications (e.g., [2–4]), it is difficult to determine worst-case execution-time as a priori, since it may depend on the run-time demands like the execution time overruns and unpredictable task arrival patterns, causing short term and/or permanent overloads. Whenever overloads occur, such applications require graceful timeliness degradation, thus fulfilling as many deadlines of importance tasks as possible, irrespective of the urgency of tasks.

Heterogeneous systems are dependable computing platforms. Many a times heterogeneous distributed computing (HDC) systems can achieve higher performance with lesser cost than single super-systems on account of resource optimization. Besides performance, fairness is also a very important requirement for the scheduling approaches.

However, in these systems, processors and networks are not free from failure and thus tend to increase the criticality of the running applications. In order to deal with failures, a reliable scheduling algorithm is an important requirement. Unfortunately, most scheduling algorithms applied for scheduling tasks in HDC systems do not sufficiently consider inter-dependent reliability requirements of tasks [5–9].

Reliability of a wireless distributed system is an important parameter in real time scheduling of tasks [10–12]. The failures occurring in a distributed system can result in

anything from minute errors to disastrous halts. When a reliable distributed system is designed it should as fault tolerant as possible. The proposed work focuses on finding a solution at the software level to wireless core-computing with due attention to the factor of concern i.e. power utilization along with scheduling and reliability. On the lines of scheduling with heuristics, a strategy for the same is applied to wireless distributed systems wherein WDSs are more prone to instability.

The work aims in proposing a reliable task scheduling strategy so that a job submitted can be reliably and successfully completed [13–17]. It also focuses on the power factor of a wireless node along with reliability. Make-span is another parameter that has been worked upon, such that, both reliability and make-span can be reduced.

2 Methodology/Experimental

The basic scheduling methodology considered so far in earlier researches, takes into account the basic scheduling model of Fig. 2 in a heterogeneous cluster environment, wherein a *global scheduler* works in collaboration with a Resource Manager. It is assumed that all parallel jobs are submitted to the global scheduler in the form of a Directed Acyclic Graph. A *schedule queue* (*SQ*) for arriving jobs is maintained by the scheduler, which is responsible for scheduling real-time tasks of each job in SQ. It places an accepted job in a *dispatch queue* (*DQ*) and from here, tasks of each accepted job are sent to designated machines, also called *processing elements (PEs)*, for execution. The scheduler also executes in parallel with PEs, each of which maintains a *local queue* (*LQ*) to which real-time tasks are transmitted from *DQ*. [7, 18].

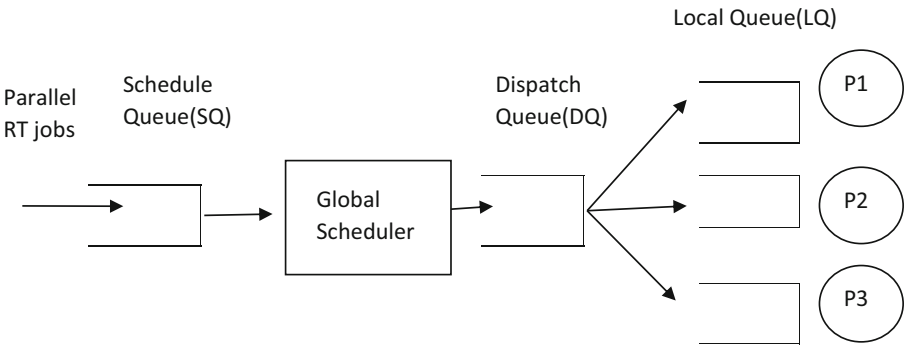

Fig. 2. A Basic Task Scheduling Model

However, keeping tasks in queue doesn't fulfill the reliability concept since the tasks are in a waiting state and also the amount of computations done by the earlier task might no longer retain the reliability of that wireless system to the same extent for other tasks in the queue. Hence, the local queues have been eliminated as in Fig. 3. According to the proposed strategy, a SQ contains all the tasks in a job defined in terms of dependency levels here. The proposed scheduling algorithm is applied to these levels and one task at a time is assigned to each scheduler [7].

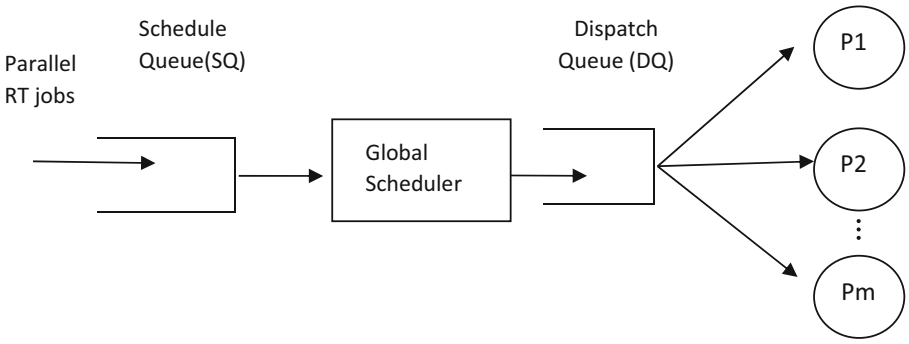

Fig. 3. A Proposed Task Scheduling Model

The input to the algorithm is a job which is modeled through a directed acyclic graph (*DAG*). G = {*V, E*}, where $V = \{v_1, v_2, \ldots, v_n\}$ represents a set of real-time tasks and E represents a set of directed edges among the real-time tasks. $M_{ij} = (v_i, v_j) \in E$ denotes a message transmitted from task v_i to v_j, and $|m_{ij}|$ is the volume of data transmitted between these tasks. A heterogeneous cluster is represented by a set $P = \{p_1, p_2, \ldots\ldots, p_m\}$ of processors, where p_i is a processor with local memory. Machines in the heterogeneous cluster are connected with one other by a high-speed network. A machine communicates with other machines through message passing, and time between two tasks assigned to the same machine is assumed to be zero [20, 21].

2.1 Generation of DAG

The task graph as well as the WDS is assumed to be static, meaning, it remains unchanged during execution. The DAG is a set of nodes $V = \{u_1, u_2, \ldots\ldots, u_n\}$ with each node representing a task. All the work is carried out in OPEN-MPI. The tasks are generated using the DAG from the upper triangular representation of an adjacency matrix of tasks. Here in the proposed work, a DAG generator module is used which applies a DOT parser. The job along with the tasks to be scheduled is stored in the form of a DOT file. The same representation can be used for a real time job too.

2.2 Battery Status and Discharge Power

The status of power remaining plays an important role in scheduling of tasks that are being shifted. The amount of power remaining with a wireless node determines its acceptability for extra tasks. This power is considered in the calculation of the discharge rate for allotted tasks.

Discharge power= 1-remaining power

2.3 Reliability of the System

Reliability is defined to be the probability that the system will not fail during the time that it is executing the tasks. The failure rate helps in determining the reliability of the

system to complete a particular task. The higher the reliability cost, the less reliable the system is.

The reliability of task execution of each process on each processor is given as in Eq. 1

$$R_{ij} = dp_j t_i(j)^{\cdot} + f_j c_i t_j(i)^{\cdot} + \sum_{p \varepsilon Pred(i)}^{\cdot} \sum_{k=1}^{M} g_{kj} w_{pi} d_{kj} \tag{1}$$

Rate of power discharge + computation time + data transfer rate.

Where,

dp_{ij} = discharged power by $task_i$ on $processor_j$ = 1 – (value / 100).

t_{ij} = Computation time of $task_i$ on $processor_j$

f_j = failure rate of $processor_j$

c_i = criticality of $task_i$

g_{kj} = communication link failure rate of kth task on j^{th} processor

w = weight matrix

d = time delay to send unit length data from Pk to Pj

n = no. of tasks

m = no. of processors

R = Reliability matrix

Most of the earlier work focuses on one or two parameters only. If reliability is the issue to be dealt, scheduling is not stressed upon, if scheduling is the issue to be dealt, reliability is ignored. Also amount of power associated with a wireless node has a crucial role to play. The current work focuses on all parameters. The estimation of the failure rate of a wired and wireless system listed below:

Failure rate for.

1. **Wireless node = 8% = 0.08**
2. **Wired node = 4% = 0.04**

To maximize system reliability, it is needed to minimize the Reliability Cost. Thus Reliability Cost is used as the indicator of how reliable a given system is when a group of tasks are scheduled and executed on to it. The lower the reliability cost is, the higher the reliability is. The Overall Reliability cost is given as

$$RC = \sum_{j=1}^{M} R_j \tag{2}$$

where, Rj is the reliability cost of task ui on processor Pj

2.4 Phases of Task Scheduling in WDS

As shown in Fig. 4, task scheduling is carried out in 3 phases:

1. System Analysis
2. Scheduling

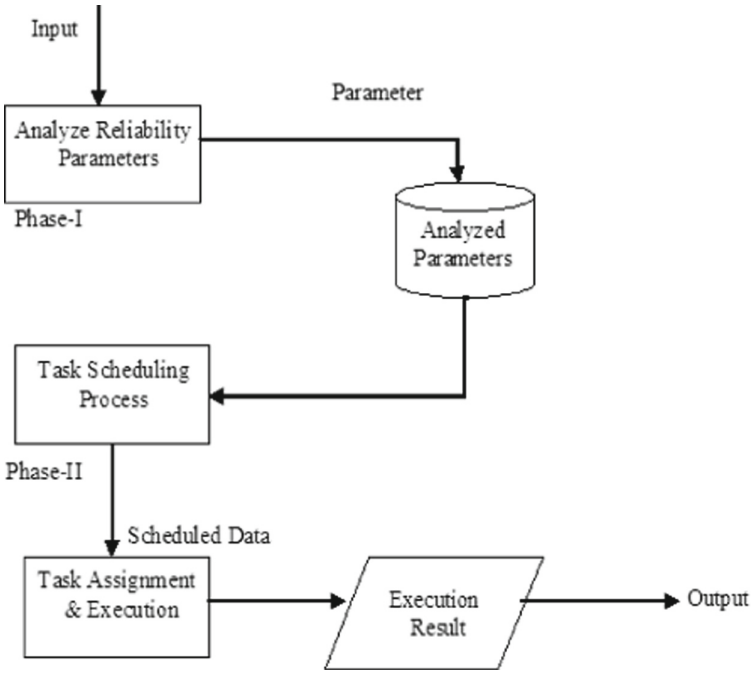

Fig. 4. Phases of task scheduling in Wireless Distributed Systems.

3. Task Assignment & execution

The first phase is the *System Analysis* phase. The input to this phase is a DAG. Depending on the number of active hosts, which is stored in a *hostfile*, this phase evaluates the various parameters necessary to compute the reliability and scheduling of the submitted job. The output of this phase is stored in a file.

The second phase is the *Task Scheduling* phase. The input to this phase is a DAG which is given as input to ASAP/ALAP. These algorithms produce as output, the various levels and the dependency among the tasks at each level. Depending on the availability of existing and active processors/systems, tasks are then scheduled/ assigned to various processors in the wireless systems by combining the reliability using RBTSA and discharge power factors.

The third phase is the *Task Assignment & Execution* phase. This phase counts the number of tasks in each level, sends each task to the identified processor for execution and evaluates the total execution time for the submitted job. The algorithms for the 3 phases are given below:

Phase-1
Compute System Analysis()

1. Read the DAG file.
2. Identify active hosts/systems and processors.
3. Evaluate the communication link failure.
4. Evaluate computation time of ***n*** tasks on ***m*** processors.
5. Evaluate communication delay for each processor.
6. Evaluate the failure rate of each processor.
7. Display the parameters computed from steps 2 to 6.
8. Store the parameters in step 7 into a file for next phase.

Phase-2
Reliability Based Distributed Task Scheduling Algorithm-RBDTSA
Compute Task Scheduling ()

Takes as input the various analyzed parameters & DAG.

1. Schedule ASAP or ALAP on the input DAG and compute dependency levels.
2. Execute fragmentation on the output of ASAP/ALAP.
   a. Read the number of active processors.
   b. Depending on the number of active processors, split the created level, keeping only as many processes in the split as the number of active processors.
3. Read various parameters of system analysis.
4. Compute discharge power Dp_k for each active processor k.
5. Compute reliability cost RC matrix using Dp_k.
6. Send task to the identified processor on the basis of reliability cost matrix in 5.
7. Schedule tasks according to processors identified in step 6.

Phase-3
Compute Assignment-Execution ()

1. Depending on the number of active processors, the scheduling strategy, count the number of tasks assigned to each processor after fragmentation.
2. Send these tasks to each slave for execution.
3. Read the HM, MI, DT values for each task and send them to the respective processor.
4. Depending on DT & MI, the corresponding task will execute. Record the execution time of each slave.
5. Calculate the total execution time of the submitted job.
6. Display ET as output.

The output of testing and execution are discussed in the next section.

3 Results

The WDS was tested with 06 machines-03 wired, 03 wireless so that heterogeneity could be included. The configuration included is listed below.

System configuration:

Machine Configuration						
	M1	M2	M3	M4	M5	M6
Cores	4	4	2	4	4	4
Speed per Processor (GHz)	3.2	3.07	2.2	3.2	3.2	3.2
RAM (GB)	2	2	1	2	2	2

Number of Machines = 6 Total processors = 22.
Number of Slaves = 21 Master = 1 No of nodes in graph = 10

The testing has been carried out on graphs with 10, 20, 30, 40, 50, 100, 150 & 200 nodes (tasks). However, the results have been discussed only on the basis of a graph with 10 nodes. The first phase produces values for parameters like computation time, communication delay, link failure rate, failure rate, etc. After the first phase, the task scheduling is carried out. In this phase the ASAP/ALAP are used to find the dependency levels among tasks and fragmentation is carried to limit the tasks in each level to the number of processors. At the same time, the reliability of each task on each processor is also evaluated. The matrices of evaluated parameter values have not been included here since 22 processors were used, which resulted in large matrix size. A sample graph is shown in Fig. 5 and its task dependency using ASAP/ALAP is given below. The output of fragmentation is also listed for an overview.

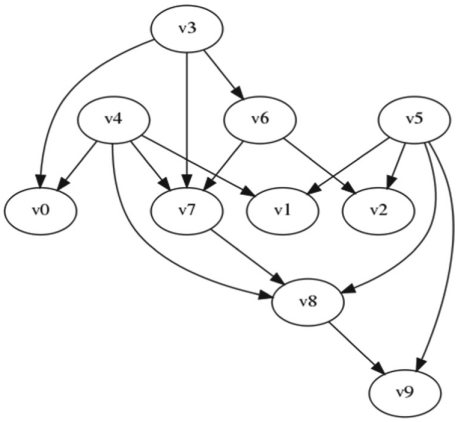

Fig. 5. An input DAG of 10 tasks

ASAP Time Matrix:

nVertex	nTask	Scheduled Time
0	1	2
1	2	2
2	3	3
3	4	1
4	5	1
5	6	1
6	7	2
7	8	3
8	9	4
9	10	5

ALAP Time Matrix:

nVertex	nTask	Scheduled Time
0	1	5
1	2	5
2	3	5
3	4	1
4	5	2
5	6	3
6	7	2
7	8	3
8	9	4
9	10	5

Output of fragmentation:

Fragmented Time Matrix:

Level	Task(s)	No. of Processors
1	4 5 6	3
2	1 2 7	3
3	3 8	2
4	9	1
5	10	1

The total time required to evaluate various parameters for reliability cost calculation for various graphs is given in the Table 1 and its graphical representation in Fig. 6

As shown in Fig. 7 and Table 2 it can be seen that the reliability cost of the proposed scheduling strategy is lower as compared to the other scheduler implemented. Also, it can be clearly stated from the shape of the representation that as the number of tasks increase the graph would form a conical structure wherein the reliability cost of random scheduler is more prone to increase indicating less reliable scheduling. Figure 8 gives the bar graph representation of Table 2.

The makespan values of the two algorithms have also been compared in Table 3 and indicate that makespan of proposed RBDTS algorithm for scheduling a DAG is better as compared to RS. The reliability values of existing implemented methods can also be compared as shown in Table 4. However there are wide variations in the way reliability is referred in the existing and proposed techniques. Most of the existing methods use

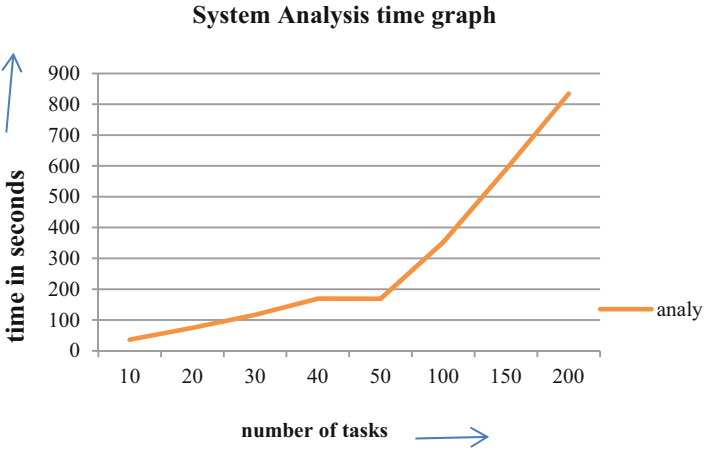

Fig. 6. Plot of time required for system analysis for varying nodes/tasks

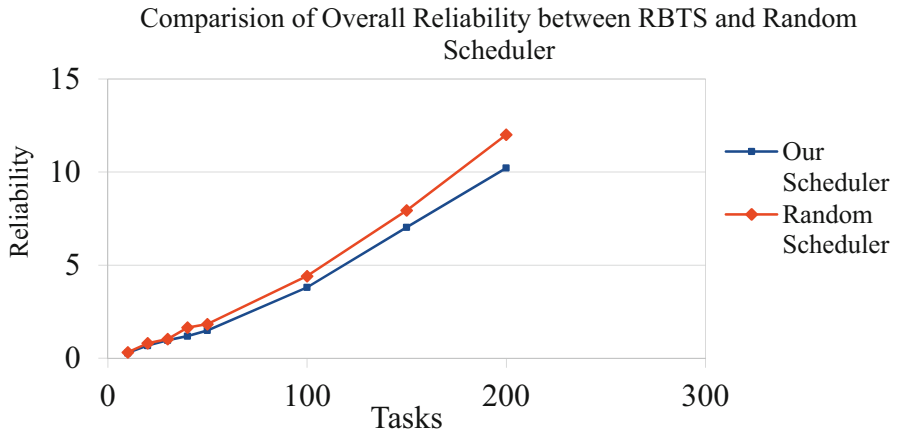

Fig. 7. Comparison of Overall Reliability between the RBTS and Random Scheduler (line graph)

simulation and have assumed values. Also a large number of methods are first assigning tasks and then evaluating reliability whereas the proposed method first evaluates reliability of processors and then assigns tasks Fig. 9.

It can be clearly seen that the overall reliability values for different set of tasks higher as compared to the two implemented methods (RBDTS, RS). The higher the reliability, the less reliable the system is. Thus it can be clearly said that the proposed method is a reliable method for task scheduling. A graphical comparison of the reliability values of Table 4 is shown in Fig. 10.

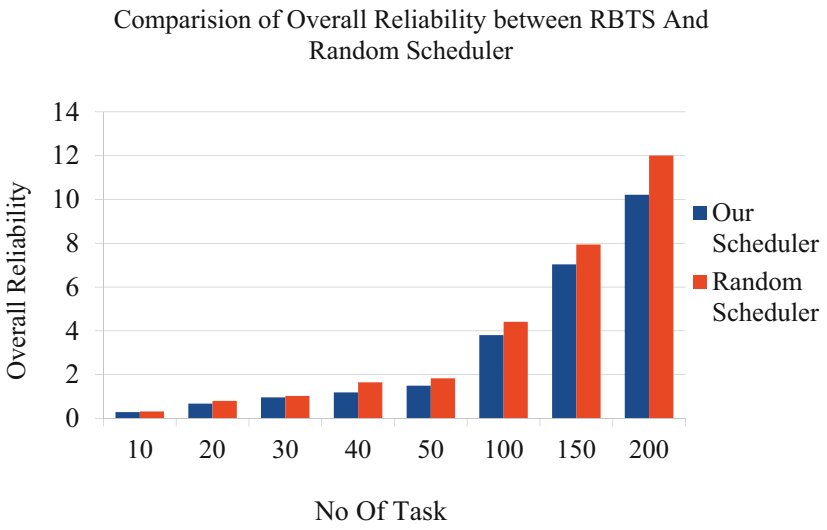

Fig. 8. Comparison of Overall Reliability between the RBTS and Random Scheduler (bar graph)

Table 1. System Analysis

Task	Time (sec)
10	36.2988
20	74.347
30	116.704
40	169.099
50	168.824
100	352.258
150	588.1
200	834.502

Table 2. Comparison of Scheduling

Task	Our Scheduler(RBTS)		Random Scheduler	
	Overall Reliability	Total Makespan	Overall Reliability	Total Makespan
10	0.283821	0.912037	0.317246	0.877697
20	0.677561	1.9276	0.801174	1.93772
30	0.962152	2.38857	1.0264	2.60701
40	1.18722	4.103537	1.64644	6.93073
50	1.48946	3.86138	1.82865	3.8406
100	3.80217	7.47043	4.41203	7.65011
150	7.03167	12.4979	7.94194	11.6151
200	10.2187	15.8561	12.008	16.1109

Table 3. Comparison of makespan values of the two schedulers

Task	Our Scheduler (RBDTS)	Random Scheduler
	Total Makespan	Total Makespan
10	0.912037	0.877697
20	1.9276	1.93772
30	2.38857	2.60701
40	4.103537	6.93073
50	3.86138	3.8406
100	7.47043	7.65011
150	12.4979	11.6151
200	15.8561	16.1109

The earlier implementations have used ASAP, ALAP for scheduling and compared the values of these standard methods with their proposed method-RCD. In the proposed work, RS and RBDTS have been implemented. RS uses a random scheduling strategy whereas RBDTS is the proposed reliable task scheduling method. It can be clearly stated that RBDTS algorithm when applied for task scheduling, gives a more reliable task schedule since the overall reliability values of RBDTS over a range of tasks is significantly less as compared to other methods thus indicating that the wireless distributed system is more reliable Fig. 11.

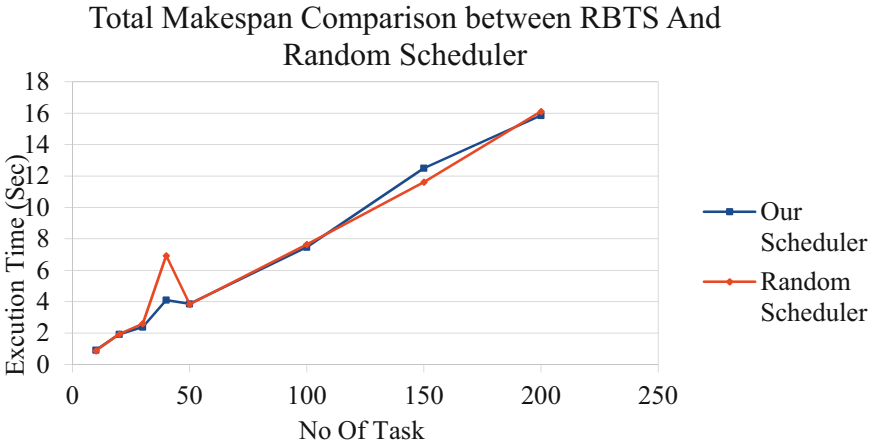

Fig. 9. Comparison of make-span of Random & Our Scheduler (RBDTS)

Table 4. Comparison of reliability values of existing and proposed implementation

No. of Tasks Reliability evaluation methods		10	30	50	70	90
Existing	**ASAP**	10.04	29.14	46.58	64.21	79.56
	ALAP	8.95	23.17	35.58	45.75	54.58
	RCD	0.83	2.5	4.18	5.82	7.44
	RS	0.31725	1.0264	1.82865	---	---
Proposed	**RBDTS**	0.28382	0.96215	1.48946	2.05524	2.60572

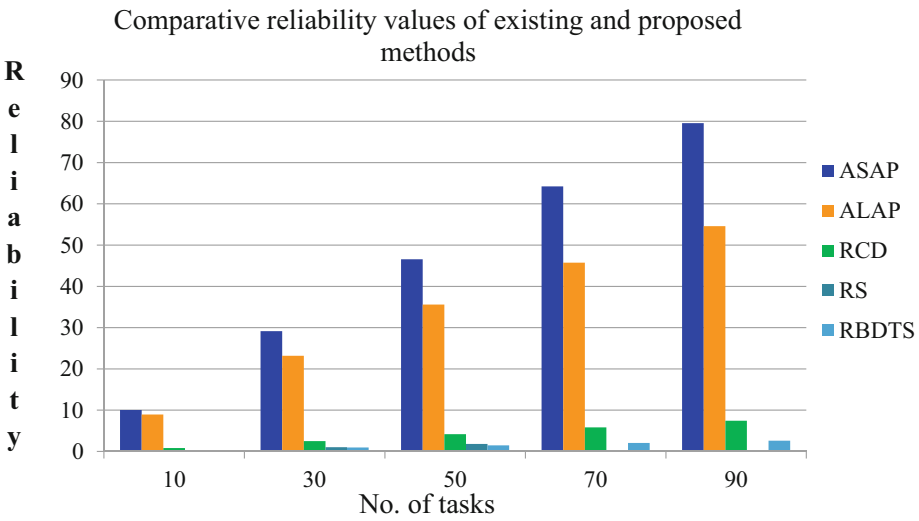

Fig. 10. Percentage reduction in the utilization of resources required during scheduling

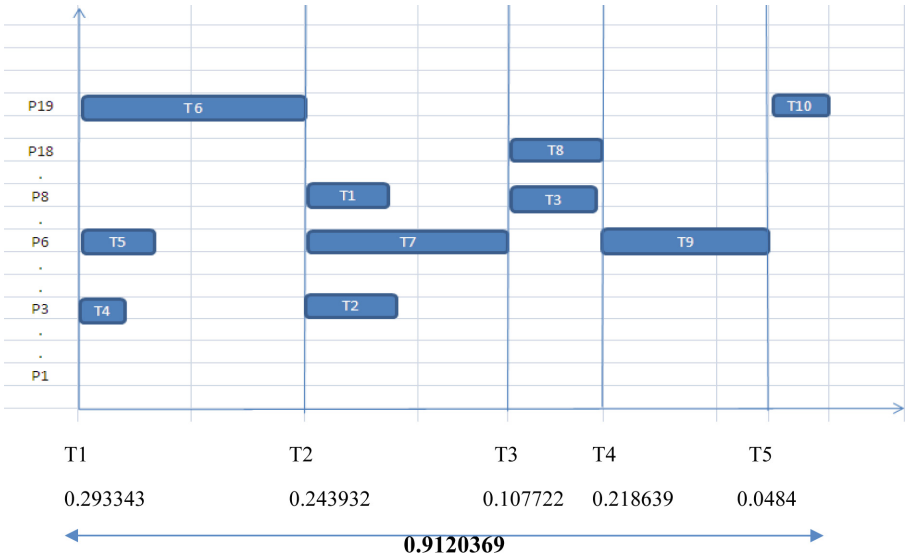

Fig. 11. Total Makespan in terms of task allocation & execution sequence.

4 Discussion

From the above sections, it can be concluded that till recent years, most research work in the area of real-time task scheduling in distributed systems was based on the following considerations:

1. It did not consider reliability issues prior to scheduling.
2. It only considered homogeneous systems, or assumed independent tasks.
3. Power discharged wasn't an issue, since wired networks were considered.

In this proposed research, an attempt has been made to address all of these issues by implementing a reliability cost driven algorithm that schedules real-time tasks with precedence constraints in a heterogeneous distributed system.

In the proposed algorithm, called RBDTS, reliability cost is used as one of the objective functions for scheduling tasks. Two other algorithms, ASAP and ALAP, that do not consider reliability cost, have been used here for finding the dependency among tasks. However, they haven't been mentioned here since they have implemented without any modification. The focus in this work was totally on a reliable task scheduling method for which two strategies have been implemented and explained and their results have been compared. These algorithms have been used here for comparison purposes. Simulation results show that RBDTS is significantly better than either ASAP, ALAP, RCD, random scheduling or list scheduling in terms of the system reliability cost and make-span of processors which are a measure of schedulability. Further, the results suggest that higher computational heterogeneity is favorable for improving RBDTS's schedulability, while the opposite is true for ALAP (and most likely for ASAP as well).

5 Conclusion

The proposed algorithm evaluates system reliability and accordingly generates a schedule for the tasks of a particular program/job submitted in the distributed system. Here a dynamic wireless distributed system has been assumed so as to justify the real purpose of scheduling. Also, two algorithms of a normal distributed system are tested by executing in wireless distributed systems.

However, the newly proposed algorithm could not be compared since strategies related to reliable scheduling in wireless distributed systems are unavailable. The methods applied earlier to a normal distributed system and their results are compared here with the two new approaches used for comparison.

In conclusion, the main features of the design proposed in the current research work can be summarized as:

- The proposed strategy considers parameters of normal as well as wireless distributed systems.
- 02 existing algorithms of a normal heterogeneous system have been used for scheduling and the results for reliability can be compared using 2 existing strategies for this proposed algorithm in the wireless distributed system.
- The newly designed algorithm aims to give a better scheduling strategy after being tested for existing algorithms.
- Scheduling using ASAP gives better results as compared to ALAP in WDS in terms of reliability cost.

Data Availability Statement. The task graphs, processor timings and other statistics used to support the findings of this study are available from the corresponding author upon request. (email-sonali.mahure@gmail.com).

References

1. Carpenter, J., Funk, S., Holman, P., Srinivasan, A., Anderson, J., Baruah, S.: A categorization of real-time multiprocessor scheduling problems and algorithms. In: Handbook on Scheduling Algorithms, Methods, and Models. Chapman Hall, CRC, Boca (2004)
2. Raymond Clark, E., et al.: An adaptive, distributed airborne tracking sysem: Process the right tracks at the right time. Presented at the (1999). https://doi.org/10.1007/BFb0097917
3. Tan, T.G., Hsu, W.: Scheduling multimedia applications under overload and nondeterministic conditions. IEEE RTSS, **0**, 178 (1997)
4. Vallidis, N.M.: Whisper: a spread spectrum approach to occlusion in acoustic tracking. PhD thesis at the University of North Carolina at Chapel Hill (2002)
5. Beaumont, O., Legrand, A., Robert, Y., ENS Lyon, L. Carter., Ferrante, J.: Bandwidth-Centric Allocation of Independent Tasks on Heterogeneous Platforms. In: Proceedings International Parallel and Distributed Processing Symposium (2002)
6. Doğan, A., Özgüner, F.: Reliable matching and scheduling of precedence-constrained tasks in heterogeneous distributed computing. In: Proceedings of the International Conference on Parallel Processing, pp. 307–314 (2000)
7. Kebbal, D., Talbi, E.G., Geib, J.M.: Building and scheduling parallel adaptive applications in heterogeneous environments. In: Proceedings of the IEEE International Workshop Cluster Computing, pp.195–201 (1999)

8. Radulescu, A., van Gemund, A.J.C.: Fast and effective task scheduling in heterogeneous systems. In: Proceedings of the Euromicro Conference on Real-Time Systems, pp. 229–238 (2000)
9. Tang, X.Y., Chanson, S.T.: Optimizing static job scheduling in a network of heterogeneous computers. In: Proceedings of the International Conference on Parallel Processing, pp. 373–382 (2000)
10. Kalogeraki, V., Melliar-Smith, P.M., Moser, L.E.: Dynamic scheduling for soft real-time distributed object systems. In: Proceedings of the IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp.114–116 (2000)
11. Lundqvist, T., Stenstrom, P.: Timing anomalies in dynamically scheduled micromachines. In: Proceedings of the IEEE Real-Time Systems Symposium, pp.12–21 (1999)
12. Palis, A.: Online Real-Time Job Scheduling with Rate of Progress Guarantees. In: Proceedings of the 6th International Symposium on Parallel Architectures, Algorithms, and Networks, Manila, Philippines, pp. 65–70 (2002)
13. Beaumont, O., Boudet, V., Robert, Y.: Realistic Model and an Efficient Heuristic for Scheduling with Heterogeneous Processors. In: Proceedings of the 11th Heterogeneous Compting Workshop (2002)
14. Doğan, A., Özgüner, F.: Reliable matching and scheduling of precedence-constrained tasks in heterogeneous distributed computing. In: Proceedings of the International Conference on Parallel Processing (ICPP), Toronto, Canada, August 21–24, pp 307–316 (2000)
15. Braun, T.D., Siegel, H.J., Maciejewski, A.A.: Static Mapping Heuristics for Tasks with Dependencies, Priorities, Deadlines, and Multiple Versions in Heterogeneous Environments. In: Proceedings of the International Parallel and Distributed Processing Symposium (2002)
16. Qin, X.: Hong Jiang, A dynamic and reliability-driven scheduling algorithm for parallel realtime jobs on heterogeneous clusters. J. Parallel Distrib. Comput. **65**(8), 885–900 (2005)
17. Raj, J.S., Vasudevan, V.: Intelligent Reliable Schedule with Budget Constraints in Grid Computing. Res. J. Appl. Sci., Eng. Technol. **7**(4), 650–655 (2014)
18. Kalogeraki, V., Melliar-Smith, P.M., Moser, L.E.: Dynamic scheduling for soft real-time distributed object systems. In: Proceedings of the IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp.114–121 (2000)
19. Bodkhe, S.T., Mahajan, A.R.: Reliability Aware Task Scheduling In: Wireless Hetrogeneous Systems, published in International Journal of Computer Applications, ISSN: 0975–8887, Vol. 140 – No.8, April 2016, pp. 20–26 (2016)
20. Zomayya, A.Y., Ward, C., Macey, B.S.: Genetic scheduling for parallel processor systems: comparative studies and performance issues. IEEE Trans Parallel Distrib Syst **10**(8), 795–812 (1999)
21. Dai, Y.-S., Levitin, G., Trivedi, K.S.: Performance and reliability of tree-structured grid services considering data dependence and failure correlation. IEEE Trans. Comput. **56**(7), 925–936 (2007)