KINETIK, Vol. 4, No. 1, February 2019, Pp. 45-54
ISSN : 2503-2259
E-ISSN : 2503-2267

A Modified Real-Time Fault-Tolerant Task Allocation Scheme for Wireless Sensor Networks

F. F. Marshall[*1], M. B. Mu'azu[2], I. J. Umoh[3], A. T. Salawudeen[4], B. O. Sadiq[5], D. E. Ikpe[6]
[1,2,3,4,5,6]Ahmadu Bello University/Department of Computer Engineering
frankdidam14@yahoo.com[*]

Abstract

In WSNs, the sensor nodes are at risk of failure and malicious attacks (selective forwarding). This may have a profound negative effect when you consider real-time WSNs, making them challenging to deploy. When there is a delay in tasks allocation execution processes in real-time WSNs because of sensor nodes failures, this will cause disastrous consequences if the systems are safety-critical, e.g. aircraft, nuclear power plant, forest fire detection, battlefield monitoring, thus the need to developed a real-time system that is fault-tolerable. This paper developed a modified real-time fault-tolerant task allocation scheme (mRFTAS) for WSNs (wireless sensor networks), using active replication techniques. mRFTAS and RFTAS performance were compared using time of execution of the task, network lifetime and reliability cost. The mRFTAS performance showed an improvement over that of RFTAS when it comes to reducing the time it takes for task execution by 45.56% and reliability cost of 7.99% while prolonging the network lifetime by 36.35%.

Keywords: *WSNs, Real-time, Fault-tolerant, Task Allocation*

1. Introduction

A WSN comprises of numerous resource-constrained sensor nodes that, are frequently deploy in unattended environments. Consequently, the sensor nodes are at risk of failure and/or malicious attacks. The failed nodes can have a profoundly negative effect on real-time WSNs. WSNs have a major constraint, which is the low power consumption requirement of sensor nodes[1][2]. Parallel processing between sensor nodes is an innovative technology, which supports the essential computation capacity in WSNs while ensuring low power consumption by the sensor nodes.

A key issue to consider in real-time systems is the time delay in task processing (which can arise from issues associated with nodes failures). It is crucial that network failures are identify early, and the proper actions taken, to ensure network reliability and network lifetime is prolong.

Task allocation plays a significant role in parallel processing. Assigning a task to the suitable sensor nodes and simultaneously balancing the network load in the context of the uncertain and dynamic network environments are essential to parallel processing[1]. In WSNs, the issue of task allocation involves assigning tasks logically within sensor nodes, to reduce the general power utilization while the tasks are complete before the set deadlines, thus prolonging the lifetime of the sensor network [2][3][4]. Load balancing is an essential factor for prolonging the network lifetime. In the absence of proper task allocation techniques, every sensor node will work independently of others [5]. WSNs have challenges such as instability of wireless communication links and dynamically changing topologies, there are potentially additional uncertainties and vulnerabilities for real-time applications. A sensor node failure should not necessarily affect the entire tasks processing of the sensor network, especially for safety and security critical applications [6].

In order to sustain the performance of the sensor networks without interruption due to failures of the sensor node, fault tolerance turns out to be a valuable initiative. For instance, if sensor nodes are being deploy in a battlefield or military-camp for surveillance and detection, the fault tolerance has to be high because the sensed data are critical for security and safety reasons. Hence, a fault-tolerant mechanism is mandatory for such safety and time-critical applications [7] [8].

Marshall, F., Mu'azu, M., Umoh, I., Salawudeen, T., Sadiq, B., & Ikpe, D. (2018). A Modified Real-Time Fault-Tolerant Task Allocation Scheme for Wireless Sensor Networks. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 4(1). doi:http://dx.doi.org/10.22219/kinetik.v4i1.659
Receive June 11, 2018; Revise August 03, 2018; Accepted August 30, 2018

The replication backup techniques which involves the use of primary/backup (P/B) system is the widely used technique for fault-tolerant task allocation as it tolerates copies of a task to be processed on separate sensor nodes.

There is a need for a real-time WSN that is fault-tolerant and safety-critical, thus the need for the development of a modified real-time fault-tolerant task allocation scheme (mRFTAS), such as to avoid the breakdown of a system due to failure of some sensor nodes.

Literature reviewed show that research has been carried in the area of task allocation/scheduling schemes, and fault tolerant task allocation scheme in WSNs. Successful task allocation execution in systems that are real-time with little or no delay in task processing time, reduced task waiting time is still an open-ended research area for real-time WSNs.

The real-time fault-tolerant task allocation scheme (RFTAS) is the kind of scheme that is used to prevent system failure or system breakdown and has mostly employed the passive replication backup technology [9]. However, the passive replication scheme has the problem of delay in task processing time. Delay in task processing time can be disastrous for real-time systems that are critical in term of safety.

This research developed a modified real-time fault-tolerant task allocation scheme (mRFTAS) using the active replication backup scheme, to address the issue of delay in task processing time associated with the RFTAS.

The modified real-time fault-tolerant task allocation scheme will help a system to keep operating even in the presence of some failures, inevitably reducing the total time of an entire task allocation process, thus prolonging the network lifetime.

2. Research Method

2.1 Total Time of Task Execution

The total execution time to finish all tasks depends on the time of execution the primary copies and the active backup copies, execution time for *m* tasks and can be defined Equation 1 [2].

$$T = \sum_{i=1}^{m} et_{i,p(t_i^P)} + \sum_{i=1}^{m} S(t_i^B) \times \left(t_i^B - lst_{i,p(t_i^B)}\right) \quad (1)$$

Where, *T* is time, $S(t_i^B)$ represents the backup mode (t_i^B). If l_i is not less than the ($et_{i,p(t_i^B)}$), t_i^Bneed not to be execute before (f_i^P) for fault-tolerant. Even $n_{S(t_i^P)}$ fails before the time f_i^P , t_i^B also can be finished before d_i. Thus, it uses passive mode. Otherwise, t_i^B will be carry before f_i^P for fault-tolerant. Thus, it employs active mode. Let $S(t_i^B)$ is equal 1, which means t_i^B uses active backup mode. Otherwise, t_i^B uses passive mode. $S(t_i^B)$ will also mean as Equation 2 [2].

$$S = \begin{cases} 1, & if\left(l_i(t_i^B) < et_{i,p(t_i^B)}\right) \\ 0 & else \end{cases} \quad (1 \le i \le m) \quad (2)$$

Generally, primary copy when executed successfully, the corresponding backup copy needs not to execute anymore.

2.2 Reliability

Reliability in WSNs is the probability that none of real-time tasks will failed due to sensor nodes failures in the network. Therefore, the reliability model is given similar to Equation 3,4,5 [2].

$$RC = 100 \times \left(RC(zp) + RC(zb)\right) \quad (3)$$

$$RC(zp) = \sum_{i=1}^{m} \lambda_{p(t_i^P)} \times et_{i,p(t_i^P)} \quad (4)$$

$$RC(zp) = \sum_{i=1}^{m} S(t_i^B) \times \lambda_{p(t_i^P)} \times \left(f_i^P - lst_{ip(t_i^B)}\right) \quad (5)$$

$$Rel = \exp(-RC) \quad (6)$$

where *Rel* denotes the reliability in a network, RC denotes reliability cost, *RC(zp)* and *RC(zb)* are the total reliability costs of primary copy and a copy used for backup of *m* tasks, and λ_i denotes the failure probability of n_i. Take Figure 1 as an example, if the failure probability of n_2 is relatively large and it also needs to work for some time; this clearly shows that system reliability will surely decline. Thus, network reliability decreases with the increase of the reliability cost as represented in Equation 6.

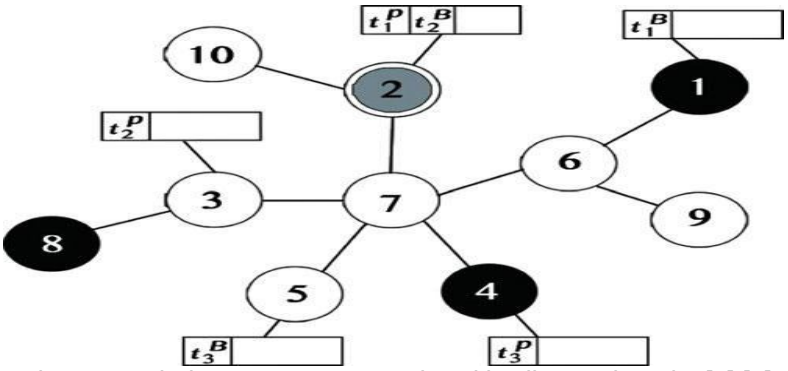

Figure 1. Wireless Sensor Networks with Allocated Tasks [2] [5]

Generally, let *m* tasks be allocated satisfactorily to *n* sensors in order to minimize task execution time, balance the network load, prolong the network lifetime, certify that the task would not fail by unexpected failure of nodes and improve the dependability of task supervision [3].

2.3 Network Lifetime

Task allocation is an optimization problem that can be formalized as seen Equation 7 [2].

$$Maximize(it)\ S.T. = \begin{cases} \sum_{i=1}^{n} y_i \le num \times 20\% \\ \forall t_i T, n_{p(t_i^P)} \neq n_{p(t_i^P)} \end{cases} \quad (7)$$

where *lt* denotes WSNs lifetime and *num* symbolizes the number of sensor nodes total used. $y_i = 1$ shows that node n_i is a failure and $y_i = 0$ indicate that node n_i is normal. Assume that WSNs are paralyzed when a number of failure nodes exceed 20% of the total number. For example, if n_1, n_4, and n_8 break down in Figure 1, then $\sum_{i=1}^{10} y_i = 3$ is larger than *num*×20%= 2, the network becomes paralyzed.

2.4 Calculation Process of the Earliest Start Time of the Task's Primary Copy

Let's assume there is a task t_j, the primary start time of t_j^P which is designated to node n_i can be calculated, we examine every node's idle time space $\left[0, S_1^{'}\right], \left[f_1^{'}, S_2^{'}\right], \left[f_2^{'}, S_3^{'}\right], ..., \left[f_n^{'}, +\infty\right]$ from left to right, where $S_i^{'}$ and $f_i^{'}$ stands for the start time and finish time of the *ith* task in the task queue of n_i, respectively. For predictable expression, let $f_0^{'} = 0$, In the event that the first idle time space $[f_k^{'}, S_{k+1}^{'}]$, which can meet $\max\left(a_j, f_k^{'}\right) + et_{j.i} \le d_j$, is discovered, the earliest start time of t_j^P on node n_i would be noted as $est_{ji} = f_k^{'}$ generally est_{ji} would be noted as $+\infty$.

2.4.1 Calculation Process of the Latest Start Time of the Task's Backup Copy

Let us assume there is a task t_j, the primary start time of t_j^B which is apportioned to node n_i can be determined as follows:

1. Step 1
   Initialize time $space = \left[d_j - et_{ji}, d_j \right]$, for comfort, the start time and finish time of space can be set apart as s_start and s_finish, i.e. $space = \left[s\_begin, s\_find \right]$.
2. Step 2
   Scan the task queue of n_i. If the *slot* is set in its idle times, let lst_{ji} be s_start and after that, the procedure is finished. Else, it implies that space overlaps with another task in the line, hence, stamp the overlapped task as t_x for further consideration. In any case, if t_x is a primary copy or an active backup copy, update space with $s\_finish =$ the start time of t_x, $s\_start = s\_finish - et_{ji}$. And after that, if $s\_finish < 0$, let lst_{ji} be $+\infty$ and the procedure is finished, else goes to Step 2 once more.

2.4.2 Allocation Process of the Task's Primary Copy

Plan a utility function $U^P(i,j)$ to quantify the comprehensive execution of every node processing t_j^P and allot t_j^P to a superior node in Equation 8 [2].

$$U^P(i,j) = wt_1 \times UB(i,j) + wt_2 \times UE(i,j) + wt_3 \times UR(i,j), \quad (8)$$

where wt_1, wt_2 and wt_3 are weight coefficient, $U^P(i,j)$ is the utility function of primary copy, the slighter the estimation of $U^P(i,j)$ is, the better n_i execute t_j^P completely. $UB(i,j)$, $UE(i,j)$ and $UR(i,j)$ signify the load degree, energy utilization degree and failure proportion level of n_i which executing t_j^P contrasted and different nodes which participate in t_j The present load b_i, energy utilization $ene_{j,i}$ and failure proportion λ_i of n_i are mapped in the scope of 0 and 0.5 by utilizing a comparable *sigmoid* function as information standardization function to achieve $UB(i,j)$, $UE(i,j)$ and $UR(i,j)$. The specific estimation equations are presented by the following Equation 9, 10, 11 [2].

$$UB(i,j) = \begin{cases} 0, & if \quad (b_{max} - b_{min}) = 0 \\ \dfrac{1}{e - \dfrac{x_{ji} \times b_i - b_{min}}{b_{max} - b_{min} + 1}} - 0.5, & else, \end{cases} \quad (9)$$

$$UE(i,j) = \begin{cases} 0, & if \quad (ene_{max} - ene_{min}) = 0 \\ \dfrac{1}{e - \dfrac{x_{ji} \times ene_i - ene_{min}}{ene_{max} - ene_{min} + 1}} - 0.5, & else, \end{cases} \quad (10)$$

$$UR(i,j) = \begin{cases} 0, & if \quad (\lambda_{max} - \lambda_{min}) = 0 \\ \dfrac{1}{e - \dfrac{x_{ji} \times \lambda_i - \lambda_{min}}{\lambda_{max} - \lambda_{min} + 1}} - 0.5 \end{cases} \quad (11)$$

where $b_{\max}$ and $b_{\min}$ indicate the high load and the lightest load of nodes which take an interest in t_j, separately, the smaller the $UB(i,j)$, the lighter the load of n_i when executing t_j^P. $ene_{\max}$, and $ene_{\min}$ signify the biggest and the slightest energy utilization of nodes which take

an interest in t_j, separately. The smaller the *UE (i,j),* the less the energy utilization of n_i when executing t_j^P. $\lambda_{\max}$, and $\lambda_{\min}$ mean the biggest and the minimum failure proportion of nodes which take an interest in t_j, separately. The two steps can present the specific allocation procedure of a task's primary copy:

1. Step 1.
   For every node n_i which takes part in t_j, calculate the total of est_{ji}, and et_{ji}, for foreseeing time required of t_j^P to finished on nodes n_i. In the event that the deadline requirement of t_j is met, $U^P(i,j)$ will be calculated by Equation 8. Generally, $U^P(i,j)$, will be noted $+\infty$ until each one of those nodes has taken into consideration.
2. Stage 2.
   Select a node with the less $U^P(i,j)$, and afterwards allot t_j^P to the chose node and update it's comparing $U^P(i,j)$ and $UB(i,j)$ for next estimation convenience.

2.4.3 Allocation Process of the Task's Backup Copy

1. Step 1
   There exists a backup copy. For every node n_i which takes part in t_j, compute the total of lst_{ji} and et_{ji} to determinate the perform method of reinforcement copy and forecast the completion time of t_j^B on node n_i. In the event that the deadline requirement of t_j^B is met, $U^B(i,j)$ will be computed using Equation 12, else, $U^B(i,j)$ will be observed as $+\infty$ pending when all nodes are taken into account.
2. Step 2
   Choose a node with the least value of $U^B(i,j)$, assign t_j^B to the chosen node and update the relating $U^B(i,j)$ and $UB(i,j)$ [2].

$$UE'(i,j) = \begin{cases} 0, & if \quad (enér_{max} - enér_{min}) = 0 \\ e - \dfrac{1}{\dfrac{x_{ji} \times S(t_j^B) \times ene_{ji} \times per_{ji} \times enér_{ji}}{enér_{max} - enér_{min} + 1}} - 0.5 \end{cases} \tag{12}$$

$$U^{B'}(i,j) = wt_1 \times UB'(i,j) + wt_2 \times UE'(i,j) + wt_3 \times UR'(i,j), \tag{13}$$

Presented by Equation 13, where wt_1, wt_2 and wt3, are weight coefficients, $U^B(i,j)$ is the utility function of reinforcement copy. The slighter the quantity of $U^B(i,j)$ is, the better n_i execute t_j^B widely. $ene'_{\min}$, and $ene'_{\max}$ represent the minimum and the maximum energy utilization of the nodes which take part in t_j, correspondingly. $ene'_{\min}$ and $ene'_{\max}$ can be estimated by the following Equation 14 and 15 [2].

$$enér_{min} = \min_{i=1}(x_{ji} \times S(t_j^B) \times ene_{ji} \times per_{ji}) \tag{14}$$

$$enér_{min} = \max_{i=1}(x_{ji} \times S(t_j^B) \times ene_{ji} \times per_{ji}) \tag{15}$$

2.5 The Modified Real-Time Fault-Tolerant Task Allocation Schemes

The mRFTAS employed a modified form of Particle Swarm Optimization (PSO) called discrete Particle Swarm Optimization (dPSO) for the generation of particle position and velocity, this also optimizes the parameter metrics.

PSO is patterned according to the communal behaviour of a flock of birds [10]. It comprises of a swarm of *s* candidate solutions called particles, which search an n-dimensional hyperspace in the exploration of a global solution (*n* represents an optimal number of parameters to be determined). A particle *i* accepts a position of X_{id} with a velocity V_{id} in the d^{th} which is a hyperspace dimension, $1 \cdot i \cdot s$ and $1 \cdot d \cdot n$. In an objective function every single particle is estimated $f(x_1; x_2; \ldots; x_n)$, where $f: R^n \rightarrow R$. Therefore, its cost (fitness) for every particle near its global solution is lesser (higher) when you consider any particle that is farther away. PSO succeeds to minimize (maximize) (fitness) cost function [10]. Global-best version of PSO considers positions of particle *i* has its lowest cost which is stored as $(pbest_{id})$. Also, it's worth knowing that, $gbest_d$, which showcases position particle best used. Each iteration *k*, velocity *V,* and position *X* are updated applying by the following Equation 16 and 17 [10] [11].

$$\mathrm{V}_{id}(k+1) = w\,\mathrm{V}_{id}(k) + \varphi_1 r_1(k)(pbest_{id} - X_{id}) + \varphi_2\, r_2(k)(gbest_d - X_{id}) \quad (16)$$

$$X_{id}(k+1) = X_{id}(k) + V_{id}(k+1) \quad (17)$$

Here, φ_1 and φ_2 are constants, and $r_1(k)$ and $r_2(k)$ are random numbers uniformly distributed in [0,1]. The process which is update iteratively is repeated until acceptable *gbest* is attained or a grand number of iterations k_{max} is gotten[10]. The benefits of PSO over many other optimization algorithms are its ease of implementation and its ability to reach a reasonable good solution quickly [10]. PSO is also more efficient in preserving population diversity to avoid premature convergence issue.

In the PSO method is also employed in mRFTAS scheme in a wireless sensor network environment. The flow chart of the scheme as in Figure 2 shows how a sink node collects tasks firstly, and then it generates a position including its velocity in parallel for one generation of dPSO. Next, it begins a series of operations. After that, it looks at the termination condition whether it has been obtained before the next iteration begins. Otherwise, it publishes tasks and waits. Figure 2 on the right part shows the tasks process execution. If the primary version is finished successfully, no matter the mode of corresponding backup version is active or not, it does not have to be executed [2]. If a backup copy mode is passive, it only executed when there is a failure in the primary copy.

3. Research Results and Discussion

This chapter will discuss the simulation experiment carried out on the real-time WSNs scenarios. The simulations were carry out using visual studio 2015 professional. The sensor network, simulation area was 100 by 100. The number of sensor nodes considered are 100-400 nodes. The total number of tasks considered are 400 tasks. Ten (10) Iteration were obtain and the average of the ten results were obtain for all the parameters under consideration. The obtained results are the plots as shown in subsections 3.1, 3.2 and 3.3.

3.1 Simulation Results of the Task Execution Time of the Sensor Nodes

The task execution time of a task is the time taken for the sensor nodes to execute their given task to completion. The simulation starts with consideration of 100 nodes and with increments of 25 nodes at interval until the last, which is 250 nodes. The simulations were carry concurrently, for the RFTAS and the mRFTAS scenarios, and the results obtained compared by the plots of Figure 3.

Figure 3 shows the comparative plots of tasks execution time against the number of nodes for both RFTAS and mRFTAS. The ranges of nodes considered for the simulations are; 100, 125, 150, 175, 200, 225 and 250 nodes. It can be seen from the graphs, the more the number of sensor nodes employed for the execution of 400 tasks, the lesser the time required by the nodes to complete the entire tasks. It is also evident that the task execution time required by the mRFTAS is less than that required by the RFTAS to complete the given task.

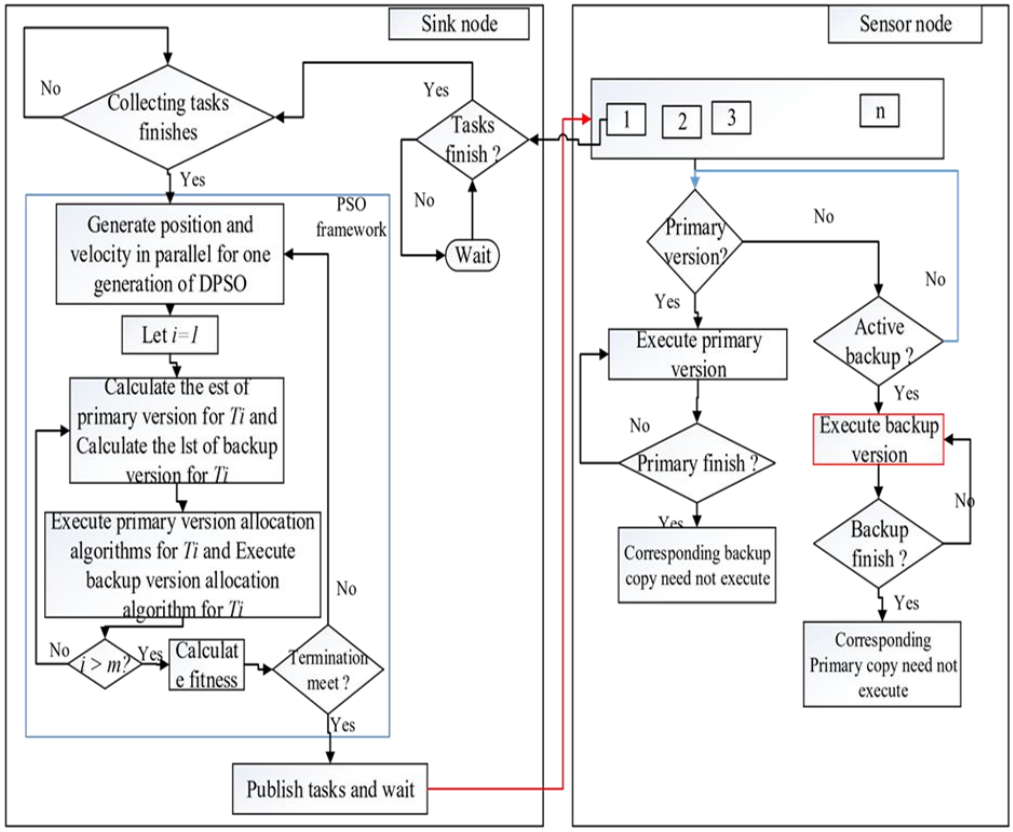

Figure 2. Modified Real-time Fault-tolerant Tasks Allocation Scheme (mRFTAS) Flowchart

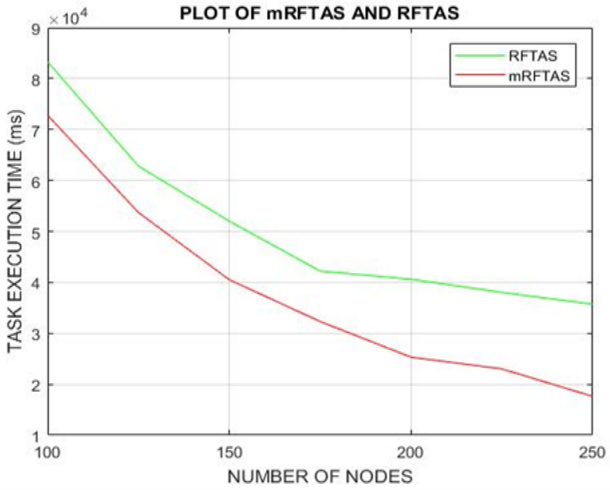

Figure 3. Task Execution Time for mRFTAS and RFTAS

3.2 Simulation Results of the Reliability Cost of the Sensor Networks

The simulation starts with consideration of 100 nodes and with increments of 25 nodes at interval until the last, which is 250 nodes. The simulations were carry concurrently, for the RFTAS and the mRFTAS scenarios, and the results obtained compared by the plots of Figure 4.

Figure 4 is the comparative plot of reliability cost against the number of nodes for the RFTAS and mRFTAS. The ranges of nodes considered for the simulations are; 100, 125, 150, 175, 200, 225 and 250 nodes It can be seen that the more the number of sensor nodes employed for the execution of 400 tasks, the lesser the reliability cost associated with the nodes. It is also evident that the reliability cost associated with the mRFTAS is less than that required by the RFTAS.

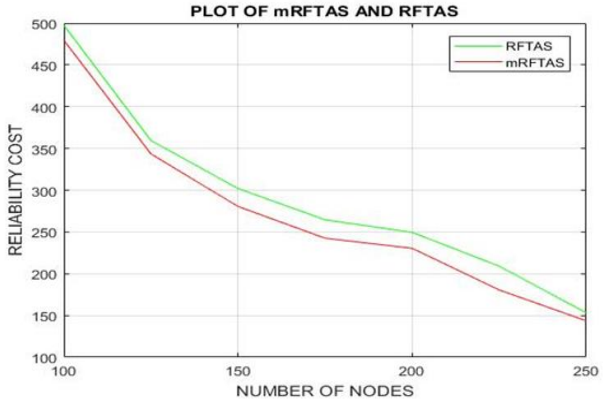

Figure 4. Reliability Cost for mRFTAS and RFTAS

3.3 Simulation Results of the Network Lifetime of the Sensor Nodes

The simulations were carry concurrently, for the RFTAS and the mRFTAS scenarios, and the results obtained compared by the plots of Figure 5.

Figure 5 is the comparative plot of a number of failed nodes against the tasks execution time (s) for the RFTAS and mRFTAS. The ranges of time considered for the simulations are; 50, 55, 60, 65, 70, 75 and 80. It can be seen, the more the time is taken for the execution of the 400 tasks, the more the number of failed nodes. This affects performance and inevitably network lifetime. It is also evident that the mRFTAS had lesser number of failed nodes recorded and invariably a longer network lifetime as compared to the RFTAS.

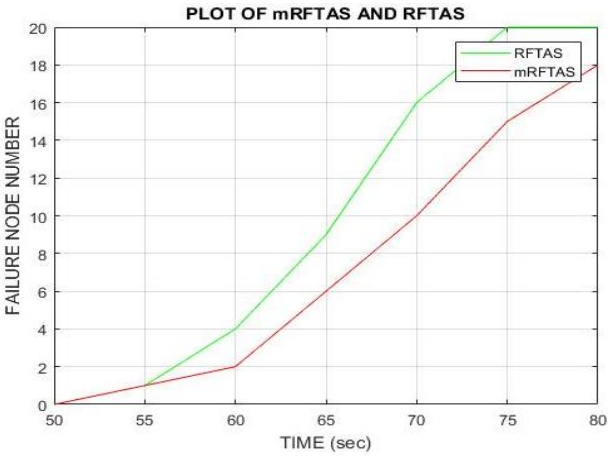

Figure 5. Network Lifetime for mRFTAS and RFTAS

4. Conclusion

This paper developed a mRFTAS using active replication backup technique, for addressing the processing time delay and real-time fault tolerance in WSNs by minimizing the time used for task execution and cost in reliability and maximizing network lifetime. The comparisons of performances for both mRFTAS and RFTAS were carried out for time used for task execution, network lifetime and reliability cost. The results obtained from the experimental simulation showed the developed mRFTAS outperformed the RFTAS. From the research carried out, it been realized that in task allocation backup on a system that is real-time and with time constraints, the modified real-time fault-tolerant task allocation scheme is a better technique to be employed. The mRFTAS performance showed an improvement over that of RFTAS when it comes to reducing the time it takes for task execution by 45.56% and reliability cost of 7.99% while prolonging the network lifetime by 36.35%.

The following areas of further work are recommended for consideration for future research: The research conducted did not put into consideration malicious nodes presence, thus further work can be carry out by implementing security. The modified Real-time Fault-tolerant task allocation scheme can be modified for minimization of energy consumption in WSNs.

Notations

M	= number of tasks
Et	= early start time
i & j	= allocated tasks
lst	= latest start time of backup task
t_i^B	= backup task copy
t_i^P	= primary task copy
f_i^P	= finish time of primary copy of task
d_i	= deadline of t_i
$\mathrm{P}(t_i^B)$, $\mathrm{P}(t_i^P)$	= the nodes t_i^B and t_i^P are allocated
$S(t_i^B)$	= mode of t_i^B
l_i	= laxity of t_i^P , ($l_i = d_i - f_i^P$).
Ene	= energy, *i & j* are allocated tasks

References

[1] C. Chen, W. Guo, and G. Chen, *"A New Task Allocation Algorithm Based on Dynamic Coalition in WSNs,"* 2012 IEEE 26th International Parallel Distributed Processing Symposium Workshops & PhD Forum, Pp. 1243–1248, 2012.

[2] W. Guo, J. Li, G. Chen, Y. Niu, and C. Chen, *"A PSO-Optimized Real-Time Fault-Tolerant Task Allocation Algorithm in Wireless Sensor Networks,"* IEEE Transactions on Parallel Distributed Systems, Vol. 26, No. 12, Pp. 3236–3249, 2015.

[3] W. Guo, Y. Chen, and G. Chen, *"Dynamic Task Scheduling Strategy with Game Theory in Wireless Sensor Networks," New Math. Nat. Comput.*, Vol. 10, No. 03, Pp. 211–224, 2014.

[4] W. Guo, N. Xiong, H.-C. Chao, S. Hussain, and G. Chen, "Design And Analysis Of Self-Adapted Task Scheduling Strategies In Wireless Sensor Networks," Sensors, Vol. 11, No. 12, Pp. 6533–6554, 2011.

[5] P. Suganya and N. Jayanthi, *"Sensor Task Reassignment in Wireless Sensor Networks Using Bio-Inspired Algorithm,"* Int. J. Futur. Innov. Sci. Eng. Res., Vol. 2, No. 1, Pp. 217, 2016.

[6] M. Priyanka, S. Anisha, and S. P. R, *"Vlsi Design for a Pso-Optimized Real-Time Fault-Tolerant,"* Vol. 11, No. 13, Pp. 8226–8230, 2016.

[7] X. Zhu, X. Qin, and M. Qiu, *"QoS-Aware Fault-Tolerant Scheduling for Real-Time Tasks on Heterogeneous Clusters,"* IEEE Transactions on Computers, Vol. 60, No. 6, Pp. 800–812, 2011.

[8] X. Zhu, J. Zhu, M. Ma, and D. Qiu, *"QAFT: A QoS-Aware Fault-Tolerant Scheduling Algorithm for Real-Time Tasks in Heterogeneous Systems,"* Proc. - 2010 13th IEEE International Conference Computational Science and Engineering (CSE), Pp. 80–87, 2010.

[9] Q. Han and T. Wang, "*Enhanced Fault-Tolerant Fixed-Priority Scheduling of Hard Real-Time Tasks on Multi-Core Platforms,"* Pp. 21–30, 2015.

[10] S. M. Islam, M. Anisur, R. Reza, and A. Kiber, *"Wireless Sensor Network using Particle Swarm Optimization,"* Pp. 1–5, 2013.

[11] R. S. Vaishnavi and M. Sukumar, *"Survey on Task Allocation using Particle Swarm Optimization in Wireless Sensor Network,"* International Journal of Innovative Research in Computer and Communication Engineering, Vol. 3297, No. 11, 2015.

ISSN: 2503-2259; E-ISSN: 2503-2267