APPLICATIONS OF
MODELLING AND SIMULATION
http://arqiipubl.com/ams eISSN 2600-8084 VOL 3, NO. 3, 2019, 160-167

A GUI Simulator for Analysis of Real-Time Tasks Assignment on Modified Fault-Tolerance Scheme by Means of Active Backup Replication Technology

Francis Franklin Marshall*, Emmanuel Adewale Adedokun, Ahmed Tijjani Salawudeen, Oduadu Ngbede Salefu, Ajayi Ore'ofe and Umar Abubakar

Department of Computer Engineering, The Ahmadu Bello University, Zaria, Nigeria

*Corresponding author: frankdidam14@gmail.com

Submitted 28 June 2019, Revised 15 July 2019, Accepted 01 August 2019.

Abstract: A graphical user interface (GUI) called the task allocation scheme simulator (TASS) for simulation of sensor nodes in wireless sensor networks (WSNs) using the real-time fault-tolerance task assignment scheme (RFTAS) and the modified real-time fault tolerance task assignment scheme (mRFTAS) is developed to carry out the performance evaluation of tasks or resources assignment in networks that are for sensor nodes. This paper focuses on the development of mRFTAS using the technology of backup (active-backup) for simulation of WSNs. Malicious attacks and the risk of sensors node failures are known to create a profoundly negative consequence on WSNs considering real-time events. The RFTAS is developed to address these issues, however, it has the problem of processing time delay. This is attributed to the characteristic of the passive backup copy technique adopted for the RFTAS in which the copies of backups tasks are activated when the copies of the primaries tasks have failed. Delay in the activation of copies of the backups tasks, of the primary tasks in tasks allocation execution processes as a result of a failure of sensor nodes or the primary tasks, will consequently lead to disastrous penalties if the systems under observation are safety-critical, such as aircraft, detecting fire burning in the forest, nuclear power plant, monitoring military battlefield. The mRFTAS is therefore enhanced using the active replication backup technique where both the primary and backup copies of tasks are executed concurrently. The analyses of the RFTAS and mRFTAS are conducted using total execution time of the task and energy consumption. The performance of mRFTAS shows an improvement over RFTAS in terms of minimizing task execution time by 28.65% and a trade-off in energy consumption by -17.32%.

Keywords: Active backup; Fault-tolerance; Graphical user interface; Real-time; Wireless sensor network.

1. INTRODUCTION

The collection of a group of sensing nodes is known as the wireless sensor networks (WSNs). These are mostly used in obtaining important information in some target areas [1]. WSNs have multiple areas of implementations, such as battlefield [2], military intelligence sensing and tracking, environmental tracking [3], emergency response and disaster management [4], agricultural precision, detection of flooding areas, medical telemonitoring, chemical and structural monitoring [5]. WSNs are also invaluable in application areas such as the Internet of Everything (IoE), systems that are cyber-physical, the vehicle with intelligent systems and smart connectivity cities [6]. WSNs are known with the most important constraint, the utilization of low power constraint requirements of nodes [7]. The technological innovation essential for the support of WSNs computational capacity of sensors nodes in term the parallel processing [4,7] and allocation of tasks performs a momentous part in processing that is in parallel. Tasks are to be suitably assigned to the sensor nodes and simultaneously balance the load of the network [4]. However, studies carried out on the issues of allocation of tasks in distributed systems indicate that the drawbacks encountered in task allocation in WSNs were different compared with those of conventional distributed systems [4,7].

Generally, assigning resources/tasks within sensing nodes during resources or tasks assignment is of great concern in WSNs. Tasks are assigned in ways that can ease the power utilization of nodes while still guaranteeing that the tasks are capable of being finished earlier than the set deadlines, thus prolonging the network lifetime [7-8]. For network lifetime to prolong, the essential factor is load balancing [7]. Proper technology needs to be put in place for the allocation of tasks such as independent processing of task will be carried out by individual nodes [7]. The failures of sensor nodes should not interfere with the processing of the entire tasks assigned to the network, especially for applications which are security or safety-critical. Sustaining the performance of the nodes in the networks without any form of interruption as a result of failures of the nodes, fault tolerance is the most invaluable inventiveness [7]. The replication technique is the principle that utilizes the

primaries/backups (P/B) classification, and is the most extensively utilized technologies for tolerance of fault during task/resources assignments as it tolerates nodes to process copies of a task separately [7]. There are two categories of the backup scheme namely, passive and active [9]. The active is the process in which both the primary and the backup copies are executed concurrently, while the process in which the backup copies are activated only after there is an incorrect result obtained from the primary copies is known as the passive [7].

Tasks/resources allocation in WSNs is the process that gives rise to a particular number of sensor nodes being involved in certain tasks, for the purpose of carrying out or executing such given tasks. Due to the several issues existing in the WSNs, the method of tasks/resources allocation is highly essential in demand to uphold the operations of failure-free resources assignment in WSN [2,14]. Task allocation by means of fault tolerance is an effectual way of enriching WSNs reliability. Allocating a number of copies of resources/tasks on diverse sensor nodes is an easy and effectual way of actualizing a fault-tolerance system [7]. In this paper, the P/B model is put to use. The model of the P/B of a lone task has two copies that are allocated to two individual sensor nodes [1]. For lone tasks, its primaries copies should be scheduled in conjunction with its backup copy and both are executing concurrently [7]. For the system to proffer high reliability, the research considers the active backup copies so that for a particular task with partial laxity, the copies concurrently are executed in an equal slot of time [1,7].

Nearly every tasks execution in WSNs are entreated to track in a real-time manner [15]. Real-time schemes involve computing schemes that are focus to real-time limitations where the precision of an output depends on its logic accuracy and when the output is delivered [11]. Applications in conjunction with timings constraints are called “real-time”. As soon as events occur, the application has to deal with it inside a recognized and confined delay. An application with a longer processing time can possibly have a negative impact [16]. For a sensor node to be used in the military operation, it is required that the fault tolerance has to be of high precision 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]. The replication backup techniques which involves the use of 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 mRFTAS, such as to avoid the breakdown of a system due to failure of some sensor nodes.

Literature review showed that there exist several researches 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 research area for real-time WSNs. The real-time fault-tolerant task allocation scheme (RFTAS) is a preventive scheme for 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 remains a challenge for real-time systems that are critical in term of safety. This work presents a GUI simulator for analysis of real-time tasks assignment on modified fault-tolerance scheme by means of active backup replication technology to address the issue of delay in task processing time associated with the RFTAS. The mRFTAS 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. MATERIAL AND METHODS

The experimental simulation on the scenarios of real-time WSNs was carried out by utilizing the Visual Studio 2016 Professional. The simulating area considered was 100 by 100 sensors networks. The parameter used in designing the sensor network and simulator are the network size, sensor radius, sensor delay, sensor power, transmitter delay, initial energy, residual energy, update frequency, energy cost, receive cost, transmission cost, transmission radius. The number of considered nodes were from 100-400 sensors nodes. The performance metrics used for this paper work are total time of task execution, and the energy consumption of task.

2.1 Development of Task Allocation Scheme Simulator (TASS) GUI for the WSN Simulations

The developed GUI for the WSN simulation is known as tasks allocation schemes simulator, and was developed in Visual Studio professional 2016 by using C# programming language. The research built up a simulating system to address the issue of real-time fault-tolerant task assignment. It incorporates node deployment, node discovery, energy management mechanism, and the routing protocol.

The research simulator was designed and implemented using to simulate the sensor node network which comprised of interconnected sensor nodes exchanging data over the wireless communication channel. The sensor (n) nodes were placed in random but uniformly 100 by 100 meters in a rectangular region. The system designed using a specified sum of nodes in the sensors network, and the initial energy of every node is arranged as [18]:

- X element of the map (for arbitrarily placing nodes)
- Y element of the map (for arbitrarily placing nodes)

Sensor nodes were added randomly to the network. They were sorted by x with no other nodes with the x/y coordinate in the list of node in the network. This provides an unambiguous for establishing routing connections [18].

a) Deployment of Node: This is carried out in a random manner. To cover a vast area, instead of deployment of nodes at a given moment, an irregular organization technique was used.

b) Discovery of Node: This protocol communicates by a broadcast "hello" message, if a node is inside the transmitting/broadcast limit, it would accept the message, and then send a feedback reaction. At the nodes where the massages are started, a recognizing node is added to the rundown of neighbors lists. At the point when a node completes a broadcast and gets no affirmation, the node will expand the broadcasting radius until negligible numbers of neighbors are found. The nodes utilize a von Neumann Neighborhood to check out the deployment framework [17-18]. To guarantee that a point in the network is in the range, the distance equation is given by an algorithm. The distance equation utilizes the Pythagorean Theorem to decide the separation of two distinct nodes. The algorithm utilizes the x-y directions of the broadcasting node and a neighboring node to decide the sides of the triangle. The hypotenuse is then calculated. In the event that the hypotenuse does not surpass the nodes broadcasting radius and the node is right now wakeful then the node is a substantial neighbor [18]. That node is then added to the rundown list of neighbors of the broadcasting node. This procedure is repeated for all the nodes not in sleep mode, in the simulation.

c) Utilization and Power Production: The loss of energy is a fundamental cause by a node because of message sending and accepting. Energy lost because of the number of messages sent from the broadcasting nodes radius and the loss of energy per the sent messages can be obtained as [17-18].

$$Energy\_Lost = Energy - \frac{Broadcasting\_Range \times Energy\_Per\_Msg}{100} \tag{1}$$

Loss of energy is more when delivering a package, however accepting a message makes the utilization of energy. Equation (2) exhibits a formula for loss of energy because of a message being accepted. It is noted that only a little segment of the measure of the message per energy is being lost. Different types of energy lost relate to the nodes computational overhead. To represent this kind of loss, another equation is utilized. Equation (3) indicates the amount of energy being lost because of overhead per time period [17-18].

$$Energy\_Accepted = Energy - \frac{Energy\_Per\_Msg}{100} \tag{2}$$

$$Energy\_Overhead = Energy - Energy\_Per\_Step \tag{3}$$

At any point the level of the energy of nodes drops past a specific threshold, usually, the node enters a sleep/rest mode. The node will stay in the rest mode until it recovers sufficient energy to start functioning again. In a node, energy is delivered essentially using solar panels. The measure of energy that is being delivered is constrained and is reliant on the node's environment. By utilizing a normal node energy creation formula in Equation (4), one can evaluate the measure of energy delivered at each time allotment [17-18].

$$Energy\_Delivered = Energy + Energy\_Produced \tag{4}$$

At each time allotment, the node will lose energy from messages being sent and received, in addition to node overhead. While the measure of energy that the node is delivering would not be sufficient to allow the node to run continually, appropriate energy management allows the node to extend its life, thus enabling the network to keep operating.

2.1.1 Programming

The simulator was composed in C# utilizing Microsoft Visual Studio.NET 2015 Professional. The simulator application has two sections: a module for the WSNs and the simulator that has the WSN objects. The modules that involve the WSN are Sensors nodes, iSensor Radius, iSensor Delay, aPackets, aConnections, Connection Current, iResidual Energy, Wireless Sensor Connection, sSender and sReceiver, Packet, Wireless Sensor Network, VectorList, and Vector. This simulator supports tremendous networks of 400×400 nodes. Simulating transmissions over these networks, and at that point drawing the entire network quite a lot of times per second, needs a non-trivial pact of processing. The CPU of a standard 3.2 GHz machine is not sufficient for this task with the message pump that controls window occurrences. Henceforth, packing the two functions into one thread brings about a totally, impassive simulator window. Rather, the simulation thread is begun in a different thread, which runs consistently until the point that the client stops it (clicking Stop Button) [18].

2.1.2 WSN Simulation

These applications are the simulations of the WSN. The networks are deployed in light of parameters: size of network (number of nodes), transmitting/communication range, transmitting cost of energy and accepting messages. The network was being utilized for the simulation of the discovery of vectors traversing the field in sensors networks. In the simulations, the minute a vector trips the sensors of system nodes, the nodes generate information packets and directs them via a down-stream system node. The routed packets are suitably transmitted until the point a sensor is reached in the "uplink zone". Every node additionally simulates stored energy, that is exhausted by transfer and accepting packets, by recognizing vectors. Nodes consist of energy that is limited, they will at long-runout of the networks communications because of shut down, causing system failure [18].

The simulation comprises of two phases: network deployment and running simulations. Prior to network deployment, the properties of the system ought to be set by utilizing the configuration sliders. The network setup properties are assembled into two classifications [18].

- The Configuration of the Network: These components decide the hardware properties of the system. The accompanying factors can be configured as Receiving Cost, Size of Network, Sensor nodes Radius, Sensors nodes Period, Sensors Cost, Radius for Transmission, Transmitter Period and Transmitting Cost.

- Parameters for Routing: These variables decide the properties of the systems software. Basically, the routing of packets strategy to be utilized, if routing is set to "Arbitrary," every node chooses a downstream connection arbitrarily for every packet.

a) Start time execution process of calculation of the task's copies (primary and backup)
As in [14] the model, start time of primary copy ought to be early schedule whereas the backup copies start time ought to be late as viable. Thus, it can give sufficient laxity for the conforming backup tasks copies to receive passive (uninvolved) modes, primary's copies and active's copies of backup to overlap as slightly as conceivable, and along these lines enhance the usage of system resources. In this way, it is important for the earliest computation of the start time of tasks primary copies and the most recent start time of a tasks reinforcement copy.

b) The calculation process of the earliest start time of the task's primary copy
Assuming there is a t_j (a task), the time to start the primary task t_t^P which is designated to n_i the node can be calculated. Every idle time-space of every node is examined $[0, S_1'], [f_1', S_2'], [f_2', S_3'], \dots, [f_n', +\infty]$ starting from the left to the right, where S_i' and f_i' represent the starts time and finish time of the ith tasks in the tasks queue of n_i, correspondingly. For probable appearance, let $f_0' = 0$. In the event that the initials idle time-space $[f_k', S_{k+1}']$, which can encounter $\max(a_{j,f_k'}) + et_{j.i} \leq d_j$, is discovered, the earliest start time of t_t^P on node n_i would be noted as $est_{ji} = f_k'$ generally est_{ji} would be noted as $+\infty$ [14,18].

c) The calculation process of the latest start time of the task's backup copy
Assuming 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:

Step 1. Initialize time $Space = [d_j - et_{ji}, d_j]$, for comfort, the start time and finish time of space can be set apart as S_{Start} and S_{finish} i.e. $Space = [S\_begin\ , S\_finish]$;
Step 2. Scan the task queue of n_i. If the $slot$ is situated in its idle times, let lst_{ji} be S_start and after that, the procedure is finished. Else, it implies that space overlaps with other tasks in the line, hence, stamp the overlapped task as t_x for further consideration. If t_x is a passive reinforcement copy, let lst_{ji} likewise, be S_start and after that, the procedure is finished. In any case, if t_x is a primary copy or an active reinforcement copy, update space with S_finish is equal to the start time of t_x and $S\_start = S\_finish\ - et_{ji}$. Then, if $S\_finish < 0$, let lst_{ji} be $+\infty$ and the procedure is finished, else goes to Step 2 once more [14,18].

d) Allocation process of the task's primary copy
In this area, the work plans a task designation process for the primary task's copy. Accepting that a task primary duplicate to be assigned is t_j^P, the procedure considers task deadline limitation, energy utilization, and the failure proportion. Those elements are quantified and weighting collected by the function of information standardization. Then 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 [14,18]:

$$U^P(i,j) = w_{t1} \times UB(i,j) + w_{t2} \times UE(i,j) + w_{t3} \times UR(i,j) \tag{5}$$

where w_{t1}, w_{t2} and w_{t3} are weight coefficients, $U^P(i,j)$ is the utility function of primary duplicate, the slighter the estimation of $U^P(i,j)$ results in a better n_i execute t_j^P completely. $UB(i,j)$, $UE(i,j)$ and $UR(i,j)$ imply the degree of load, the degree of utilization of energy and the proportion of failure level of n_i which executing t_j^P distinguished and differentiate nodes that took part in executing t_j. The energy utilization $ene_{j,i}$ and failure proportion $\prec_i$ of n_i are planned in the scope of 0 and 0.5 by utilizing an equivalent *sigmoid* function as packets adjustment function to achieve $UB(i,j)$, $UE(i,j)$ and $UR(i,j)$. The estimation of some specific equations are offered by [14,18]:

$$UE(i,j) = \begin{cases} 0, \quad if \quad (ene_{max} - ene_{min}) = 0 \\ \dfrac{1}{e - \dfrac{x_{ji} \times ene_i - ene_{min}}{ene_{max} - ene_{min} + 1}} - 0.5, \text{else}, \end{cases} \tag{6}$$

Smaller $UB(i,j)$ gives lighter the load of n_i when executing t_j^P. ene_{max} and ene_{min} imply the biggest utilization of energy by the nodes and the slightest utilization of energy by the nodes which take an interest in t_j, separately. The smaller $UE(i,j)$

shows less energy utilization of n_i when executing t_j^P. $\prec_{max}$ and $\prec_{min}$ mean the biggest and the minimum failure proportion of nodes which take an interest in t_j, separately. The individual's allotment procedure of a task's primary copies can be presented by the two steps [14,18]:

Step 1. For every node n_i which takes part in t_j, calculation of the total, est_{ji} and et_{ji}, foresee the required time off t_j^P to be finished on the node n_i. In the event that the deadline requirement t_j is met, $U^P(i,j)$ will be calculated by Equation (6). Generally, $U^P(i,j)$ will be noted $+\infty$ until every one of those nodes has been taken into deliberation.

Step 2. Select the nodes with less $U^P(i,j)$, and afterward, allot t_j^P to the chosen node and updates by comparing $U^P(i,j)$ and $UB(i,j)$ for the next estimation suitability.

e) Allocation process of the task's backup copy

Step 1. There exists a reinforcement copy t_j^B . 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 (7), else, $U^B(i,j)$ will be observed as $+\infty$ pending when each of the nodes is taken into the interpretation.

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 $UB(i,j)$ [14,18]:

$$UE'(i,j) = \begin{cases} 0, \quad if \quad (ene'_{max} - ene'_{min}) = 0 \\ \dfrac{1}{e - \dfrac{x_{ji} \times S(t_j^B) \times ene_{ji} \times per_{ji} \times ene'_{ji}}{ene'_{max} - ene'_{min} + 1}} - 0.5, \text{else}, \end{cases} \tag{7}$$

ene'_{min} and ene'_{max} represent the minimum and the maximum utilization of the energy of the nodes which take part in t_j, correspondingly. Theycan be estimated by [14,18]:

$$ene_{min} = \min_{i=1}\left(x_{ji} \times S(t_j^B) \times ene_{ji} \times per_{ji}\right) \tag{8}$$

$$ené_{min} = \min_{i=1}\left(x_{ji} \times S(t_j^B) \times ene_{ji} \times per_{ji}\right) \tag{9}$$

2.2 The Modified Fault-tolerant Real-Time Task Allocation Scheme

The modification of the fault-tolerant real-time task allocation scheme will be carried out by utilizing the active replication techniques for the backup systems. In the active backup's replication scheme both the primary and the backup's copies of tasks are processed concurrently. This takes care of the time for waiting and listening for the failure of the primary task copy. The waiting and the listening time of the backup's copies is the time of laxity. The modification is removing the laxity of the backup task copy [18].

2.2.1 The Flowchart of the mRFTAS

In this paper, a dPSO scheme is also hired in an mRFTAS in a WSNs framework. The scheme flow chart is presented in Figure 1 that shows the sink node at first collects the processed tasks. It then generates the velocity, the position in parallel for a single generation of the dPSO. After which, a series of operations begins. The terminal conditions are checked to know if the conditions are met, then it begins with the next iteration or else, tasks are published and waits. The right part of Figure 1 is the execution tasks process, where both primary and backup's tasks are running concurrently on the separate sensor node. There is no waiting time for the primary task to fail first before running the backup copy of the task. Also, when one of the tasks is completed, either the primary or the backup copy a query is sent to stop the corresponding copy from executing [18].

3. RESULTS AND DISCUSSIONS

The experimental simulations were done on scenarios of real-time WSNs, with the simulations design area of the sensors networks was 100 by 100 meter. The quantity of sensor nodes considered is 100-400 nodes. The absolute number of tasks considered is 400 tasks. 10 iterations were considered and the average of the ten results was calculated.

3.1 Task Allocation Scheme Simulator

The interface of the developed TASS, which is used for all the simulations, is shown in Figure 2. The two scenarios of 100 by 100 meter area with equal numbers of sensor nodes were simulated, and the level of energy depletion was observed and compared between the RFTAS and mRFTAS. The execution time of both models was also taken into consideration.

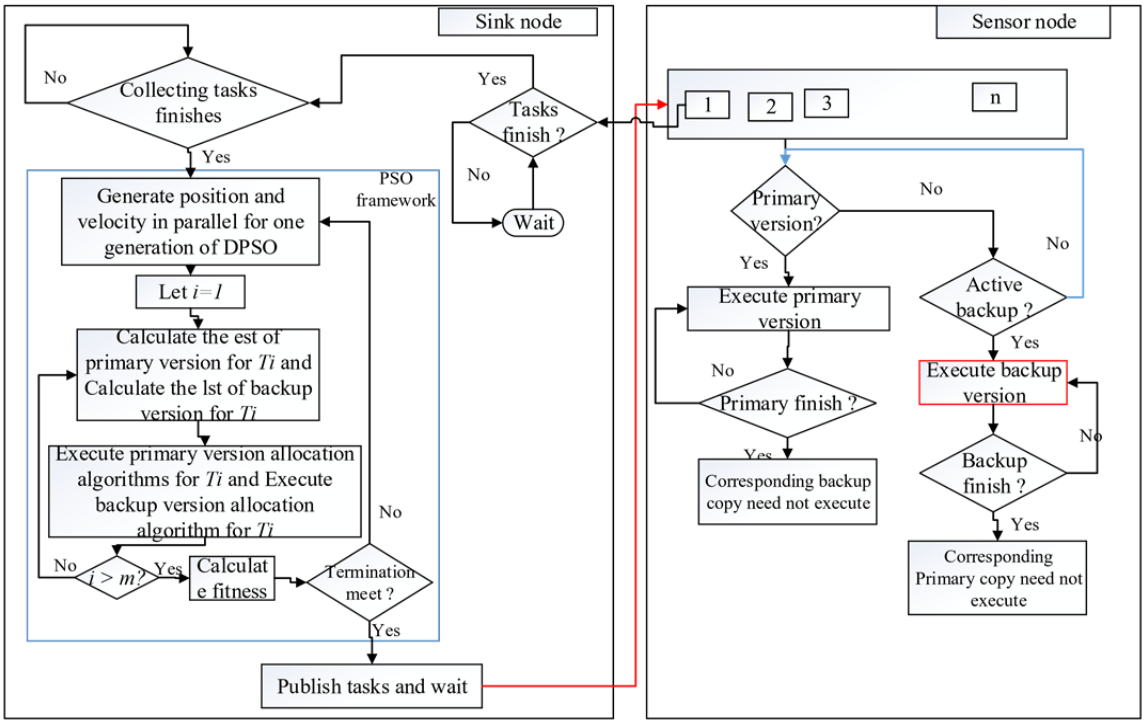

Figure 1. Flow chart of mRFTAS

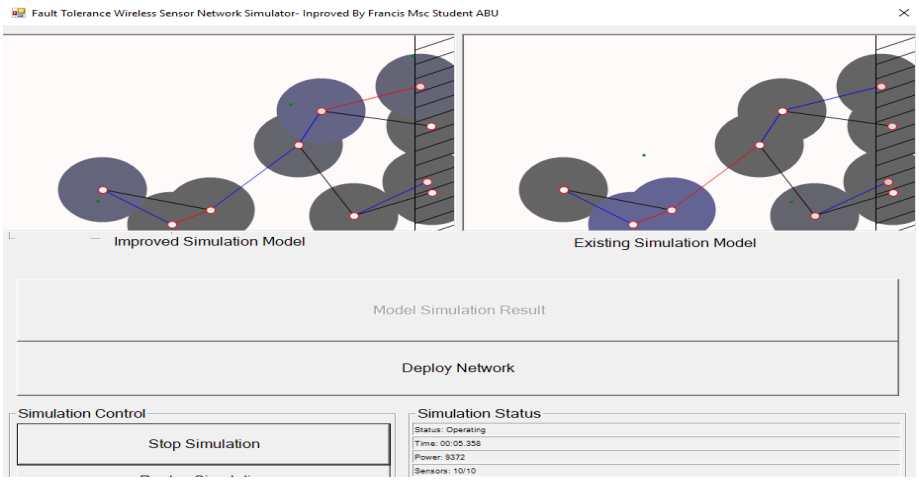

Figure 2. Simulation scenario using TASS

3.2 Task Execution Time of the Sensor Nodes

The tasks/resources execution time of a given task is the time taken for the sensor nodes to execute their tasks assigned to its completion. The experimental simulations begin with 100 nodes and with additions of 25 nodes at interim until the last, which is 250 nodes. The experimental simulations were conveyed simultaneously, for the RFTAS and the mRFTAS situations, and the outcomes can be seen by the plots of Figure 3.

Figure 3 demonstrates the similar plots of tasks execution time against the numbers of nodes for both RFTAS and mRFTAS. The scopes of nodes considered for the experimental simulations are 100, 125, 150, 175, 200, 225 and 250 nodes. It can be seen from the diagrams, the more the number of sensor nodes utilized for the execution of 400 tasks, the lesser the time required by the nodes to finish the whole tasks/resources execution. It is likewise clear that the tasks/resources execution time required by the mRFTAS is not as much as that required by the RFTAS to finish the given tasks.

3.3 Energy Consumptions of the Sensor Nodes during Task Execution

The energy utilization of the sensors nodes is the utilized energy in executing their tasks assigned to its completion. The experimental simulations begin with 100 nodes and with additions of 25 nodes at interim until the last, which is 250 nodes. The experimental simulations were conveyed simultaneously, for the RFTAS and the mRFTAS situations, and the outcomes can be seen by the plots of Figure 4.

Figure 4 demonstrates the relative plots of energy utilization against the range of nodes used for both RFTAS and mRFTAS. The scopes of nodes considered for the experimental simulations are 100, 125, 150, 175, 200, 225 and 250 nodes.

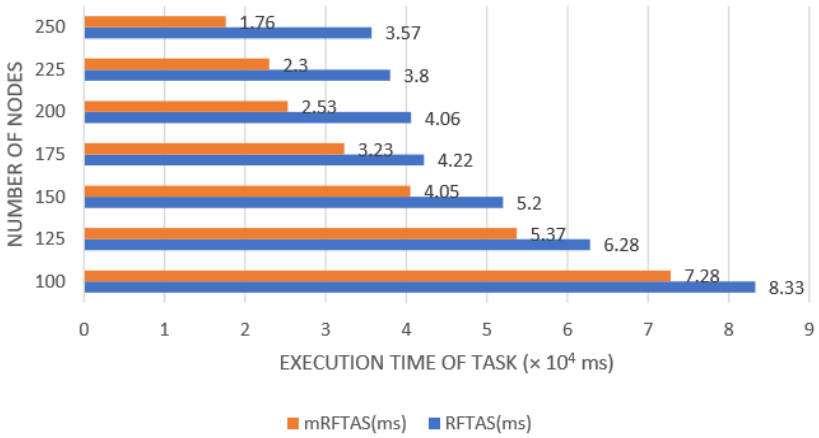

Figure 3. Task execution time for mRFTAS and RFTAS

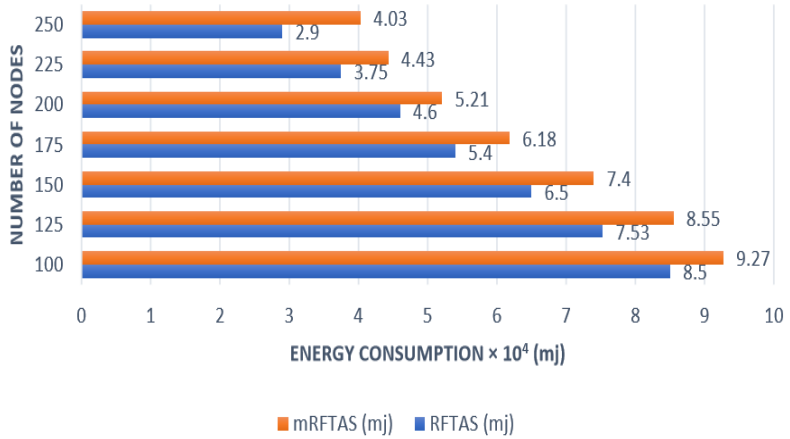

Figure 4. Energy consumption for mRFTAS and RFTAS

It can be seen from the diagrams, the more the numbers of nodes utilized for the execution of 400 tasks/resources allocation, the lesser the energy required by the sensors nodes to finish the whole assignments. It is additionally clear that the energy utilized by the mRFTAS is higher than that required energy by the RFTAS to finish the given tasks.

4. CONCLUSION AND FUTURE WORK

The results obtained from the experimental simulation showed the developed mRFTAS outperformed the RFTAS. 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%. The mRFTAS failed to perform better as compared to the RFTAS in term of energy consumption. The RFTAS percentage performance on energy consumption obtained when compared to mRFTAS was 17.32%. RFTAS took more time in its execution and assigning task/resources due to the time delay or the waiting period for activating the passive backup copy of tasks assigned for execution by the scheme. The mRFTAS has the fastest execution time due to the fact that the time delay or the waiting period of the backup task copies was removed considering the models were considered in real-time. mRFTAS utilized more energy than RFTAS, due to the fact that both the active and the backup copies of tasks are executed simultaneously. The future work will look into reducing both the energy consumption and the execution time of the task allocation.

REFERENCES

[1] C. Chen, W. Guo and G. Chen, A new task allocation algorithm based on the dynamic coalition in WSNs, *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & Ph.D. Forum*, Shanghai, China, 2012, pp. 1243–1248.

[2] S. Gangadharaiah, U. M. Hallur and S. S. Jamadar, Soft real-time auction scheme for task allocation in wireless sensor networks, *International Journal of Research in Engineering and Technology*, 3(4), 274–280, 2014.

[3] L. Mei, H. Dao-ping and X. Xiao-ling, Node task allocation based on PSO in WSN multi-target tracking, A*dvances in Information Sciences and Service Sciences*, 2(2), 13–18, 2010.

[4] W. Guo, Y. Chen and G. Chen, Dynamic task scheduling strategy with game theory in wireless sensor networks, *New Mathematics and Natural Computation*, 10(3), 211–224, 2014.

[5] H. -Y. Shi, W. -L. Wang, N.-M. Kwok and S.-Y. Chen, Game theory for wireless sensor networks: A survey, *Sensors*, 12(7), 9055–9097, 2012.
[6] J. Zhang and J. Long, An energy-aware hybrid ARQ scheme with multi-ACKs for data sensing wireless sensor networks, *Sensors*, 17(6), 1–28, 2017.
[7] W. Z. Guo, J. Y. Chen, G. L. Chen and H. F. Zheng, Trust dynamic task allocation algorithm with nash equilibrium for heterogeneous wireless sensor network, *Security and Communication Networks*, 8, 1865–1877, 2015.
[8] 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*, 11(7), 6533–6554, 2011.
[9] M. Priyanka, S. Anisha and R. S. Prabha, VLSI design for a PSO-optimized real-time fault-tolerant task allocation algorithm in wireless sensor network, *ARPN Journal of Engineering and Applied Sciences*, 11(13), 8226–8230, 2016.
[10] X. Zhu, J. Zhu, M. Ma and D. Qiu, QAFT: A QoS-aware fault-tolerant scheduling algorithm for real-time tasks in heterogeneous systems, *2010 IEEE 13th International Conference on Computational Science and Engineering*, Hong Kong, China, 2010, pp. 80–87.
[11] Q. Han, *Energy-aware fault-tolerant scheduling for hard real-time systems*, FIU Electronic Theses and Dissertations, Florida International University, USA, 2015.
[12] A. Bröring, J. Echterhoff, S. Jirka, I. Simonis, T. Everding and C. Stasch, New generation sensor web enablement, *Sensors*, 11(3), 2652–2699, 2011.
[13] M. Martin and M. Islam, Overview of wireless sensor network, in *Wireless Sensor Networks-Technology and Protocols*, InTech, 2013, pp. 1–5.
[14] 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 and Distributed Systems*, 26(12), 3236–3249, 2015.
[15] D. G. Harkut, M. S. Ali and P. Lohiya, Real-time scheduling algorithms for wireless sensor network, *Circuits and Systems: An International Journal*, 1(1), 11–18, 2014.
[16] I. Augé-Blum, F. Yang and T. Watteyne, Real-time communications in wireless sensor networks, in *Next Generation Mobile Networks and Ubiquitous Computing*, IGI Global, 2011, pp. 69–78.
[17] K. A. Marsal, I. Abdullah, W. Ismai and K.A. Rahim, Controlling algorithm for energy-consumption, radio bandwidth and signal strength deploying single fitness function to solve coverage area problems, *Advanced Science Letters*, 20(10-11), 2147-2151, 2014.
[18] F. Marshall, *Development of a modified real-time fault-tolerant task allocation scheme for wireless sensor networks*, Master Degree Thesis, Department of Computer Engineering, Ahmadu Bello University Zaria, Nigeria, 2018.