Improving Dynamic TDMA for Wireless Sensor Networks

Bharadwaja M Chittapragada, Rohit Kumar, Violina Doley, B. R. Chandavarkar
Department of Computer Science and Engineering
National Institute of Technology Karnataka, Surathkal
Mangalore, India
meher.211cs216@nitk.edu.in, rohitkumar.211cs244@nitk.edu.in, violinadoley.211it081@nitk.edu.in, brc@nitk.edu.in

***Abstract*—Dynamic TDMA is a multiple access control protocol that allocates specific time slots for each device/user to transmit or receive data (dynamically to those nodes which have data to send), providing predictable access to the communication medium. This mechanism has proven to efficiently reduce collisions during data transmission through Wireless Sensor Networks (WSNs). This paper aims to further enhance the existing algorithm and hence propose a Dynamic TDMA slot assignment protocol for wireless sensor networks that only allocates slots to nodes that have data to send and then dynamically allocate variable length time slots, thereby, reducing vacant slots and increasing channel usage. The goal is to improve the current node detection algorithm by considering the other important factors, incorporating advanced node tracking methodology, and increasing the flexibility of the slot allocation process.**

***Index Terms*—Dynamic TDMA, Wireless Sensor Network(WSN), Advanced Node state tracking, Dynamic threshold, Dynamic slot sizing**

I. Introduction

A wireless sensor network (WSN) [1] is a collection of sensor nodes that are deployed in large numbers to monitor the environment or system. When two or more devices attempt to send data concurrently on the same wireless channel, a collision happens, wasting both data and energy. Collisions in WSNs can be prevented by designing various access mechanisms, such as Medium Access Control (MAC) protocols [2]. Achieving fairness and efficiency in WSNs is particularly challenging because of the enormous number of linked devices and the network's dynamic structure. The limited energy and computing capabilities of these devices also place restrictions on the design of the various access modes. Several MAC protocols, including TDMA (Time Division Multiple Access) [3], FDMA (Frequency Division Multiple Access) [4], CDMA (Code Division Multiple Access) [5], and CSMA (Carrier Sense Multiple Access) [6] have been proposed for WSNs to address these issues. These protocols have differing degrees of fairness and efficiency and use different methods to channel access.

TDMA (Time Division Multiple Access) systems [7] allocate specific time slots to each user, and the user can only transmit or receive data during their assigned time slot. Several TDMA scheduling algorithms that employ both static and dynamic techniques have been implemented [8], [9]. The most typical TDMA techniques are static and distributed or static and centralized [10]. However, in static TDMA, users are assigned a fixed time slot regardless of whether they have any data to transmit during that time. This results in inefficient bandwidth use as resources are allocated to users who may not be transmitting any data.

Dynamic-TDMA is highly effective in enhancing channel utilization and improving standard metrics, including throughput, latency, and energy efficiency as shown in paper [11]. By the use of a Dynamic allocation algorithm, slots will only be assigned to nodes that have data to transmit. In [11], Dynamic TDMA for Wireless Sensor Networks, a technique for identifying nodes without data to send was devised. This paper intends to enhance the existing algorithm by making the following 3 significant changes:

- Proposing an Enhanced Algorithm for Dynamic TDMA allotment
- Incorporating more advanced node state tracking methodology and using it for dynamic slot sizing.
- Increasing the flexibility of the slot allocation process by optimizing the thresholds introduced in our algorithm.

The rest of this paper is organized in the following manner, In section II, discusses studies related to the existing Static and Dynamic TDMA methods. In section III, the methodology of identifying the nodes which do not have data to send and then dividing the time slot size relatively to generate the slot matrix is presented. Section IV shows the results of the experimentation. Finally, section V concludes the paper along with future work.

II. Literature Review

Several variations of TDMA can be found in the literature. A few of the important ones and standard definitions are mentioned below:

TDMA is a multiple access control protocol (MAC) [2] that allows multiple stations to transmit data by dividing signals of a frequency into several time slots hence reducing the chances of data collisions.
Static TDMA is used to allot a time slot to each user to transmit data. This performs greatly while avoiding the collision of data during transmission but it is not that effective in dealing with varying network conditions [8]. In Static

TDMA If a node doesn't have data to transmit it still gets a time slot allotted which results in a waste of bandwidth.

Dynamic TDMA is a technique used to allocate time slots to users based on changing network conditions or demands i.e. dynamically. The concept of Dynamic TDMA has been used in the original research paper [11] as unlike Static TDMA it is very flexible that helps to reduce waste of bandwidth drastically.

Random Static TDMA(Rand-ST) combines the phrases random and static i.e., the selection of nodes to begin the slot allocation is random, and once the scheduling is obtained, it remains the same for the duration of the network.

Random Dynamic TDMA(Rand-DT) in the WSN system allocates each user a specific time slot within a TDMA frame, allowing users to transmit data without the risk of collision. This allocation of time slots is done dynamically based on the traffic patterns of users i.e. by the demand of the system.

In a **Centralized TDMA** a central controller oversees the distribution of time slots to various users. The network is completely visible to this controller, which assigns time slots based on user demand and traffic patterns. To make sure users transmit and receive data at the appropriate time slots, synchronization signals are delivered [7], [12].

Distributed TDMA does not need a centralized controller. In this method, the allocation of time slots between multiple users is done by communicating with each other. The users maintain their time slots and share information about it [7].

The paper [11] currently implements 2 algorithms Dynamic-TDMA-BSPS and Dynamic-TDMA-BAPS which are explained below.

- **Dynamic-TDMA-BSPS** (Based on a single previous state):
  A node's use of its slot for data transmission during the prior TDMA frame is indicated by its previous state. The general concept is as follows: if a node has used its time slot in a prior TDMA frame, we give it a time slot in the following frame. The node will lose its slot in the current frame if it didn't use it in the prior frame. And after the entire frame, the node is not allocated a slot in the previous frame it is given a slot in the next frame.

- **Dynamic-TDMA-BAPS** (Based on all previous states): According to this method, a slot is assigned to a node that has used it for at least half of the previous TDMA frames based on all of the node's prior states. For instance, if a TDMA scheduling was constructed throughout 10 frames, the approach checks in the 11th frame to see if the node has used its slot to send data at least 5 times throughout the previous 10 frames (i.e., 10/2 times); if so, the node will be given a slot in the 11th frame.

III. Proposed Methodology

A. Overview of the Enhanced Algorithm

This paper proposes an improved algorithm to detect nodes that may not transmit data in the slot provided to them. The initial algorithm as provided in [11] detects whether a node is active or not based on past data, but we plan to make 3 major changes to the algorithm:

- **Increase the flexibility of the slot allocation process:**
  The current existing approach only tracks whether a node has transmitted data in the past and allocates slots based on a fixed threshold (i.e., at least half of the previous frames), which may not be optimal in all cases. A more flexible approach could take into account the priority of the current node based on the level of the node in the route tree (distance from the sink) and allocate slots accordingly.

- **Incorporate more advanced node state tracking:**
  The current existing approach only tracks whether a node has transmitted data in the past. However, we could use the other properties that can be pre-determined and kept track of in the WSN to give relative time slots.

- **Using dynamic slot sizes based on relative priority score:**
  The current approach assumes that each node requires a fixed slot size, but in reality, different nodes may have different data transmission requirements. A more advanced approach could use dynamic slot sizes, where the size of the slot allocated to each node is adjusted based on its current data transmission requirements.

This paper adopts the restrictions that have been used in paper [11] and generates Slot Matrix dynamically using an optimal slot generator similar to the one in paper [11].

B. The Flow of the Enhanced Algorithm

The enhanced algorithm for a dynamic and centralized WSN as each node keeps track of its allotted slot whose structure is described in the flow chart as shown in Fig. 1.

We take the Routing Tree information and the current node information(such as the battery level, data rate, etc) as the input and perform the following:

- Real-time Data Analysis: Case-Based
- The Generic PriorityScore Approach
- Dynamic Slot Sizing and Slot Removal

C. Real-time Data Analysis(Case Based)

- Go through the architecture and the network topology of the WSN and analyze the amount of data each node needs to transmit while considering all the factors such as processing power, battery life, etc. If the node has no data to send, remove its slot in the frame.
- Else calculate the amount of data each active node has to send and do a dynamic slot size allocation, i.e., give more time to nodes with more data to send and less time to nodes with fewer data to send. Therefore, we can sort the nodes in descending order of how much data they send and distribute the time among nodes linearly.

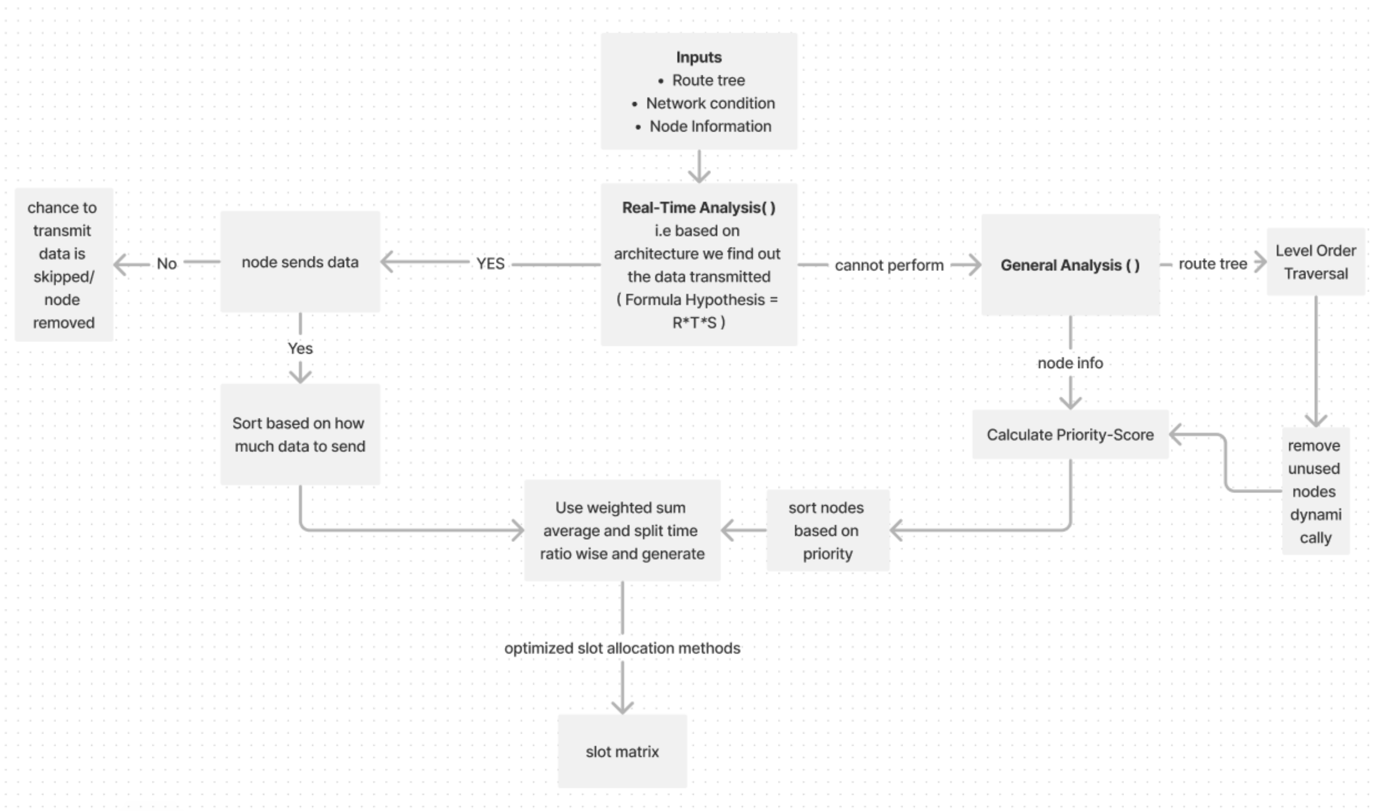
Inputs
Route tree
Network condition
Node Information
Real-Time Analysis( )
i.e based on architecture we find out the data transmitted (Formula Hypothesis = R*T*S)
YES
node sends data
No
chance to transmit data is skipped/ node removed
Yes
Sort based on how much data to send
cannot perform
General Analysis ( )
route tree
Level Order Traversal
node info
Calculate Priority-Score
remove unused nodes dynamically
sort nodes based on priority
Use weighted sum average and split time ratio wise and generate
optimized slot allocation methods
slot matrix

Fig. 1. The Flow of the Enhanced Algorithm

- If the calculation of the amount of data to be transmitted is not possible or if a more general approach is preferred, the generic analysis can be implemented.

This step is very application specific. i.e. For example, if the WSN is deployed to monitor environmental conditions such as temperature, humidity, and air quality, the data collection schedule may be based on the time of day, the weather conditions, or other factors that affect the data readings. Based on all the above factors, we can estimate the amount of data that each node may collect and transmit at a given time. It is also important to consider the inherent properties of the sensor nodes, such as their processing power, memory capacity, and battery life, when estimating the amount of data to be transmitted. Additionally, for a brief estimation, the following **empirical formula** can be used for calculating the amount of data that a node can deliver.

$$D \propto (R \times T \times S) \tag{1}$$

where:

- D: Amount of data that needs to be transmitted by each node
- R: Data rate (in bits/second) required by the application or protocol
- T: Transmission time (in seconds) required for transmitting the data
- S: Sampling rate (in samples/second) of the sensor data

Some related work to this empirical formula is in the cited papers [13]–[15].

D. The Generic Analysis Approach

In this subsection, the paper describes the enhanced algorithm that can be scaled and applied to a wide range of WSNs without doing a specific analysis. The Generic Analysis approach can be broadly classified into mainly 3 steps.

- Level order-based traversal and Removal based on past data regarding the utilization of channel
- Priority Score Calculation for the remaining nodes and Sorting of the nodes based on the priority score
- Linearly varying dynamic time slot size allocation.

1) Level order Traversal and Removal based Past usage Data: The routing tree information of the WSN is considered as the input where the sink is taken as the root and a level order traversal is performed on the tree from the leaf node to the root node to obtain a list of lists containing nodes of each level which are described in the algorithm 1 shown below.

In algorithm 2, the threshold values for the leaf and top levels of the tree are initialized. Then the threshold is calculated for each level dynamically based on a linear decrease between the leaf and top levels.

Algorithm 1 Level order traversal starting from the leaf nodes for all trees

```
Input: root node of a tree
Output: a list of lists containing nodes of each level, starting
        from the leaf nodes
$result$ ← empty list;
$q$ ← empty queue;
foreach leaf node in the tree do
    enqueue $q$ with the leaf node;
end
while the queue $q$ is not empty do
    $size$ ← size of the queue $q$;
    $level\_nodes$ ← empty list;
    for $i \leftarrow 1$ to $size$ do
        $node$ ← dequeue a node from the queue $q$;
        append $node$ to $level\_nodes$;
        foreach child of $node$ do
            enqueue $q$ with the child;
        end
    end
    append $level\_nodes$ to $result$;
end
return $result$ ;
```

The output of algorithm 1 is used as the input for algorithm 2

Algorithm 2 Dynamic threshold-based node filtering algorithm

```
Input: a list of lists containing nodes of each level, starting
       from the leaf nodes
Output: a list of lists containing nodes that have sent data
        more than K percent of the time slots allocated to
        them
$K\_leaf \leftarrow 60\%$;
$K\_top \leftarrow 30\%$;
$n$ ← number of levels in the tree;
for $l \leftarrow 1$ to $n$ do
    $K \leftarrow K\_leaf + \frac{(K\_top - K\_leaf) \times (n-l)}{n-1}$;
    foreach node in the $l$th level of the tree do
        $count$ ← number of times the node has sent data in
          all of its previous time slots; $total$ ← total number
          of time slots allocated to the node;
        if $count/total \geq K/100$ then
            append the node to the result list;
        end
    end
end
return result;
```

Algorithm 2 iterates over all nodes in each level of the tree, calculates the number of times each node has sent data, and checks whether it is greater than or equal to the corresponding threshold for that level. If so, a time slot is allocated to that node, else we remove the time slot to that node for this particular cycle (However, those nodes which are not allocated slots in this cycle are provided the slots in the next cycle)

After using algorithms 1 and 2, the subset of nodes involved in the WSN is obtained as the nodes that are nodes that are most likely to not have sent data are eliminated based on past information.

2) PriorityScore Calculation and Sorting for the remaining nodes: **PriorityScore** is a metric that is based on factors like Battery Level, Packet queue length, Data rate requirement, and Channel Quality. The assumption as stated before, is that the data transmission is between the parent and child, hence this comes with the accompanying fact of ignoring the network traffic factor to be considered for the priority score calculation.

The formula for calculating the **PriorityScore** for each node i can be defined as:

$$PriorityScore(i) = w_B \cdot \frac{B_i}{\max_{j\in N}(B_j)} + w_Q \cdot \frac{Q_i}{\max_{j\in N}(Q_j)} + w_R \cdot \frac{R_i}{\max_{j\in N}(R_j)} + w_C \cdot \frac{C(i,p_i)}{\max_{j\in N}(C(j,p_j))} \tag{2}$$

where:
B_i is the **battery level** of node i,
Q_i is the **length of the packet queue** at node i,
R_i is the **data rate** requirement of node i,
$C(j,p_j)$ is the **channel quality** between nodes i and p_i(parent(i)),

w_B, w_Q, w_R, and w_C are weighting factors that determine the relative importance of each factor in the PriorityScore calculation, and $\max_{j\in N}$ denotes the maximum value of the corresponding factor across all nodes in the network. The weights would generally be in the order of $w_B \geq w_Q \geq w_R \geq w_C$. The weight selection can be done manually or using deep learning techniques.

3) Linear Dynamic Time Slot Sizing: The linear dynamic time slot sizing approach has several benefits. Firstly, it ensures that nodes with higher priority receive more time slots, allowing them to communicate more frequently and effectively. This can be particularly useful for critical nodes that require timely and reliable data transmission. Secondly, it allows for efficient utilization of network resources. Lastly, it allows for flexibility and adaptability to changing network conditions, as the length of time slots can be adjusted dynamically based on the priority of nodes, accommodating variations in node requirements or network conditions.

Algorithm 3 Node scheduling algorithm based on PriorityScore

```
Require: Set of nodes $N$, weights $w_B$, $w_Q$, $w_R$, and $w_C$, and total time $T$
Ensure : Allocated time slots for each node $i$
State  : Calculate the PriorityScore for each node $i$ using the formula:
State  : $PriorityScore(i) = w_B \cdot \frac{B_i}{\max_{j\in N}(B_j)} + w_Q \cdot \frac{Q_i}{\max_{j\in N}(Q_j)} + w_R \cdot \frac{R_i}{\max_{j\in N}(R_j)} + w_C \cdot \frac{C(i,p_i)}{\max_{j\in N}(C(j,p_j))}$
State  : Sort the nodes in descending order of their PriorityScore.
State  : Allocate time slots to each node $i$ proportional to its normalized PriorityScore:
State  : $Time(i) = \frac{PriorityScore(i)\cdot T}{\sum_{j\in N} PriorityScore(j)}$
State  : Pass the allocated time slots through an optimal slot generator() function to generate a standard slot matrix.
```

This algorithm takes in a set of nodes and weights and calculates a PriorityScore for each node. It then sorts the nodes by their priority score and allocates time slots to each node based on its normalized priority score multiplied by the total time. Finally, a standard slot matrix is obtained after being passed through an optimal slot generator function.

E. Justification

- **Level Order based traversal of the Route Tree:** The communication model used here is parent-child-based communication. The data sent by the leaf node will not be able to reach the sink if one of the relaying parent nodes is inactive. By inductive logic, we see that the node becomes a more important relay node as it is higher up in the routing tree. As shown in Fig. 2, if we remove node 2 from the tree, node 3 and node 4 cannot transmit data. Also, the base case would be that if the leaf node itself is not collecting data, then it makes perfect sense to not allocate it a slot as it has no data to send, unlike the parent nodes that act as relay nodes.

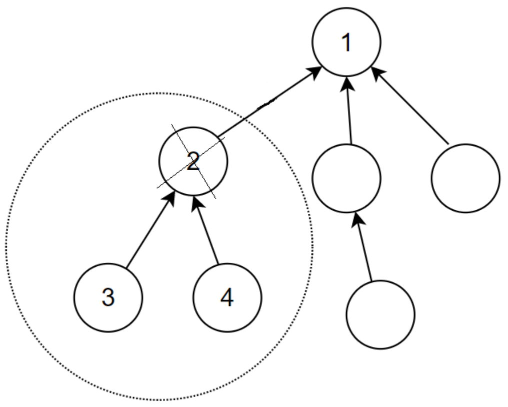

Fig. 2. Illustrating the importance of relay nodes

- **Priority order for the PriorityScore calculation** The relative importance of the factors is in the following order:
  1) **Battery level** is a critical factor because nodes with low battery levels may not be able to transmit data completely due to low energy so they would just need a smaller time slot. Therefore, nodes with a higher battery level should be given priority (more time) as they are more certain to send data.
  2) **Packet queue length** is important because nodes with longer queues may imply more data that is to be transmitted. For example, in a monitoring application, a node may have a long queue of critical data that needs to be transmitted urgently.
  3) **Data rate requirement**: Allocating slots to nodes with higher data rate requirements first can help ensure that they receive the bandwidth they require. For example, in a multimedia streaming application, nodes that are streaming high-quality video may require more slots to ensure a smooth and uninterrupted stream.
  4) **Channel quality** is less important because obtaining accurate channel-quality information can be challenging, especially in dynamic environments. While channel quality can affect transmission quality and reliability, it may not always be the most critical factor to consider.
- **Various Threshold and Methodology Selection Criteria** The Threshold values for K_leaf and K_top, are set to 60% and 30% respectively, based on the previously chosen value in paper [11] and the lower limit was updated based on the test results. Lower K_top resulted in higher lenience which in turn results in a lot of potentially data-less nodes being allocated time. The values of K_leaf and K_top can be varied based on the specific network structure and requirements.
  Our mathematical formulation considered an underlying linear relationship between different entities, mainly because of two factors i.e. direct proportionality and simplicity.

$$\text{PriorityScore} \propto \text{BatteryLevel}$$
$$\text{PriorityScore} \propto \text{PacketQueueLength}$$
$$\text{PriorityScore} \propto \text{DataRateRequirement}$$
$$\text{PriorityScore} \propto \text{ChannelQuality}$$

Although proportionality hints only towards increasing function as a relationship, linearity can be considered a simplified approximation. Hence we considered weighted linearity for our formula. However, we made sure that our mathematical approaches are scalable because different WSNs will have a different number of nodes, different environmental settings, different units, scales, or ranges, and hardware features, Hence we normalized the priority scores by dividing by the maximum corresponding value to prevent biased or skewed results and making them scale-invariant and more interpretable. There are several other choices for normalization methods, such as dividing by the sum of values or using z-score normalization but it would increase the calculation involved in large-scale networks.

- **Impact of removing nodes from allotting time slots [15]** Removing nodes that do not have data to send can have a significant impact on the overall performance, energy efficiency, and throughput of a wireless sensor network. It can affect each of these:
  - **Performance**: This is because the remaining nodes can operate more efficiently, with less interference and fewer collisions. This can lead to better packet delivery rates which can improve the overall performance of the network.
  - **Energy Efficiency**: When nodes are in an idle state, they consume energy while waiting for incoming data or when given a time slot but not sending data. By removing idle nodes, the overall energy consumption of the network can be reduced, leading to longer battery life for each node and potentially extending the lifetime of the network as a whole.
  - **Throughput**: This can lead to higher throughput, as more data can be transmitted in a given period we save time by assigning the slots to nodes that may have data to send.

IV. Results and Comparison

The results of a comparison between the proposed algorithm and the Dynamic-TDMA-BSPS and Dynamic-TDMA-BAPS algorithms were obtained using Python's matplotlib library and random value generation. The values were averaged over 100 cycles for WSNs of sizes 10, 100, 1000, and 10000 to obtain practical results. It was observed that the proposed algorithm performed similarly to, and even slightly better than, the original algorithm.

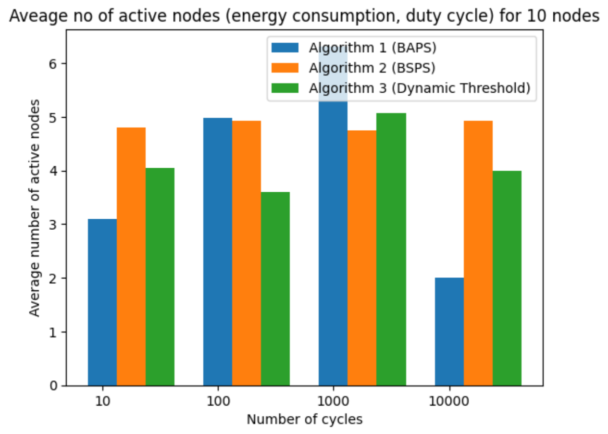

Fig. 3. Average no of active nodes(energy consumption, duty cycle) for 10 nodes.

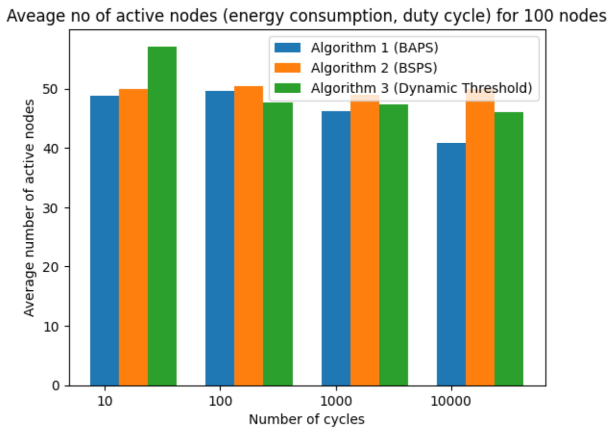

Fig. 4. Average no of active nodes(energy consumption, duty cycle) for 100 nodes.

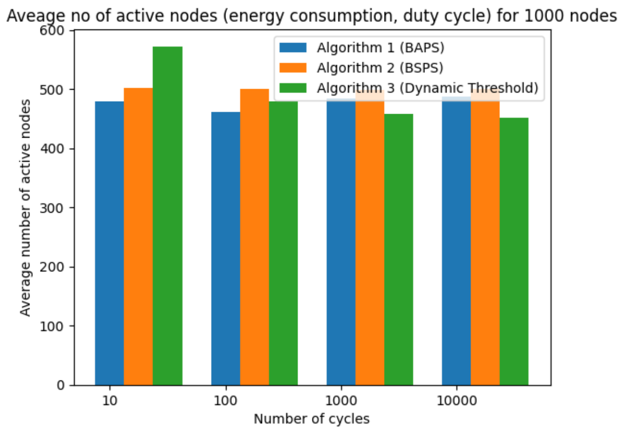

Fig. 5. Average no of active nodes(energy consumption, duty cycle) for 1000 nodes.

- **Fig 3,4,5** : The average number of removed nodes is compared across WSNs of sizes 10, 100, and 1000 over 100 cycles. Several conclusions can be drawn from these graphs. The fixed total time implies that **energy consumption** is directly proportional to the number of active nodes. It was observed that the proposed algorithm performed similarly to the Dynamic-TDMA-BAPS and outperformed the Dynamic-TDMA-BSPS. Therefore, this algorithm shows an improvement in **energy efficiency** as the number of active nodes is reduced. Also, the number of active nodes is not too less to result in inadequate data transfer. These factors also imply an improvement in **performance**.
- **Figure 6**: The time slot allocated to each node is shown when using the standard TDMA to split the time and then using our Linear Dynamic Time Slot Sizing and from the bar graph it is evident that more important nodes are being given more time when the enhanced algorithm is being used. Hence allowing more data to be sent in the same time frame which implies higher **throughput**.

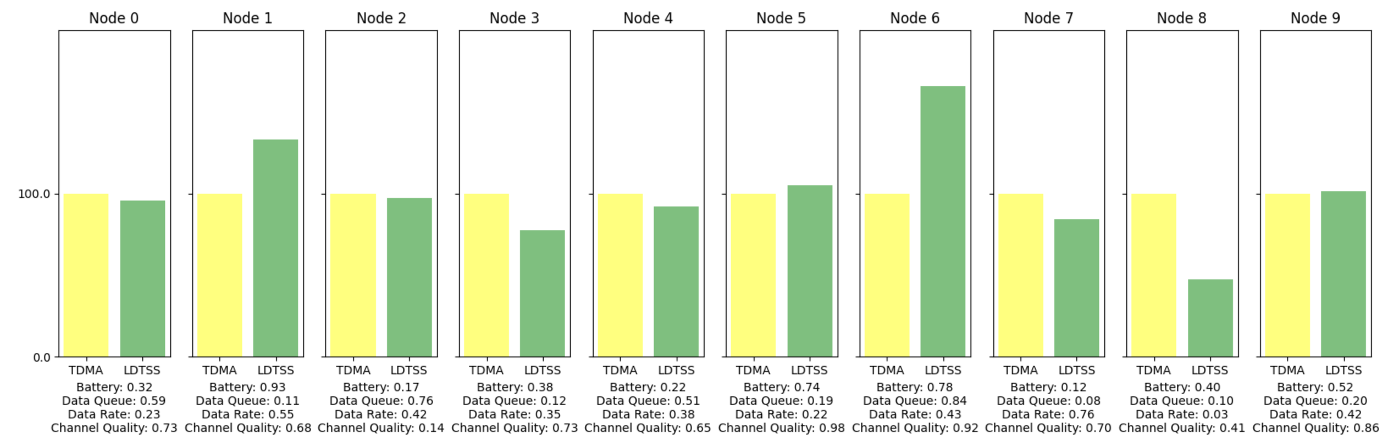

Fig. 6. Time distribution to each of the nodes for 10 nodes using LDTSS

V. Conclusion and Future Work

This paper presented dynamic, distributed, and contention-free TDMA scheduling approaches using routing tree information. The objective was to build a TDMA schedule that adapts its frames to the needs of the nodes, without affecting energy consumption. Our goal was achieved by considering many factors and by dynamically changing the TDMA slot size according to the network needs. The comparative study between our approaches and existing similar approaches [11] showed that our enhanced algorithm generates better throughput and energy efficiency.

The future work includes obtaining a more rigorous mathematical formulation using deep learning. Also, the ideal weights for the PriorityScore formula can be found using deep learning techniques.

References

[1] Z. Guan, L. Bian, T. Shang, J. Liu, When Machine Learning meets Security Issues: A survey, https://ieeexplore.ieee.org/abstract/document/8535799?casa_token=hanRTGahTMsAAAAA:p7V93Dzv7BqB_iurmeCma7t-FvwMycgb4B9Yd3DkebnsjSMtsJUlnHYFpi3lHFwUyyLGigjwXXo, [Accessed: 02-09-2023] (2018).

[2] P. Huang, L. Xiaoa, S. Soltani, M. W. Mutka, N. Xi, The evolution of mac protocols in wireless sensor networks: A survey. IEEE communications surveys & tutorials, https://www.researchgate.net/publication/260710169_The_Evaluation_of_MAC_Protocols_in_Wireless_Sensor_Networks_A_Survey, [Accessed: 12-03-2023] (2012).

[3] T. Kaur, D. Kumar, Tdma-based mac protocols for wireless sensor networks: A survey and comparative analysis, https://www.researchgate.net/publication/318751958_TDMA-based_MAC_protocols_for_wireless_sensor_networks_A_survey_and_comparative_analysis, [Accessed: 12-03-2023] (2016).

[4] FDMA(Frequency Division Multiple Access), https://www.sciencedirect.com/topics/engineering/frequency-division-multiple-access#:~:text=FDMA%20is%20the%20most%20basic,first%20and%202G%20cellular%20systems., [Accessed:27-04-2023].

[5] S. Ndungu, CDMA (Code-Division Multiple Access), https://www.techtarget.com/searchnetworking/definition/CDMA-Code-Division-Multiple-Access, [Accessed:27-04-2023] (2021).

[6] Network Protocol: Carrier Sense Multiple Access (CSMA), https://www.scaler.com/topics/csma/, [Accessed:27-04-2023].

[7] A. Bhatia, R. Hansdah, A classification framework for tdma scheduling techniques in wsns., https://arxiv.org/pdf/2002.00458v1.pdf, [Accessed: 12-03-2023] (2020).

[8] S. U. Hashmi, J. H. Sarker, H. T. Mouftah, N. D. Georganas, An efficient tdma scheme with dynamic slot assignment in clustered wireless sensor networks, https://www.researchgate.net/publication/261116546_An_Efficient_TDMA_Scheme_with_Dynamic_Slot_Assignment_in_Clustered_Wireless_Sensor_Networks, [Accessed: 12-03-2023] (2010).

[9] Z. De-min, L. Yun-jiang, Z. C. LI Man, A Dynamic TDMA Protocol Utilizing Channel Sense, https://drive.google.com/file/d/1PTtNqnRzJHLz-a7YjUzCCije1JeNVZqi/view?usp=sharing, [Accessed: 13-03-2023] (2015).

[10] S. Garg, V. S. S. Kuchipudi, E. S. Bentley, S. Kumar, A real-time, distributed, directional tdma mac protocol for qos-aware communication in multi-hop wireless networks, https://ieeexplore.ieee.org/document/9349361, [Accessed: 12-03-2023] (2021).

[11] E. A. Alkeem, S.-K. Kim, C. Y. Yeun, M. J. Zemerly, K. F. Poon, G. Gianini, P. D. Yoo, An Enhanced Electrocardiogram Biometric Authentication System Using Machine Learning , https://ieeexplore.ieee.org/abstract/document/8812730, [Accessed: 02-09-2023] (2019).

[12] A. Kumar, M. Zhao, K.-J. Wong, Y. L. Guan, P. H. J. Chong, A comprehensive study of iot and wsn mac protocols: Research issues, challenges and opportunities, https://ieeexplore.ieee.org/abstract/document/8543861, [Accessed: 12-03-2023] (2018).

[13] John, T. Ogbiti, H. C. Ukwuoma, S. Danjuma, M. Ibrahim, Energy Consumption in Wireless Sensor Network, https://core.ac.uk/download/pdf/234645193.pdf, [Accessed: 05-04-2023] (2016).

[14] B.-S. Kim, H. Park, D. G. 1Kyong Hoon Kim, K.-I. Kim, Real-Time Communication in Wireless Sensor Networks, https://www.hindawi.com/journals/wcmc/2017/1864847/, [Accessed: 05-04-2023] (2017).

[15] Q. Sun, Y. Qiao, J. Wang, S. Shen, Node importance evaluation method in wireless sensor network based on energy field model, https://jwcneurasipjournals.springeropen.com/articles/10.1186/s13638-016-0693-2, [Accessed: 05-04-2023] (2016).