# A Hybrid based Distributed Slot Scheduling Approach for WSN MAC

Manas Ranjan Lenka, Amulya Ratna Swain and Biraja Prasad Nayak

Original scientific paper

***Abstract*—In Wireless Sensor Networks(WSNs), collision handling during transmission of data is an important challenge. MAC protocol plays a vital role in handling those collisions. Among different types of MAC protocols, schedule based MAC protocol is one where a valid schedule is prepared to handle the collision. The existing schedule based MAC protocols focus on preparing either a feasible schedule or an optimal schedule. In order to satisfy both feasibility as well as optimality feature, in this paper, we proposed a hybrid approach for slot scheduling that prepares a feasible schedule in a distributed manner and at the same time reduces the number of slots in the feasible schedule to achieve optimality. In this paper, we named this as Hybrid based Distributed Slot Scheduling (HDSS) approach. The proposed HDSS algorithm initially prepares a feasible schedule which is further tuned in quick time to prepare a valid schedule with a reduced number of slots. The reduction of the number of slots in the schedule improves the efficiency of data transmission in terms of latency. The simulation results show that the HDSS algorithm outperforms RD-TDMA with respect to both the number of slots allotted for a feasible schedule as well as the data transmission latency.**

***Index Terms*—Wireless Sensor Network, Media Access Control, Slot Scheduling, feasible schedule, correlated contention.**

## I. Introduction

WSNs consist of thousands of tiny sensor nodes that sense the environment, collect the data, process it, and send the information to the sink node. The sink node behaves as a gateway which provides the information to the outer world. In this current age of advancement, WSN plays a vital role in various application areas such as Health Monitoring, Habitat Monitoring, Home automation, Environmental Monitoring, Military Surveillance, Bridges and building Monitoring, and Real-time Monitoring of Nuclear Plants. Each sensor node in WSN is powered by a battery, and these are mostly deployed in the hostile environment therefore not possible to recharge. Hence in WSNs, one of the most important goals is to conserve the energy as the battery power is limited. Along with limited battery power, some of the other constraints in WSNs are processing capability, memory, and bandwidth.

In order to conserve energy it is indeed required to identify various sources of energy wastes and handle them properly.The various sources of energy wastes in WSNs are basically categorized into four types [1], [2].

*Idle listening*: Any node that has kept its radio on but no data has been sent from other nodes to it then that situation is referred as idle listening. This is one of the major source of energy waste in WSN. Various protocols [1], [2] have been designed to reduce idle listening by switching off the radio from time to time.

*Overhearing*: Another source of energy waste is overhearing. In the case of overhearing, a node receives data which is not meant for it there by wasting energy.

*Control message overhead*: In WSN, it is unavoidable to stop the exchange of control messages among sensor nodes before the actual communication. The exchange of control messages also consumes energy. Hence in order to conserve energy, WSN protocols must exchange minimum number of control messages before the actual data communication.

*Collision*: During data communication, if a data packet gets collided with another packet then the transmission of the data packet must be stopped and retransmission of the same packet must be carried out which consumes energy. Also, due to the collision of packets, the latency in the transmission of the packets has been introduced.

Among various sources of energy wastage, reducing collision during the data transmission is one of the important consideration to conserve energy in WSN. Various slot scheduling algorithms [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [24] and MAC protocols [1], [2], [14], [15], [16], [17], [18], [19], [20], [25], [26] has been proposed to fulfill the above requirements. Nevertheless, none of the existing literature focuses on achieving both feasible schedule and optimal schedule in quick time. So, in order to achieve both feasibility as well as optimality, here, we proposed a distributed approach for hybrid slot scheduling algorithm that not only prepares a feasible schedule but also further reduce the number of slots along with handling collision.

The remaining part of the paper is structured as follows. Section II describes the related works. The proposed algorithm is briefly illustrated in Section III. Section IV analyzes the proposed algorithm. The simulation results and its analysis is presented in Section V. The conclusion and future work is given in Section VI.

## II. Related Work

The MAC protocols for WSNs can be broadly classified into 3 categories [21] viz., Schedule-based, Contention-based, and Hybrid. The Schedule-based MAC protocols prepare a valid schedule for data communication to handle the collision, which is one of the prominent source of energy wastes in WSNs [1].

Manuscript received January 16, 2019; revised April 9, 2019. Date of publication May 16, 2019. Date of current version June 3, 2019.

Authors are with the KIIT University, Bhubaneswar, India (e-mails: {manasy2k3, swainamulya, brjnayak}@gmail.com).

Digital Object Identifier (DOI): 10.24138/jcomss.v15i2.695

Although this category of MAC protocols handle the collision efficiently, still it does not adapt well to the topology changes in WSN. The Contention-based MAC protocol adjusts to the changes in topology (i.e. addition or removal of any sensor nodes to WSN) very easily. Nevertheless, handling collision in this category of protocols is costlier as compared to the Schedule-based MAC protocols. Furthermore, the hybrid MAC protocols are those protocols that try to combine the advantages of both schedule based MAC protocol and contention based MAC protocol while counteracting their weaknesses.

In many sensor network applications, the sensor nodes remain idle for a long time which consumes a lot of energy. Ye et al. [1] proposed a contention based MAC protocol called S-MAC that minimizes the wastage of energy due to idle listening and also taking care of other forms of energy wastage due to collision, overhearing, and control message overhead. According to S-MAC protocol, each sensor node goes to sleep state from time to time by turning off its radio. After timeout, the sensor node wakes up and listens if any neighbor nodes wants to send data or the same node has some data to sent to its neighbors. In order to achieve the above mentioned benefits, the following steps are carried out.

*Periodic listen and sleep*: Before each node goes to periodic listen and sleep, they must synchronize with each other by broadcasting their schedule to its Neighbors. After synchronization, each node has a schedule table which keeps the schedules of all its neighbors.

The below steps are followed to prepare and synchronize the schedule among the neighbor nodes.

- During start up, a sensor node first listens for certain amount of time to receive a schedule from it's neighbors. If it does not hear anything from it's neighbors within the timeout period, then it selects its own schedule and immediately broadcasts the same to its neighbors.
- After receiving this schedule, the sensor nodes use the same without creating their own schedule and then, they wait for a random amount of time and broadcasts the received schedule to their neighbors.
- The sensor nodes, who receive a different schedule other than their own schedule will follow both the schedules but broadcasts only their own schedule before going to sleep.

*Collision avoidance*: After synchronization, nodes whoever try to transmit data, first sense the carrier and exchanges RTS/CTS message to avoid a collision before any data communication.

*Overhearing avoidance*: The neighbor sensor nodes of both the sender and the receiver of an RTS/CTS packet are brought to the sleep state to avoid overhearing.

S-MAC [1] protocol makes a trade-off between the energy consumption and throughput/latency. The throughput is reduced as the data transmission occurs only during the active period. In addition the latency for data transmission may increase, as there is a chance that an event might occur during the sensor node is in sleep mode and data transmission has to be deferred till the sensor node has been brought back to the active mode. S-MAC employs a fixed sleep/active period which also affects the performance of the protocol in varying load condition.

Dam et al. [2] proposed the T-MAC protocol which is a contention based MAC protocol and works very much alike the S-MAC protocol. The T-MAC protocol improves the performance of S-MAC by adapting a variable duty cycle in a novel way to cope up with the variable load condition.

Z-MAC(Zebra MAC) proposed by Rhee et al. [14], is a hybrid MAC protocol which combines both the advantages of CSMA and TDMA to enhance the channel utilization. In the worst case, the performance of Z-MAC falls back to CSMA. This protocol works in 2 phases as follows.

*Set-up phase*: During the set-up phase, first 2-hop neighbor list of each sensor node is prepared. Then each node is assigned a time slot using DRAND [22] in such a way that no 2-hop neighbors get the same slot number. The node which is assigned to a slot is known as the owner of the slot and all other nodes are the non-owners of that slot.

*Transmission phase*: In the transmission phase, a node can transmit data in any time slot. Before transmitting the data, the node first sense the carrier using CSMA and if it finds the carrier to be free, then transmits the data using that time slot. However, the owner of that slot always gets the priority over the non-owners for sending the data.

Slama et al. [15] proposed a hybrid MAC protocol called I-MAC that takes the advantage of both CSMA and TDMA technique with an adaptive priority scheme for channel access. Like Z-MAC, this protocol also follows 2 phases, i.e. setup phase and transmission phase. During the setup phase, it performs neighbors discovery, TDMA slot assignment, framing and synchronization operations only once. These steps are also carried out when there is a significant change in the network topology.

The TDMA slot scheduling is done using the proposed distributed neighborhood information based algorithm (DNIB) [23] that prepares the schedule based on 2-hop neighbors and handles the dynamic topology changes with low complexity.

In the data transmission phase, the owner nodes always get priority over the non-owner nodes for data transmission. In case, the owner nodes have no data to sent at a particular point of time then the non-owner nodes can use that slot for data transmission. However, the non-owner nodes with higher priority compete with one another to use that slot for data transmission. Hence, the chance of collision is reduced as the number of competing nodes are less. The nodes having more data to transmit are assigned with higher priority so as to utilize the channel in a better way.

Zhuo et al. [16] proposed the Queue-MAC protocol which maintains a queue of nodes those who are willing to transmit data. This protocol dynamically adapts the duty cycle using the queue length of the nodes. When the queue length gets increased i.e. traffic increases, the protocol extends the active CSMA period by adding dynamic TDMA time slots. Hence, at low load it preserves energy and at the same time at high load, it provides high throughput.

This adaptive CSMA/TDMA hybrid MAC is an improvement over the IEEE 802.15.4 protocol. IEEE 802.15.4 is a standard for Low-rate wireless personal area networks (LR-WPANs). The IEEE 802.15.4 standard operates in 2 modes i.e. beacon enabled mode and non-beacon mode.

In non-beacon mode, CSMA/CA is used as the MAC layer protocol. In beacon enabled mode, a special node called the PAN coordinator sends beacon frames in a regular time interval (i.e. beacon interval) to identify its PAN. The beacon interval consists of 2 periods

- Contention Access Period (CAP): During this period nodes get access of the channel using slotted CSMA/CA technique.
- Contention Free Period (CFP): During this period the nodes required to access the channel, and sent a request to the coordinator which gives a time slot to the node.

In this protocol, during the Contention Access Period (CAP), nodes can access the channel using an adaptive CSMA/TDMA hybrid MAC protocol rather than slotted CSMA/CA protocol. The channel utilization and amount of data pending in the queue of each node decides the border between CSMA/CA and TDMA protocol during the Contention Access Period(CAP). The coordinator checks the queue state of each node and assigns a TDMA slot to the node in the descending order of their amount of pending data in the queue. Once a TDMA slot is assigned to a node for communication, it is no more going to participate in the contention for getting the channel using CSMA/CA and thereby reduces the number of nodes contending for the channel. Hence, reduces the collision and improves the performance in terms of energy consumption and throughput.

Rhee et al.[22] proposed a randomized distributed TDMA slot scheduling algorithm called DRAND. In this algorithm, each node can be either in one of the four states i.e. IDLE, REQUEST, GRANT, or RELEASE. Initially, each node is in the IDLE state. Then each node goes for a lottery, and whichever node wins the lottery that node goes into the REQUEST state and gets the chance to negotiate with their neighbors to choose a time slot. If a node is either in IDLE or RELEASE state and receives a request message from its neighbors then the receiving node sends back a grant message and itself goes to GRANT state. While a node receives a request message being in the REQUEST or GRANT state then the receiver sends back a reject message. After receiving the reject message the sender node goes back to IDLE state. When a node, who is in the process of choosing a time slot, receives grant message from all of its neighbors then it enters into RELEASE state and then broadcasts the release message containing the information regarding the chosen slot.

Bhatia et al. [4] proposed an RD-TDMA scheduling algorithm for handling collision in WSNs. Most of the existing TDMA scheduling algorithm tries to find out an optimal schedule. However, this scheduling algorithm prepares a feasible schedule to handle the correlated contention. When two or more nodes transmit data at the same time and collision occurs at any of the receivers, then this situation is referred as a correlated contention. In the case of correlated contention, the contention occurs for a very short duration. Hence, RD-TDMA algorithm prepares a feasible schedule within a short period of time, which helps in handling the correlated contention in a better way. The RD-TDMA algorithm proceeds in the following way.

- The whole time line is divided into a number of fixed sized frames and each frame is consisting of no. of fixed time slots (i.e. the maximum of all the 2-hop neighbors) , under the assumption that nodes are not synchronized with reference to a global clock.
- Initially, each node tries to occupy a slot by broadcasting a request message to its one-hop neighbor.
- When a node receives this request message, either it sends a grant or reject message to the sender.
- It sends a grant message to the sender if the node itself has not requested for the same slot or it has not granted the same slot to any other node. Otherwise, send a reject message.
- When a node requests for a particular slot and it is granted by all of its neighbors through the grant message then the requested node is assigned to that slot. In case, it receives a reject message from any of its one-hop neighbor nodes then it stops there and reiterates the same process with another available slot.
- Once a node is allotted to a slot then it broadcasts the node allotment information to its two-hop neighbors from time to time. This helps the two-hop neighbors of the node to refrain from either request for getting that slot or grant that slot to others.

This paper is mainly focusing on preparing both feasible as well as optimal schedule, which will handle the correlated contention as well as minimize the number of slots to reduce the latency. In order to do so, our proposed algorithm initially prepares a feasible schedule and then fine tuned the feasible schedule in a novel way that reduces the number of slots to achieve the desired objective.

## III. Proposed HDSS Algorithm

The proposed HDSS algorithm initially prepares a feasible schedule to avoid the collision. Once a feasible schedule gets prepared, it is further fine tuned in quick time to minimize the number allotted slots with maintaining the feasibility of the newly prepared schedule intact. In order to do so, the proposed HDSS algorithm goes through following three phases.

- Calculate the maximum two-hop neighbors count at each node by exchanging the number of two-hop neighbor among each other.
- Allotment of slot to each sensor node in such a way that no two-hop neighbors are alloted the same slot thereby handling the collision during data transmission.
- Reduce total number of slots in a quick time and also handle the collision.

Our proposed algorithm use the following assumptions to work correctly.

- All the sensor nodes are static and homogeneous in nature.
- Uniform random deployment of sensor nodes.
- Local clocks of all the sensor nodes are synchronized.
- Packet transmission happens under ideal condition i.e. during packet transmission, there is no loss or corruption of packets.

The set of common notations used in our algorithm is summarizes in Table I. Table II summarizes the set of notations used for various message exchanges and Table III summarizes various data structures maintained at each node.

TABLE I
SET OF COMMON NOTATIONS USED

| Notation | Description |
|---|---|
| $S_{A_i}$ | Original slot number allotted to the $i^{th}$ node |
| $S_{R_i}$ | Reallotted slot number to the $i^{th}$ node |
| N | Total number of nodes |
| $N_{sq\_unit}$ | Number of nodes per square meter |
| $A_{tot}$ | Total deployment area in square meter |
| $S_{TA}$ | Total number of slots allotted for a feasible schedule |
| $N_{Max}$ | Maximum 2-hop neighbors count |
| $N_{Avg}$ | Average 2-hop neighbors count |
| $S_{RA}$ | Reduced number of allotted slots as per the ratio of average 2-hop neighbors count to maximum 2-hop neighbors count |
| $S_{FA}$ | Total number of slots allotted finally after the reallotment process |
| $O_{(i,j)}$ | $i^{th}$ node is the owner node of $j^{th}$ slot |
| $N_{(i,j)}$ | $i^{th}$ node is the non-owner node of $j^{th}$ slot |
| $RS_i$ | Re-allotment slot information at $i^{th}$ node |

TABLE II
SET OF NOTATIONS USED FOR VARIOUS MESSAGE EXCHANGES

| Notation | Description |
|---|---|
| $ND1_{Req}$ | One-hop neighbor discovery request message |
| $ND1_{Res}$ | One-hop neighbor discovery response message |
| $ND2_{Req}$ | Two-hop neighbor discovery request message |
| $NC2_{Max}$ | Max two-hop neighbor count message |
| $SA_{Req}$ | Slot allotment request message |
| $SG_{Res}$ | Slot grant response message |
| $SR_{Res}$ | Slot reject response message |
| $SA_{Succ}$ | Slot allotment success message |

TABLE III
SET OF INFORMATION MAINTAINED AT EACH NODE

| Notation | Description |
|---|---|
| $N1_i$ | List of one-hop neighbors of $i^{th}$ node |
| $N2_i$ | List of two-hop neighbors of $i^{th}$ node |
| $N2_{Max}$ | Maximum two-hop neighbors count of the whole network |
| $AS_i$ | list of available slots at node $i$ |
| $SS_{(i,j)}$ | Status of slot $j$ at node $i$ |

### A. Phase1: Two-hop Neighbor Discovery

In this phase, we follow the steps as given below to calculate the two-hop neighbors count at each sensor nodes. The two-hop neighbors count further helps in preparing the feasible schedule.

*[Step 1:]* Initially each sensor node $i$ broadcasts an one-hop neighbor discovery request ($ND1_{Req}$) message.

*[Step 2:]* Nodes that receives $ND1_{Req}$ message generate an one-hop neighbor discovery response ($ND1_{Res}$) message and send it back to the $i^{th}$ node.

*[Step 3:]* The nodes from which the node $i$ has received the $ND1_{Res}$ message are considered as the list of one-hop neighbors ($N1_i$) of node $i$.

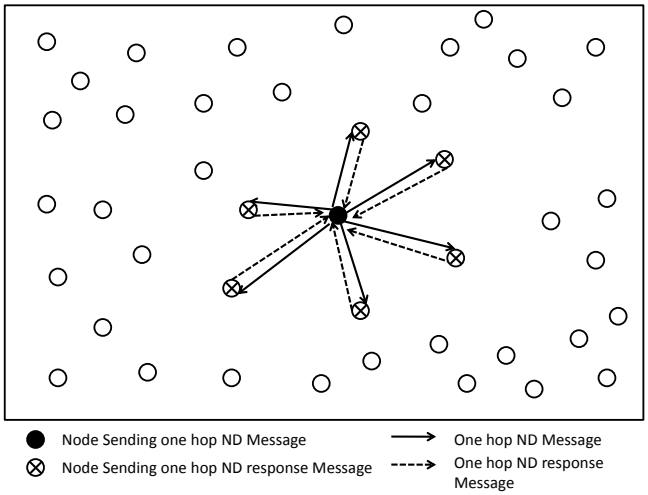


Fig. 1. One-hop neighbor discovery

*[Step 4:]* After discovering the list of one-hop neighbors, i.e. $N1_i$, a two-hop neighbor-discovery request ($ND2_{Req}$) message that consists of the list of one-hop neighbor's for node $i$.

*[Step 5:]* Each node $j$ that receives the two-hop neighbor discovery request ($ND2_{Req}$) message populates its two-hop neighbor list ($N2_j$) using the one-hop neighbor list ($N1_i$) of node $i$.

*[Step 6:]* After few such iterations each node $i$ populates their final two-hop neighbors list($N2_i$) and then broadcasts a $NC2_{Max}$ message.

*[Step 7:]* Each node $i$ that receives the $NC2_{Max}$ message verifies whether the received maximum two-hop neighbors count ($N_{Max}$) is more than the maximum two-hop neighbors count ($N_{Max}$) currently held at its own end. If so, then $N_{Max}$ is updated at node $i$ with the received maximum two-hop neighbors count. Then node $i$ broadcasts the $NC2_{Max}$ message over the whole network. Along with this, the node also calculates the average two-hop neighbors count($N_{Avg}$) with the help of received maximum two-hop neighbors count($N_{Max}$) from a particular node. During the calculation of average two-hop neighbors count($N_{Avg}$), a node rejects all the duplicate maximum two-hop neighbors count ($N_{Max}$) of a particular node received from its different neighbors. Initially, the feasible schedule is prepared with requisite number of slots using $N_{Max}$, where $N_{Avg}$ helps in further minimizing the number of slots in the feasible schedule.

As per figure 1, the node that represented with black color broadcasts an $ND1_{Req}$ message to find its one-hop neighbors. The cross marked nodes (who receives this $ND1_{Req}$ message) send back a $ND1_{Res}$ message to the black color node. Hence, the cross marked nodes become the one-hop neighbors of the black color node.

As per figure 2, after the cross marked nodes find their one-hop neighbors, the dotted nodes become the one-hop neighbors of the cross marked nodes. Then, the black color node broadcasts a $ND2_{Req}$ message to their neighbors for the discovery of the two-hop neighbors. Each cross marked nodes sends a $ND2_{Res}$ message that contains the list of its one-hop

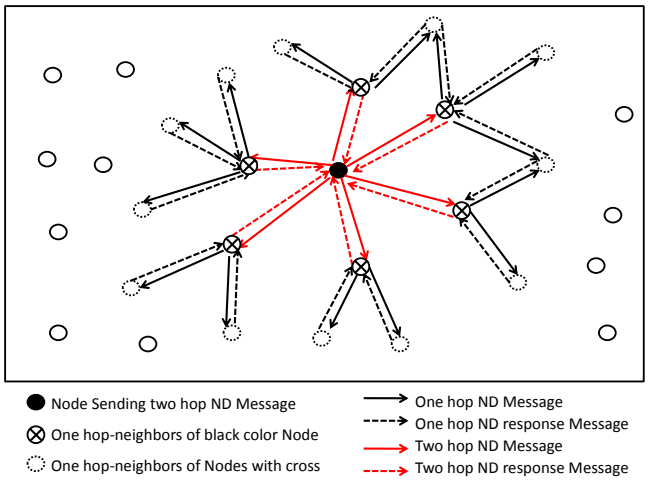


Fig. 2. Two-hop neighbor discovery

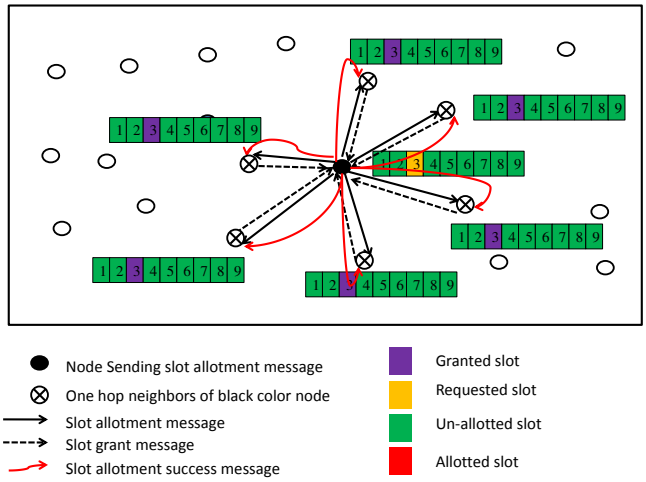


Fig. 3. Success in slot allotment for nodes

neighbors. Finally, the cross marked nodes and dotted nodes form the two-hop neighbors of the black color node.

### B. Phase2: Allotment of Slots

After discovery of two-hop neighbors and $N2_{Max}$ information, each node $i$ prepares a feasible schedule with number of slots ($S_{TA}$) equals to the $N2_{Max}$. During the slot allotment process each slot is associated with one of the states represented in Table IV.

TABLE IV
VARIOUS STATE NOTATIONS OF EACH SLOT

| Notation | Description |
|---|---|
| SU | Slot is in Un-allotted state |
| SR | Slot is in Requested state |
| SG | Slot is in Grant state |
| SA | Slot is in Allotted state |

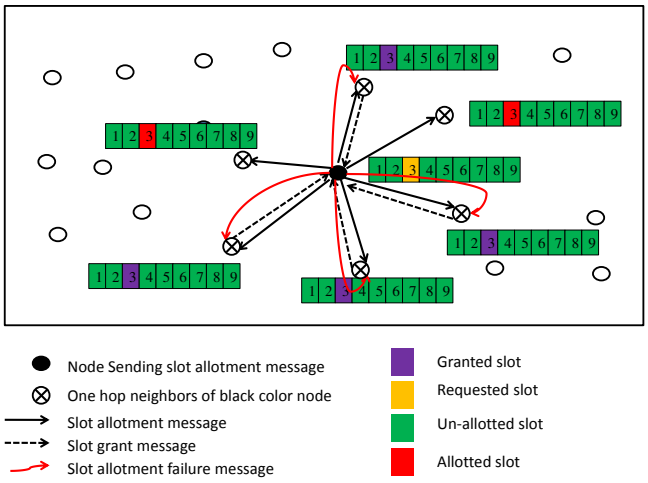


Fig. 4. Failure in slot allotment for nodes

The details of the slot allotment process proceeds as follows.
*[Step 1:]* Every slot in a feasible schedule is assigned with a state "un-allotted" (SU).
*[Step 2:]* Each node $i$ selects a slot randomly from the currently available "un-allotted" list. Then the selected slot is updated with state as "requested" (SR). Finally, node $i$ sends a slot allotment request message ($SA_{Req}$) that carries the requested slot information to its neighbors.
*[Step 3:]* Each node that receives the $SA_{Req}$ message grant the requested slot if either of the following criteria matches or else rejects the request for slot allotment.

- The receiver node itself has not requested for the same slot earlier.
- The receiver node has not yet granted the same slot to any other node.

*[Step 4:]* In case the slot is granted then the slot status is updated to granted(SG) and a $SG_{Res}$ message is sent to its neighbors. The intended receiver of the $SG_{Res}$ message keeps the node id of the sender.
*[Step 5:]* If node $i$ receives $SG_{Res}$ message from all of its neighbors then the state of the requested slot is updated to allotted (SA) and broadcasts a $SA_{Succ}$ message to its neighbors or else, the state of the slot is updated to un-allotted(SU) and then go back to step-2 again.
*[Step 6:]* Each node $i$ that receives the $SA_{Succ}$ message update state of the slot to be allotted(SA). In case, node $i$ did not receive the slot allotment success message ($SA_{Succ}$) within a certain period of time, after giving a grant to a particular slot, then node $i$ update the slot status to be "un-allotted" (SU) at its own end considering the slot allotment fails.

As per figure 3, the node represented with black color chooses a slot randomly and broadcasts a $SA_{Req}$ message to its neighbors. Once the black color node receives the $SG_{Res}$ message from all of its neighbors, it allots the requested slot for itself and inform the same to its neighbors through $SA_{Succ}$ message. The same requested slot will be updated at the neighbors end to state SA when the neighbor node receives the $SA_{Succ}$ message.

figure 3 shows an illustration of success in slot allotment where as figure 4 shows a case of failure in slot allotment.

According to figure 4, the node represented with black color has not received $SG_{Res}$ message from two of its one-hop neighbor nodes as the requested slot at these nodes are already in the allotted(SA) state before receiving the $SA_{Req}$ message. As a consequence, the node with black color updates the state of that slot to un-allotted(SU). The nodes who have given grant if did not receive the $SA_{Succ}$ message within a certain period of time then updates the state of the granted slot back to un-alloted.

### C. Phase3: Reallotment of Slots

In order to minimize the length of the feasible schedule, the initially allotted slots($S_{TA}$) are reduced as per the ratio of average two-hop neighbor count ($N_{Avg}$) to the maximum two-hop neighbor count ($N_{Max}$). For example, If the ratio of $N_{Avg}$ to the $N_{Max}$ is 1:2 then the number of reallotted slots to be half of the currently alloted slots, i.e. $S_{TA}$ . The process of slot reallotment consists of following steps.

*[Step 1:]* Reduce the number of slots, i.e. $S_{RA}$, based on the ratio of average two-hop neighbor count ($N_{Avg}$) to the maximum two-hop neighbor count ($N_{Max}$). Eventually, each node $i$ checks whether it is eligible for reallotment or not by comparing $S_{A_i}$ with $S_{RA}$. In case, $S_{A_i} > S_{RA}$, the node $i$ is reallotted to a slot $S_{A_i} = (N_{max} - S_{A_i})+1$.

*[Step 2:]* A node is referred as owner node to a slot, i.e. $O_{(i,j)}$, where $i^{th}$ node alloted to $j^{th}$ slot, if it is initially alloted to that slot. If a node is later reallotted to a slot then it is referred as non-owner node for that slot, i.e. $N_{(i,j)}$.

*[Step 3:]* After the reallotment of slots, data transmission is carried out by each node $i$ to verify which non-owner nodes, i.e. $N_{(i,j)}$ are in a collision during the same. In order to do so, each non-owner sensor node checks whether the medium is free or busy using CSMA before their data transmission. Once the non-owner node finds the medium is busy then the collision will happen if the same node will transmit data packet along with the owner nodes.

*[Step 4:]* The non-owner nodes, i.e. $N_{(i,j)}$ which are in a collision are reallotted back to their original slots. The reallotted slot information, i.e. $RS_i$ is broadcasted by the non-owner nodes to its one-hop neighbors. A sensor node that receives the reallotted slot information updates the same at its own end if the received information is a new one, otherwise, it ignores the message. In case, the information is a new one, it again broadcasts the updated information to its one-hop neighbors and so on. Finally, every node has the updated information of how many slots are currently available in the feasible schedule.

As per Figure 5, originally nine slots are available in the feasible schedule. After reallotment, the total number of slots reduced to half, as the ratio of "average two-hop density" to the "maximum two-hop density" is 1:2. During data transmission, out of the five allotted slots two of them i.e. slot number two and four are in collision. Finally, the non owner nodes to whom the slot number two and four get allotted are now reallotted with slot number six and seven. Hence, the final number of slots allotted to handle collision become seven.

The whole process has been carried out once at the beginning to prepare the schedule. This schedule will handle collision during data transmission, and the reduction in the length of the schedule (as compared to the feasible schedule) allows transmitting data with reduced latency.

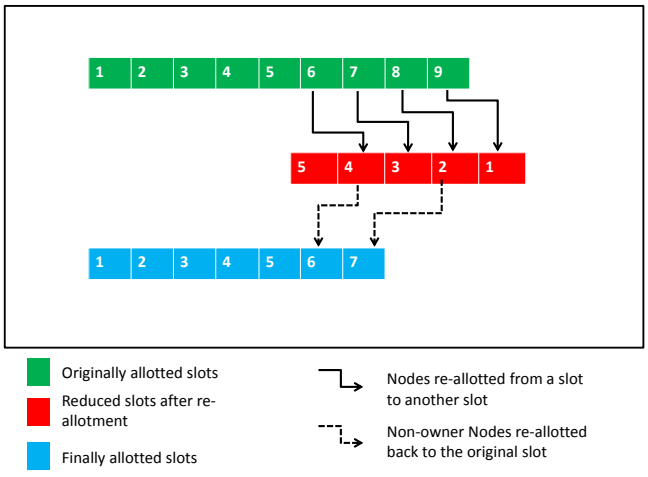


Fig. 5. Slot reallotment for nodes

## IV. Analysis of the HDSS Algorithm

We analyze our HDSS algorithm to show that the number of slots allotted originally to prepare a feasible schedule can be reduced and after such reduction in the number of allotted slots still the schedule remain feasible.

Let us consider a node $N_i$ from the given figure 6. The cross marked nodes are considered as the one-hop neighbors of $N_i$. The nodes with dots are considered as the two-hop neighbors of $N_i$. Let us assume that the number of two-hop neighbors of $N_i$ are $Cnt_{2N}$. Initially, while preparing the feasible schedule, the maximum of two-hop neighbors counts, i.e. $N_{max}$ is taken in to consideration. By considering the transmission range( $T_r$) and with basic assumption, i.e. the nodes are deployed in a random uniform manner, the $N_{max}$ can be computed as follows.

$$N_{\text{sq\_unit}} = N/A_{\text{tot}}. \tag{1}$$

$$N_{\text{max}} = (\pi * 2T_r)^2 * N_{\text{sq\_unit}}. \tag{2}$$

Initially, we assume that $N_{max} = Cnt_{2N}$ as per the figure 6, so there will be at least $N_{max}$ number of slots are required to prepare a feasible schedule, where $S_{TA} = N_{max}$.

As per the figure 6, Let us consider two nodes named as $N_j$ and $N_k$ which are two-hop neighbors of node $N_i$ and are assigned to the same slot in a feasible schedule. If both these nodes start transmitting data packets at the same time still these packets will not collide with each other even if they are the two-hop neighbors of each other as they are almost four-hop away from each other. Hence a feasible schedule can be prepared with $S_{TA} = N_{max}$-1 number of slots.

It proves that there is a possibility of reducing the number of originally allotted slots and at the same time it provides a feasible schedule.

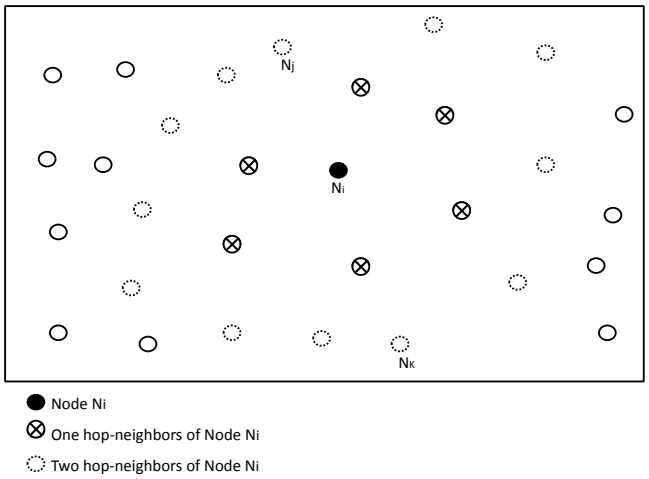


Fig. 6. Two-hop Neighbors of a node

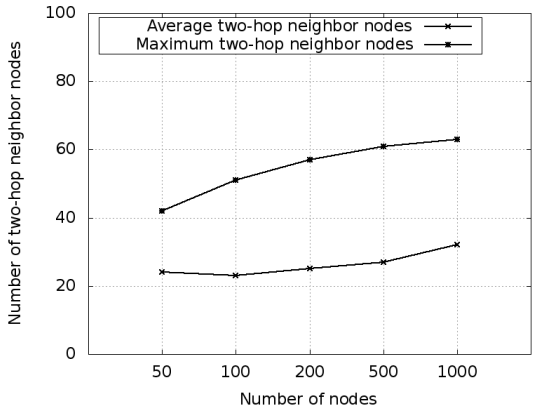


Fig. 7. With fixed node density, average two-hop neighbors compared to maximum two-hop neighbors

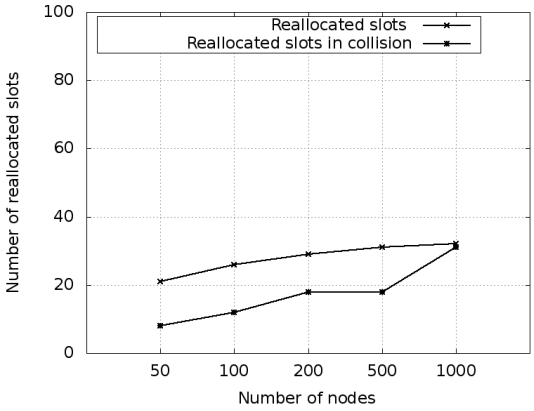


Fig. 8. With fixed node density, number of slots in collision after reallotment as compared to the total number of slots after reallotment

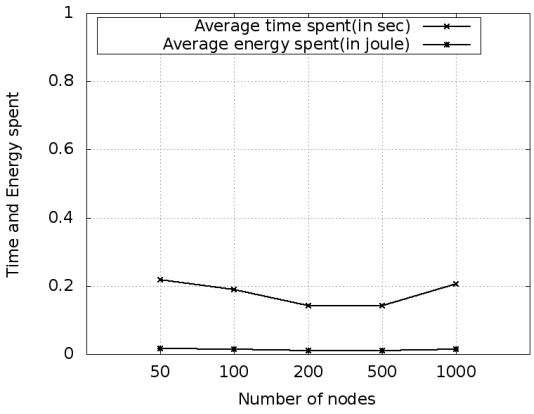


Fig. 9. With fixed node density, time and energy spent for reallotment of slots

## V. Simulation Experiments and Results

The performance of our proposed algorithm is carried out using castalia simulator. In this simulation, the sensor nodes are uniformly distributed in a random manner. For this simulation, the number of nodes deployed is varied from 50 to 1000 with a fixed as well as varying density. Various sensor node parameters such as power transmission level, data transmission rate,and energy consumption are taken into consideration as per the information available in cc2420 data sheet and TelosB data sheet.

Figure 7 shows the computed average two-hop neighbors vs the actual two-hop neighbors in different size network. In this figure, the average two-hop neighbors remain almost 50% of the actual two-hop neighbors. As the average two-hop neighbors are almost half of the actual two-hop neighbors, hence reduction of the originally allocated slots to it's half will improve the performance. Our proposed algorithm proves the same.

Figure 8 shows the number of slots are in collision after reallotment. From this figure, it shows that the number of slots in collision increases with the increase in number of sensor nodes. With the increase in number of sensor nodes, the number of nodes allotted to a particular slot increases. This leads to the possibility of the increase in collision due to increase in number of sensor nodes.

Figure 9 shows the time and energy spent during slot reallotment. This result shows that the average time and energy spent almost remains same as the number of node increases as the proposed algorithm is distributed in nature.

Figure 10 shows a comparison of the number of slots allotted to the final schedule in our proposed HDSS algorithm with the existing RD-TDMA and DRAND algorithm. Here, the result shows that the number of slots allotted to handle the collision in our proposed HDSS algorithm is less as compared to the RD-TDMA and DRAND algorithm, i.e. our proposed HDSS algorithm performs better than RD-TDMA and DRAND algorithm w.r.to the number of slots allotted to handle collision.

Figure 11 shows a comparison of the number of allotted slots, reallotted slots, and number of slots in collision after reallotment. It shows that the number of slots which are in collision almost remains the same with varying area. As the number of nodes remains same hence, the chance of collision also remains the same with varying number of nodes.

Figure 12 shows that the latency in data transmission is

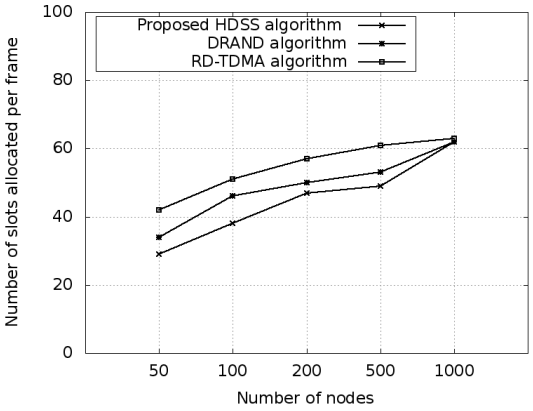


Fig. 10. With fixed node density, number of slots allotted for the feasible schedule using proposed HDSS, DRAND, and RD-TDMA algorithm

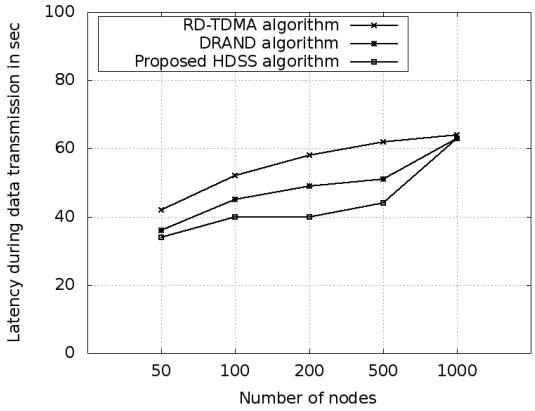


Fig. 12. Latency in data transmission in proposed HDSS, DRAND, and RD-TDMA algorithm

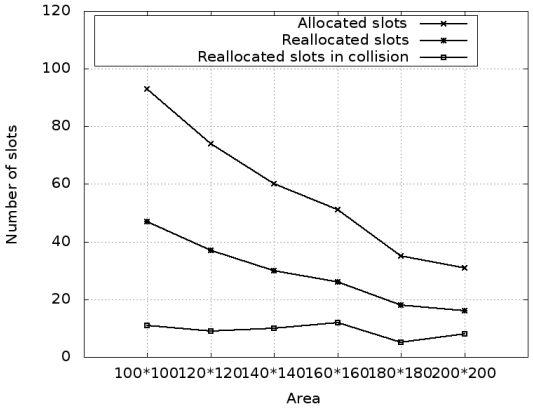


Fig. 11. With varying node density, comparison of the number of allotted slots, reallotted slots, and slots in collision after reallotment

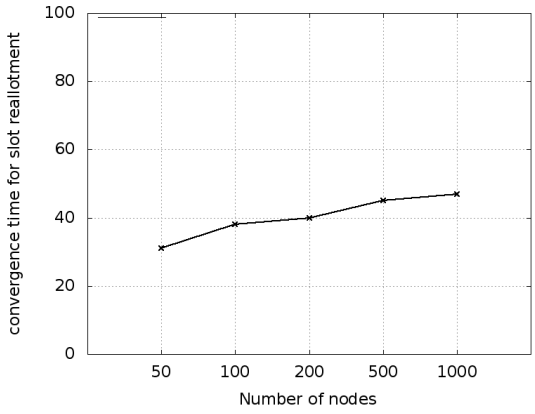


Fig. 13. Convergence time for slot reallotment

less in our proposed HDSS algorithm as compared to the RD-TDMA and DRAND. In case of the HDSS algorithm as less number of slots are used to prepare the feasible schedule as compared to RD-TDMA and DRAND hence the result.

TABLE V
AVERAGE ENERGY SPENT(IN JOULES) FOR VARIOUS PHASES DURING PREPARATION OF THE FEASIBLE SCHEDULE

| Phases | HDSS | DRAND |
|---|---|---|
| Neighbor Discovery | 1.01 | 0.93 |
| Slot Allotment | 5.12 | 5.15 |
| Slot Re-allotment | 0.02 | - |

Table V shows that the proposed HDSS algorithm consumes almost the same amount of energy as compared to the DRAND. Most importantly, with the expense of same amount of energy, the proposed algorithm can able to reduce the number of slots to prepare the feasible schedule as compared to DRAND.

Figure 13 shows that with increase in number of sensor nodes the convergence time for reallotment of slots increase very slowly.

## VI. CONCLUSION

In this paper, we have proposed a slot scheduling algorithm, named HDSS, for wireless sensor networks to handle collision during communication. In the first phase of the proposed algorithm, two-hop neighbors are calculated for each node in WSN. Based on the maximum two-hop neighbors count over the whole network, the total number of slots for a feasible schedule is decided. In the next phase, reallotment of slots is done by reallocating the nodes from the last to first , last but one to the second and so on. Finally, after reallotment, the number of slots in collision is found out and accordingly the final schedule is prepared. This reduces the number of slots allocated for handling collision. The efficiency of the proposed algorithm has been evaluated with fixed as well as varying node density. Our proposed algorithm is compared with an existing distributed slot scheduling algorithm called RD-TDMA. Comparison results show that the proposed HDSS algorithm outperforms the existing RD-TDMA algorithm to handle the collision with reduced number of slots and at the same time improves the efficiency of data communication in the channel with reduced latency.

## REFERENCES


[1] Ye W, Heidemann J, Estrin D. An energy-efficient MAC protocol for wireless sensor networks. In: Proceedings of the IEEE Infocom; 23-27 June 2002; New York, NY, USA. New York, NY, USA: IEEE. pp. 1567-1576, DOI:10.1109/INFCOM.2002.1019408.

[2] Dam T, Langendoen K. An adaptive energy-efficient MAC protocol for wireless sensor networks. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems; 05-07 November 2003; Los Angeles, California, USA. New York, NY, USA: ACM. pp. 171-180, DOI:10.1145/958491.958512.

[3] Li S, Qian D, Liu Y, Tong J. Adaptive distributed randomized TDMA scheduling for clustered wireless sensor networks. In: Proceedings of 2007 International Conference on Wireless Communications, Networking and Mobile Computing; 21-25 September 2007; Shanghai, China. New York, NY, USA: IEEE. pp. 2688-2691, DOI:10.1109/WICOM.2007.668.

[4] Bhatia A, Hansdah R. RD-TDMA: A randomized distributed TDMA scheduling for correlated contention in WSNs. In: Proceedings of 2014 28th International Conference on Advanced Information Networking and Applications Workshops; 13-16 May 2014; Victoria, BC, Canada. New York, NY, USA: IEEE, DOI: 10.1109/WAINA.2013.23.

[5] Lenka M, Swain A, Sahoo M. Distributed slot scheduling algorithm for hybrid CSMA/TDMA MAC in wireless sensor networks. In: Proceedings of 2016 IEEE International Conference on Networking, Architecture and Storage (NAS); 8-10 August 2016; Long Beach, CA, USA. New York, NY, USA: IEEE, DOI:10.1109/NAS.2016.7549396.

[6] Ergen S, Varaiya P. TDMA scheduling algorithms for wireless sensor networks. WIREL NETW 2010; 16: 985–997, DOI:10.1007/s11276-009-0183-0.

[7] Dobslaw F, Zhang T, Gidlund M. End-to-End reliability-aware scheduling for wireless sensor networks. IEEE T IND INFORM 2016; 12: 758-767, DOI: 10.1109/TII.2014.2382335.

[8] Mao J, Wu Z, Wu X. A TDMA scheduling scheme for many-to-one communications in wireless sensor networks. COMPUT COMMUN 2007; 30: 863-872, DOI: 10.1016/j.comcom.2006.10.006.

[9] Yang D, Xu Y, Wang H, Zheng T, Zhang H, Zhang H, Gidlund M. Assignment of segmented slots enabling reliable real-time transmission in industrial wireless sensor networks. IEEE T IND ELECTRON 2015; 62: 3966-3977, DOI: 10.1109/TIE.2015.2402642.

[10] Guo L, Li Y, Cai Z. Minimum-latency aggregation scheduling in wireless sensor network. J COMB OPTIM 2016; 31: 279–310, DOI: 10.1007/s10878-014-9748-7.

[11] Ramanathan S, Lloyd E. Scheduling algorithms for multihop radio networks. IEEE ACM T NETWORK 1993; 1: 166-177, DOI: 10.1109/90.222924.

[12] Wang Y, Henning I. A deterministic distributed TDMA scheduling algorithm for wireless sensor networks. In: Proceedings of 2007 International Conference on Wireless Communications, Networking and Mobile Computing; 21-25 September 2007; Shanghai, China. New York, NY, USA: IEEE. pp. 2759-2762, DOI: 10.1109/WICOM.2007.685.

[13] Gandham S, Zhang Y, Huang Q. Distributed time-optimal scheduling for convergecast in wireless sensor networks. COMM COM INF SC 2008; 52: 610-629, DOI: 10.1016/j.comnet.2007.10.011.

[14] Rhee I, Warrier A, Aia M, Min J, Sichitiu M. Z-MAC: a hybrid MAC for wireless sensor networks. IEEE ACM T NETWORK 2008; 16: 511-524, DOI: 10.1109/TNET.2007.900704.

[15] Slama I, Jouaber B, Zeghlache D. Priority-based hybrid MAC for energy efficiency in wireless sensor networks. Wireless Sensor Network 2010; 2: 755-767, DOI: 10.4236/wsn.2010.210091 .

[16] Zhuo S, Song Y, Wang Z, Wang Z. Queue-MAC: A queue-length aware hybrid CSMA/TDMA MAC protocol for providing dynamic adaptation to traffic and duty-cycle variation in wireless sensor networks. In: Proceedings of 2012 9th IEEE International Workshop on Factory Communication Systems; 21-24 May 2012; Lemgo, Germany. New York, NY, USA: IEEE, DOI: 10.1109/WFCS.2012.6242552.

[17] Ergen S, Varaiya P. PEDAMACS: Power efficient and delay aware medium access protocol for sensor networks. IEEE T MOBILE COMPUT 2006; 5: 920–930, DOI: 10.1109/TMC.2006.100.

[18] Sun Y, Du S, Gurewitz O, Johnson D. Dw-mac: a low latency, energy efficient demand-wakeup mac protocol for wireless sensor networks. In: Proceedings of the 9th ACM international; 26-30 May 2008; Hong Kong, Hong Kong, China. New York, NY, USA: ACM. pp. 53-62, DOI; 10.1145/1374618.1374627.

[19] Oller J, Demirkol I, Casademont J, Paradells J, Gamm G, Reindl L. Has time come to switch from duty-cycled MAC protocols to wake-up radio for wireless sensor networks?. IEEE ACM T NETWORK 2016; 24: 674-687, DOI: 10.1109/TNET.2014.2387314.

[20] Fafoutis X, Mauro A, Vithanage M, Dragoni N. Receiver-initiated medium access control protocols for wireless sensor networks. COMM COM INF SC 2015; 76: 55-74, DOI: 10.1016/j.comnet.2014.11.002.

[21] Segun A, Ao A, Eo O. A survey of medium access control protocols in wireless sensor network. International Journal of Computer Applications 2015; 116: 1-8, DOI: 10.5120/20465-9980.

[22] Rhee I, Warrier A, Aia M, Min J, Xu L. DRAND: Distributed randomized TDMA scheduling for wireless ad hoc networks. IEEE T MOBILE COMPUT 2009; 8: 1384-1396, DOI: 10.1109/TMC.2009.59.

[23] Slama I, Shrestha B, Jouaber B, Zeghlache D, Erke T. DNIB: Distributed neighborhood information based TDMA scheduling for wireless sensor networks. In: Proceedings of 2008 IEEE 68th Vehicular Technology Conference; 21-24 September 2008; Calgary, BC, Canada. New York, NY, USA: IEEE, DOI: 10.1109/VETECF.2008.25.

[24] Ahmad A, Hanzálek Z. Distributed Real Time TDMA Scheduling Algorithm for Tree Topology WSNs. IFAC-PapersOnLine 2017. DOI: 10.1016/j.ifacol.2017.08.1484.

[25] Lee J, Cho S. Tree TDMA MAC Algorithm Using Time and Frequency Slot Allocations in Tree-Based WSNs. Wireless Personal Communications 2017. DOI: 10.1007/s11277-016-3938-9.

[26] Danmanee T, Nakorn K, Rojviboonchai K. CU-MAC: A Duty-Cycle MAC Protocol for Internet of Things in Wireless Sensor Networks. ECTI Transactions on Electrical Engineering, Electronics, and Communications 2018.


**Manas Ranjan Lenka** received his M.Tech degree in Computer Science and Engineering from Biju Patnaik University of Technology (BPUT), Bhubaneswar, India in 2010. Currently, he is working as an assistant professor in the School of Computer Engineering, KIIT Deemed To Be University, Bhubaneswar, India. His research interests include wireless sensor networks, Internet of Things.

**Amulya Ratna Swain** received his M.E. Degree in Software Engineering from Jadavpur University, Calcutta, India in 2006. He received the PhD degree in Computer Science from Indian Institute of Science, Bangalore, India, in 2013. Currently, he is working as an associate professor in the School of Computer Engineering, KIIT Deemed To Be University, Bhubaneswar, India. His research interests include wireless sensor networks, distributed computing and operating systems.

**Biraja Prasad Nayak** received his M. Tech. degree in Computer Science and Engineering from School of Computer Engineering, Kalinga Institute of Industrial Technology (KIIT) Deemed To Be University, Bhubaneswar, Odisha, India in 2017. His research interests are Wireless Sensor Networks and Mobile Computing.