IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 5 July 2023, accepted 1 August 2023, date of publication 9 August 2023, date of current version 15 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3303876

RESEARCH ARTICLE

GS-MAC: A Scalable and Energy Efficient MAC Protocol for Greenhouse Monitoring and Control Using Wireless Sensor Networks

MIKE MAJHAM[1], MARCO P. MWAIMU[1], THOMAS KIVEVELE[2], AND RAMADHANI S. SINDE[1], (Member, IEEE)

[1]School of Computational and Communication Science and Engineering, The Nelson Mandela African Institution of Science and Technology, Arusha 44740, Tanzania

[2]School of Materials, Energy, Water, and Environmental Sciences, The Nelson Mandela African Institution of Science and Technology, Arusha 44740, Tanzania

Corresponding author: Mike Majham (majhamm@nm-aist.ac.tz)

ABSTRACT Wireless sensor networks (WSNs) are widely used in agricultural greenhouses to monitor and control farming-related parameters. These networks are composed of multiple sensor nodes, usually deployed in an ad hoc fashion. But the nodes mostly run on batteries. So if a node fails or loses power, it creates areas in the network with no sensor coverage, consequently affecting the entire system. Therefore minimizing energy use is essential for extending the network lifetime. Researchers have proposed multiple energy-saving schemes in the past but the majority have not eliminated the sources of energy waste, and are not suitable for greenhouse applications. In light of this, we propose GS-MAC, a scalable and energy-efficient medium-access-control (MAC) protocol specialized for greenhouse monitoring and control: Unlike previous designs, our technique is applicable to both homogenous and heterogeneous settings. To minimize power use, nodes periodically sleep, but GS-MAC avoids periodic node synchronizations, unlike traditional duty cycling mechanisms. Instead, nodes use coordinated universal time (UTC) to maintain strict schedules to avoid energy waste and maintain constant low-duty cycles even with increasing node density. GS-MAC also uses short node addresses to reduce packet overheads. Finally, GS-MAC adopts a contention approach on reserved time slots scheduled between communication rounds to maintain scalability. Our work is evaluated on MATLAB, with simulation parameters obtained from actual hardware. The experiment results show that GS-MAC is more efficient by at least 2.7 times compared to previous research in terms of duty cycling, energy consumption, and network lifetime, in exchange for increased delays.

INDEX TERMS Duty cycle, energy efficiency, heterogeneous, homogenous, MAC protocol, network lifetime, scalability, wireless sensor network.

I. INTRODUCTION

In recent years Wireless Sensor networks (WSNs) have been widely used in agricultural greenhouses to monitor and control farming-related parameters to attain precision agriculture [1]. Without having to heavily rely on the weather or climate of the surroundings, this type of farming usually attains the best quality and crop yield. Therefore multiple studies have focused on devising wireless networks for greenhouse parameter monitoring and control [2], [3], [4], [5].

But despite its great results, WSNs' energy efficiency is still difficult to solve. As the majority of WSN nodes are battery-powered, if one node runs out of power or fails, it leaves gaps in the network's sensor coverage, which decreases the useful lifespan of the entire system. Since replacing or recharging the batteries is difficult and expensive, reducing energy waste is essential to extending the WSN's usable lifetime. This task is made easier by the authors of [6] who correlated the power use from different sources

The associate editor coordinating the review of this manuscript and approving it for publication was Arun Prakash.

within a wireless sensor node using wireless nodes type eZ430-RF2500 from Texas Instruments and SHT11 temperature/humidity sensor as an external unit. It was found that when active, the microcontroller and sensing units only use 0.9 μA when in low-power mode whereas the radio unit consumes 21.2 mA when transmitting and 12.8 mA when receiving or idle listening. However, the radio unit only uses about 0.4 μA when in sleep mode. Therefore to minimize power use, the nodes should spend less time transmitting, receiving, or idle listening and more time in sleep mode but without compromising the communication process. We can achieve this by first identifying the sources of energy waste.

According to [7], the five main causes of energy waste in a WSN are *collisions*, *overhearing*, *control packet overheads*, *zidle listening*, and *over-emitting*. Collisions occur when multiple nodes transmit at the same tim; as a result, the transmitted packets get corrupted at the receiving end. So the packets must be retransmitted, increasing transmission time and power use. Control packet overheads are data that is included in the packets to help the proper functioning of the communication process but are not part of the intended information. Having many overheads increases the length of the packets, leading to more transmitting time, and consequently using more power. Idle listening happens when a node listens for potential incoming packets that are not sent. As we explained earlier, nodes consume about the same power when idle listening as when receiving. Overhearing occurs when a node receives packets intended for other nodes, wasting more energy. And finally, over-emitting, the final source of energy waste, occurs when a node sends packets when the receiving node is either out of range, sleeping, or otherwise not able to receive them. The average life of a WSN node is shortened by all these aforementioned energy losses. Therefore, minimizing them is very important. To reduce energy waste, a good Medium Access Control (MAC) protocol is required.

There are so many energy-saving MAC protocols that comply with IEEE 802.15.4, a standard that governs wireless communications of low-power, low-data-rate, short-range devices [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24]. These protocols use either a contention mechanism [7], [8], [9], [10], [11], time schedules [12], [13], [14], [15], [16], [17], or a combination of the two [18], [19], [20], [21], [22], [23], to access the shared medium. The schedule-based protocols are more energy-preserving than contention mechanisms because they have a duty cycle built-in with an inherent collision-free nature; however, they often have high complexity in design due to synchronization requirements [25]. On the other hand, contention-based schemes consume more energy but are less complex and have better scalability. Hybrid systems attempt to obtain the energy efficiency of schedule-based schemes while maintaining the scalability of contention methods. However, we have noticed that none of the existing studies has fully eliminated the sources of energy waste (see section II). Moreover, most of the proposed solutions are not suitable for greenhouse monitoring.

Therefore, this paper proposes Greenhouse Sensor MAC (GS-MAC), a new MAC protocol specifically designed for greenhouse monitoring. The protocol achieves better scalability with collision avoidance while also minimizing energy consumption by utilizing combined scheduling and contention schemes. GS-MAC adopts the scheduling approach by allowing nodes to form *clusters* where a specially selected node called *cluster head* collects the data and forwards it to the sink node or base station. All other nodes only communicate with the cluster head. The cluster head assigns all nodes strict time schedules for transmitting. Therefore nodes can adaptively sleep and wake up only when they have to transmit or expect to receive packets. All time stamps are based on the Universal Coordinated Time (UTC) [26]. Nodes can maintain this time with the help of real-time clock (RTC) modules embedded in them [27]. This technique eliminates collisions and minimizes energy waste associated with contention for the medium. However, maintaining scalability is not simple. To ensure that new nodes entering the network or nodes transferred to a different place may be smoothly synchronized with the network, GS-MAC utilizes a contention method. To accomplish this, the cluster head must wake up for a brief period during each sleep-wake-up cycle to monitor potential traffic from new nodes seeking to join. If additional nodes are found, GS-MAC uses a contention strategy to synchronize them with the existing nodes in the cluster. During this time, all other nodes are in their sleep states. As a result, GS-MAC is energy efficient and also retains excellent scalability.

The main contributions of this work include the following:

1) We avoid collisions, overhearing, over-emitting, and idle listening similar to modern scheduling schemes while also maintaining as good scalability as contention designs.
2) We reduce control packet overheads by eliminating synchronization overheads and also using short node addresses of 1 byte instead of the conventional 8-byte address.
3) We reduce transmission delays by avoiding periodic node synchronizations.
4) Unlike previous research, member nodes in our protocol consume less energy and maintain constant low power use even with the increase in node density.
5) Our protocol is applicable to both homogenous and heterogeneous networks.

The following is how the rest of the paper is structured: Prior works related to GS-MAC are described in Section II. The article then moves on to part III, which has a thorough description of GS-MAC. The performance of GS-MAC is then evaluated and compared to that of other protocols in Section IV. Finally, Section V concludes the paper.

II. RELATED WORKS

As sensor nodes inevitably expire if their batteries run out, increasing network longevity is a popular goal of WSN research. To reduce the main sources of energy waste, several MAC protocols have been suggested. These systems are

broadly classified into three groups: Time Division Multiple Access (TDMA) based protocols, contention-based schemes, and hybrid mechanisms [25]. Contention-based systems use carrier sense multiple access (CSMA) protocol before transmitting any traffic. They are divided into synchronous and asynchronous MAC. In synchronous MAC protocols, nearby sensor nodes periodically synchronize to sleep and wake up at the same time. For example, S-MAC forms virtual clusters among nearby nodes to synchronize their sleep periods [7] (i.e., to minimize idle listening, nodes periodically sleep). So, S-MAC keeps WSN nodes' radios in sleep mode when other nodes are transmitting. Also, as data flows, for applications that require store-and-forward processing, S-MAC uses message passing to minimize contention delays. One of its drawbacks is that S-MAC only supports one-hop data forwarding per communication cycle. This problem is solved in [8] by adding an adaptive listening approach to S-MAC. With adaptive listening, a sensor node that overhears its neighbor can briefly wake up after the broadcast. In this technique, if a node is the next hop for its neighbor, it can receive its neighbor's packets without having to wait for its scheduled listening time. Moreover, the time between transmissions is shortened. However, in [8], data packets can only be forwarded by a maximum of two hops per cycle. Another disadvantage of both [7] and [8] is an increase in latency, especially when the network experiences high traffic. Not only that but since nodes are scheduled to sleep and wake up at the same time, they stay awake (even if they do not expect to communicate) until other neighboring nodes finish communicating as well, before going back to sleep. This results in more energy waste. Reference [9] proposes to minimize the waste by using a regression technique to continually vary the nodes' duty cycles based on the traffic density, to minimize collisions and queuing delays. Moreover, [9] rotates the role of the cluster head among the member nodes based on their residual energies, to increase the network lifetime. Alternatively, in [10], nodes with the most important data or with the least residual energy transmit first to extend the WSN's useful life. Additionally, [10] uses variable contention windows to minimize collisions. However, all the aforementioned schemes, i.e., [7], [8], [9], [10] suffer from periodic node synchronization delays.

Asynchronous schemes, on the other hand, eliminate synchronization overheads because a node can choose its active periods without having to synchronize with the neighbors. A good example is FAWR, proposed by [11]. In FAWR, all nodes are embedded with a main radio for data transmission and a wake-up radio (WuR) for demand-driven node activation. The main radio is usually in sleep mode to minimize energy waste, but the WuR, which uses ultra-low energy consumption is always active to detect possible incoming wake-up calls (WuCs) and minimize delays. When a node wants to transmit data, it first sends a WuC through its WuR, which may be detected and processed by the WuR of the intended destination node. Then both the sending and receiving nodes turn on their main radios before data transmission begins. When communication ends, all nodes, except their WuRs resume their sleep states. To overcome the limited range of WuCs, a multi-hop scheme is proposed, to enable asynchronous transmissions between the nodes and a base station. FAWR outperforms modern duty-cycling schemes in terms of power usage, transmission delay, and collision avoidance. However, because the WuR is constantly listening to the medium, more power is used for idle listening. As a result, this method is unsuitable for applications like greenhouse monitoring where nodes are expected to remain idle for the majority of the time. Additionally, latency is increased because a sending node must first send a WuC to the receiving node to wake it up before sending data.

Unlike contention schemes, TDMA protocols preserve more energy by eliminating energy waste from collisions. In TDMA, time is split into frames and then into several slots. Afterward, every sensor node is given an assured time slot for transmitting or receiving. At other times, it switches off the radio. Consequently, TDMA-based protocols enable collision avoidance and can offer better energy efficiency than contention schemes. However, traditional TDMA schemes keep their radios on, even when they have no data to send or receive, resulting in energy inefficiency. For instance, LEACH applies TDMA within a cluster [12], where the network is split into clusters, each containing a cluster head. Instead of sending data directly to the base station, cluster nodes transmit to their cluster heads. The cluster heads then forward the received data to the base station. The role of the cluster head is randomly rotated among cluster nodes to evenly distribute power use. But, LEACH assumes all nodes have data to send at all times. This is unfavorable because a node wastes energy by operating in idle listening mode. To reduce idle listening, E-TDMA improves LEACH: if a node has no data to send during its allotted time, it turns off its radio [13]. BEST-MAC further extends LEACH and E-TDMA by proposing multiple smaller-size time slots which are more than the number of available sensor nodes to handle adaptive traffic [14]. The time slots are scheduled by the knapsack algorithm to reduce job completion time and also to minimize average packet delay. Additionally, BEST-MAC uses short node addresses of 1 byte to minimize control overheads. We also adopt these short node addresses in our protocol to minimize overheads. References [15] and [16] add to BEST-MAC by rotating the role of the cluster head among the member nodes based on their residual energies. However, [16] calculates the nodes' residual energies at the start of each communication round, whereas [15] makes predictions about the energy reserves using artificial intelligence. Additionally, [15] increases throughput via an artificial neural network using a back-propagation method. Reference [17] also offers more contributions; it varies its duty cycle based on latency and packet arrival rate to reduce energy waste. But like all TDMA schemes, [14], [15], [16], [17] experience periodic synchronizations accompanied by their overheads.

In recent years, scholars have proposed hybrid protocols, to combine the best features of both contention-based and

TDMA schemes to balance out their weaknesses. A good example is IHMAC [18]. IH-MAC employs both TDMA and CSMA schemes. To reduce power use, the network uses both link scheduling and broadcast scheduling, i.e., it dynamically transitions from link scheduling to broadcast scheduling and vice versa based on traffic loads. Schedules are obtained with a decentralized scheme where nodes use local clock arithmetics to find their allocated time slots. Moreover, IH-MAC uses RTS/CTS handshakes to obtain the minimum required transmit powers to reach intended destinations, thus further lowering the power use. IH-MAC also uses parallel transmissions to minimize delays. EDS-MAC uses a similar approach to IH-MAC but minimizes the network overheads by proposing short node addresses of 1 byte on the sensor nodes [19]. References [20], [21], and [22] are some of the most recent work. To reduce delays and collision probability, [20] employs many channels instead of the medium's single main channel. Additionally, depending on traffic density, [20] uses variable duty cycles and backoff algorithms. In the synchronous CSMA/TDMA protocol used by [21], during periodic communication rounds, nodes that intend to transmit send a Ready-To-Send (RTS) frame using CSMA. The cluster head then responds with a TDMA schedule that the nodes are to adhere to, which saves energy. Unfortunately, this method only functions for stationary nodes. Reference [22] continuously alters its operating mode between CSMA and TDMA schemes to adjust to the changing data traffic. However, the drawback is that in every communication round, every node listens for a certain period before transmitting, thereby increasing delays. Elsewhere, in emergencies, and disaster management situations, the focus is usually made on the effective use of unmanned aerial vehicles (UAVs) to gather real-time data whereas energy efficiency comes secondary. Therefore, to minimize energy use, [23] proposed a technique that starts by using a CSMA mechanism for the registration of sensor nodes to the UAV, then schedules are generated and a TDMA scheme with variations in slot times for data transmissions is followed. However, due to their high battery consumption, this method requires frequent UAV recharge.

All the aforementioned protocols have greatly improved energy conservation and quality of service, however, most of them have drawbacks, as explained in each protocol, and have not fully solved the challenge of energy inefficiency in WSNs. Therefore we propose GS-MAC, which uses the best features from each protocol to eliminate all the sources of energy waste in a system that is suitable for greenhouse monitoring applications.

III. THE PROPOSED SYSTEM

Since the nature of the operation of greenhouse monitoring systems are different from other monitoring systems, assumptions about the GS-MAC protocol must be outlined.

A. NETWORK AND APPLICATION ASSUMPTIONS

The network may consist of multiple portable nodes organized in clusters. The nodes are stationary most of the time, although, they may be moved from one location to another within the greenhouse with new nodes also able to join the network at any time.

The protocol design is applicable even for areas that cover about 125,600 m^2 of land, which is six times larger than the world's largest greenhouse at the time of writing this article [28].

Different crops may have different irrigation and other farming schemes. For example, maize crops may require different irrigation patterns compared to tomatoes, or tomatoes at one stage of farming may require different irrigation schemes compared to tomatoes at a different stage. For this reason, the GS-MAC protocol is not designed for any specific crop type, instead, the user may customize network parameters to suit the growing conditions of the available crops.

RTC modules should be embedded in all sensor nodes to maintain UTC [27]. RTC modules are time and date-remembering systems which have battery setups that even in the absence of external power, keep the modules running. These modules will be used for network synchronization as explained in the upcoming sections.

The network is dedicated to a single application of monitoring and control of farming-related parameters. Therefore all sensor nodes are embedded with the same source code and coordinate together. Also, similar to S–MAC [7], rather than per-node fairness, the focus is made on maximizing system-wide application performance.

This design does not take into account what is to be done to any specific actuator node as all crops have different requirements. Instead, time slots are reserved for control packets from the user to operate actuator nodes. The control action will depend on the type of crop, the number of environmental parameters being monitored and the available actuators on the greenhouse.

Finally, the application is expected to be idle for long durations, to minimize power use, and then become active after certain periods. This is because weather parameters do not change frequently; so it is not necessary to monitor the environment at all times.

These assumptions strongly influence the protocol design and distinguish the proposed work from other energy-saving WSN mechanisms.

B. GS-MAC PROTOCOL DESIGN

Since environmental weather parameters do not change frequently, it is not necessary to keep nodes listening at all times. So, similar to TDMA schemes, the listening time is reduced by letting the nodes go into periodic sleep. During the sleep mode, nodes consume less energy, thereby prolonging the network lifetime. To maintain schedules, the use of *clusters* is adopted. This involves dividing the entire network into separate groups known as clusters. Each cluster consists of one node randomly elected as a *cluster head* and the rest of the nodes are identified as *member nodes*. When member nodes have data to send, they transmit it to the cluster head.

The cluster head is responsible for collecting data from all its member nodes and forwarding it to the *sink node*. A sink node is the only node that has a gateway to the internet. The cluster heads also transmit messages from the sink node to the member nodes. These are usually control instructions from the user to dictate what a specific node or group of nodes should do. Many clustering techniques have been proposed in the past but GS-MAC adopts a clustering network topology similar to the one proposed in [12]. The entire operation of GS-MAC consists of *network initialization* and *communication rounds*. Each communication round is further split into the *data phase*, *control phase,* and *sleep phase*.

1) NODE DEPLOYMENT

Network initialization starts with node deployment. Nodes are deployed to form a ring topology with a radius not exceeding 200 m as shown in Fig. 1. The user randomly selects one node from each cluster and sets it as a cluster head before placing it at a distance of not more than 100 m from the center of the greenhouse to form a virtual ring around the sink node, which is placed at the center of the greenhouse. Then each member node is placed in its cluster region. A cluster region, in this case, is a circular area with a radius of less than 100 m surrounding a particular cluster head. Therefore the distance of separation between a cluster head and any of its member nodes does not exceed 100 m. With this arrangement, a 2-hop network is formed with the distance from the sink node to the farthest node in any direction being less than 200 m. The hop distance between nodes is limited to 100 m because the range of 802.15.4 standard radio is approximately 100 m for 250 kbps data rates. Therefore, the GS-MAC network coverage spans a region of 125,600 m^2, six times larger than the current largest greenhouse in the world. But not all greenhouses have circular regions. In the case of rectangular, square, or any other environmental shape, the distance between the edges of the environment and the center, where a sink node is placed should not exceed 200 m. With the optimum number of cluster regions and proper placement of nodes, there is complete network coverage on a 200 m radius, with no dead zones.

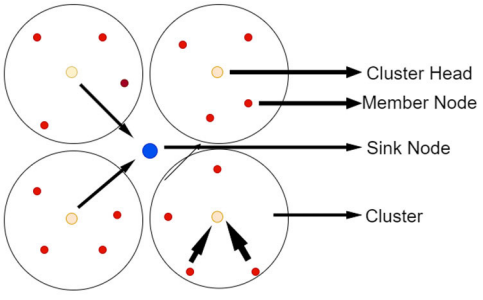


FIGURE 1. Node deployment.

After deployment, nodes follow three steps to synchronize and prepare for communication. At first, the cluster heads broadcast an announcement message (*CH_BROAD*) to its members. Then the member nodes reply with a request-to-join message (REQ_JOIN) Finally, the cluster head assigns schedules to its member nodes by broadcasting a *schedule message*. The member nodes can now transmit based on restrictions from the schedule message.

2) BROADCASTING CH_BROADS

During deployment, once a node is turned on for the first time, it checks its *network status* to determine if it already belongs to the network or whether it has been deployed for the first time. The network status is 2 bits long and indicates whether a node belongs to a network or not; if a node already belongs to a network, the value of its network status will be 1, therefore it will proceed to communicate as per its schedule. But if a node does not belong to a network, its network status value will be 0. During the initialization phase, the network status of all nodes including the cluster heads is 0. Therefore all nodes will immediately go to sleep, but wake up to communicate at the first second of the adjacent minute after when they were first turned on. For example, if a node was turned on at 08:30:21.40 AM, it will immediately go to sleep and wake up at 08:31:00.00 AM to communicate. All nodes can maintain UTC with the help of embedded RTC modules [27].

During communication, all member nodes go to listen mode, i.e., turn on their radio receivers. This is when cluster heads broadcast their CH_BROADs for all the member nodes to hear. A CH_BROAD is a 3 Bytes frame that contains a short address of the cluster head (1 Byte), and a Frame check sequence (2 Bytes). This announcement period takes place for a period of 1 minute. All cluster heads broadcast using the same frequency and channel to make it possible for all neighboring member nodes to hear. By default, all nodes operate using the same frequency and channel at this stage. To avoid collision of the CH_BROAD frames, each cluster head broadcasts only once at a time that depends on when it was turned on. Therefore the time at which a cluster head broadcasts its CH_BROAD is given by:

$$T^{CH\_BROAD} = T^{ON} + T^{minute} \tag{1}$$

where T^{CH_BROAD} represents the time at which a particular cluster head sends its CH_BROAD and T^{ON} represents the time at which that cluster head was first turned on. T^{minute} indicates 1 minute. For example, if a cluster head was turned on at 08:30:21.30 AM, then it will broadcast its CH_BROAD at 08:31:21.30 AM, the same second and millisecond as when it was turned on but the minute has an increment of 1. Assuming all the nodes were turned on at slightly different times by one person or a few people, then the chances of a collision are very minimal.

3) SENDING REQ_JOINS

The duration of time for broadcasting CH_BROADs is exactly one minute. Thereafter during the following minute, the nodes which want to join a particular cluster respond with a REQ_JOIN. Each REQ_JOIN is 5 Bytes long and

contains short addresses each for the member node (1 Byte) and cluster head (1 Byte), Frame Check Sequence (2 Bytes), and data length (1 Byte). The data length is 256 bits long and represents the length of data expected to be transmitted by a node. For example, if a node expects to transmit 30 bytes of data then its data length will be 00011110; a 1-byte binary number that represents the decimal number 30. Then the cluster head will know to reserve data slots for 30 bytes of data from that particular node. As a result, the cluster head is prepared to handle traffic from nodes of different loads. Using this technique, nodes can request to transmit up to 256 bytes of data at a time, which is more than enough for recorded environmental weather parameters. How the member nodes send their REQ_JOINs follows a contention approach similar to RTS/CTS mechanism to avoid the hidden node problem. If a member node finds itself in overlapping clusters and receives advertisements from more than one cluster head, then it will respond to the cluster head with the highest signal strength, because that is the node nearest to its location. This period of sending REQ_JOINs takes place for exactly two minutes. Two minutes are selected to allow enough time for each member node to communicate with the cluster head. Since cluster heads broadcast CH_BROADs at a time that depends on when it was turned on, if this period was only one minute or shorter, and a cluster head was turned on at 08:30:59.50, it would broadcast its CH_BROAD at 08:31:59.50, leaving only 10 ms for member nodes to communicate with it, which would not be enough. Therefore by selecting two minutes, all member nodes have more than 1 minute to communicate with the cluster head. The duration of time for broadcasting CH_BROADs and REQ_JOINs is long and since all nodes are awake at this stage, they consume a lot of energy. However, this process only takes place once, during the initialization process, therefore its impact on the network lifetime is negligible.

4) ALLOCATING SCHEDULES

Finally, the cluster head assigns schedules to all the member nodes. At first, the cluster head calculates the total number of received request messages to determine the number of member nodes. Then it computes the time it takes to complete communication exchange with each separate member node to determine slot durations. The slot duration, T_n^{dur} of a node, is the total time it takes for that node to transmit all its data and receive an acknowledgment from the cluster head. It can be computed as:

$$T_n^{dur} = \frac{(L_n^{MN} + L^{CH})}{S^{node}} + T^{Del} \tag{2}$$

where L_n^{MN} is the length of data expected to be transmitted by node n, L^{CH} is the length of an acknowledgment from the cluster head to the node, S^{node} is the transmission speed of the nodes, and T^{Del} is the sum of the time taken by a member node in preparing to transmit, and the time taken by a cluster head to process the received frame before sending an acknowledgment. The total time required by a cluster head to communicate with all member nodes is given as:

$$T^{DPP} = \sum_{n=1}^{N-1} T_n^{dur} \tag{3}$$

where T^{DPP} represents the time all member nodes take to communicate with a cluster head. Afterward, the cluster head calculates the member node transmission times (*T_MN*). T_MN is the time in a communication round when a particular node is required to start data transmission, and it is computed as:

$$T^{MN} = \begin{cases} T^{CR} & n = 1 \\ T^{CR} + \sum_{n=1}^{n-1} T_n^{dur} & n > 1 \end{cases} \tag{4}$$

where T^{CR} represents the time at which a communication round starts and T^{MN} represents T_MN. The cluster head labels the nodes based on the first come first served rule, i.e., the node that initially sent their REQ_JOIN first would be the first to transmit their data in every communication round. Afterward, cluster heads assign new short addresses to all its associated member nodes. Remember, we use short node addresses to minimize overheads. But new addresses are assigned to eliminate the possibility of more than one member node possessing the same address. Before deployment, each node usually has a random 1 Byte short address assigned to it, therefore more than one member node may possess the same address. To avoid this, cluster heads must assign new addresses to all member nodes. So each node in a particular cluster has a short address that is unique from all its members; however, to maximize node count, member nodes from separate clusters may use the same addresses. But one may wonder how the protocol distinguishes between two member nodes of different clusters but using the same address. To understand this, assume multiple member nodes using the same address send their data. To distinguish the nodes, the protocol will check from which cluster heads the packets originated. And if the user is to send control packets, each reply will address a particular node through the cluster head from which it previously sent its last data. Member nodes only send their data to the cluster heads, which forward it to the sink. So member nodes from different clusters may share the same addresses but all cluster heads must have addresses distinct from each other. With this arrangement, our protocol can support a maximum of 256 nodes per cluster and 65,536 nodes throughout the entire greenhouse. But communications between the user and cluster heads do not occur individually. Instead, in each communication round the cluster heads usually combine all data from their members and forward it as one bulk message. Similarly, the user control instructions are usually broadcasted as one aggregated message where each member node picks the information corresponding to its address. With this method, we can further lower the nodes' duty cycles.

Therefore, before the initialization phase's conclusion, the cluster head broadcasts a schedule message to all its member nodes as shown in Fig. 2.

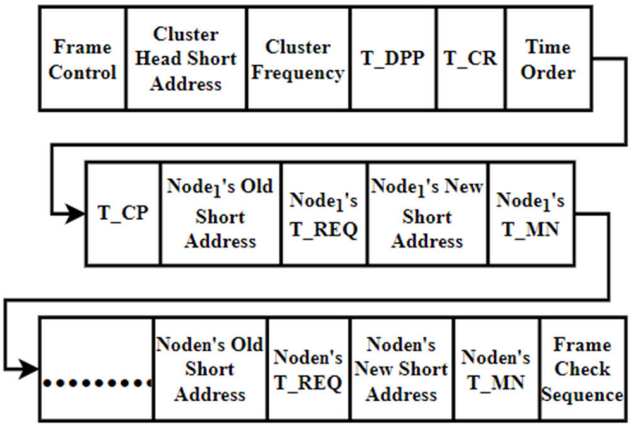


FIGURE 2. Schedule message.

The schedule message contains cluster head short address, *cluster frequency*, all member nodes' old and newly assigned short addresses, the times when member node sent their REQ_JOINs (*T_REQ*), member nodes transmission time (*T_MN*), Data Phase Period time (*T_DPP*), communication round time *(T_CR)*, *time order,* control phase duration *(T_CP)* and Frame Check Sequence. The length of the schedule message depends on the number of member nodes in the cluster. Short node addresses are assigned to the cluster heads and member nodes to reduce energy consumption associated with excess control overheads during communication. Cluster frequency is the frequency at which member nodes communicate with their cluster head during communication rounds. Each cluster operates at a different frequency compared to all other clusters to avoid co-channel and adjacent channel interference. T_CR, T_MN, and time order are used by a particular member node to determine the start of a communication round and when it is required to start data transmission. T_CR is 6 bits long and is used to show the commencement of a communication round. The time order is 1 bit long and it indicates whether the data on T_CR is in minutes or seconds, i.e., the time order is 0 to represent minutes and 1 to represent seconds. For example, if T_CR is 000010 which means decimal number 2, and the time order is 0 which means minutes, this would mean that a communication round starts after every two minutes from the beginning of every hour. However, if T_CR is 000010 which means decimal number 2, and the time order is 1 which means seconds, this would mean that a communication round starts after every two seconds from the beginning of every hour. A length of six bits is allocated for T_CR since 6 bits can be used to represent up to 64 decimals; this is sufficient to represent 60 minutes in an hour or 60 seconds in a minute. Therefore GS-MAC allows a user to control sleep and wake-up durations from a few seconds to several minutes. By default, T_CR is 000001, and the time order is 0, meaning member nodes communicate with the cluster head once every minute before going to sleep. However, these settings can be changed by the user at any time to suit the nature of crops in the greenhouse.

T_MN is used by a particular node to determine at what time during a communication round, it should start data transmission. T_DPP is the time it takes for all member nodes to complete data transmission. It is used by member nodes to determine the end of the data phase and the

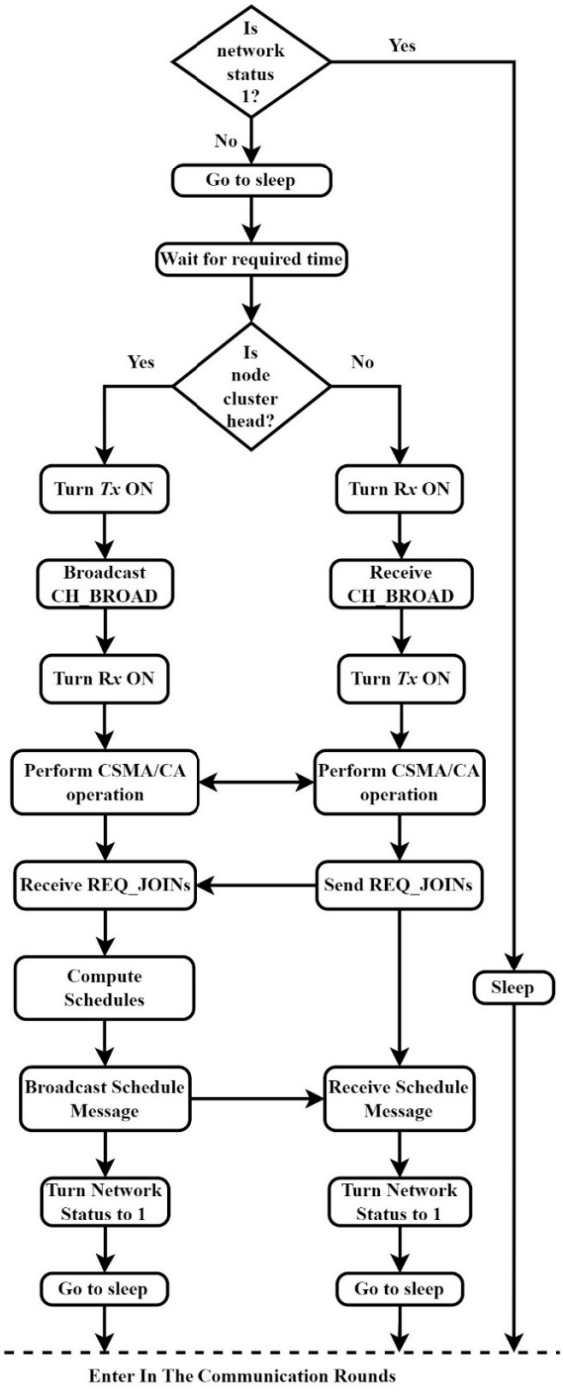


FIGURE 3. Network initialization.

beginning of the control phase. It is computed from (3). T_CP indicates the duration of the control phase. It is used by the cluster head to go to sleep at the end of the control phase and wait for the beginning of a new communication round. The T_REQ and the old and newly assigned short addresses are used by each member node to determine the portion of the schedule message that is intended for it. The schedule message is one long frame that contains messages for every member node. To decode its part of the data on the schedule message, a member node selects a portion of the information that corresponds to its old short address. Then it gathers all the information that corresponds to that address. If a node identifies more than one old address resembling its own, this would mean that more than one member node had the same old address during deployment. In this case, the member node will proceed with comparing the T_REQs corresponding to the associated old addresses and pick the T_REQ that resembles its own. The T_REQ is the time that a particular member node sent its REQ_JOIN to the cluster head. Since the T_REQs of all the nodes are different, the member node will now proceed to record all the information in the schedule message that corresponds to its T_REQ. Then the node will replace its old address with the newly assigned short address.

During schedule allocation, all nodes stay awake for the entire period and go back to sleep immediately after they receive the schedule message. Once a node has received its schedule, it changes its network status to 1, meaning it now belongs to a network, and then goes to sleep, awaiting a new communication round. The cluster head at this point, also updates its network status to 1 and goes to sleep. The entire network initialization process is summarized in Fig. 3.

If a node is not successfully initialized in a cluster during the network initialization phase, it will go to sleep and wake up at the control phase to attempt to join the network once more.

5) BASIC SCHEME

The communication rounds phase begins after the successful completion of network initialization. The basic scheme is shown in Fig. 4.

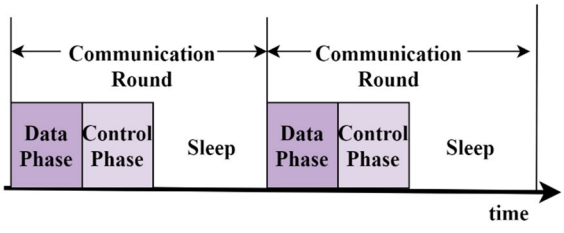


FIGURE 4. Basic scheme.

Each communication round consists of a data phase, a control phase, and a sleep phase.

a: DATA PHASE

The data phase is the period in a communication round when member nodes are required to transmit their data. Each member node follows a strict schedule obtained from the cluster head through the schedule message. Only one member node transmits to the cluster head at a time, with no synchronization requirements as shown in Fig. 4 and Fig. 5. As a result, all nodes except the cluster heads maintain constant low-duty cycles even with increasing node density, thereby minimizing energy use. Additionally, even if nodes have different traffic loads, through (2), cluster heads can compute the time it takes to complete communication exchange with each separate member node without affecting other nodes. Therefore GS-MAC applies to both homogeneous and heterogeneous applications.

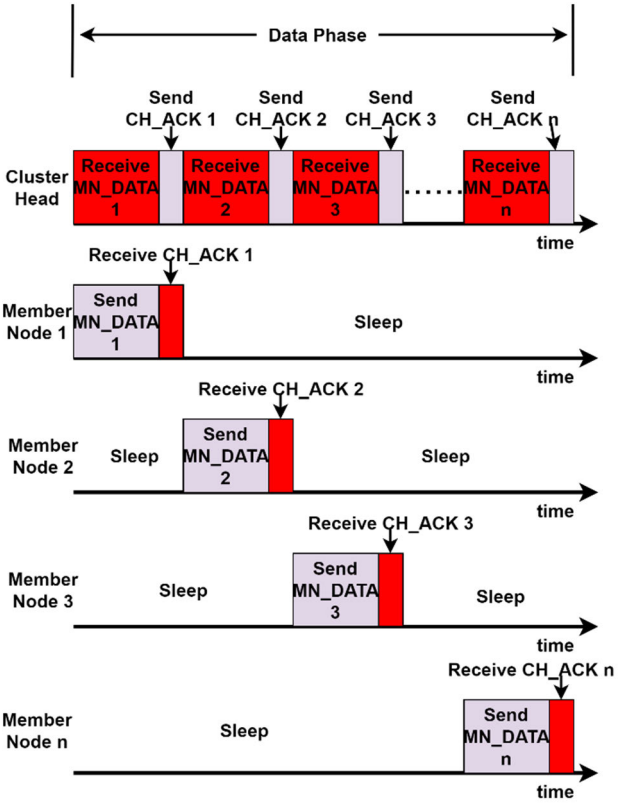


FIGURE 5. Data phase process.

During its schedule, a member node will wake up, sense the environment and send the recorded data to the cluster head. Then the cluster head replies with a cluster head acknowledgment packet (*CH_ACK*) to indicate acknowledgment of the packet before the schedule of another node begins. The data transmitted to the cluster head at this stage is called member node data (*MN_DATA*) It contains environmental data (*ENV_DATA*) and energy level data (*E_LEVEL*). ENV_DATA consists of information related to the environmental weather parameters being monitored and E_LEVEL contains the energy level of the battery powering the member node. The E_LEVEL is used by the cluster head to select the node

suitable to replace it as the new cluster head when its energy level falls below an allocated threshold.

The role of the cluster head is rotated among all nodes in a cluster to maintain uniform drops of energy levels on all the nodes. If the cluster head's responsibilities are not shared among all nodes, the elected cluster head will deplete its energy level much earlier than the member nodes and the network lifetime of the system will be decreased.

The CH_ACK is 2 bits long and contains an update message (1 bit) and an acknowledgment message (1 bit). The acknowledgment message signals a member node whether or not its data has been received by the cluster head; its value is 1 to indicate acknowledgment and 0 to indicate no acknowledgment. The update message informs the member node if the cluster head has additional information to provide; its value is 1 to indicate the presence of additional information and 0 to indicate no additional messages. The additional information may be control instructions from the user or control messages from the cluster head indicating changes that need to take place. If cluster heads have no additional information, a member node will skip the control phase, go to sleep, and set a timer to awaken itself during the next communication round. The time at which a member node wakes up to transmit in the next communication round, T^{MN+1} is calculated by:

$$T^{MN+1} = T^{MN} + T^{CR} \tag{5}$$

However, if there is additional information from the cluster head, the node will go to sleep and set a timer to awaken itself at the beginning of the control phase. The time at which a node wakes up to listen at the control phase, T^{MN_DP} is given as:

$$T^{MN\_DP} = T^{CR} + T^{DPP} \tag{6}$$

Here T^{DPP} represents T_DPP. After the control phase, the member node will go back to sleep and set a timer to awaken itself at the next communication round. This time is calculated as shown in (5). Alternatively, a node may calculate this time concerning the control phase as:

$$T^{MN+1} = T^{MN\_DP} + T^{CR} - (T^{DPP} - T^{MN}) \tag{7}$$

The communication exchange at the data phase has no additional control overheads apart from E_LEVEL and CH_ACK. This is because member nodes transmit based on strict schedules, eliminating the possibility of collisions. Also, the cluster head maintains the schedules of all member nodes and can identify the senders without requiring additional overheads. So there is no need for the nodes to add their address overheads. By minimizing control overheads, more energy is saved. The algorithm used for the data phase may be summarized as shown in Fig. 6.

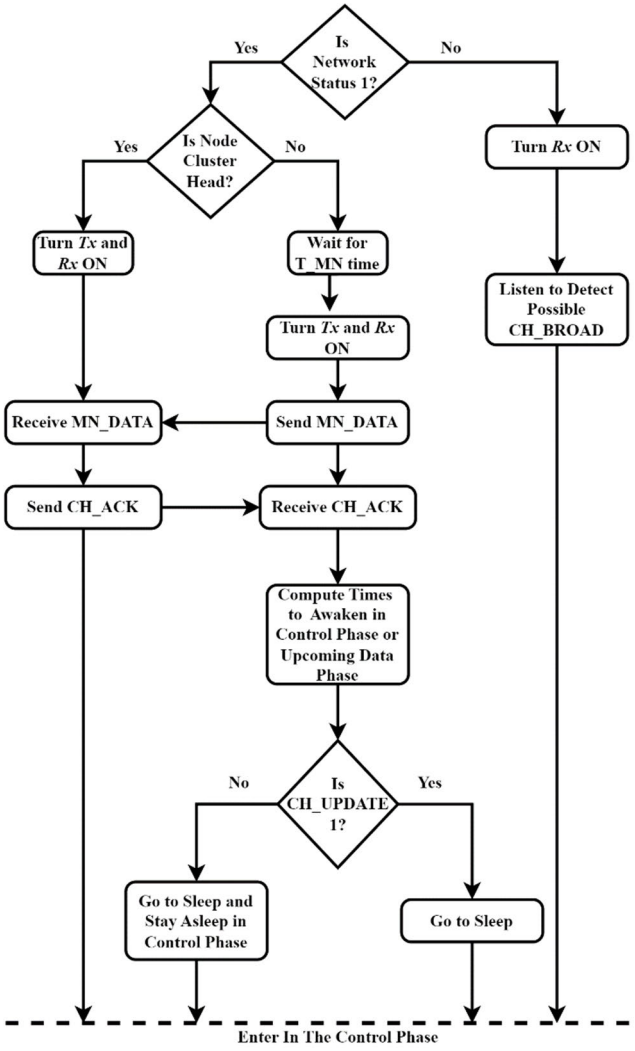


FIGURE 6. Data phase algorithm.

b: SLEEP PHASE

The duration of time for sleeping for a member node, T^{dur}_{sleep} is given as:

$$T^{dur}_{sleep} = T^{CR} - \left(T^{dur}_{n} + T^{dur}_{CMP}\right) \tag{8}$$

Here T^{dur}_{CMP} represents the length of time of the control message phase. Since greenhouse sensor nodes are stationary most of the time, and the weather parameters monitored are the same, then rarely is there a need to go to the control phase. When a node skips the control phase, it consumes less energy per communication round. The default GS-MAC settings allow communication rounds to begin after every 1 minute. However, these settings can be changed based on the requirements of a particular crop. Some studies have shown that for certain crops, the system works efficiently even if nodes wake up to communicate every thirty minutes [6]. By increasing the sleeping durations, the network lifetime increases. All user instructions will be updated on the nodes during the control phase as described in the next subsection.

c: CONTROL PHASE

The control phase is a period in a communication round reserved for member nodes that have not yet been initialized in the network to join a cluster, for member nodes to receive control information from the user or cluster head, for new member nodes to join the network, or for nodes that have been transferred from one location to another within the network to find a suitable cluster. The control phase is split into the control message phase (*CMP*) and the scalability phase.

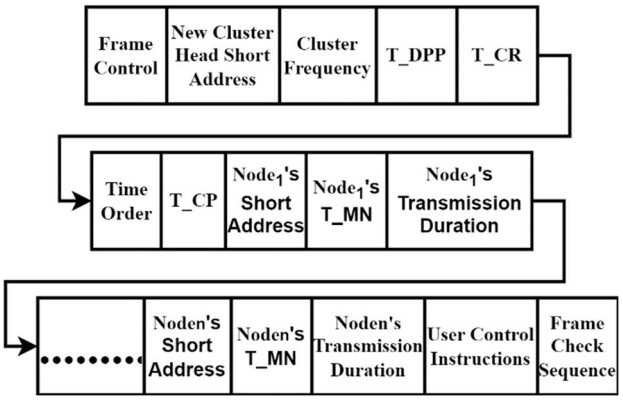


FIGURE 7. CH_UPDATE.

The CMP starts after the completion of the data phase. If member nodes are directed to enter the control phase, they wake up at this time to receive an update message (CH_UPDATE) from the cluster head. The CH_UPDATE contains a new cluster head address, a new schedule message, and *user control instructions*. The new cluster head indicates the address of the member node that has been selected as the new cluster head. If the cluster head has not selected any replacement, then the new cluster head address will be the same as its current one. In this case, the member nodes will ignore the new schedule message and maintain the current schedules they already have. However, if the cluster head has selected its replacement, then each member node will update its schedule according to the new schedule message and send its data to the newly selected cluster head in the upcoming communication rounds. The new schedule message, as shown in Fig. 7 contains all member nodes' short addresses, cluster frequency, T_MNs, T_DPP, T_CR, time order, T_CP, member nodes transmission duration (*T_MNDs*), and Frame Check Sequence. The T_MNDs indicate how long a member node is expected to transmit during the data phase and are used by the new cluster head to know when to send a CH_ACK and when another member node should begin transmitting. All other symbols carry meanings similar to those defined in the initial schedule message as indicated in Fig. 2. The current cluster head also includes its details as a member node in the new schedule message and operates as a member node afterward. Other Member nodes decode their portion of the data by comparing all addresses and selecting one that corresponds to its own.

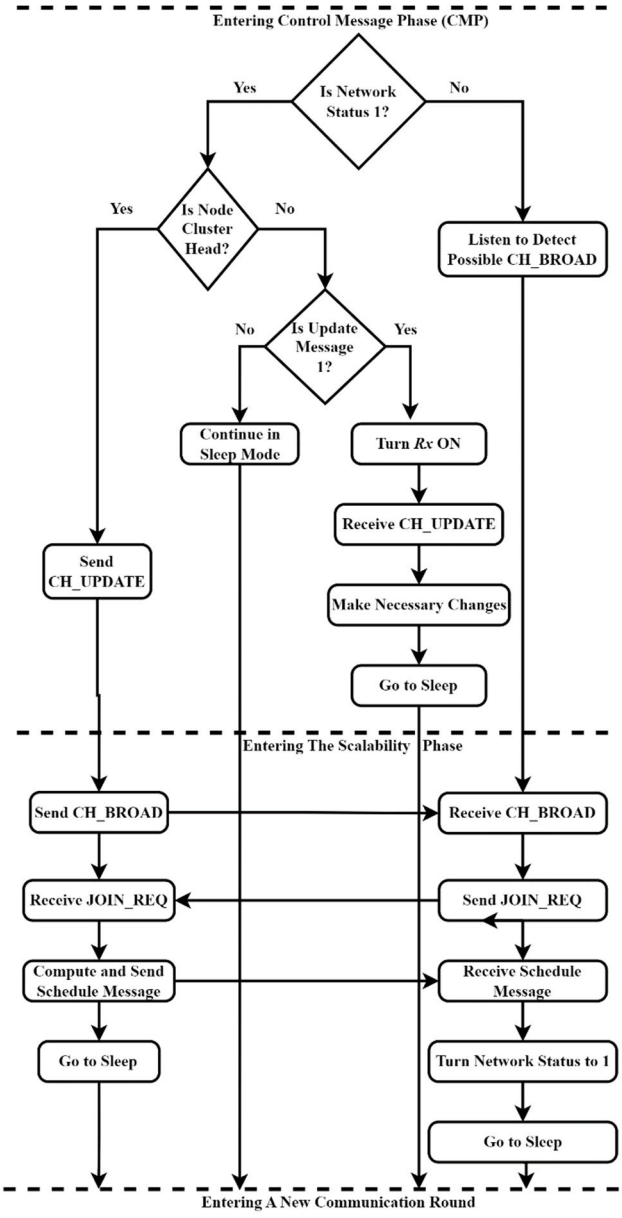


FIGURE 8. Control phase.

The user control instructions are from the user to a particular member node or all nodes. For example, the user may want to change the sleep and wake-up durations or may require a particular node to control a certain actuator. This phase is important because GS-MAC enables the user to customize the default system settings or perform control actions at any time. The length and content of the user control message depend on crop type and the number of monitored environmental parameters.

The scalability phase is allotted for nodes that have been transferred from one location to another within the network. Nodes that have not been successfully initialized to the network and new nodes may also join a cluster during this period. During the scalability phase, all member nodes are usually in sleep mode whereas the cluster head stays awake to

detect possible traffic from new members. The initialization process is similar to the one discussed in the initial network initialization process but without RTS/CTS mechanism to minimize delays, and the durations for CH_BROAD and REQ_JOINs in the scalability phase may be shorter, based on the communication round's sleep and wake durations. This is because the scalability phase must be completed before a successive communication round commences. After the successful initialization of new nodes, the network proceeds to the successive communication rounds and transmits as per the schedules. The entire control phase is summarized as shown in Fig. 8.

IV. RESULTS AND DISCUSSION

A. TESTBED

Actual hardware is used to determine current characteristics during transmitting, receiving, idle listening, and sleeping modes. Actual hardware is chosen because theoretical current consumption estimates may differ from real-world data significantly [6]. Each sensor node is composed of an MSP430 family of microcontroller units embedded with a CC2500 radio unit and a DHT11 sensing unit. The controller units are also embedded with RTC modules to keep track of time [27]. To determine the average current consumption, an oscilloscope voltage probe is connected in series with the power supply over a 1 Ω resistor. Therefore current consumption values are displayed as a function of time on the oscilloscope. Next, a node is made to transmit to another, and measurements of current consumption are taken at both the sending and receiving nodes. Afterward, readings are taken when the nodes are in idle listening and sleeping states respectively. The recorded data are then used as simulation parameters on MATLAB to represent current consumption characteristics in transmit, receive, idle listening, and sleeping modes.

B. SIMULATION ENVIRONMENT

The simulations are carried out in a 3D setting on MATLAB as shown in Fig. 9.

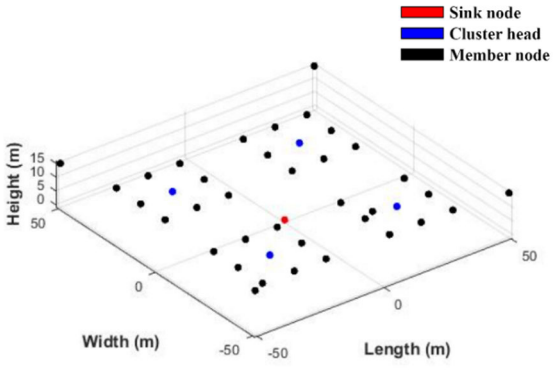


FIGURE 9. Simulation environment.

We estimate the greenhouse has dimensions of 100 m in both length and width and 15 m in height. We deploy four cluster heads around the sink node, placed in the center. Then we randomly place 10 member nodes in each cluster. Afterward, we simulate with traffic loads sent every 60 seconds. We select 60 seconds because environmental parameters do not vary frequently. Therefore our system can tolerate large delays. Then we increase member nodes to 20, 30, and 40. We then adjust the delays to range from 1 to 20, 40, and 60 seconds with 10 member nodes. We do not recommend any delay value; the ideal farming conditions are known only by the user, or farming experts. Our protocol merely provides a platform for the user to choose delays anywhere between 1 second and 1 hour. Remember that some researchers have even proposed 30-minute delay settings for certain crop situations [6]. We model the system to operate as though it has been running for 360 days. When using CSMA, we set the maximum contention window size to 1024, but vary the minimum contention window from 10 to 20, 30, and 40, depending on the node count. Nodes may retransmit if a collision occurs, however, only a maximum of 4 retries are allowed. The lengths of the RTS, CTS, and ACK packets are each 30 bytes, but the RTS/CTS mechanism is only used during the initial network initialization phase. We assume that every member node transmits at a speed of 250 kbps when active, and has a payload (i.e., packet length) of 300 bytes for each communication round. We estimate 300 bytes because we do not expect environmental parameters to have large loads of data. For example, after 5 communication rounds each member node will have transmitted a total of 1500 bytes. This is because, despite any changes in the environmental conditions (such as temperature, humidity, wetness, etc.), the packet lengths of such payloads remain constant. Other simulation parameters are summarized as shown in Table 1, most of which are obtained from experiments conducted on the testbed. We use the same parameters (i.e., where applicable) for all the protocols involved in the analysis. Finally, we do a comparison of the chosen protocols.

TABLE 1. Simulation parameters.

Parameters	Values
The initial energy of nodes	2000 J
Slot length	20 µs
SHT11 sensor current (low power mode)	0.9 µs
Microcontroller current (low power mode)	0.9 µs
Microcontroller sleep-wake-up switching time	1 µs
Radio unit current in transmitting mode	21.2 mA
Radio unit current in receiving mode	12.8 mA
Radio unit current in idle listening mode	12.8 mA
Radio unit current in sleeping mode	0.4 µA
Radio unit sleep-wake-up switching time	240 µs

C. PERFORMANCE METRICS

The proposed scheme is assessed using four performance metrics: *duty cycle*, *energy consumption*, *network lifetime*,

and *delay*. A node's duty cycle , DuC is calculated as [25]:

$$\text{DuC} = \frac{(T^{transmit} + T^{receive} + T^{idle})}{T^{total}} \tag{9}$$

where,

$$T^{total} = T^{\text{transmit}} + T^{\text{receive}} + T^{idle} + T^{sleep} \tag{10}$$

Here T^{total} denotes the total simulation time whereas T^{transmit}, T^{receive}, T^{idle}, and T^{sleep} represent the times spent by a node in transmitting, receiving, idle listening, and sleeping respectively. Assuming E^{sensor}, $E^{controller}$, $E^{transmit}$, $E^{receive}$, E^{idle}, and E^{sleep} represent the energy used by the node's sensing and microcontroller units, and when transmitting, receiving, idle listening, and sleeping respectively, then the nod's total energy consumption, E^{node} after T^{total} is computed as [6]:

$$\begin{aligned} E^{node} &= E^{sensor} + E^{controller} + E^{transmit} \\ &\quad + E^{receive} + E^{idle} + E^{sleep} \end{aligned} \tag{11}$$

$$\begin{aligned} E^{node} &= (P^{sensor} x T^{total}) + (P^{controller} x T^{total}) \\ &\quad + \left(P^{transmit} x T^{transmit}\right) + \left(P^{receive} x T^{receive}\right) \\ &\quad + \left(P^{idle} x T^{idle}\right) + \left(P^{sleep} x T^{sleep}\right) \end{aligned} \tag{12}$$

where P^{sensor}, $P^{controller}$, $P^{transmit}$, $P^{receive}$, P^{idle}, and P^{sleep} denote the power consumption by the node's sensing and microcontroller units, and when transmitting, receiving, idle listening, and sleeping respectively. As explained in Section I, the sensor and microcontroller units consume almost the same power when active in low-power mode as when sleeping. So, in this study, we assume these units are active throughout the entire simulation period. However, the radio units may periodically sleep. Therefore if the node has an initial energy capacity of $E^{initial}$, then the time taken for the node to deplete its battery (i.e., its lifetime), $\text{Node}_{\text{life}}$ is computed as:

$$\text{Node}_{\text{life}} = \frac{(E^{initial})}{(E^{node}/T^{total})} \tag{13}$$

Concerning delays, a packet moving through a multi-hop network experiences the following delays at each hop [7]:

Carrier sense delay happens when a sending node performs carrier sense. The size of the contention window determines its value.

Backoff delay occurs when carrier sense fails, either due to collisions or when a sending node detects other transmissions.

Transmission delay is affected by the packet length, channel bandwidth, and the coding method applied.

Propagation delay is affected by the distance of separation between the transmitting node and the receiver. In WSNs, it is normally ignored since the distance between the sending and receiving nodes is usually very small.

Processing delay happens because before sending a packet to the following hop, the receiver must process it. This delay is mostly determined by a node's processing capacity and the effectiveness of the algorithm applied for data processing.

Queuing delay is affected by the traffic load. In heavy traffic scenarios, queuing delay is usually very significant.

The sleeping delay occurs when a sending node intends to transmit but the receiving node is in a sleeping state.

All of the aforementioned delays, except the sleeping delay, are characteristics of multi-hop networks and apply equally to GS-MAC, BEST-MAC, FAWR, EDS-MAC, and S-MAC. Therefore, all protocols assume the same delay values in the simulation environment. However, each method experiences a different sleep delay, depending on the time spent in sleeping mode. We assume a complete cycle of listening and sleeping to be a frame. Suppose packets arrive at the sending node with equal time probability within a frame, and T^{listen} and T^{sleep} denote the times in a frame when a node spends in listening and sleeping modes respectively, then the average sleep delay (D_s) can be computed as:

$$D_s = \frac{T_{frame}}{2} \tag{14}$$

where,

$$T_{frame} = T^{listen} + T^{sleep} \tag{15}$$

D. COMPARATIVE ANALYSIS

The performance of the proposed protocol is compared to that of S-MAC [7], BEST-MA [14], FAWR [11], and EDS-MAC [19]. We select S-MAC because it is one of the first energy-saving systems. It is also the first WSN protocol to implement duty cycling and has paved the way for the creation of all other WSN duty-cycling methods. We choose BEST-MAC, FAWR, and EDS-MAC because they are current works in scheduling, contention, and hybrid methods respectively, and have outperformed many other protocols.

1) IMPLEMENTATION OF PROTOCOLS

Five protocols—GS-MAC, BEST-MAC, FAWR, EDS-MAC, and S-MAC—have been simulated. In real world, nodes communicate simultaneously, however, in simulation environments, codes are executed sequentially. Therefore, to mimic a contention scheme on MATLAB, each node starts by selecting a random number from a set of integers distributed uniformly from 1 to a value of the assumed contention window. The chosen numbers act as backoff counters. Then an array of the selected backoff counters is created to indicate the time slots when nodes are expected to transmit. This array represents first-try transmissions. Afterward, each value in the created array is compared with every other array member. We assume a collision if more than one entry in the array contained the same value. As a result, each node involved in a collision will choose an extra random integer that will be stored on an extra array created. The second array represents transmissions on the first retry. This process of selecting and comparing backoff counters is repeated until the maximum number of retries is met. Afterward, sleep durations for each node are determined by subtracting a node's active time from the total simulation time, concluding the simulations of the contention-based algorithms. Schedule-based protocols also

follow the same approach in simulating their setup phase, but then transmit sequentially, each node communicating in its allocated time slot. The duration of simulations depends on the chosen number of simulation rounds. Finally, performance metrics are computed. Animations are also made on the same script based on the sequence of operations.

2) ANALYSIS OF ENERGY EFFICIENCY

Equation (11) shows that to decrease a node's energy use, we need to minimize E^{sensor}, $E^{controller}$, $E^{transmit}$, $E^{receive}$, E^{idle}, and E^{sleep}. As explained in Section I, the sensor and microcontroller units consume almost the same power when active in low-power mode as when sleeping. So, in this study, we assume E^{sensor} and $E^{controller}$ are active (i.e., not sleeping) throughout the entire simulation period. However, the radio units may periodically sleep. Therefore, this leaves us the task of minimizing $E^{transmit}$, $E^{receive}$, E^{idle}, and E^{sleep}. $P^{transmit}$, $P^{receive}$, P^{idle}, and P^{sleep} are constant but T^{transmit}, T^{receive}, T^{idle}, and T^{sleep} may vary depending on the MAC protocol used. Since $P^{transmit}$, $P^{receive}$, and P^{idle} are significantly high compared to P^{sleep} (see section I), the best way to reduce E^{node} is by minimizing T^{transmit}, T^{receive}, and T^{idle} whereas T^{sleep} is increased. For example, if we neglect E^{sensor} and $E^{controller}$ and if T^{total} is 80 seconds where T^{transmit}, T^{receive}, T^{idle}, and T^{sleep} are each 20 seconds, then using readings from [6] where $P^{transmit}$, $P^{receive}$, P^{idle}, and P^{sleep} are 21.2 mA, 12.8 mA, 12.8 mA, and 0.4 μA respectively, we get E^{node} of 0.936008 J. But if T^{total} is 80 seconds where T^{transmit}, T^{receive}, T^{idle}, and T^{sleep} are 10 seconds, 10 seconds, 0 seconds, and 60 seconds respectively, we get E^{node} of 0.340024 J. So minimizing T^{transmit}, T^{receive}, and T^{idle} is crucial to lowering power use. Equation (9) shows that reducing T^{transmit}, T^{receive}, and T^{idle} results in a low duty-cycle, and (13) shows that a decrease in E^{node} leads to an increase in $\text{Node}_{\text{life}}$. Therefore, we can conclude that a protocol with the lowest duty cycle will have the least energy consumption, resulting in the highest network lifetime. This is achieved by GS-MAC; our protocol eliminates T^{idle} by avoiding idle listening, and lowers T^{transmit} and T^{receive} by avoiding over-emitting and over-hearing respectively (see Fig. 5). GS-MAC further decreases T^{transmit} and T^{receive} by avoiding collisions and synchronization delays (see Fig. 5), while also reducing packet overheads (see section III, part B.4). As a result, figures 10 and 12 show that GS-MAC has the lowest duty cycle and energy consumption respectively, resulting in the highest network lifetime as shown in Fig. 14.

On the other hand, other protocols have higher duty cycles because they synchronize periodically. With S-MAC [7], nodes experience the largest duty cycle because they wake up and go to sleep together. As a result, a node will continue listening even if it has completed communicating until every other node has also finished transmitting. Nodes in FAWR [11], BEST-MAC [14], and EDS-MAC [19] have lower duty cycles than S-MAC because they go to sleep after transmitting or receiving, regardless of whether other nodes are still communicating. BEST-MAC has an even lower duty cycle than EDS-MAC and FAWR because it uses a TDMA method whereas EDS-MAC and FAWR use a contention strategy during data transmissions. In contention schemes, nodes usually stay awake for longer periods due to carrier sense and backoff delays. But in TDMA designs, nodes follow schedules and avoid carrier sense and backoff delays. FAWR does not require periodic node synchronizations, but the WuR must send a WuC before transmitting any data. WuC transmissions take almost as long as periodic synchronization packet transmissions since they use a contention process. Therefore FAWR has a comparable duty cycle to EDS-MAC.

Figures 10 and 12 also show that the duty cycle and energy use respectively of GS-MAC remain at constant low levels leading to a constant high network lifetime despite the variations in node density. GS-MAC has a 2.7 times higher network lifetime than BEST-MAC, the closest energy-saving scheme, and this gap increases with traffic density as shown in Fig. 14. This is because the nodes only wake up to communicate during their allotted time slots, even if the node density is increased, and that all other nodes stay in a state of sleep when a given node is transmitting (see Fig. 5). Only the cluster head's duty cycle increases as a result. But the duty cycles of other protocols rise with the increase in

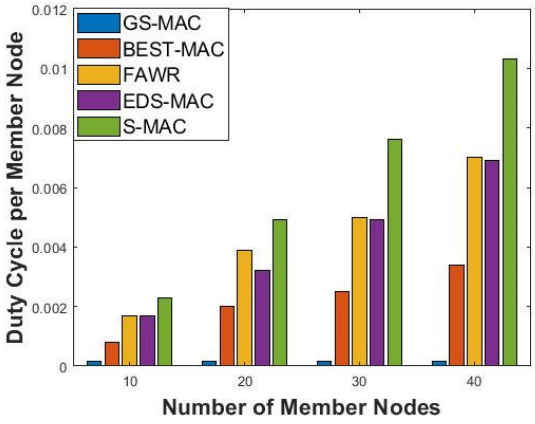


FIGURE 10. Analysis of duty cycle in the first scenario.

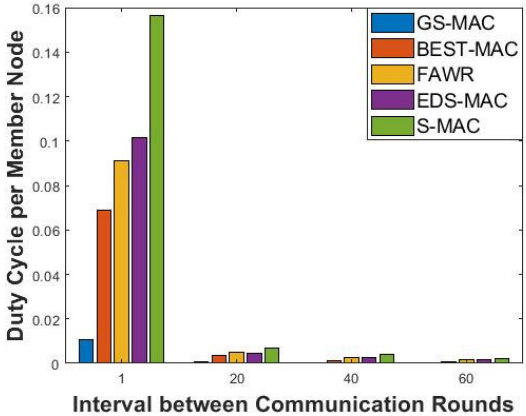


FIGURE 11. Analysis of duty cycle in the second scenario.

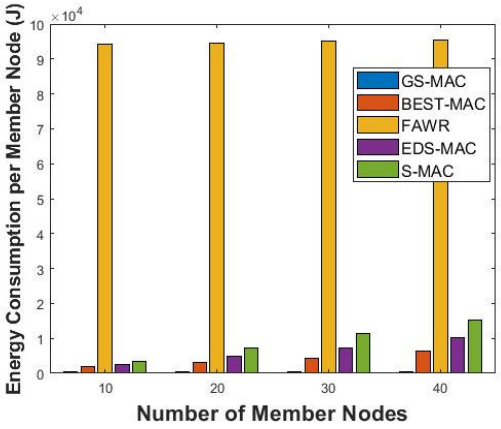


FIGURE 12. Analysis of energy consumption in the first scenario.

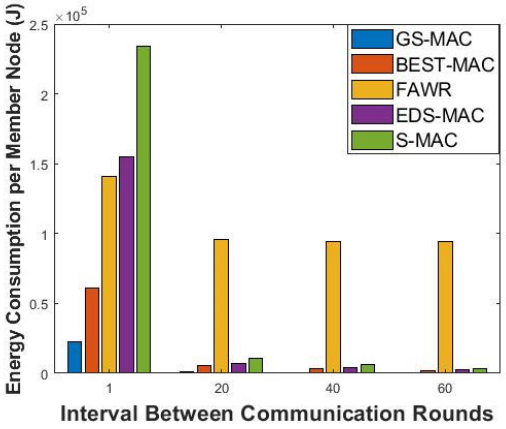


FIGURE 13. Analysis of energy consumption in the second scenario.

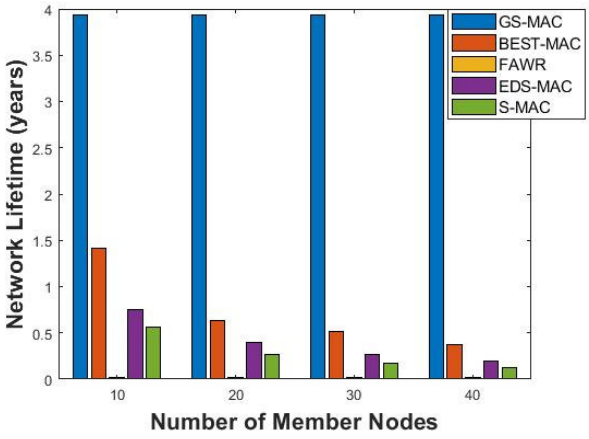


FIGURE 14. Analysis of network lifetime in the first scenario.

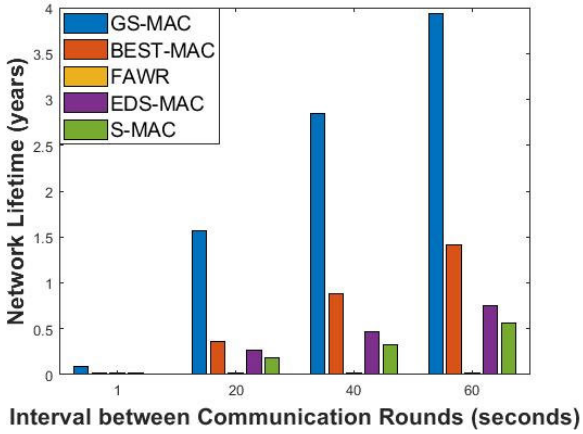


FIGURE 15. Analysis of network lifetime in the second scenario.

node density because they use a contention approach during data transmissions. BEST-MAC is the least affected because it only uses a contention approach in the setup phase. With an increase in node density, contention mechanisms experience more collisions, which lengthen carrier sense and backoff delays. Increased node density also lengthens queuing delay. These delays prolong the length of idle listening. As a result, the duty cycles of all other protocols increase with high node densities. Since the WuR in FAWR is always listening, we do not include it in duty cycle calculations but involve it in the computations of energy use and other metrics. Figures 12 and 14 show that FAWR uses comparable energy and has a similar network lifetime compared to other protocols in cases where the sleeping periods are relatively short (for instance, if set at 1 second). But the energy use becomes notably higher than all other protocols when the sleeping time is relatively long (for example, if it is set to 60 seconds), leading to the lowest network lifetime. This is because the WuR in FAWR is always listening for any incoming signals.

The WuR typically uses only 1 mW of power, which is very small, but if nodes sleep for extended periods, then other protocols will save considerably more energy than FAWR. Figures 11 and 13 show that the duty cycles and energy use of all protocols decrease with an increase in the interval between communication rounds leading to higher network lifetimes as shown in Fig. 15. This is because when the length of the interval is increased nodes spend more time in sleeping states than in active modes.

3) ANALYSIS OF DELAYS

Fig. 16 demonstrates that, of all the protocols, FAWR has the lowest delay, followed by S-MAC, EDS-MAC, and BEST-MAC, with GS-MAC having the largest delay. In FAWR, the WuR is always on and can detect incoming transmissions at any time, hence FAWR has the lowest latency because it does not experience sleep delays.

FAWR experiences all other forms of delay but avoids sleep delay, which is the most significant. All other protocols have larger latencies because they suffer sleep delay. Other protocols have higher duty cycles than GS-MAC, therefore suffer fewer sleep delays. Because GS-MAC has the lowest duty cycle, it experiences the highest sleep delay, consequently suffering the highest latency, which is the only downside of this work. However, the resulting delay has a negligible impact on the performance of greenhouse

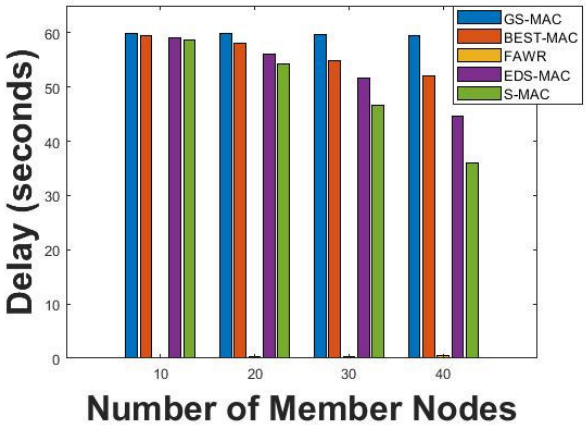


FIGURE 16. Analysis of delay.

monitoring and control activities. Therefore, GS-MAC is still the best option for greenhouse applications.

V. CONCLUSION

This research presents a new MAC protocol for wireless sensor network-based greenhouse monitoring and control. The protocol has better energy efficiency than previous research. As a result, it provides a longer network lifetime. Experiment results show that the proposed work has a network lifetime that is 2.7 times better than BEST-MAC and substantially higher than other protocols. The findings also indicate that this difference grows even more as the sensor node density is raised. This is because the protocol has eliminated most of the energy waste sources associated with wireless sensor networks that were identified by previous researchers. As a result, the proposed protocol has an energy efficiency superior to both TDMA and contention schemes. The protocol also maintains good scalability, comparable to contention mechanisms, within the network range. Thus, the main goal of developing a protocol with energy efficiency superior to TDMA schemes while also preserving the scalability of contention mechanisms is achieved.

A. LIMITATIONS

The main drawback of the proposed protocol is that it adds more delays. Duty cycling is the primary method of reducing energy loss, but the longer nodes spend sleeping, the more delays there are. As a result, there is a trade-off between energy conservation and delays. However, given the nature of how greenhouse farms operate, the resulting delays barely affect how well monitoring and control operations are carried out. Therefore the proposed protocol is still the best choice for greenhouse applications, despite the ensuing delays.

Another limitation of the proposed protocol is that it only supports a 2-hop network, resulting in a maximum network coverage of about 125,600 m^2 when using the IEE 802.15.4 standard. At the moment, this is not a problem since the network coverage is six times larger than the largest available greenhouse in the world. But if larger greenhouses are to be built in the future, more work would be required to adapt the proposed technique.

B. RECOMMENDATIONS FOR FUTURE RESEARCH

Most of the parameters used in the simulation environments had assumed values. However, each crop has unique farming requirements, necessitating the use of different kinds of sensors and actuators. So this research may be further improved by studying the actual parameters used in greenhouse systems for particular crops. Then the observed values may be integrated with the GS-MAC to provide accurate estimates of the anticipated network lifetime.

In addition, for systems using an energy harvesting approach, such as solar energy systems, the minimum daily sunshine expectations of a certain place may also be researched. Then ideal and precise sleeping times may be determined using the observed data and the GS-MAC protocol to reduce delays. For instance, there would be no need of keeping nodes in the sleep state for very long periods if there was plenty of sunlight.

Finally, the algorithms used in the GS-MAC protocol need to be improved to increase the network hopping capability to more than 2. Presently constructed, GS-MAC only supports a 2 hop network, but by enabling multiple hops, the protocol may be applied in open agriculture farms, which is the primary farming method used globally, thereby providing greater impact.

REFERENCES

[1] A. Kochhar and N. Kumar, "Wireless sensor networks for greenhouses: An end-to-end review," *Comput. Electron. Agricult.*, vol. 163, Aug. 2019, Art. no. 104877.

[2] H. Darmono, R. Perdana, and W. Puspitasari, "Observation of greenhouse condition based on wireless sensor networks," in *Proc. IOP Conf. Ser., Mater. Sci. Eng.*, 2020, Art. no. 012107.

[3] A. Ali and H. S. Hassanein, "Wireless sensor network and deep learning for prediction greenhouse environments," in *Proc. Int. Conf. Smart Appl., Commun. Netw. (SmartNets)*, Dec. 2019, pp. 1–5.

[4] Z. Wang, "Greenhouse data acquisition system based on ZigBee wireless sensor network to promote the development of agricultural economy," *Environ. Technol. Innov.*, vol. 24, Nov. 2021, Art. no. 101689.

[5] C. Visvesvaran, S. Kamalakannan, K. N. Kumar, K. M. Sundaram, S. M. S. Vasan, and S. Jafrrin, "Smart greenhouse monitoring system using wireless sensor networks," in *Proc. 2nd Int. Conf. Smart Electron. Commun. (ICOSEC)*, Oct. 2021, pp. 96–101.

[6] M. Srbinovska, V. Dimcev, and C. Gavrovski, "Energy consumption estimation of wireless sensor networks in greenhouse crop production," in *Proc. IEEE 17th Int. Conf. Smart Technol.*, Jul. 2017, pp. 870–875.

[7] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. 21St Annu. Joint Conf. IEEE Comput. Commun. Societies*, Jun. 2002, pp. 1567–1576.

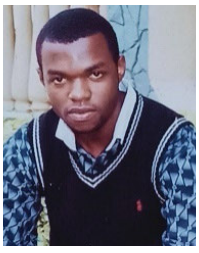
[8] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, pp. 493–506, Jun. 2004.

[9] G. Sakya and V. Sharma, "ADMC-MAC: Energy efficient adaptive MAC protocol for mission critical applications in WSN," *Sustain. Comput., Informat. Syst.*, vol. 23, pp. 21–28, Sep. 2019.

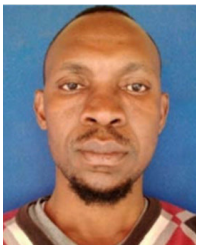
[10] M. Rasheed, I. U. Din, M. Adnan, A. Tariq, S. Malik, and I. Syed, "ECM-MAC: An efficient collision mitigation strategy in contention based MAC protocol," *IEEE Access*, vol. 9, pp. 62880–62889, 2021.

[11] A. Pegatoquet, T. N. Le, and M. Magno, "A wake-up radio-based MAC protocol for autonomous wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 56–70, Feb. 2019.

[12] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci.*, 2000, p. 10.
[13] J. Li and G. Y. Lazarou, "A bit-map-assisted energy-efficient MAC scheme for wireless sensor networks," in *Proc. 3rd Int. Symp. Inf. Process. Sensor Netw.*, Apr. 2004, pp. 55–60.
[14] A. N. Alvi, S. H. Bouk, S. H. Ahmed, M. A. Yaqub, M. Sarkar, and H. Song, "BEST-MAC: Bitmap-assisted efficient and scalable TDMA-based WSN MAC protocol for smart cities," *IEEE Access*, vol. 4, pp. 312–322, 2016.
[15] R. Kumar and M. Gangwar, "Improved BEST-MAC protocol for WSN using optimal cluster head selection," *Int. J. Inf. Technol.*, vol. 15, pp. 859–875, Nov. 2019.
[16] K. Debasis and M. Singh, "An enhanced low duty cycle MAC protocol for wireless sensor networks," in *Proc. Int. Conf. Soft Comput. Signal Process.*, vol. 2020, pp. 423–431.
[17] M. Tolani and R. K. Singh, "Adaptive duty-cycle-enabled energy-efficient bit-map-assisted MAC protocol," *Social Netw. Comput. Sci.*, vol. 1, p. 144, May 2020.
[18] M. Arifuzzaman, M. Matsumoto, and T. Sato, "An intelligent hybrid MAC with traffic-differentiation-based QoS for wireless sensor networks," *IEEE Sensors J.*, vol. 13, no. 6, pp. 2391–2399, Jun. 2013.
[19] V. Sundararaj, S. Muthukumar, and R. S. Kumar, "An optimal cluster formation based energy efficient dynamic scheduling hybrid MAC protocol for heavy traffic load in wireless sensor networks," *Comput. Secur.*, vol. 77, pp. 277–288, Aug. 2018.
[20] K. Wang, X. Zhao, Y. Shi, D. Xu, and R. Li, "The energy-efficient MDA-SMAC protocol for wireless sensor networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, pp. 1–10, Dec. 2020.
[21] A. Roy and N. Sarma, "A synchronous duty-cycled reservation based MAC protocol for underwater wireless sensor networks," *Digit. Commun. Netw.*, vol. 7, no. 3, pp. 385–398, Aug. 2021.
[22] A. Boukerche and X. Zhou, "A novel hybrid MAC protocol for sustainable delay-tolerant wireless sensor networks," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 4, pp. 455–467, Oct. 2020.
[23] S. Poudel and S. Moh, "Energy-efficient and fast MAC protocol in UAV-aided wireless sensor networks for time-critical applications," *Sensors*, vol. 20, no. 9, p. 2635, May 2020.
[24] M. Atto and C. Guy, "Wireless sensor networks: MAC protocols and real time applications," in *Proc. 13th Annu. Post Graduate Symp. Converg. Telecommun., Netw. Broadcast.*, Liverpool, U.K., 2012, pp. 1–6.
[25] R. Braun and F. Afroz, "Energy-efficient MAC protocols for wireless sensor networks: A survey," *Int. J. Sensor Netw.*, vol. 32, no. 3, p. 150, 2020.
[26] Timeanddate. (Jul. 5, 2023). *UTC—The World's Time Standard*. [Online]. Available: https://www.timeanddate.com/time/aboututc.html
[27] SparkFun. (2023). *SparkFun Real Time Clock Module*. [Online]. Available: https://www.sparkfun.com/products/12708
[28] CNN. (2022). *Designs Unveiled for the World's Largest Single-Domed Greenhouse*. [Online]. Available: https://edition.cnn.com/style/article/largest-greenhouse-venice-biennale/index.html



MIKE MAJHAM received the bachelor's degree in electronics and communication engineering from St. Joseph University, Tanzania, in 2015. He is currently pursuing the master's degree in wireless and mobile computing with The Nelson Mandela African Institution of Science and Technology. His research interests include system security, wireless sensor networks, artificial intelligence, the Internet of Things, television white space, and wireless and mobile computing.



MARCO P. MWAIMU received the B.Eng. degree in electronics and communication engineering from St. Joseph University, Tanzania, in 2010. He is currently pursuing the master's degree in wireless and mobile computing with The Nelson Mandela African Institution of Science and Technology. He is a Researcher with the Arusha Technical College. His research interests include cognitive radio, the Internet of Things, television white space, wireless community networks, and wireless security.

THOMAS KIVEVELE received the B.Sc. degree in electro-mechanical engineering and the M.Sc. and Ph.D. degrees in mechanical engineering. He is currently a Senior Lecturer with The Nelson Mandela African Institution of Science and Technology (NM-AIST), Arusha, Tanzania, where he teaches with the School of Materials, Energy, Water, and Environmental Sciences (MEWES). He coordinates the Ph.D. and M.Sc. programs in sustainable energy science and engineering (SESE) and teaches energy-related courses. He is a very active Researcher and the Group Leader of the Clean Energy Technologies Research Group. He is also the Lead Researcher on five studies related to solar energy uses and biofuels. The projects are funded by USAID, the Regional Scholarship and Innovation Fund (RSIF), the EPFL and UM6P-led Excellence in Africa Program, and the African Research Initiative for Scientific Excellence Pilot Program (ARISE-PP), which is funded by the European Union and coordinated by the African Union and the African Academy of Sciences (AAS).

RAMADHANI S. SINDE (Member, IEEE) received the B.Sc. and M.Sc. degrees in engineering and technologies in telecommunication from the Moscow Technical University of Communication and Informatics, the Postgraduate Diploma degree in wireless and mobile computing from the Center for Development of Advanced Computing, India, and the Ph.D. degree in information and communication science and engineering from The Nelson Mandela African Institution of Science and Technology (NM-AIST). He specializes in electronics and telecommunication engineering. He has authored or coauthored more than 20 papers in internationally refereed journals and conferences. His research interests include telecommunications and informatics, wireless and mobile communication, wireless sensor networks, the Internet of Things, and embedded systems.