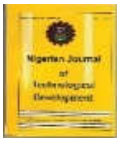
A Testbed for Group Acknowledgement Circular Shift Communication Model in LoRa-based Wireless Sensor Networks



A. A. Kasali[1*], C. O. Akanbi[2], L. O. Omotosho[2], and I. K. Ogundoyin[2]

[1]Department of Computer Engineering Technology, Federal Polytechnic Ede, Osun State, Nigeria.
[2]Department of Computer Science, Osun State University, Osogbo, Nigeria.

ABSTRACT: This paper introduces a testbed for a Group Acknowledgement Circular Shift (GACS) MAC protocol model designed to balance current consumption in LoRa-based Wireless Sensor Networks. The model leverages a time-slotted approach to allocate transmission times for end devices, using ATmega328p microcontrollers and Ebyte 22 LoRa Modules. The testbed includes a gateway, an application server, and initially four, then fifteen, end devices. Current monitoring circuits were attached to certain end devices to measure current usage during both GACS and Group Acknowledgement (GA) without Circular Shift (CS) models. In a network of four end devices, the GACS model extended battery life uniformly to 463 days with a 3000 mAh battery, while inconsistent battery life was observed in GA without CS. Similarly, the GACS model enabled 434 days of battery life across 15 devices, promoting uniform battery longevity. Results show that the GACS model enhances current consumption fairness, reduces collisions, and improves packet delivery to 99.81%.

KEYWORDS: Testbed, Time-slotted, LoRa, GACS, Current Consumption, and WSN

[Received Nov. 9, 2024; Revised Nov. 25, 2024; Accepted Dec. 11, 2024] Print ISSN: 0189-9546 | Online ISSN: 2437-2110

I. INTRODUCTION

Wireless Sensor Networks (WSNs) consist of disposable micro-sensors that detect physical and environmental parameters and relay data to a computer (Ketshabetswe et al., 2019). Advances in material science, semiconductors, and networking have made WSNs essential for various Internet of Things (IoT) applications. Low Power Wide Area (LPWA) technologies like LoRa, Narrow Band IoT, Dash7, and Sigfox provide low-power, cost-effective solutions suitable for remote, battery-operated WSNs, allowing end devices to reach gateways with a single hop and without route computing. These technologies achieve long range at the expense of low data rate and high latency (Migabo et al., 2017). They are suitable for delay-tolerant, non-critical infrastructure monitoring and environmental monitoring IoT applications which require high coverage distance. LPWA technologies have been used in supporting intelligent sensor-based smart irrigation system (Lakshmi et al., 2023), smart grid systems (Sharma et al., 2023), asset trackers, smart home appliances, patient monitors, utility meters, smart parking system (Khan et al., 2023), water meters and pipeline, smart palettes, smart street and highway lights (Čelarević & Mrakić, 2023).

LoRa, an open-source LPWA technology, enables affordable network establishment using Chirp Spread Spectrum (CSS) modulation, operating in the sub-GHz ISM band for bi-directional half-duplex communication (Gresl et al., 2021; Haxhibeqiri et al., 2018; Zorbas, 2020). LoRa's protocol stack includes physical, MAC, and application layers.

Several LoRa MAC protocols, like FREE (Abdelfadeel et al., 2019), TS-LoRa (Zorbas et al., 2020), LoRaBlink (Bor, 2020), TSCH (Haubro et al., 2020), synchronous LoRa mesh (Ebi et al., 2019) and many others, address scalability and energy usage, but fairness in current consumption was overlooked. Kasali et al., (2024) introduced a Group Acknowledgement Circular Shift (GACS) MAC protocol, which promotes fairness in current consumption using a time-slotted approach for data transmissions and acknowledgments. Using time-slotted eliminates transmission collisions among the end devices and thus, improves scalability and reliability (Zorbas, 2020).

The GACS model operates as illustrated in Figure 1. When an end device joins the network, the gateway assigns it a position and places it into a group. Each group of end devices transmits data to the gateway according to their assigned positions. Once the gateway receives the data, it sends an Acknowledgement (ACK) along with the last device's position to the group. Upon receiving the ACK, the end devices apply the circular shift algorithm to update their positions. Details of the algorithm is in Figure 5 of the work of (Kasali et al., 2024) and it is presented in this paper by equations 1 and 2. It governs the behaviour of 10 devices, as shown in Figure 2, with a group slot size (SN) of 4. pos denotes the current position of each node.

*Corresponding author: gabdulwakil@gmail.com

doi: http://dx.doi.org/10.4314/njtd.v21i4.3069

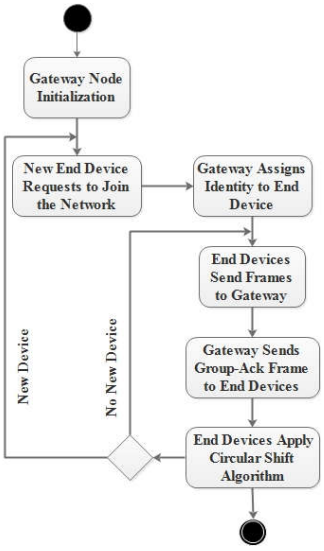
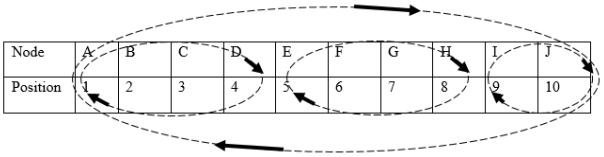


Figure 1 Activity Diagram of Group-Ack Circular Shift Model

$$new\ pos = \begin{cases} ((pos\ \%\ SN) + 1) + \left(INT\left(\frac{pos}{SN}\right) \times SN\right) & if\ pos\ \%\ SN > 0 \\ (pos - SN) + 1 & if\ pos\ \%\ SN = 0 \\ slot\_begin & if\ pos > lastnode \end{cases} \quad (1)$$



$$slot\_begin = pos \quad if\ pos\ \%\ SN = 1 \quad (2)$$

Figure 2: Behaviour of the GACS Algorithm

A working end device (node) transits from transmit to receive and to sleep state, and each state has different current consumptions. The state transitions, as illustrated in Table 1, determine the states of the underlying components, including the microcontroller unit (MCU) and the transceiver unit, that constitute a node. Changing positions, in a rotational manner, by nodes varies the time they stay in receive mode (Kasali et al., 2024) which in turn balances current consumption among them.

Table 1: States of Sensor Node and its Components

Node States	MCU States	Transceiver States
Transmit	Running	Transmit
Receive	Running	Receive
ACK/Join Reply		
Sleep	Sleep	Sleep

This section has introduced GACS model and it behaviour. Section II discusses how the model was implemented on real hardware. The account of the results obtained was discussed in section III while the conclusion and future directions were stated in section IV.

II. METHODOLOGY

In this work, a test bed for the GACS model is implemented. The model is implemented on ATmega328p MCUs and LoRa transceiver (Ebyte 22) modules. The prototype is made up of four sensor nodes, a gateway node, and an application server as depicted in Figure 3. An additional current monitoring circuit was constructed to monitor electrical current consumed by the sensor nodes. Each sensor node comprises hardware and software components. Figure 4 shows the circuit connection of the hardware component. The hardware include an Ebyte 22 module, a power supply unit and a microcontroller unit that houses the software component (control program).

The circuit is powered by a 5V DC. The positive and the negative terminals of the Ebyte module are connected to the positive and negative rails of the power supply. Pin 15 and 16 (which are equivalent to pin D9 and D10 on the Arduino UNO board respectively) of the ATmega328p are configured as serial port using softwareSerial library. Pin 15 was configured as Rx and pin 16 as Tx. Ebyte library, in conjunction with the softwareSerial library, is used by the microcontroller to communicate with the LoRa module.

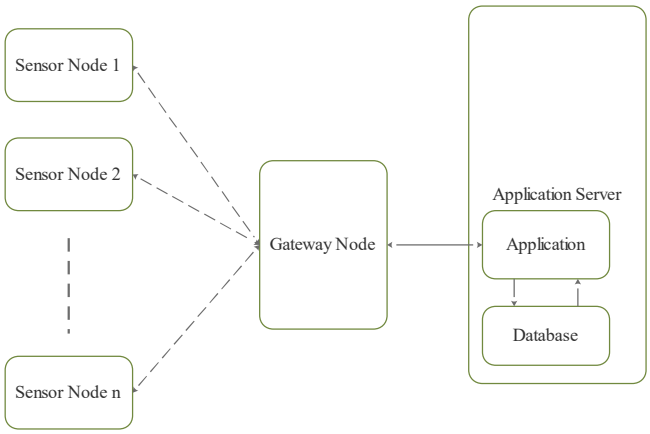


Figure 3: Block Diagram of Time-slotted LoRa MAC Protocol Model Test Bed

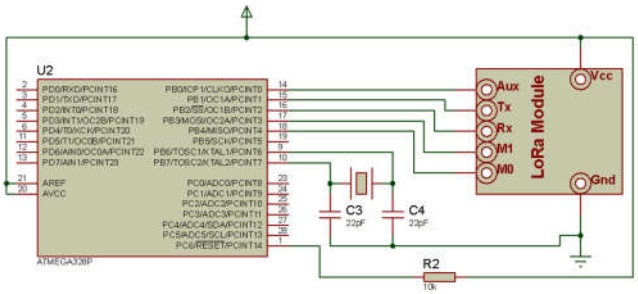


Figure 4: Circuit Diagram of Sensor Node

The configured Tx of the ATmega328p is connected to Rx of the LoRa module while the configured Rx is connected to Tx of the LoRa. Pin 17 and pin 18 (which are equivalent to pin D11 and D12 on the Arduino UNO board respectively) of

the ATmega328p are configured as output pins and are connected to the M1 and M0 pins of the LoRa module respectively. Logic zero '0' is sent out of these pins in order to set the LoRa to operate in transparent mode (which make messages to be sent to all device in the same channel). Pin 14 (which is equivalent to pin D8 on the Arduino UNO board) of the ATmega328p is configured as an input pin and connected to the AUX pin of the LoRa module. It helps to monitor the transmit buffer of the LoRa module.

The gateway is similar in hardware connection to the end device except that it is connected to a computer system (Application server) through a USB cable which provides power and RS-232 link to it. The circuit diagram is as depicted in Figure 5. A control program runs on the microcontroller and directs it to receive join requests, data, send acknowledgements, and forward data to the application server.

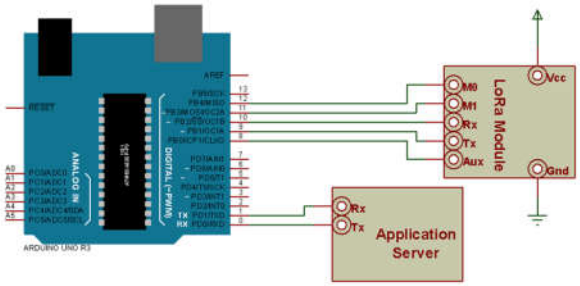


Figure 5: Circuit Diagram of Gateway Node

The application server comprises hardware and software components. The hardware is a core i5 16GB RAM computer system that runs a 64-bit Windows 11 Pro Operating System. The software components consist of a MySQL database and a desktop application written in Microsoft Visual Basic 2010. The schema of the data table is as depicted in Table 2. The fields (Nodeid, Nodepos, Date, Time and Data) are used to hold each sensor node's identity, the node's current position, the data transmission's date, the time of data transmission and the actual data, respectively. It was implemented in PHPMyAdmin of WAMP server. MySQL connector (mysql-connector-odbc-3.51.30-winx64) was installed to link the database and the desktop application.

Table 2: Data Table Schema

Column	Data Type
Nodeid	Varchar(20)
Nodepos	Varchar(20)
Date	Varchar(20)
Time	Varchar(20)
Data	Varchar(255)

The application (interface as depicted in Figure 6) is divided into three (3) modules namely: the connection, the data, and the download module. The connection module is responsible for binding the application server with the gateway through the communication (COM) port assigned to the gateway node. SerialPort, an RS-232 compatible, control is used to detect all active COM ports and populate the list in a drop-down box. Once the COM port assigned to the gateway is selected and the connect button is clicked, the gateway is attached to the application. In the data module, incoming data available at the attached COM port is read by the SerialPort control at every tick (every 1000 ms) of a tmer control. These data are then tokenized and saved into their respective fields in the database. The download module is responsible for building a simple query to fetch information from the database. A drop-down box is populated with the list of date information that were saved into the database. On selection of a date and the download button is pressed, an SQL query is generated and used to select all records that match the query and export the records into an Excel document.

Figure 6: Interface of the GACS Application

As shown in Figure 7, a current monitoring circuit is attached to a sensor node. The upper section of the Figure is the monitoring circuit, while the lower section is the sensor node. The monitoring circuit comprises an Arduino UNO board and a current sensor module (INA219). The module is connected to the Arduino board through its Inter-Integrated Circuit (I2C) interface. The SCL pin and the SDA pin on the module are connected to pin A5 and pin A4 of the Arduino board respectively. The Vcc and the Gnd pins are connected to the 5V and Gnd pins on the Arduino board respectively. The Vcc of the sensor node, to be monitored, is connected to the Vin- of the INA219 while the Vin+ is connected to the positive rail of the power supply unit. An 0.1Ω shunt resistor is connected across the Vin+ and Vin- pins. Current flows through the resistor first before the sensor node and this setup is called high-side. The control program that runs on the Arduino UNO board is as shown in Algorithm 1.

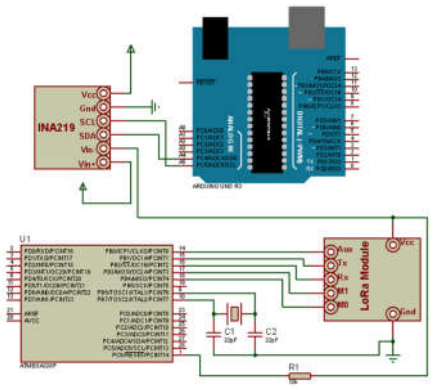


Figure 7: Circuit Diagram of Current Monitoring (upper section) and Sensor Node (lower section) Circuits

Algorithm 1: Control Program for Monitoring Current Consumption of Sensor Node

Control Program for Sensor Node Current Measurement

```
#include <Wire.h>
#include <Adafruit_INA219.h>
Adafruit_INA219 ina219;
  float current_mA = 0;
  float sum = 0.0, avg = 0.0;
void setup(void)
{
  Serial.begin(9600);
  while (!Serial)
  {
    delay(1);
  }
  // Initialize the INA219.
  if (! ina219.begin())
  {
    Serial.println("Failed to find INA219 chip");
    while (1)
    {
      delay(10);
    }
  }
}
void loop(void)
{
  for(int i = 1; i <= 50;i++){
    current_mA = ina219.getCurrent_mA();
    sum = sum + current_mA;
    delay(10);
    }
  avg = sum / 50;
  Serial.println(avg);
  sum = 0.0;
}
```

The program included and instantiated an Adafruit library for INA219. The getCurrent_mA() method of the library was used to extract the amount of current passing through the shunt resistor which is invariably the current drawn by the sensor node because they are connected in series. Fifty (50) samples are taken for each measurement and the average is calculated.

III. RESULTS AND DISCUSSION

This section gives accounts on how GACS algorithm affects sensor nodes' battery life and the overall network packet delivery ratio (PDR). The battery life determines how long rated batteries can sustain the sensor nodes while the PDR determines the efficiency and reliability of data transmission. Using the transmission parameters in Table 3, each node spent 2.17s in sending 240 bytes of data in a transmission cycle and expended 154 mA of current.

It drew 24 mA for varying time in receiving state based on its position in the transmission queue. It also drew 0.25mA for the period of its stay in sleep state. Each node was configured to transmit data four (4) times daily. Two scenarios were explored namely: Group Acknowledgement without Circular Shift (GA without CS) and Group Acknowledgement Circular Shift (GACS). During the GA without CS, nodes did not change their positions and hence the total energy (in mAh) consumed per day during the transmitting state is calculated based on equation 3.

Table 3: Transmission Parameters used for the GACS and GA without CS Scenarios

Parameters	Value
Spreading Factor (SF)	12
Bandwidth (BW)	500
Coding Rate (CR)	1
Header (H)	0
Data Optimizer (DE)	1
Number of Preamble (NP)	8
Payload	240 bytes | 54 bytes
AckCycleTime	10 s
Duty Cycle	10%
Nos of End Devices	4 | 15
Interval between Cycle	360 minutes

$$E_{Tx}\ (\text{mAh/day}) = I_{Tx}\ (\text{mA}) \times \frac{N_{Tx} \times ToA\ (seconds)}{3600\ seconds\ /hour} \quad (3)$$

Where I_{Tx} is the instant current consumed in mA during the transmiting state. N_{TX} is the number of transmissions per day and ToA is the Time on Air in second. Substituting the recorded values into the equation, the total energy consumed per day during the transmitting state is 0.3713 mAh.

$$154\ \text{mA} \times \frac{4\ \times 2.17}{3600\ seconds\ /hour} = 0.3713\ \text{mAh/day}$$

Current consumption during the receiving state is calculated based on equation 4.

$$\boldsymbol{E_{Rx}\ (\text{mAh/day}) = I_{Rx}\ (\text{mA}) \frac{N_{Tx} \times T_{receive-state}\ (seconds)}{3600\ seconds\ /hour}} \quad (4)$$

Where I_{Rx} is the instant current in mA during the receiving state. N_{Tx} is the number of transmissions per day and T_(receive-state) is the time (in second) a node stays in receiving state during a transmission cycle. It is calculated by using equation 5.

$$T_{receive-state}\ (\text{s}) = (T_{slot} \times Slot_{remain}) + T_{Ack} \quad (5)$$
$$T_{receive-state}\ (\text{s}) = (2.25 \times 3) + 1 = 7.75s$$

By using node 1 and substituting the recorded values into the equation, the total energy consumed per day during the receiving state is 0.2066 mAh.

$$24\ \text{mA} \times \frac{4\ \times 7.75\ seconds}{3600\ seconds\ /hour} = 0.2066\ \text{mAh/day}$$

Current consumption during the sleep state is calculated based on equation 6.

$$\boldsymbol{E_{sleep}}\ \textbf{(mAh/day)} = \boldsymbol{I_{sleep}}\ \textbf{(mA)} \times \frac{\boldsymbol{N_{Tx}} \times \boldsymbol{T_{sleep}}\ \boldsymbol{(seconds)}}{\textbf{3600 seconds /hour}} \quad (6)$$

Where I_{sleep} is the instant current in mA during the sleep state. N_{Tx} is the number of transmissions per day and T_{sleep} is the time (in second) a node stays in sleep state during a transmission cycle. It is calculated by using equation 7.

$$T_{sleep}(s) = \left(T_{slot} \times \left(Slot_{before-node-pos} + Slot_{after-GACK}\right)\right) + \left(Ack_{other-group} \times T_{Ack}\right) + T_{interval-bet-cycle} \quad (7)$$

By using node 1 and substituting the recorded values into the equations, the T_{sleep} and total energy consumed per day during the sleep state is 21615.5s and 6.0043 mAh/day respectively.

$$T_{sleep} = \left(2.25 \times (0 + 6)\right) + (2 \times 1) + 21600 = 21615.5\text{s}$$

$$0.25\ \text{mA} \times \frac{4 \times 21615.5\ seconds}{3600\ seconds\ /hour} = 6.0043\ \text{mAh/day}$$

Summing the energy consumed in the three (3) states together as expressed in equation 8 gives the total energy consumed by node 1 which is 6.5822 mAh/day.

$$E_{node}\ (\text{mAh/day}) = E_{Tx}\ (\text{mAh/day}) + E_{Rx}\ (\text{mAh/day}) + E_{sleep}\ (\text{mAh/day}) \quad (8)$$

$$E_{node}\ (\text{mAh/day}) = 0.3713 + 0.2066 + 6.0043\ \text{mAh/day} = 6.5822\ \text{mAh/day}$$

The battery life is calculated using equation 9. Using a battery of 3000mAh capacity, the battery life is 456 days.

$$\text{Battery Life (day)} = \frac{Battery\ capacity\ (mAh)}{E_{node}\ mAh/day} \quad (9)$$

$$\text{Battery Life (day)} = \frac{3000\ mAh}{6.5822\ mAh/day} = 456\ \text{days}$$

Under the scenario of GA without CS, node 1, 2, 3 and 4 spent 456, 460, 465, and 469 days respectively when operated on a battery rated 3000 mAh. Table 4 shows the battery life of the nodes during GA without CS and GACS scenarios.

In the GACS scenario, nodes dynamically change their positions, allowing each node to occupy all four positions (1, 2, 3, and 4) within a day, as there are four transmissions per day. Consequently, the total energy consumption (in mAh) during the transmitting state per day is determined using Equation 10.

$$E_{Tx}\ (\text{mAh/day}) = I_{Txi}\ (\text{mA}) \times \frac{\sum_{i=1}^{n} ToA_i\ (seconds)}{3600\ seconds\ /hour} \quad (10)$$

Where i ranges from one to n cycles. I_{Txi} is the instant current consumed in mA during the ith transmitting state. ToA_i is the ith Time on Air in second. Substituting the recorded values into the equation, the total energy consumed per day during the transmitting state is 0.3713 mAh.

$$154\ \text{mA} \times \frac{8.68}{3600\ seconds\ /hour} = 0.3713\ \text{mAh/day}$$

Current consumption during the receiving state is calculated based on equation 11.

$$E_{Rx}\ (\text{mAh/day}) = I_{Rxi}\ (\text{mA}) \times \frac{\sum_{i=1}^{n} T_{receive-state_i}\ (seconds)}{3600\ seconds\ /hour} \quad (11)$$

Where I_{Rxi} is the instant current in mA during the ith receiving state. $T_{receive\text{-}statei}$ is the time (in second) a node stays in the ith receiving state during a transmission cycle. It is calculated by using equation 5. By using node 1 and substituting the recorded values into the equation, the total energy consumed per day during the receiving state is 0.1167 mAh.

$$24\ \text{mA} \times \frac{7.75 + 5.5 + 3.25 + 1\ seconds}{3600\ seconds\ /hour} = 0.1167\ \text{mAh/day}$$

Current consumption during the sleep state is calculated based on equation 12.

$$E_{sleep}\ (\text{mAh/day}) = I_{sleepi}\ (\text{mA}) \times \frac{\sum_{i=1}^{n} T_{sleep_i}\ (seconds)}{3600\ seconds\ /hour} \quad (12)$$

Where I_{sleepi} is the instant current in mA during the ith sleep state and T_{sleepi} is the time (in second) a node stays in sleep state during the transmission cycle. It is calculated by using equation 7. By using node 1 and substituting the recorded values into the equations, the total energy consumed per day during the sleep states is 6.0043 mAh/day.

$$T_{sleep} = 21617.75 + 21620 + 21622.25 + 21615.5 = 86475.5\text{s}$$

$$0.25\ \text{mA} \times \frac{86475.5\ seconds}{3600\ seconds\ /hour} = 6.0052\ \text{mAh/day}$$

Summing the energy consumed in the three (3) states together as expressed in equation 13 gives the total energy consumed by node 1 which is 6.4932 mAh/day.

$$E_{node}\ (\text{mAh/day}) = E_{Tx}\ (\text{mAh/day}) + E_{Rx}\ (\text{mAh/day}) + E_{sleep}\ (\text{mAh/day}) \quad (13)$$

$$E_{node}\ (\text{mAh/day}) = 0.3713 + 0.1167 + 6.0052\ \text{mAh/day} = 6.4932\ \text{mAh/day}$$

The battery life is calculated using equation 9. Using a battery of 3000mAh capacity, the battery life is 463 days.

$$\text{Battery Life (day)} = \frac{3000\ mAh}{6.4932\ mAh/day} = 463\ \text{days}$$

Table 4: Battery life calculation for four nodes under GA without CS and GACS

	GA without CS Nodes				GACS Nodes
Transmission Cycle	1	2	3	4	
Position	1	2	3	4	
Rx - Mode Remaining Slot before ACK	3	2	1	0	
ACK Time (s)	1	1	1	1	
ToA (s)	2.17	2.17	2.17	2.17	8.68
TG (s)	0.08	0.08	0.08	0.08	
Slot Time (s)	2.25	2.25	2.25	2.25	
Rx- Mode Time (s)	7.75	5.5	3.25	1	17.5
Instant Rx - Current (mA)	24	24	24	24	24
Instant Tx - Current (mA)	154	154	154	154	154
Sleep Time (s)	21617.75	21620	21622.25	21615.5	86475.5
Instant Sleep - Current (mA)	0.25	0.25	0.25	0.25	0.25
Number of Transmissions per day	4	4	4	4	
Rx - mAh Consumed per day	0.206666667	0.146666667	0.086666667	0.026666667	0.116666667
Tx - mAh Consumed per day	0.371311111	0.371311111	0.371311111	0.371311111	0.371311111
Sleep - mAh Consumed per day	6.004930556	6.005555556	6.006180556	6.004305556	6.005243056
Total mAh Consumed per day	6.582908333	6.523533333	6.464158333	6.402283333	6.493220833
Battery Capacity (mAh)	3000	3000	3000	3000	3000
Battery Life (day)	456	460	465	469	463

Figure 8 shows that all nodes achieved a battery life of 463 days when powered by a 3000 mAh battery and operating under the GACS algorithm. However, varying battery lifespans were observed when the nodes executed the GA without CS algorithm.

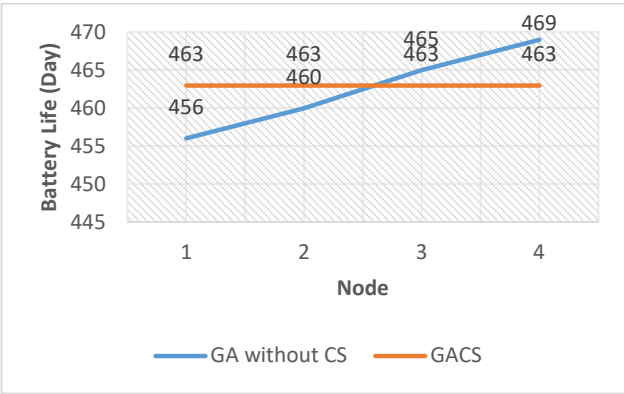


Figure 8: Comparison of battery life for GA without CS and GACS scenarios in a network of 4 nodes

Just like it was done by Kasali et al. (2024) during the simulation stage of this work for the two scenarios, all the transmission parameters remain the same except the payload which was changed to 54 bytes and slot number of the network increased to fifteen (15) nodes. Each node was configured to make fifteen (15) transmissions daily. It was observed, as shown in Figure 9, that the nodes have varying battery life under the GA without CS but the same battery life (434 days) under the GACS scenario.

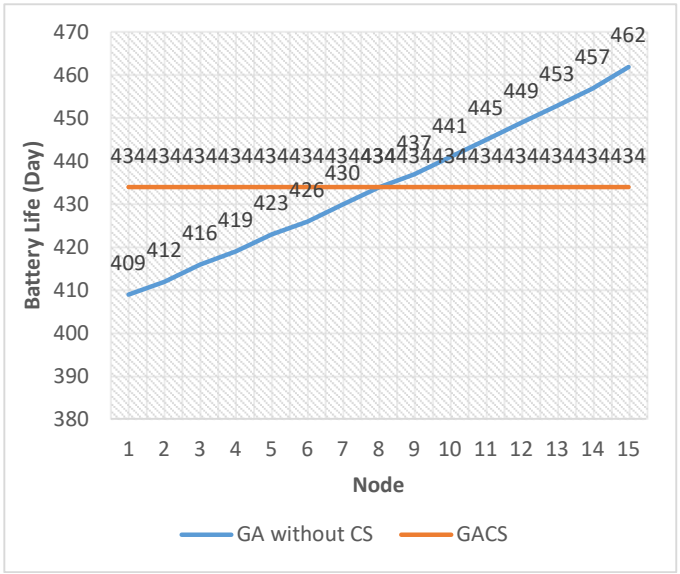


Figure 9: Comparison of battery life for GA without CS and GACS scenarios in a network of 15 nodes

From Figure 8 and Figure 9, it is observed that battery life (which is a function of the current consumption) of nodes under GACS scenario is the (or almost the) same as the battery life of the node at the mid position in the network. Although, nodes after the mid position in the GA without CS scenario

always have higher battery life than nodes that execute GACS algorithm because they are close to the acknowledgement time but nodes before the mid position always have lower battery life when compare to the nodes running GACS algorithm. This imbalance in current consumption is what the proposed GACS model is addressing. Therefore, it can be deduced that GACS model always brings in fairness in current consumptions of node in a time-slotted network.

Comparing experiment 1 and 2, it can be deduced that battery life is inversely proportional to the number of nodes in a group slot (SN). The model maintains fairness and good battery life irrespective of the network size.

To determine the effect of GACS model on Packet Delivery Ratio (PDR) of the network, the GACS code was deployed on the nodes and the nodes were allowed to operate for 19 hours. During this period, each node sent 385 data and were stored in the tbl_data of the gasc database. Examining these data, it was observed that node 1 and node 4 sent all their data correctly to the gateway. However, out of 385 data sent by node 2 and node 3, only one (1) and two (2) data were not properly sent respectively. Table 5 shows the number of correctly and incorrectly received data while Figure 10 shows a sample of incomplete data sent by node 3. The PDR, according to Nandy et al. (2019), can be calculated as expressed in equation 14.

$$PDR = \frac{N_{rp}}{N_{sp}} \tag{14}$$

Where Nsp denotes the total amount of packets transmitted, and Nrp is the total amount of packets received. The PDR of this network is 99.81%.

Table 5: Number of correctly and incorrectly received data

Node Identity	Total Data Sent	Correctly Received Data	Incorrectly Received Data
1	385	385	0
2	385	384	1
3	385	383	2
4	385	385	0
Total	**1540**	**1537**	**3**

Figure 10: Incomplete record sent by node 3

IV. CONCLUSION

This study constructed a prototype using ATmega328p microcontrollers and Ebyte 22 LoRa modules to validate the performance of the GACS protocol in Time-Slotted LoRa WSN. Two scenarios were tested: GA without CS and GACS. The GACS model reallocates transmission time to minimize collisions, achieving a packet delivery ratio of 99.81%. It enhances fairness in current consumption, ensuring uniform battery usage and prolonging the network's overall lifetime. To further optimize battery life, it is recommended to configure transmission parameters such that the number of nodes per group slot remains small. These findings highlight the importance of energy-efficient communication protocols in Time-Slotted LoRa-based WSNs. The prototype can be employed in delay-tolerant applications like multipoint weather stations, energy meters, waste management and precision agriculture. A single channel gateway was constructed in this work. This limit the capability of the model developed. Future work will explore intelligent slot number selection, the impact of multi-channel gateway(s) on fairness, and the re-transmission mechanism.

AUTHOR CONTRIBUTIONS

A. A. Kasali: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. **C. O. Akanbi:** Supervision, Methodology, Writing – original draft & editing. **L. O. Omotosho:** Supervision, Writing – original draft & editing, Writing – review & editing. **I. K. Ogundoyin:** Writing – review & editing.

REFERENCES

Abdelfadeel, K. Q.; D. Zorbas; V. Cionca; and D. Pesch. (2019). FREE - Fine-Grained Scheduling for Reliable and Energy-Efficient Data Collection in LoRaWAN. *IEEE Internet of Things Journal*, *7*(1): 669–683.

Bor, M. C. (2020). Towards the efficient use of LoRa for Wireless Sensor Networks. Unpublished PhD Thesis, School of Computing and Communications, Lancaster University, Lancaster, United Kingdom.

Čelarević, A., and Mrakić, I. (2023). Progress and Perspectives: Lpwa and Lora Networks in the Era of the Iot Revolution. *Nauka i Tehnologija*, *11*(2):104–113.

Ebi, C.; F. Schaltegger; A. Rust; and F. Blumensaat. (2019). Synchronous LoRa Mesh Network to Monitor Processes in Underground Infrastructure. *IEEE Access*, *7*: 57663–57677.

Gresl, J.; S. Fazackerley; and R. Lawrence. (2021). Practical precision agriculture with LoRa based wireless sensor networks. Paper presented at *SENSORNETS 2021 - Proceedings of the 10th International Conference on Sensor Networks*, *Sensornets*, LDA, Portugal, 131–140.

Haubro, M.; C. Orfanidis; G. Oikonomou; and X. Fafoutis. (2020). TSCH-over-LoRA : long range and reliable IPv6 multi-hop networks for the internet of things . *Internet Technology Letters*, *3*(4): 1-6.

Haxhibeqiri, J.; E. De Poorter; I. Moerman; and J. Hoebeke. (2018). A survey of LoRaWAN for IoT: From technology to application. *Sensors (Switzerland)*, *18*(11): 2-38.

Kasali, A. A.; C. O. Akanbi; L. O. Omotosho; and I. K. Ogundoyin (2024). Integrating Fairness in Current Consumption of End Devices in Time-Slotted LoRa-based

Wireless Sensor Network. *Annals of the Faculty of Engineering Hunedoara*, *22*(2): 153–162.

Ketshabetswe, L. K.; A. M. Zungeru; M. Mangwala; J. M. Chuma; and B. Sigweni. (2019). Communication protocols for wireless sensor networks: A survey and comparison. *Heliyon*, *5*(5): 1-43.

Khan, M. A.; M. Anjum; S. A. Hassan; and H. Jung. (2023). Applications of LPWANs. In Low-Power Wide-Area Networks: Opportunities, Challenges, Risks and Threats (171-209). Cham: Springer International Publishing.

Migabo, E.; K. Djouani; A. Kurien; and T. Olwal. (2017). A Comparative Survey Study on LPWA Networks: LoRa and NB–IoT. *Proceedings of the Future Technologies Conference (FTC)*, *November 2017*, Vancouver, Canada, 29–30.

Lakshmi, G. P.; P. N. Asha; G. Sandhya; S. V. Sharma; S. Shilpashree; and S. G. Subramanya. (2023). An intelligent IOT sensor coupled precision irrigation model for agriculture. Measurement: Sensors, 25, 100608.

Nandy, T.; M. Y. I. B. Idris; R. M. Noor; M. L. M. Kiah; L. S. Lun; N. B. A. Juma'at; I. Ahmdy; N. AbdulGhani; and B. Bhattacharyya. (2019). Review on Security of Internet of Things Authentication Mechanism. *IEEE Access*, *7*, 151054–151089.

Sharma, K.; A. Malik; I. Batra; A. S. M. Sanwar Hosen; M. A. Latif Sarker; and D. S. Han. (2023). Technologies Behind the Smart Grid and Internet of Things: A System Survey. *Computers, Materials and Continua*, *75*(3), 5049–5072.

Zorbas, D. (2020). A Testbed for Time-Slotted LoRa Communications. Proceedings - 21st IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2020, Ireland, 182–184.

Zorbas, D., Abdelfadeel, K., Kotzanikolaou, P., and Pesch, D. (2020). TS-LoRa: Time-slotted LoRaWAN for the Industrial Internet of Things. *Computer Communications*, *153*, 1–10.