Integrating Fairness in Current Consumption of End Devices in Time-Slotted LoRa-based Wireless Sensor Network

*[1]Abdulwakil A. Kasali, [2]Caleb O. Akanbi, [2]Lawrence O.Omotosho, and [2]Ibrahim K. Ogundoyin
[1]Department of Computer Engineering Technology, Federal Polytechnic, Ede, Nigeria
[2]Department of Information and Communication Technology, Osun State University, Osogbo, Nigeria
gabdulwakil@gmail.com|{akanbico | lawrence.omotosho | ibraheem.ogundoyin}@uniosun.edu.ng

Received: 29-MARCH-2024; Reviewed: 21-APRIL-2024; Accepted: 29-APRIL-2024
http://dx.doi.org/10.46792/fuoyejet.v9i2.9

ORIGINAL RESEARCH

Abstract— LoRaWAN, a Low Power Wide Area (LPWA) networking protocol, allows devices to share a communication link and transmit data randomly. Its scalability is hindered by high collision rates and duty cycle restrictions, leading to uneven power usage and network instability. In contrast, time-slotted communications offer a solution by dividing time into fixed slots, ensuring fair and efficient network access. However, fairness in current consumption of end devices has been overlooked in previous studies, potentially causing network instability. This research introduces Group Acknowledgement Circular Shift (GACS) algorithm, combining group acknowledgment with a new Circular Shift method. Using MATLAB simulations and Jain's Fairness Index, two scenarios were tested: one with Group Acknowledgement (GA) without Circular Shift (CS) and another with GACS. Results on a network with ten devices and one gateway over ten cycles showed fairness improved from 98.32% with GA without CS to 99.75% with GACS. Even with 100 transmission cycles and varied parameters, GACS consistently outperformed GA without CS, highlighting its strong fairness index and ability to maintain uniform current consumption across nodes. Overall, the study emphasizes the robust fairness index of the GACS algorithm, irrespective of the number of nodes (SN) within each group slot, with nodes consistently exhibiting uniform current consumption at every SN^{th} cycle.

Keywords— Current Consumption, Jain's Fairness Index, LoRa, Time-slotted, WSN.

1 INTRODUCTION

Wireless Sensor Networks (WSNs) are networks of a large set of disposable unattended sensors, which are formed by the deployment of micro sensors that are equipped with data processing and communication capabilities (Ketshabetswe et al., 2019). In recent years, WSNs have proven to be effective solutions for a wide range of Internet of Thing (IoT) applications. The primary function of this network is to sense various physical and environmental parameters (such as temperature, vibration, humidity, and others), and transmit them to a sink (Guravaiah et al., 2021; Oluwaranti & Ayanda, 2011).

A natural and suitable communication link that allows these microsensors (nodes) to collaborate among themselves is a wireless network. Wireless communications, according to Murdyantoro et al., (2019), come in various forms, technologies, and delivery methods which include cellular, short range and long range. Among these technologies, some also belong to the Low Power Wide Area (LPWA) technologies. Examples are Narrow Band IoT (NB-IoT), Long Range (LoRa), Sigfox, and others. They offer low-power, low-cost and low-complexity end devices that can communicate wirelessly over large distances (Haxhibeqiri et al., 2018).

*Corresponding Author

Section B- ELECTRICAL/COMPUTER ENGINEERING & COMPUTING SCIENCES
Can be cited as:
Kasali A. A., Akanbi C. O., Omotosho L. O., and Ogundoyin I. K. (2024). Integrating Fairness in Current Consumption of End Devices in Time-Slotted LoRa-based Wireless Sensor Network, FUOYE Journal of Engineering and Technology (FUOYE**JET), 9(2), 213-220.**
http://dx.doi.org/10.46792/fuoyejet.v9i2.9

This makes them suitable for WSNs because WSNs are mostly deployed in remote areas and hence, operate on batteries. As mentioned by (Migabo et al., 2017), NB-IoT and LoRa are the two most promising LPWA technologies. Unlike the NB-IoT, LoRa networking is an open-source technology that enables autonomous network setup at a low cost. It uses Chirp Spread Spectrum (CSS) modulation and has a flexible deployment model that allows creating private network (Le and Giap, 2020). LoRa typically operates in the sub GHz Industrial, Scientific, and Medical (ISM) band, is low-power and communicate bi-directionally, albeit half-duplex (Gresl et al., 2021).

LoRa transceiver offers flexible transmission parameters that impact its performance in terms of adaptability to diverse applications, balancing data rate, range, energy efficiency, and interference resilience (Bor & Roedig, 2018). These parameters are Transmission Power (TP), Carrier Frequency (CF), Bandwidth (BW), Spreading Factor (SF), and Coding Rate (CR). LoRa radios can adjust its TP from -4 dBm to 20 dBm (default is 14dBm), affecting signal quality and energy consumption. The CFs used by LoRa can be set between 137 MHz and 1020 MHz, but hardware may limit it to a narrower range (e.g., 868 MHz, 915 MHz or 433 MHz). The BW controls the frequency width. The higher BW means faster data but lower sensitivity. Common choices are 500 kHz, 250 kHz, and 125 kHz, with a range of 7.8 kHz to 500 kHz. The SF ranges from 6 to 12, affecting transmission speed, range, sensitivity, and interference resistance. LoRa CR defines Forward Error Correction rate (e.g., 4/5, 4/6), offering interference protection at the cost of airtime. Radios with

different CR settings can still communicate using a payload header.

To calculate the time required for transmitting a LoRa frame, otherwise referred to as Time on Air (ToA), from one node to another, given the BW, SF, and CR, the transmission time of the preamble ($T_{preamble}$) and payload ($T_{payload}$) must be added together as shown in equation 1. Details on how to calculate the $T_{preamble}$ and $T_{payload}$ can be seen in (Semtech Corporation, 2013).

$$ToA = T_{preamble} + T_{payload} \quad (1)$$

Radio devices (End Devices and gateways), equipped with LoRa and LoRaWAN, a standard LPWA networking Medium Access Control (MAC) that employs an ALOHA-style protocol, can access a shared communication link and transmit data randomly. However, as the number of devices increases, LoRaWAN confronts scalability difficulties stemming from the high collision probability of its MAC layer. Moreover, the performance degrades even more when using acknowledged transmissions due to duty cycle limitations at the gateway (Abdelfadeel et al., 2019). The high collision also necessitates retransmissions which in turn results in uneven and more power consumption among the affected end devices. Therefore, some nodes deplete their battery current faster than others, causing network partitioning and reducing network lifetime. This limitation has prompted researchers to explore alternative medium access approaches to support remote applications with strict network requirements.

Unlike LoRaWAN, time-slotted communications offer competing end devices fair access to the communication channel and reduces collisions which is inhibited in standard LoRaWAN MAC thus, achieving the desired level of network reliability. In time-slotted communications, a time is divided into multiple time-slots. The size of a time-slot is typically fixed and determined by factors such as payload size and radio characteristics. With this approach, multiple users can share the same radio frequency without colliding with each other, as long as they are assigned to different time-slots. The assignment of time-slots is a fundamental process in time-division protocols and is usually managed by a central coordinator, as seen in cellular networks. End devices sleep, wake up, receive, and transmit data synchronously. This synchronization ensures coordinated operation and efficient use of resources within the network while minimizing energy usage (Bor, Vidler, et al., 2016; Zorbas, 2020).
It also improves scalability, data delivery, and device lifetime as reported in the works of (Abdelfadeel et al., 2019; Ebi et al., 2019; Haubro et al., 2020; Zorbas et al., 2020). However, these authors, and many other reviewed works, did not consider fairness in current consumption which could lead to network partitioning and consequently network instability. This work brings in fairness in current consumption. It combines group acknowledgement technique used in (Zorbas et al., 2020) with a new Circular Shift algorithm forming Group Acknowledgement Circular Shift (GACS) algorithm. GACS algorithm will promote fair current consumption among nodes, thereby increasing the device lifetime and improving the overall performance of LoRa-based time-slotted WSNs. The remainder of this paper is organized as follows. Section 2 discusses how the Circular Shift Algorithm was formulated and modelled. Section 3 presents the results of the simulations. The final section presents the conclusion.

2 MATERIALS AND METHOD

The Circular Shift Algorithm can be viewed as a modular operation that rotates the elements of a sequence, where the rotation distance is determined by the size of the sequence. It can be used to schedule data transmission and reception among nodes in a sensor network. The Circular Shift Algorithm can be understood by considering it in the context of modular arithmetic, which involves performing arithmetic operations on integers within a finite range, called a modulus. When two integers are divided, the remainder is the focus of attention rather than the quotient. For example, in modulo 5 arithmetic, if we add 2 to 4, we get 1, because 4+2=6, and the remainder when 6 is divided by 5 is 1.
This work covers the communication interactions between End Devices (EDs) and a Gateway (GW). The conceptual diagram of the GACS model is as shown in Fig. 1 where a set of EDs transmit their data, each at allotted time, to the GW and receive a Group Acknowledgement (G_ACK) at the same time. The total uplinks (ULs) time, as depicted in Fig. 2, for a given set of EDs is refers to as Uplink Group Slot Time (T_{ULGS}). The T_{ULGS} can be expressed as shown in equation 2.

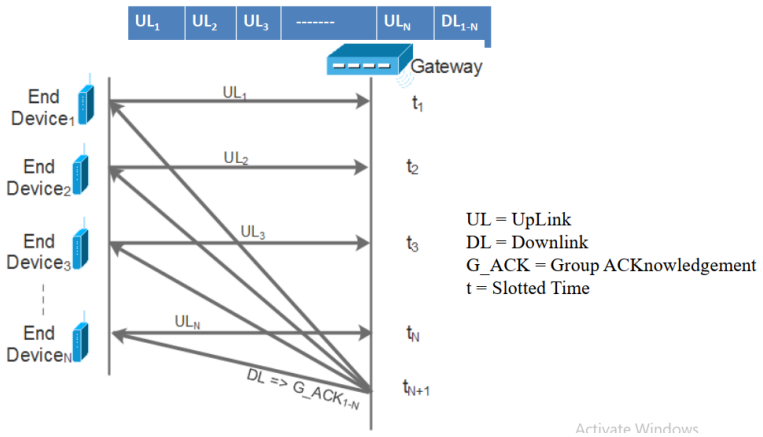

Fig. 1: Conceptual Diagram of a Single Group-Ack Circular Shift Model

$$T_{ULGS} = S_N \times T_S \quad (2)$$

Where T_S is the slot time of each ED and S_N is the number of slots in the given set. The T_S is also expressed as shown in equation 3.

$$T_S = ToA \times T_G \quad (3)$$

https://dx.doi.org/10.4314/fuoyejet.v9i2.9 engineering.fuoye.edu.ng/journal

Where T_G is the Guard Time. It is used to separate data transmissions between two EDs to avoid collision. Substituting equation 3 in equation 2 gives equation 4.

$$T_{ULGS} = S_N\,(ToA + T_G) \tag{4}$$

Therefore, T_G can be expressed as:

$$T_G = \frac{T_{ULGS} - (S_N \times ToA)}{S_N} \tag{5}$$

And S_N is reduced to:

$$S_N = INT\left[\frac{T_{ULGS}}{ToA}\right] \tag{6}$$

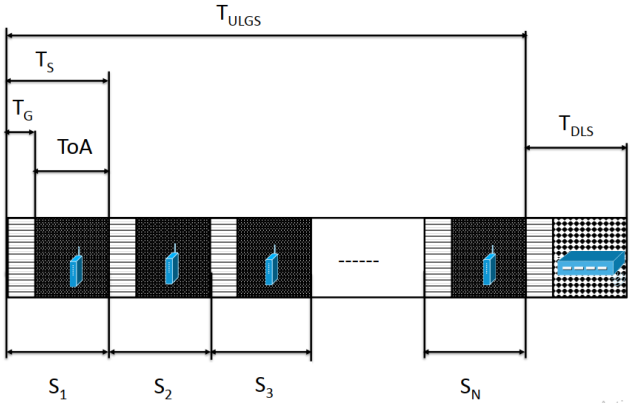

Fig. 2: A Single Group-Ack Circular Shift Model Formulation

because, S_N depends largely on the ToA. T_{DL} is the downlink time. This is the time the GW spends to send ACK to the given set of EDs and it occurs at regular interval of time known as *ACK_Cycle_Time*. During this period, the GW can also add a new ED to the network if there is a join request. For an illustration, assuming T_{DL} and T_{ULGS} are set to 1s and 9s respectively, it means the *Ack_Cycle_Time* occurs at every 10s. Therefore, *Ack_Cycle_Time* can be expressed as shown in equation 7.

$$Ack\_Cycle\_Time = T_{ULGS} + T_{DL} \tag{7}$$

The activity diagram of the GACS protocol model as shown in Fig. 3 describes the flow of communication between Gateway (GW) and End Devices (EDs). The GW (likewise the EDs) initializes itself after taking in as inputs the LoRa transmission parameters and calculate the ToA using equation 1.

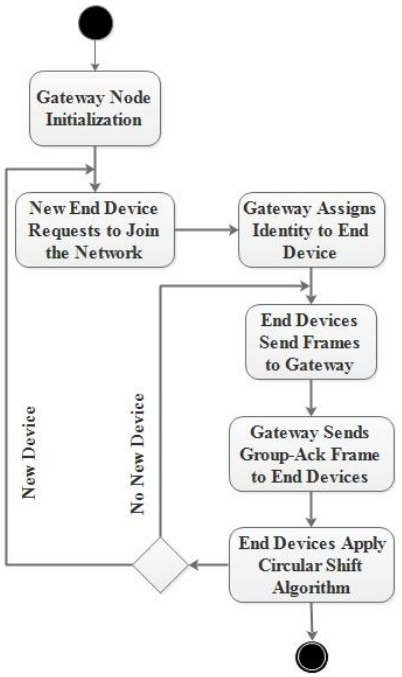

Fig. 3: Activity Diagram of Group-Ack Circular Shift Model

Then it obtains the T_G and S_N using equation 5 and 6 respectively. During the ACK_Cycle_Time, period, a new ED can send a join request to the GW. If the request is received by the GW, the GW will assign an identity (which means the node position in queue) to the ED. A set of S_N (2, 3, or more) EDs that belong to a particular group will then send their data to the GW in the order of their current node position. On receipt of these data by the GW, the GW will send an Acknowledgement (ACK) and the last ED identity to the EDs. On receipt of the ACK, the EDs will execute the GACS algorithm in Fig. 4.

The GW then check for new ED to add to the network. If there is any, it adds, and if otherwise, it listens to receive data from another set of EDs while the previous set of EDs go into sleep mode.

The GACS algorithm is a code segment that runs on each ED. The code segment, as shown in Fig. 4, requires three variables to calculate new node position. The variables are the current ED position (pos), slot number (SN) and the last ED position (lastNode). While SN is calculated by the EDs using equation 6, the pos and the lastNode are received by the ED from the GW during join request period. For an illustration, assuming there are ten (10) EDs in a network and each of the ED transmits data once in every iteration of ten (10) times. As shown in Fig. 5, Node J is the last node, therefore, the lastNode variable is equal to 10 (lastNode = 10). Assuming the SN is calculated and it is equal to 4 (SN = 4), the GACS algorithm will calculate the new position for node F which, as shown in Fig. 5, is in position 6 (that is, pos = 6) as follows:

Position 6 will be used for the first transmission. After the node receives it's ACK, line 10 of the GACS code segment will check for the remainder of the division of pos by SN. If the remainder is equal to 1, then it assigns pos to

https://dx.doi.org/10.4314/fuoyejet.v9i2.9 engineering.fuoye.edu.ng/journal

```
Group Acknowledgement Circular Shift Algorithm
1   %pos = last known position of the end node
2   %SN  = Slot Number gotten from the mathematical model developed
3   %lastNode = lastnode position assigned and known by the gateway node
4   %Tstart =  Start Time of the gateway node sent to each end node
5   %TG = Guard Time gotten from the mathematical model developed
6   %ToA = Time on Air
7   %TxCycle = Transmission Cycle
8   %Tdownlink = Downlink Time
9   Tlastnode = Tstart + (lastNode - 1)*(TG + ToA)
10   if rem(pos,SN) == 1
11    slot_begin = pos;
12  end
13  new_pos = (rem(pos,SN) + 1) + (fix(pos/SN) * SN);
14  if rem(pos,SN) == 0
15    new_pos = (pos - SN) + 1;
16  end
17  if new_pos > lastNode
18    new_pos = slot_begin;
19  end
20  pos = new_pos;
21  if mod(pos,SN) ~= 0
22    diff = pos - (G_Ack_Cycle * SN);
23 if diff > 0
24   slot_remain =  remain - diff  % remaining node(s) to the end of a grouped slot
25 else
26  slot_remain =  SN - mod(pos,SN)  % remaining node(s) to the end of a grouped slot
27 end
28 else
29   slot_remain =  mod(pos,SN)  % remaining node(s) to the end of a grouped slot
30 end
31 timebeforeGACK = (pos - 1);
32 timeafterGACK = (lastNode - pos)- slot_remain;
33 otherGACK = (G_Ack_Cycle - 1);
34 Tsleep = ((timebeforeGACK + timeafterGACK) * (TG + ToA)) + (otherGACK * Tdownlink) + ntervalbtwcycle_s
35 NodeNextTxTime = (TxCycle * Tlastnode) + Tsleep;
```

Fig. 4: A Code Segment of the Group Acknowledgement Circular Shift Algorithm

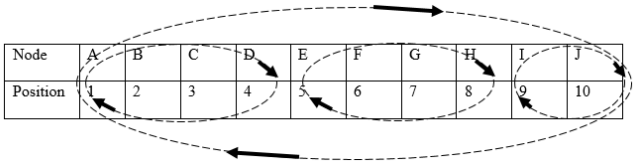

Fig. 5: Behaviour of the GACS Algorithm

variable slot_begin (line 11 executed). Otherewise, it jumps to line 13. In this case, pos = 6, SN = 4 and the remainder is not equal to 1. Therefore, line 13 is executed. On line 13, (rem(pos,SN) + 1) evaluates to 3 and (fix(pos/SN) * SN) evaluates to 4. Therefore, 7 will be returned into new_pos variable. The Evaluation of the *if* statement on line 14 is false and therefore, execution jumps to line 17. The *if* statement on that line also returns false. Hence, line 20 is executed and pos variable is updated with 7, which is the value in *new_pos* variable. Node F assumes position 7 during the second transmission. The GACS algorithm is executed again immediately after the group ACK and make node F to assume position 8. After the third transmission and the group ACK, the position of node F changes from 8 to 5 thus:

Line 10 of the GACS code segment will check for the remainder of the division of pos (which is equal to 8) by SN (which is equal to 4). The remainder is not equal to 1, therefore, it jumps to line 13. On line 13, (rem(pos,SN) + 1) evaluates to 1 and (fix(pos/SN) * SN) evaluates to 8. Therefore, 9 will be returned into *new_pos* variable. The Evaluation of the *if* statement on line 14 is true and therefore, (pos - SN) + 1 on line 11 is evaluated and updated the *new_pos* variable with 5. The *if* statement on line 17 is evaluated and returned false. Hence, line 20 is executed and pos variable is updated with 5, which is the last known value in the *new_pos* variable. Node F assumes position 5 and hence, moves in a cyclic order. The behaviour of this algorithm for ten (10) nodes is as shown in Fig. 5. Node new position (*new_pos*) is calculated based on line 10 through 20 and this can be represented mathematically as shown in equation 8 where *slot_begin* is expressed as shown in equation 9.

$$new\_pos = \begin{cases} ((pos\ \%\ SN) + 1) + \left(INT\left(\frac{pos}{SN}\right) \times SN\right) & if\ pos\ \%\ SN > 0 \\ (pos - SN) + 1 & if\ \ pos\ \%\ SN = 0 \\ slot\_begin & if\ pos > lastnode \end{cases} \quad (8)$$

$$slot\_begin = pos \qquad if\ pos\ \%\ SN = 1 \quad (9)$$

The consequence of changing positions by nodes in this network is to balance current consumption among them. A working node changes from one state to another and current consumption in each state differs from one another. A node can be in *transmit*, *idle* (Waiting for ACK), *receive* or *sleep* state. These states transitions affect the components that made up of a node, especially the microcontroller unit (MCU) and the transceiver unit. For example, current consumption of an MCU (ATmega328p) and a LoRa module (Ebyte 22) are extracted from their respective datasheets (Atmel, 2016; Ebyte, 2023) and depicted in Table 1. Table 2 shows the states of a sensor node and that of it's components. Juxtaposing Table 2 and Tables 1, it clearly shows that a node consumes highest current when it is in transmit state and least current when it is in sleep state.

Table 1: Node Components' States and their Current Consumptions (Atmel, 2016; Ebyte, 2023)

States	Atmega328P @ 5V and 16MHz	States	LoRa E22 Ebyte @ 5V and 433MHz
Run	24mA (120mW)	Tx	110 mA (550mW)
Idle	12mA (60mW)	Rx	12mA (60mW)
Sleep	0.00012mA – PowerDown (0.0006mW)	Sleep	2uA (10uW) = (0.01mW)

Table 2: State of Sensor Node and its Components

https://dx.doi.org/10.4314/fuoyejet.v9i2.9 engineering.fuoye.edu.ng/journal

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

For these reasons, authors of Time-slotted MAC protocols design their protocols to allow node to stay longer in *sleep* state in order to conserve available energy of the battery. This work adopts this approach but, unlike others, re-arrange transmission order of the nodes after every transmission to bring in fairness in their consumptions. Time taken to stay in the *idle* and *sleep* states are implemented in line 21 through 35 of the algorithm presented in Fig. 4. The idle state is represented as *slot_remain* and it is calculated by line 21 through 30 based on the conditions in line 21 and 23. The *sleep* period is calculated according to the equation in line 34 while the node next transmission time is calculated based on the equation in line 35.

3 Results and Discussion

This model is simulated in MATrix LABoratory (MATLAB) R2020a. Fig. 6 depicts the interface of the simulator developed and the transmission parameters used are as shown in Table 3. With these parameters, the Time on Air (ToA), Guard Time (TG) and Nos of Slots per ACK Cycle (SN) evaluate to 2167.36ms (2.1674s), 82.6426ms (0.0826s), and 4 respectively.

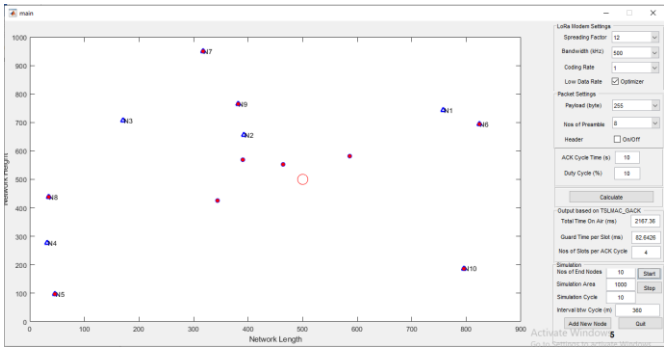

Fig. 6: Simulation Interface in MATLAB

Simulations were carried out for two scenarios namely: when there is Group Acknowledgement (GA) without Circular Shift (CS) and when there is Group Acknowledgement with Circular Shift (GACS). In the first instance, each of the simulation contains ten (10) End Devices and one (1) Gateway. Each run was repeated ten (10) times using the same parameters. After each run, data containing current consumption by each End Device is exported into an excel file and graphs were subsequently generated.

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

Parameters	Value
Spreading Factor (SF)	12
Bandwidth (BW)	500
Coding Rate (CR)	1
Header (H)	0
Data Optimizer (DO)	1
Number of Preamble (NP)	8
Payload	255 bytes
AckCycleTime	10 s
Duty Cycle	10%
Nos of End Devices	10
Simulation Area	1000
Simulation Cycle	10
Interval between Cycle	360 minutes

Later, the transmission cycles was increased from 10 to 100 in order to see the effect of the scenarios during short and long periods.

Equation 10, 11, 12, and 13 are used to calculate current consumption of each node during the *transmit*, *AckWaiting*, *receive* and *sleep* states in a transmission cycle. Equation 14 is used to computes the total current consumed by the node. The amount of currents drawn from their respective batteries during the *transmit*, *receive*, and the *sleep* states are the same at each transmission cycle except during their *AckWaiting* state.

$$I_{Node-Tx} = T_S \times \left(I_{mcu-running-state} + I_{lora-Tx-state}\right) \quad (10)$$

$$I_{Node-wait-4-ACK} = \left(slot_{remain-b4-ACK} \times T_S\right) \times \left(I_{mcu-idle-state} + I_{lora-Rx-state}\right) \quad (11)$$

$$I_{Node-Rx} = \left(T_{downlink} \times I_{mcu-running-state}\right) \times \left(GW_{ACK-ToA} + I_{lora-Rx-state}\right) \quad (12)$$

$$I_{Node-sleep} = T_{sleep} \times \left(I_{mcu-sleep-state} + I_{lora-sleep-state}\right) \quad (13)$$

$$I_{Node-Total} = I_{Node-Tx} + I_{Node-wait-4-ACK} + I_{Node-Rx} + I_{Node-sleep} \quad (14)$$

Fig. 7 shows the result for the scenario GA without CS wherein some nodes drew the same amount of currents (e.g. node 1 and 5; 2 and 6; 3, 7 and 9; 4, 8, and 10) because they have the same waiting time to the Ack period. This is the reason their lines are superimposed on one another. Node 1 and node 5 drew the same and highest current. Followed by node 2 and node 6 and so on. Hence, the longer the waiting time, the more the current consumed. Meanwhile, for scenario GACS, as depicted in Fig. 8, nodes' waiting time to the Ack period are re-arranged after each transmission cycle and this has brought a reduced and near even consumption of current from their

https://dx.doi.org/10.4314/fuoyejet.v9i2.9 engineering.fuoye.edu.ng/journal

respective batteries. At every SNth cycle, nodes in a full group slot consumed the same amount of current. It is also noticed that node 9 and node 10 which belong to the last group slot consumed far less than (almost 0.33 of) the nodes in the other two full group slots (node 1, 2, 3, 4 and node 5, 6, 7, 8). This is because, they have smaller waiting time to Ack period and the circular shift involves only the two of them.

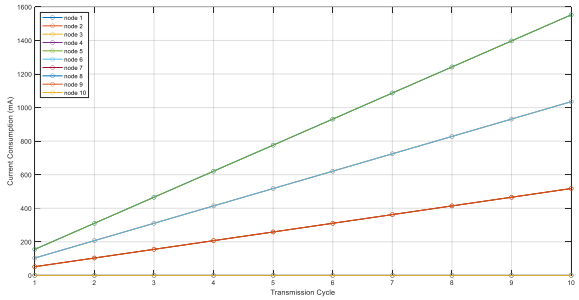

Fig. 7: Current Consumption of End Devices Waiting for Ack Versus Transmission Cycle Using Group Ack without Circular Shift

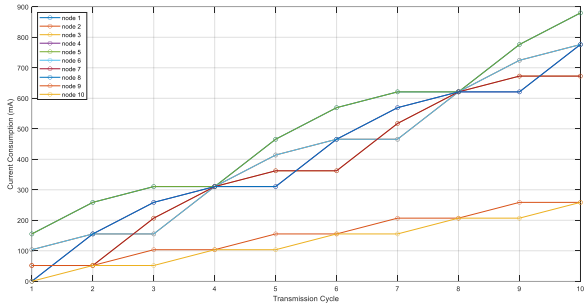

Fig. 8: Current Consumption of End Devices Waiting for Ack Versus Transmission Cycle Using Group Ack with Circular Shift (GACS)

Current consumptions during the *AckWaiting* states of the two scenarios bring in variations. While the consumption in the GACS brings in near-even consumption, the GA without CS could not achieve this. Fig. 9 and Fig. 10 show the current consumption of nodes using GA without CS and GACS respectively.

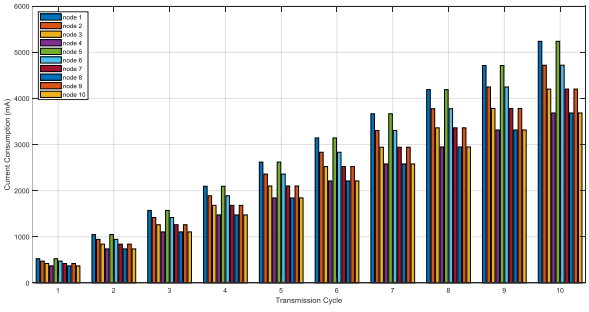

Fig. 9: Total Current Consumption of End Devices Versus Transmission Cycle Using Group Ack without Circular Shift

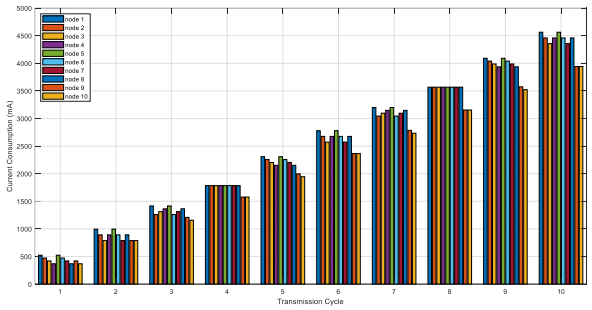

Fig. 10: Total Current Consumption of End Devices Versus Transmission Cycle Using Group Ack with Circular Shift (GACS)

Fairness in current consumption is the main goal of this work. Examining Fig. 10, it is observed that current consumption of individual End Device, is almost the same especially when the transmission cycle is equal to the SN. This has only be proven graphically. To analyse the fairness quantitatively, Jain's fairness index is employed. The final consumptions of each node, as shown in Table 4, are used to calculate the Jain's fairness index which is the ratio of the arithmetic mean of the expected shares of the current consumed and the geometric mean of the expected shares of the current consumed. Equation 15 shows the mathematical expression.

$$f(X) = \frac{[\sum_{i=1}^{n} x_i]^2}{n\sum_{i=1}^{n} x_i^2} \quad (15)$$

Where $0 \leq f(X) \leq 1$ and x_i represents current consumption at cycle i up till cycle n. The closer $f(X)$ to 1, the better the fairness. From Table 4, it is observed that fairness under GA without CS is 98.32% but was improved to 99.75% as shown in Table 5 by using the GACS.

Table 4: Jain's Fairness Index for Ten (10) Transmission Cycles using GA without CS

Node (N)	Current Consumed (mA)	Individual Share (IS)	Square of IS
1	5235.3947	0.120197035	0.014447327
2	4717.9418	0.10831707	0.011732588
3	4200.4889	0.096437106	0.009300115
4	3682.8452	0.08455276	0.007149169
5	5235.3947	0.120197035	0.014447327
6	4717.9418	0.10831707	0.011732588
7	4200.4889	0.096437106	0.009300115
8	3682.8452	0.08455276	0.007149169
9	4200.4889	0.096437106	0.009300115
10	3682.9406	0.084554951	0.00714954
Total	**43556.7707**	**1**	**0.101708054**
Arithemetic Mean	(Summation of IS) ^ 2	1	

https://dx.doi.org/10.4314/fuoyejet.v9i2.9 engineering.fuoye.edu.ng/journal

Geometric Mean	(Summation of Square of IS) x N	1.017080544
Jain's Fairness	Arithemetic Mean / Geometric Mean	98.32%

The transmission cycle of the simulation was increased to 100 while other parameters remain the same. The simulations were conducted under the two scenarios and there was no difference in the Jain's fairness index. This shows that irrespective of the transmission cycle, for a given transmission parameters, their level of fairness in terms of current consumption will remain the same. In another simulation for the two scenarios, all the parameters remain the same except the payload which was changed to 54 bytes, number of nodes set to 50 and the transmission cycle set to 50. The reduction in the payload from 255 to 54 bytes decreases the ToA to 586.63ms (0.57s) and increases the SN to 15. For the GA without CS, the Jain's fairness index recorded was 94.22% while that of the GACS was 98.68%.

Table 5: Jain's Fairness Index for Ten (10) Transmission Cycles using GACS

Node (N)	Current Consumed (mA)	Individual Share (IS)	Square of IS
1	4562.6678	0.1047522	0.010973024
2	4459.1772	0.1023762	0.010480889
3	4355.6676	0.0999997	0.009999956
4	4459.1581	0.1023757	0.010480799
5	4562.6678	0.1047522	0.010973024
6	4459.1772	0.1023762	0.010480889
7	4355.6676	0.0999997	0.009999956
8	4459.1581	0.1023757	0.010480799
9	3941.7148	0.0904960	0.008189531
10	3941.7148	0.0904960	0.008189531
Total	**43556.771**	**1**	**0.100248399**

Arithemetic Mean	(Summation of IS) ^ 2	1
Geometric Mean	(Summation of Square of IS) x N	1.002483992
Jain's Fairness	Arithemetic Mean / Geometric Mean	99.75%

Summarily, when SN (the number of nodes that receive acknowledgement together) is small, the fairness in current consumption among nodes in a network is high irrespective of the scenario adopted (that is, whether GA without CS or GACS). However, when SN is high, GACS maintains high fairness index while that of GA without CS is reduced.

4 CONCLUSION

This work has produced a GACS algorithm that brings in fairness in current consumption of end devices in a time-slotted LoRa-based WSN. The algorithm rotates/schedule data transmission and reception time of nodes based on the SN. The Algorithm can be viewed as a modular operation that rotates the elements of a sequence, where the rotation distance is determined by the size of the sequence. This work has also produce a model that divides a sensor network into group slots based on the ToA and *Ack_Cycle_Time*. Results obtained have shown that the GACS algorithm has high fairness index irrespective of the number of nodes (SN) in each group slot. Also, at every SN^{th} cycle, nodes in a full group slot consumed the same amount of current.

Effort is on-going to implement this model on real hardware. ATmega328p microcontrollers, Ebyte E22-400T30D LoRa modules and some weather sensors have been purchased. Currently the circuits are at the breadboarding and Arduino program writing stage. Result obtained will be presented in our next publication.

References

Abdelfadeel, K. Q., Zorbas, D., Cionca, V., & Pesch, D. (2019). FREE - Fine-Grained Scheduling for Reliable and Energy-Efficient Data Collection in LoRaWAN. *IEEE Internet of Things Journal*, *7*(1), 669–683. https://doi.org/10.1109/JIOT.2019.2949918

Atmel. (2016). *ATmega328/P Summary*. Retrieved September 25, 2023, from https://datasheet.octopart.com/ATMEGA328P-MU-Microchip-datasheet-65729177.pdf

Bor, M., & Roedig, U. (2018). LoRa transmission parameter selection. *Proceedings - 2017 13th International Conference on Distributed Computing in Sensor Systems, DCOSS 2017*, *2018-Janua*, 27–34. https://doi.org/10.1109/DCOSS.2017.10

Ebi, C., Schaltegger, F., Rust, A., & Blumensaat, F. (2019). Synchronous LoRa Mesh Network to Monitor Processes in Underground Infrastructure. *IEEE Access*, *7*, 57663–57677. https://doi.org/10.1109/ACCESS.2019.2913985

Ebyte. (2023). *E22-400TXXX Data Sheet*. Retrieved September 25, 2023, from https://www.cdebyte.com/pdf-down.aspx?id=2179

Gresl, J., Fazackerley, S., & Lawrence, R. (2021). Practical precision agriculture with LoRa based wireless sensor networks. *SENSORNETS 2021 - Proceedings of the 10th International Conference on Sensor Networks, Sensornets*, 131–140. https://doi.org/10.5220/0010394401310140

Guravaiah, K., Kavitha, A., & Leela Velusamy, R. (2021). Data Collection Protocols in Wireless Sensor Networks. In *Wireless Sensor Networks - Design, Deployment and Applications*. https://doi.org/10.5772/intechopen.93659

Haubro, M., Orfanidis, C., Oikonomou, G., & Fafoutis, X. (2020). TSCH-over-LoRA : long range and reliable IPv6 multi-hop networks for the internet of things . *Internet Technology Letters*, *3*(4). https://doi.org/10.1002/itl2.165

Haxhibeqiri, J., De Poorter, E., Moerman, I., & Hoebeke, J. (2018). A survey of LoRaWAN for IoT: From technology to application. *Sensors (Switzerland)*, *18*(11). https://doi.org/10.3390/s18113995

Ketshabetswe, L. K., Zungeru, A. M., Mangwala, M., Chuma, J. M., & Sigweni, B. (2019). Communication protocols for wireless sensor networks: A survey and comparison. *Heliyon*, *5*(5), e01591. https://doi.org/10.1016/j.heliyon.2019.e01591

Le, N. G. O. P., & Giap, L. N. (2020). *Remote Monitoring System For Independent Power Stations In Rural And Mountainous Areas In Vietnam*. *15*(3), 18–27. https://doi.org/10.9790/2834-1503021827

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

Murdyantoro, E., Wisnu, A., Nugraha, W., & Wisnu, A. (2019). *A review of LoRa technology and its potential use for rural development in Indonesia A Review of Lora Technology and Its Potential Use for Rural Development in Indonesia*. *020011*(April).

Oluwaranti, A., & Ayanda, D. (2011). Performance Analysis of an Enhanced Load Balancing Scheme for Wireless Sensor Networks. *Wireless Sensor Network*, *03*(08), 275–282. https://doi.org/10.4236/wsn.2011.38028

Semtech Corporation. (2013). *SX1272/3/6/7/8 LoRa Modem Design Guide, AN1200.13*. Retrieved July 9, 2023, fromhttps://www.openhacks.com/uploadsproductos/loradesignguide_std.pdf

Zorbas, D. (2020). Design Considerations for Time-Slotted LoRa(WAN). *MaDeLoRa 2020: 1st Workshop on Massive LoRa Deployments: Challenges and Solutions*.

Zorbas, D., Abdelfadeel, K., Kotzanikolaou, P., & Pesch, D. (2020). TS-LoRa: Time-slotted LoRaWAN for the Industrial Internet of Things. *Computer Communications*, *153*(October 2019), 1–10. https://doi.org/10.1016/j.comcom.2020.01.056

9999