Design of A Minimal Overhead Control Traffic Topology Discovery and Data Forwarding Protocol for Software-Defined Wireless Sensor Networks

Simon Atuah Asakipaam, Jerry John Kponyo, Justice Owusu Agyemang and Fredrick Appiah-Twum

Department of Telecommunication Engineering, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana

***Abstract*:** Software-defined networking is a novel concept that is ported into wireless sensor networks to make them more manageable and customizable. unfortunately, the topology discovery and maintenance processes generate high overhead control packet exchange between the sensor nodes and the central controller leading to a deterioration of the network's performance. In this paper, a novel minimal overhead control traffic topology discovery and data forwarding protocol is proposed and detailed. The proposed protocol requires some changes to the topology discovery protocol implemented in SDN-WISE to improve its performance. The proposed protocol has been implemented within the IT-SDN framework for evaluation. The results show reduced overhead control traffic and increase, of about 20%, data packet delivery rate over the protocol in SDN-WISE.

***Keywords*:** Software-Defined Wireless Sensor Networks, Topology Discovery Protocol, minimal control overhead, energy.

1. Introduction

Since the term “Internet of Things (IoT)” was first coined about a decade ago, the concept has been widely accepted and used in many areas [1]. The general idea revolves around interconnecting various heterogeneous devices in different geographical areas to communicate, store, or analyze data using the internet. Wireless Sensor Networks (WSNs) drive the growth of the IoT paradigm. WSNs are groups of specialized sensor nodes that are used to measure physical quantities such as sound, vibration, humidity, pressure, and temperature of an environment and convert them to electronic signals [2]. These signals are sent to a desired central location where they are used to make decisions. Each sensor node is equipped with limited resources [3] namely: one or more sensors, a processing unit, a memory, a power supply, and a Radio Frequency (RF) transceiver. The sensor nodes also communicate with one another using low-bandwidth wireless links.

There are numerous application areas of WSNs including military applications, weather forecasting, health monitoring, disaster detection, smart cities/vehicle design, pollution detection, and power management in schools and office buildings [4]. Due to the limitations of sensor node resources, WSNs are unable to satisfy the requirements of all application scenarios. For instance, some applications may require WSNs to store information for future retrieval. Others may require regular user queries to be scheduled and automatically dispatched without external operator intervention. Still, others may need to share information among themselves in real-time to aid in decision-making tasks. These limitations are considered to be significant drawbacks of the performance of WSNs and the growth of the IoT paradigm.

Considerable work has been carried out in the area of WSNs to improve the effective usage of sensor nodes' resources through the design of routing algorithms [5] and [6], the design of network architectures and topologies [7]. However, the task of designing innovative techniques, protocols, and applications to minimize the impact of WSN limitations creates the need for the usage of a Software-Defined Networking (SDN) solution to the WSN limitations. SDN is envisaged to offer better solutions to WSN limitations by decoupling the network control logic from the underlying hardware and by incorporating real-time network programmability and management [8]. Several latest recommendations in the SDN literature include: Anadiotis et al [9] presented SD-WISE, a software abstraction of sensor node's resources, to expand the SDN approach to WSN. In three cases, the key operations and characteristics of SDN were described and demonstrated to test the efficiency of the approach in WSN. The authors concluded that SDN could be considered an enabling technology for robust WSNs capable of performing complex tasks efficiently. Musa et al [10] reviewed some recent research on traditional WSN and discussed SDN-based management strategies for WSNs while at the same time emphasizing the benefits of SDN over traditional WSN.

The tight interplay between the data plane and the control plane in traditional WSN makes the sensor nodes consume high energy when performing tasks. Smitha et al [11] proposed a distributed energy-efficient Software-Defined Wireless Sensor Network (SDWSN) and implemented the controller at the base station. With knowledge of the network’s global view, the controller selects the cluster heads based on energy and distance. Similarly, green routing algorithms were proposed to maximize the lifetime of the sensor network in [12] and [13]. The communication between the sensor nodes and the controller in these works was via OpenFlow protocol. The Problem of OpenFlow [14] is that it requires a dedicated control channel. It also creates memory constraints and heavy overhead traffic; therefore, it is not suitable for SDWSN.

Generally, the SDN concept requires each node to gather details about the network topology, send these details to the central controller, and continuously update the controller about the topology status [15]. The nodes are also required to request further information from the controller for ambiguous flows. This results in continuous exchange of control messages between the central controller and the nodes. The continuous transmission of control messages impairs the quality of radio communication and drains the nodes’ energy

faster. This problem is more pronounced when the network topology is constantly changing due to node mobility or energy drainage. In any of these scenarios, additional control messages are shared to update the topology and the flow table, thus, leading to high traffic overheads. Also, when the hop count increases in the case of large networks, intermediate nodes generate flow requests to set up paths. The resultant effects include congestion, reduced network throughput, short network lifetime, and inefficient use of sensor node resources such as bandwidth, battery, CPU, and memory.

To reduce the updates of the flow table, Pineas et al [16] proposed a flow aggregation mechanism to improve energy consumption, network performance, and lifetime, but this could lead to expected errors including accessibility failures, routing loops, and network traffic isolation. The works in [17], [18], and [19] investigated the potential for distributing the control system to minimize congestion and address WSN 's perpetual limitations. The authors proposed autonomous controllers, distributed to each domain, to monitor and effectively secure the domains to prevent external and internal attacks. To ensure uniformity of network rules and accessibility of these rules by all controllers, so that a malfunction in one section of the network will not bring down the entire network, controllers continuously synchronize data resulting in high overhead control traffic and inefficient network operation.

This paper, therefore, focuses on reducing overhead traffic in topology discovery, topology maintenance, and data forwarding processes of SDWSNs to improve network operational efficiency and minimize energy consumption in the network performance by:

- Designing a distributed SDWSN in which each cluster head is a sub-controller responsible for collecting the network topology information, performing low-level control functions such as admitting or removing sensor nodes to or from the network, distributing flow table, filtering and forwarding traffic, and updating the central controller on the network topology.
- Designing a minimal overhead control traffic topology discovery, maintenance, and data forwarding protocol to minimize congestion, increase network throughput, and lifetime and improve the efficient use of sensor node resources.
- Assessing protocol performance with different key network performance metrics within the IT-SDN framework.

The rest of the paper is organized as follows: Section 2 reviews works related to the research problem. Section 3 discusses the limitations of the existing TD protocol, the modifications to the existing TD protocol, and the proposed protocol. Section 4 presents the proposed protocol and section 5 outlines the experimental method used to validate it. In section 6, the experimental results are presented and analyzed, and the conclusion and future work are provided in section 7.

2. Related Works

Some research works relating to the work in this article are presented in this section. Topology information collection is a critical component of any SDN solution for WSN due to the limitations of WSNs. This, therefore, calls for the development of an efficient Topology Discovery (TD) protocol in SDWSN without compromising the performance of the network's throughput, lifetime, and latency. To this end, Joseph et al [15] addressed how overhead control traffic can be minimized to improve the efficiency of SDWSN. In a full literature review of various methods or algorithms, the authors identified the drawbacks and strengths as well as open issues and future research directions. Babedi et al [20] proposed a Request For Comments (RFC) 7567 based SDWSN QoS resource-aware scheme to minimize overhead traffic in highly dynamic and large-scale SDWSNs. The proposed scheme improved data acquisition and network information collection time as well as network throughput, delay, and packet loss, compared to other implemented schemes.

The authors in [21] introduced programmable controller logic in the sensor nodes so that when the nodes receive a new packet, they can take decisions without often sending flow requests to the controller. Besides, a new topology management layer was introduced to collect the local topology information. The proposed solution was intended to minimize the exchange of control messages between the SDN controller and the sensor nodes and to permit the programming of such sensor nodes as finite-state machines. Building on this work, Nasim et al [22] proposed and implemented SDWSN "Fuzzy TD protocol " to increase the delivery rate of packets, reduce the loss of packets, and conserve the energy of the network to further improve the performance of SDWSN. The approach uses node cost calculated from energy, queue length, and the number of neighbors of each node to build the flow table and to distribute it to all the nodes. Though it ensures a fair distribution of energy consumption, it is computationally expensive, and it turns to generate high overhead control traffic. Theodorou et al [23] proposed a separate control channel in SDWSN to minimize the performance issues of high control messages associated with the in-band control channel of SDWSNs. However, this additional network interface and dedicated channel increase the hardware complexity and cost of the sensor nodes.

In [24], game theory and fork and join adaptive particle swarm optimization algorithm was proposed to improve network efficiency and service life. The authors stated that the algorithm automates the number and best positions of controllers, so that control messages can be exchanged with minimal overhead. While this approach ensures load balancing, it requires high computational power, and it does not improve overhead control traffic in dynamic networks. As data transmission and network communication consume a lot of resources, sensor node operations need to be altered to extend the network lifetime [25]. If this is not done properly in SDWSN, the efficiency of the controller decisions may be reduced as the controller is unable to have real-time network topology information. Faiza et al [26] investigated how the SDN approach was employed to decrease energy consumption in WSN. The authors provided the SDN architectural implementation in WSN to deal with energy consumption. However, this work was merely exploration since it did not provide implementation details and actual performance results. Sudip et al [27] proposed a situation-aware SDWSN protocol switching scheme using an information routing strategy to improve network efficiency. The proposed scheme adopts supervised learning algorithms that enable the controller to take decisions in real-time, based on network condition and application-specific requirements. However, changing routing protocols in each sensor node takes more time, thus, increasing delay and loss of packets.

To the best of our knowledge, the existing literature requires:

- Sensor nodes to collect and continuously report topology information to the controller via broadcast,
- Sensor nodes to request flow table for ambiguous flows by broadcasting the request to immediate neighbors,
- Sensor nodes to install flow table for the entire network,
- Independent controllers to man clusters in distributed SDWSNs

To fill the required gap, this paper proposes a minimized overhead traffic topology discovery, maintenance, and data forwarding protocol for SDWSN.

3. Discussion of Literature Review Findings

The literature review revealed the need for an efficient topology discovery protocol in SDWSN due to limitations of sensor node resources. we discuss the limitations of existing TD protocol and our modifications to it in this section.

3.1 Topology Discovery Process

The network status collections start with each node broadcasting Topology Discovery (TD) packet to its neighbors. A node, upon receipt of a TD packet, inserts into its neighbor table the list of its current neighbors, the current RSSI, address, and the battery level. The node then updates its route towards the controller to enable it to send the neighbor report and receive flow table. The node also sets its battery level and address in the corresponding field of the TD packet and broadcasts the updated TD packet to its neighbors [21]. This process continues until the controller receives the network status. This process poses a serious challenge to efficient network operation as the sensor nodes expend limited resources processing, broadcasting, and relaying control messages to establish the network global view.

The topology discovery process was modified by putting the network into clusters, depicted in Figure 1, using the concept in [17], allowing only the cluster head to broadcast TD packets, and restricting the sensor nodes to communicate with the controller and any other node through the cluster head. However, unlike [17], the cluster heads do not independently control the respective clusters and cannot modify the flow table. The cluster heads, here refer to as sub-controllers, (1) only rebroadcast TD packets on behalf of the central controller, (3) collate neighbor information and relay it to the central controller, and (4) distribute the neighbor table to all sensor nodes in the respective domains. The cluster heads can also admit or delete a node and update the central controller of the changes. The introduction of cluster heads, as sub-controllers, eliminates performance degradation due to multiple controllers [19], improves network operational efficiency, and minimizes latency and congestion.

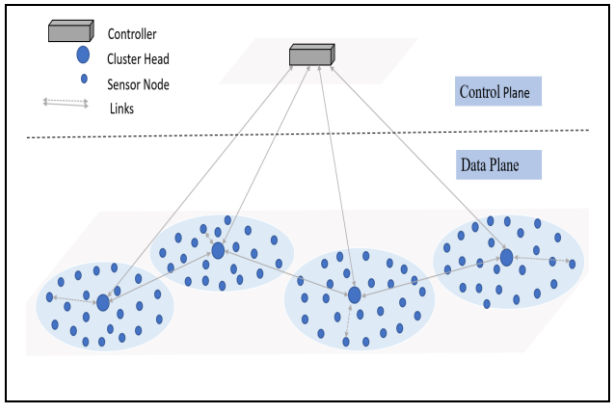

Figure 1: Network Architecture of Proposed Solution

3.2 Topology Maintenance Process

The existing TD protocol uses periodic beacon messages to keep the central controller updated on current network topology when the network is in operation. If one sensor node receives a beacon packet from another node that is not already in the neighbor table, it adds the other node's address, battery level, and RSSI value into the neighbor table and updates the controller in the neighbor/network status report which is sent periodically. However, when the topology fluctuates due to sensor node failure or mobility, the SDN controller cannot correctly compute the routing rules and install them. As a result, the sensor nodes keep sending flow request messages and dropping packets in queues to be relayed. Besides, when hop count increases, in the case of large networks, the intermediate nodes generate flow requests to set up paths. The resultant effects are congestion and inefficient use of sensor node resources. To compensate node failure and mobility, Million et al [28] implemented a keepalive counter to track and remove dead nodes. However, high dynamism in topology maintenance puts more constraints on the sensor nodes since the resources are limited.

The topology maintenance process was modified to allow only the cluster heads to periodically broadcast the beacon messages to the respective domains. A node, upon receipt of a beacon message, unicasts its response to the cluster head with an update of its current status. If a node receives two beacon messages from two different cluster heads, it responds to the one with stronger RSSI. If a node moves out of coverage range of a cluster head, it listens for broadcast beacons continuously, and upon receipt of a beacon message, it requests to join that cluster. In this way, only the cluster heads collate the changes in the network and update the controller about them and also distribute the updated flow table to all the sensor nodes in the respective domains. Advantage was taken of the small size of a cluster, compared to the entire network, to limit the impact of the keepalive counter, implemented with the broadcast beacons, so that the cluster heads use it to remove dead nodes. It was assumed that since the controller has information about every node in the network, it can automatically select a cluster head for any cluster if an existing cluster head runs out of energy/dies or cannot meet the minimum threshold, set by the central controller, to be a cluster head. The central controller then sets a new threshold and uses it to select a normal node to act while a notification is sent to the administrator.

3.3 Data Forwarding Process

Nodes farther away from the destination of packets forward them to the closest neighbors, and nodes receiving packets directed towards the central controller relay them to the closest nodes (in terms of the number of hops) to the central controller. This approach was modified by restricting sensor nodes to only sense and forward traffic to the closest sub-controller which has the entire network's flow table. The cost of sending traffic to the sub-controller (cluster head) was made lowest so that nodes still forward traffic to the cluster head even though they may have closest physical neighbors.

4. Proposed Protocol Algorithm

The burden of topology information collection was shifted to the cluster heads (sub-controllers) with enhanced resources to eliminate the exchange of control messages among the sensor nodes and between the sensor nodes and the central controller

and reduce the hop count of data packets. Algorithm 1 illustrates the logic of the proposed protocol and the flowcharts for topology discovery and data forwarding are shown in figure 2 and 3 respectively.

Algorithm 1: Procedure for CHs in topology discovery, maintenance, and data forwarding

Require: Let C = controller, n = set of sensor nodes in a cluster (neighbors to CH) and t = timer for topology updates and r = requests send to CH by n.

```
Upon receiving an event, E do
    if E==Flow Table then
        Broadcast to n
    end
    if E==data then
        Forward to destination
    end
    if E==t or r
        while t<Tmax or n<Nmax do
            Broadcast neighbor discovery message or receive a cluster head discovery message
            if a new neighbor is found from cluster head discovery message then
                Add to neighbor table
            end
            if neighbor is found and neighbor ID exists and status is the same as status data then
                Ignore cluster head discovery message for that neighbor
            end
            if the neighbor is found and status is not the same as status data then
                Update status data
            end
        send the neighbor table to C
    end
end
wait for the event, E
end procedure
```

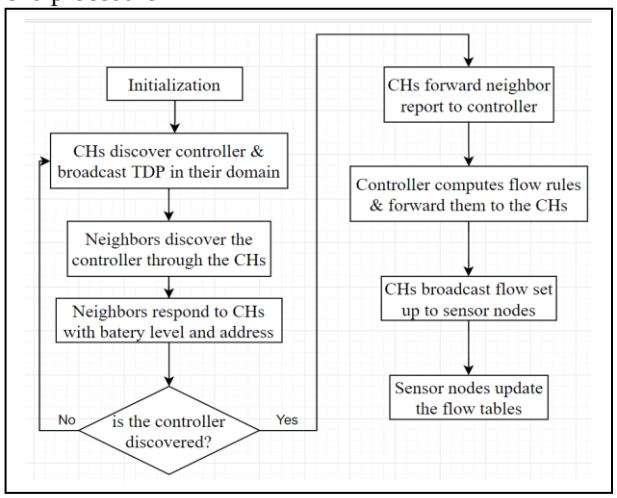

Figure 2: Controller Discovery Process

CH broadcasts neighbor discovery messages to all sensor nodes within its radius and sensor nodes send cluster head discovery messages by unicast to a CH with stronger RSSI to join that cluster.

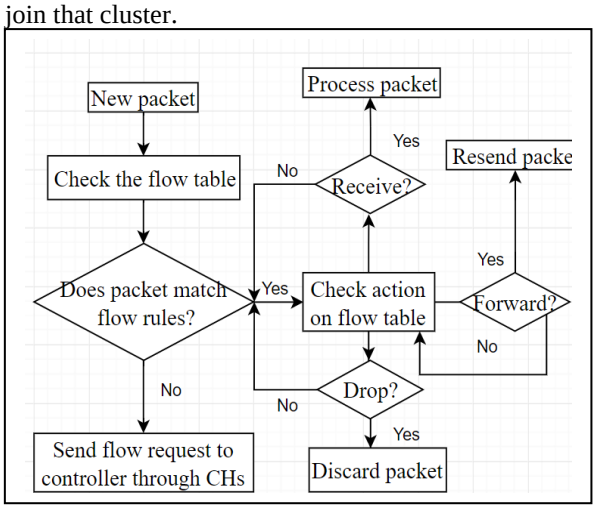

Figure 3: Data Forwarding Process

5. Experimental Setup and Evaluation Procedure

The proposed algorithm was validated using the IT-SDN framework [29]. The framework runs on COOJA, a network simulator of emulated Contiki WSN nodes [30] using the ContikiOS. Detailed description of COOJA is provided in [31]. Two experiments were performed comparatively: a distributed SDWSN with cluster heads as sub-controllers to evaluate the proposed protocol and implementation of the concept of topology discovery in SDN-WISE [21]. Figure 4 depicts the distributed SDWSN, referred herein as the proposed solution. This experiment consists of one central controller and two sub-controllers. The second experiment, shown in Figure 5, depicts the implementation of SDN-WISE, referred herein as the traditional solution. We evaluated the number of control messages and the impact of these messages on key network metrics: traffic delay, traffic delivery rate, network initialization time, and energy consumption by comparing the two experiments. Both experiments were implemented on the same platform using the same framework. Table 1 summarizes the configuration parameters for the framework and the experiments. 10 sensor nodes and 1 central controller including 2 sub-controllers for the proposed solution were considered for both experiments. The choice was reasonable enough to analyze and compare both protocols. All nodes were manually positioned at a fairly equal distance from each other in both experiments and maintained constant throughout the experiments. Following a complete network initialization, the nodes began to send data packets at constant intervals, apart from sub-controllers and the central controller. A random initial delay was applied to avoid artificial data transmission synchronization. The packet loss and overhead control messages were assumed to depend on the intensity of the generated packets, so the packets were generated at intervals of 30 seconds. Each simulation was run for 10 minutes and repeated 12 times with random seeds, increasing the time of each run by 10 minutes. Since the performance evaluation was primarily time-related, each event that occurred in the sensor nodes, and the controller was time-stamped to make the measurement possible. After the simulation was run, the sender node log, the receiver node log,

and the controller node log were saved to a file. The logged file was later processed to extract values for the performance metrics. The simulation setup and flow table distribution are shown in Figure 6 and 7 respectively .

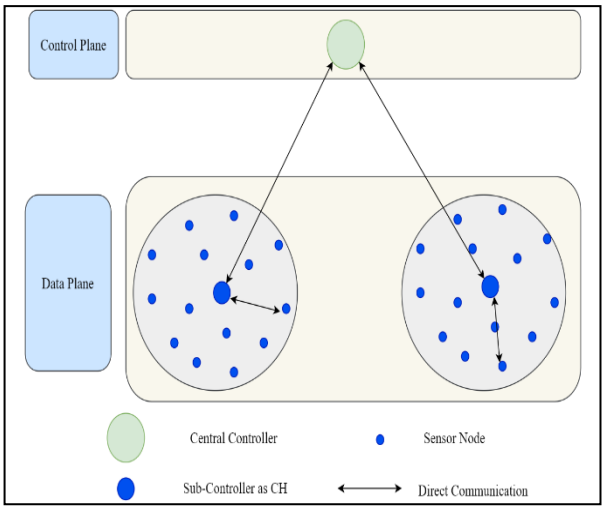

Figure 4: Experiment 1, Proposed Solution

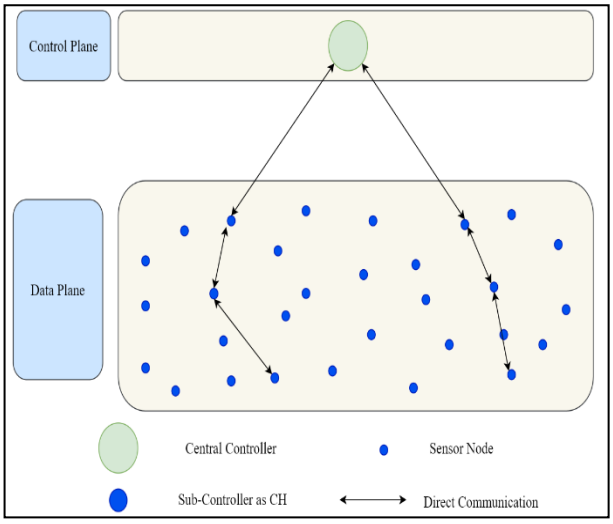

Figure 5: Experiment 2, Traditional Solution

Table 1: Simulation Parameters

Simulation parameters	
Topology used	Manually positioned
Number of sensor nodes	10
Number of sub-controllers	2
Number of controllers	1
Payload max size	116 bytes
Buffer size	10 bytes
Simulation time	120 minutes
ContikiMAC channel check rate	128Hz
MAC layer	CSMA
Radio Medium Model	UDGM
Node type	Sky mote
Transmission Range	50
Interference Range	100
IT-SDN Tool parameters	
Version number	0.4
Packet retransmission time	60s
Link metric	ETX
Flow table size	15 entries
Route calculation algorithm	Dijkstra

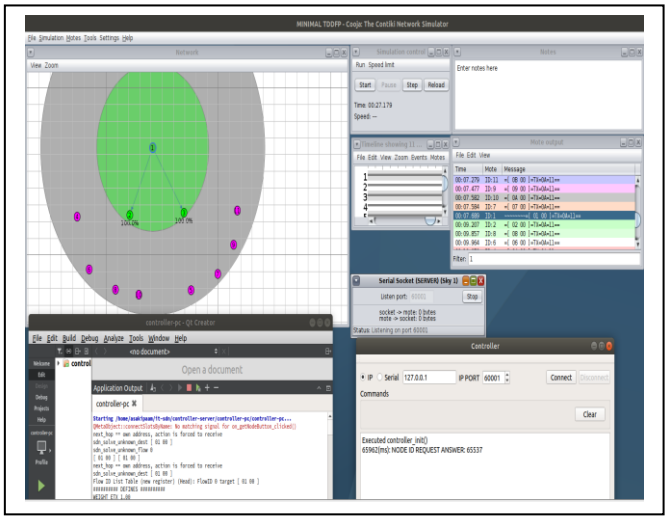

Figure 6: Simulation Setup

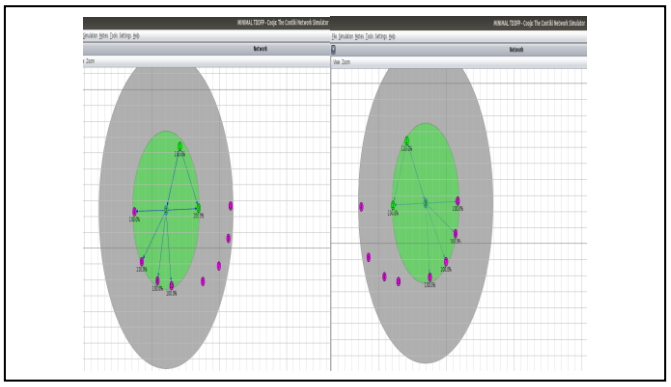

Figure 7: Controller (1) and Sub-controllers (2 & 3) distributing flow table

6. Results Analysis and Discussion

6.1 Network Initialization Time

The controller discovers the entire topology before making any routing decisions. The initialization time is the time it takes the central controller to learn of the entire network topology and install flow tables in all the nodes in the network. In the network initialization time measurement, the central controller and all the sensor nodes were started at the same time, and the instant at which a node received a data flow setup was recorded. The time at which the last node received its data flow setup marked the full network initialization time. At this time, all the nodes in the network could forward traffic.
The network initialization time for both experiments is shown in Figure 8. The graph indicates that the proposed solution has converged the network faster than the traditional solution. In the proposed solution, each sensor node only needs to discover a cluster head and send a flow request to and obtain the flow table from the central controller through that cluster head. This reduced the computation requirement and the hop count in building the global network view. The shorter convergence time could also be attributed to the introduction of sub-controllers as cluster heads with enhanced capabilities to process neighbor related information on behalf of the sensor nodes. The cluster heads keep track of all sensor nodes joining the network and update the central controller, and the controller, in turn, computes/updates the flow table promptly. However, in the traditional solution, each sensor node needs to learn of all its neighbors before discovering the central controller and receiving the flow table. This slowed down the

network initialization time. The network converged 30s faster in the proposed solution than in the traditional solution.

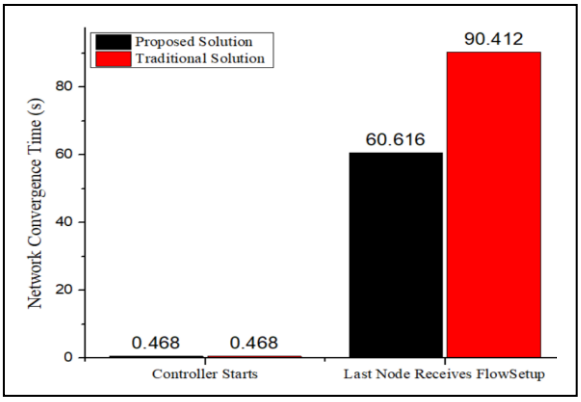

Figure 8: Network Initialization Time

6.2 Control Message Overhead

Control message overhead is defined as the number of specific control messages that are exchanged between the sensor nodes and the central controller. Control messages include the periodic topology reports by each sensor node and request and response messages for flow entries. Approximately, 200 topology report messages alone are sent every 10 minutes to the SDN controller [32].

The numbers of control messages over the simulation time is shown in Figure 9. The graph shows that the proposed solution generated fewer control messages than the traditional solution. This can be attributed to congestion or retransmission in the traditional solution. For instance, when the topology fluctuates due to the failure of a sensor node, some sensor nodes send repeated flow request messages, in addition to the usual topology report messages, to create new routes. Besides, the intermediate nodes often generate flow requests to help construct routes. This increases the overhead control message in the network. However, in the proposed solution, only the cluster heads generated flow request messages to setup new paths as intermediate nodes, and if a cluster head failed as a result of running out of a certain resource, the controller could pre-emptively select a new cluster head since the controller knows the status of each node in the network. These could be the reasons for the minimal control overhead in the proposed solution.

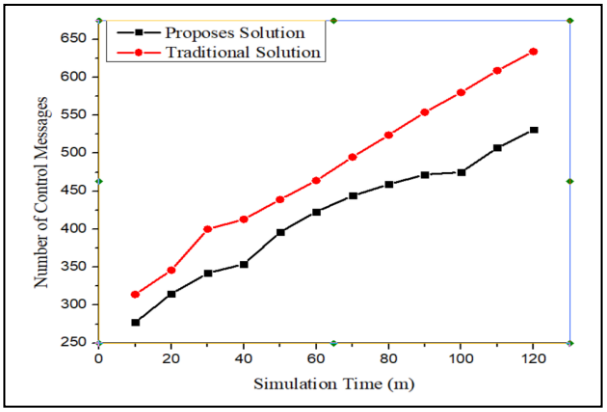

Figure 9: Average Control Overhead

6.3 Average Traffic Delay

The end-to-end Packet delay can be defined as the time difference between when a packet is sent and when it is received. Packet delay can be computed by summing transmission delay, propagation delay, the processing time of the controller, and queue delay. However, it is highly difficult to compute these delays in WSNs due to synchronization problems. The commonest approach is to use the individual time of each node and sum them [28]. The delay, in this case, considered all the above for each node and was measured by calculating the time difference between when a packet was generated and when the packet was received.

Heavy loading of requests in SDWSN may result in the central controller taking a longer time to process and respond to flow requests. This may also result in network congestion due to packets taking a longer time to reach the destination. As the proposed solution introduced sub-controllers to service low-level requests and relay packets, this could account for the lower average delay of packets than the traditional solution, as shown in Figure 10.

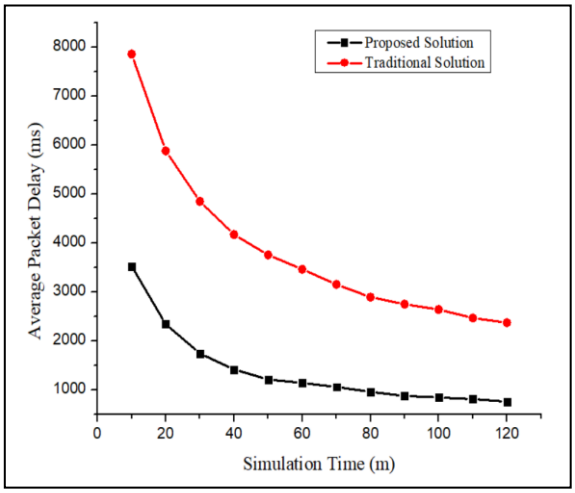

Figure 10: Average Packet Delay

6.4 Average Data Delivery Rate

The data delivery rate measures the ratio of receive data packets to send data packets at the destination. Data delivery rate is a very important network metric since it indicates network congestion, interference, or a yet to be established path in the network. Each node keeps a counter that is incremented every time the node sends or receives a packet. This counter is sent to the controller in the report message. The data delivery rate was measured by checking which and how many of the packets sent were received at the destination. The data packets were generated at a 30-second interval after the network had been converged. Figure 11 shows the distribution of the data packets. The graph shows that the proposed solution has a higher data packet delivery rate than the traditional solution. In the traditional solution, some packets were lost due to network congestion. Besides, the nodes used broadcast beacons with higher priority to establish and maintain the network global view. During the broadcasting period, data packets were buffered and lost when the underlying network could no longer accommodate incoming packets. With the proposed solution, only the cluster heads used broadcast beacons to establish and maintain the network global view, hence, packet loss would greatly relate to the capacity of the cluster head to collect topology information, update the central controller, and relay packets. Since sub-controllers have been used as cluster heads with enhanced capabilities, it is not unexpected that the proposed solution has shown a delivery rate of 20% higher than the traditional solution.

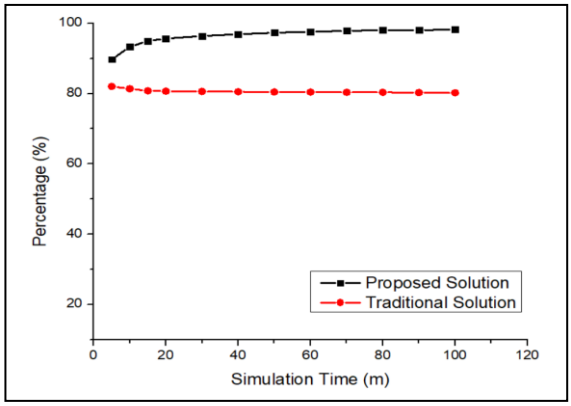

Figure 11: Average Packet Delivery Rate

6.5 Energy Measurements

Many factors affect node power consumption in WSNs. These factors include the network topology, the transmission ratio of each node, the propagation distance of the packets, and the type of transmissions such as broadcast or unicast transmission as well as synchronized or non-synchronized transmission [33]. Taking all these factors into consideration, the topology and the transmission ratio of each node were maintained constant for both experiments, and the energy of the Central Processing Unit (CPU), the energy of Low Power Mode (LPM), the energy of Radio Transmit, and energy of Radio Listen or receive were measured. The energy of CPU is defined as the total energy used for active computation, the energy of LPM refers to the total energy used when the sensor node is in the power saving condition or the idle state, and the energy of Radio Transmit (TX) and energy of Radio Listen or receive (RX) refer to the total energy used by the radio devices to send and receive data packets respectfully. Energest was used to track the energy consumption of each node, and the data were logged into a file.

Figures 12-15 show the average distribution of power consumption for the different parameters. In Figure 12, the average CPU power consumed is lower in the proposed solution than in the traditional solution. This is because the sensor nodes performed less active computation when establishing and maintaining the global network view since this burden was shifted to the cluster heads and the hop count is significantly reduced to 1. However, as explained in Figure 8, this is not the case for the traditional solution. Figure 13 also shows that the sensor nodes consume more power in the power saving or idle condition, a confirmation to the explanation for Figure12.

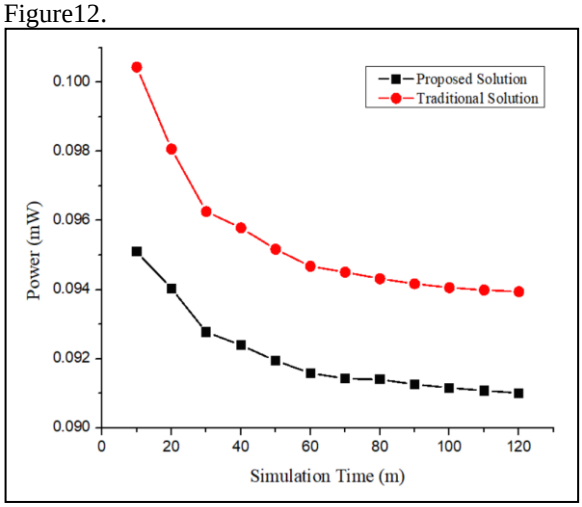

Figure 12: Average CPU Energy Consumption

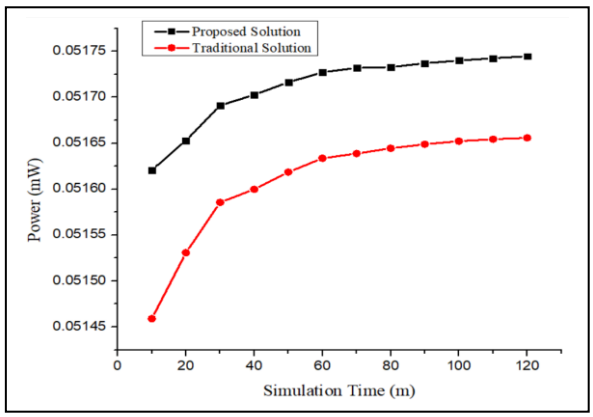

Figure 13: Average LPM Energy Consumption

Figure 14 shows that the proposed solution produced a slightly lower transmit power consumption effect on the nodes than the traditional solution. Since packets were retransmitted by fewer nodes in the proposed solution, it can be deduced that the average power consumed to transmit a packet across the network will be lower than in the traditional solution.

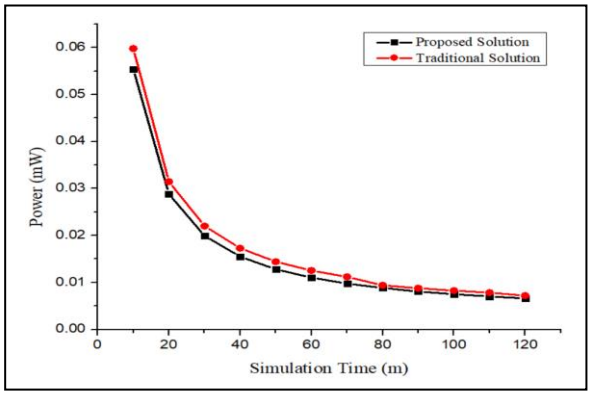

Figure 14: Average TX Energy Consumption

Figure 15 shows the average power consumed by the nodes in listening to the radio channel or receiving incoming packets. The data shows that the traditional solution outperforms the proposed solution when the network was in operation for more than 10 minutes. It can be deduced that because the average distance at which a node received a packet from a sender in the proposed solution was always longer than that of the traditional solution and since more distance means more interference, the nodes increased the received power to be able to detect incoming packets. This accounted for an increase in receive power after the 10th minute when the network was completely initialized, and all the nodes began to forward packets, thus increasing the interference in the network.

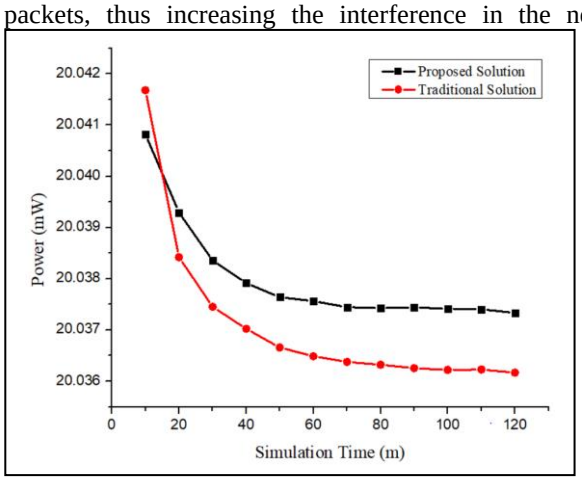

Figure 15: Average RX Energy Consumption

7. Conclusion and Future Work

The SDN concept is still at its novel stage regarding its utilization in WSN. Though the concept is envisioned to solve the various limitations of WSNs, its implementation in WSNs comes with many challenges. One of the most important challenges is how to reduce the overhead control traffic in topology discovery and maintenance. There are several proposals in literature on how to resolve this challenge. One of these proposals is SDN-WISE which seeks to minimize the exchange of control messages in the network and improve the network performance. However, its approach has serious limitations. These limitations were modified in this paper to propose an improved topology discovery protocol. The proposed protocol initializes the network faster, reduces the overhead control traffic significantly, requires less consumption of sensor node energy, and improves the SDWSN performances. The proof of concept experiments revealed that the proposed solution increased the data packet delivery rate by 20% and converges the network 49% faster than the traditional solution. Similar research on performance analysis of topology control techniques for SDWSNs using CORAL-SDN was carried out in [34]. The authors obtained an average convergence time of around 120 seconds for a linear topology of 25 sensor nodes whereas the result obtained for convergence time in this paper is about 60 seconds. This shows that the performance of the proposed solution is optimal. A correlation observed in the experiment is the effect of propagation distance on the consumption of energy to receive incoming packets. It was observed that as the propagation distance increases, the consumption of received energy increases. This has been explained as the sensor nodes having to increase sensitivity to be able to detect incoming packets as a result of increased radio interference. There is a possibility of tweaking the proposed algorithm to make routing decisions to take into account the battery levels of each sensor node. In the future, the authors intend to implement and evaluate the proposed protocol on a large-scale scenario to verify its scalability.

8. Acknowledgement

The authors would like to thank MTN Ghana for funding this research.

References

[1] J. E. Ibarra-Esquer, F. F. González-Navarro, B. L. Flores-Rios, L. Burtseva, and M. A. Astorga-Vargas, "Tracking the evolution of the internet of things concept across different application domains," *Sensors (Switzerland)*, vol. 17, no. 6, pp. 1–24, 2017.

[2] M. A. Zibouda Aliouat, "Efficient Management of Energy Budget for PEGASIS Routing Protocol," *HAL Arch.*, p. 215, 2012.

[3] M. L. F. Miguel, E. Jamhour, M. E. Pellenz, and M. C. Penna, "SDN architecture for 6LoWPAN wireless sensor networks," *Sensors (Switzerland)*, vol. 18, no. 11, pp. 1–23, 2018.

[4] K. M. Modieginyane, B. B. Letswamotse, R. Malekian, and A. M. Abu-Mahfouz, "Software defined wireless sensor networks application opportunities for efficient network management: A survey," *Comput. Electr. Eng.*, vol. 66, pp. 274–287, 2018.

[5] A. Jedidi, "Workload cluster balance algorithm to improve Wireless sensor Network performance," *Int. J. Commun. Networks Inf. Secur.*, vol. 11, no. 1, pp. 105–111, 2019.

[6] M. Razzaq, D. Devi Ningombam, and S. Shin, "Energy efficient K-means clustering-based routing protocol for WSN using optimal packet size," *Int. Conf. Inf. Netw.*, vol. 2018-Janua, no. 1, pp. 632–635, 2018.

[7] M. J. McGrath, C. N. Scanaill, M. J. McGrath, and C. N. Scanaill, "Sensor Network Topologies and Design Considerations," in *Sensor Technologies*, 2013, pp. 79–95.

[8] T. Bakhshi, "State of the art and recent research advances in software defined networking," *Wirel. Commun. Mob. Comput.*, vol. 2017, p. 36, 2017.

[9] A. Anadiotis, L. Galluccio, S. Milardoc, G. Morabito, and S. Palazzo, "SD-WISE : A Software-Defined Wireless SEnsor network "," *Elsevier*, vol. 159, pp. 84–95, 2019.

[10] M. Ndiaye, G. P. Hancke, and A. M. Abu-Mahfouz, "Software defined networking for improved wireless sensor network management: A survey," *Sensors (Switzerland)*, vol. 17, no. 5, pp. 1–32, 2017.

[11] B. Smitha and D. Annapurna, "Software defined network for conservation of energy in wireless sensor network," *2017 Int. Conf. Energy, Commun. Data Anal. Soft Comput. ICECDS 2017*, pp. 591–596, 2018.

[12] D. P. V. Neetesh Kumar, "A Green Routing Algorithm for IoT-Enabled Software Defined Wireless Sensor Network," *IEEE Sens. J.*, vol. 18, no. 22, p. 12, 2018.

[13] M. S. Azizi and M. L. Hasnaoui, "Software defined networking for energy efficient wireless sensor network," *Proc. - 2019 Int. Conf. Adv. Commun. Technol. Networking, CommNet 2019*, p. 7, 2019.

[14] D. Hasan and M. Othman, "Efficient Topology Discovery in Software Defined Networks: Revisited," *Procedia Computer Science*, vol. 116. pp. 539–547, 2017.

[15] J. Kipongo, T. O. Olwal, and A. M. Abu-Mahfouz, "Topology Discovery Protocol for Software Defined Wireless Sensor Network: Solutions and Open Issues," *IEEE Int. Symp. Ind. Electron.*, vol. 2018-June, pp. 1282–1287, 2018.

[16] G. P. H. Egidius, Pineas M., Adnan M. Abu-Mahfouz, Musa Ndiaye, "Data Aggregation in Software-Defined Wireless Sensor Networks: A Review," *IEEE Access*, vol. 9, p. 6, 2019.

[17] B. T. De Oliveira and Ć. B. Margi, "Distributed control plane architecture for software-defined Wireless Sensor Networks," *Proc. Int. Symp. Consum. Electron. ISCE*, pp. 85–86, 2016.

[18] O. Flauzac, C. Gonzalez, and F. Nolot, "Developing a Distributed Software Defined Networking Testbed for IoT," *Procedia Comput. Sci.*, vol. 83, pp. 680–684, 2016.

[19] H. I. Kobo, G. P. Hancke, A. M. Abu-Mahfouz, and G. P. Hancke, "Towards a distributed control system for software defined Wireless Sensor Networks," *Proc. IECON 2017 - 43rd Annu. Conf. IEEE Ind. Electron. Soc.*, vol. 2017-Janua, pp. 6125–6130, 2017.

[20] B. B. Letswamotse, R. Malekian, C. Y. Chen, and K. M. Modieginyane, "Software defined wireless sensor networks and efficient congestion control," *IET Networks*, vol. 7, no. 6, pp. 460–464, 2018.

[21] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," *Proc. - IEEE INFOCOM*, vol. 26, pp. 513–521, 2015.

[22] S. M. Nasim Abdolmaleki, Mahmood Ahmadi, Hadi Tabatabaee Malazi, "Fuzzy topology discovery protocol for SDN-based wireless sensor networks," *Elsevier*, vol. 79, pp. 54–68, 2017.

[23] T. Theodorou and L. Mamatas, "A Versatile Out-of-Band Software-Defined Networking Solution for the Internet of Things," *IEEE Access*, vol. 8, p. 24, 2020.

[24] L. I. Peizhe, W. U. Muqing, L. Wenxing, and Z. Min, "A Game-Theoretic and Energy-Efficient Algorithm in an Improved Software-Defined Wireless Sensor Network," *IEEE Access*, vol. 5, p. 16, 2017.

[25] J. Long and O. Büyüköztürk, "Collaborative duty cycling strategies in energy harvesting sensor networks," *Comput. Civ. Infrastruct. Eng.*, vol. 35, no. 6, pp. 534–548, 2020.

[26] N. F. Ali, A. M. Said, K. Nisar, and I. A. Aziz, "A Survey on Software Defined Network Approaches for Achieving Energy Efficiency in Wireless Sensor Network," *2017 IEEE Conf. Wirel. Sensors*, vol. 2018-Janua, pp. 28–33, 2017.

[27] S. Misra, S. Bera, M. P. Achuthananda, S. K. Pal, and M. S. Obaidat, "Situation-aware protocol switching in software-defined wireless sensor network systems," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2353–2360, 2018.

[28] M. A. Beyene, "Evaluation of SDN in Small Wireless-capable and Resource-constrained Devices," Norwegian University of Science and Technology, 2017.

[29] R. C. A. Alves, D. A. G. Oliveira, N. S. Gustavo, and C. B. Margi, "IT-SDN: Improved architecture for SDWSN," *XXXV Brazilian Symp. Comput. Networks Distrib. Syst.*, 2017.

[30] A. Dunkels, "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors," *IEEE Access*, p. 8, 2010.

[31] M. Q. Thomson, Craig, Ahmed Yassin Al-Dubai, Imed Romdhani, "Cooja Simulator Manual," *ResearchGate*, no. July, p. 26, 2016.

[32] T. C. Luz *et al.*, "In - network performance measurements for Software," *Proc. 2019 IEEE 16th Int. Conf. Networking, Sens. Control*, pp. 206–211, 2019.

[33] M. Tutunovic and P. Wuttidittachotti, "Discovery of Suitable Node Number for Wireless Sensor Networks Based on Energy Consumption using Cooja," *Int. Conf. Adv. Commun. Technol. ICACT*, p. 5, 2019.

[34] T. Theodorou and L. Mamatas, "Software defined topology control strategies for the internet of things," *2017 IEEE Conf. Netw. Funct. Virtualization Softw. Defin. Networks, NFV-SDN 2017*, vol. 2017-Janua, no. November, pp. 236–241, 2017.