Soft-WSN: Software-Defined WSN Management System for IoT Applications

Samaresh Bera, *Student Member, IEEE,* Sudip Misra, *Senior Member, IEEE,*
Sanku Kumar Roy, *Student Member, IEEE,* and Mohammad S. Obaidat, *Fellow, IEEE, Fellow, SCS*

***Abstract*—In this paper, we propose a software-defined wireless sensor network architecture (*Soft-WSN*) — an effort to support application-aware service provisioning in internet of things (IoT). Detailed architecture of the proposed system is presented involving the application, control and infrastructure layers to enable software-defined networking in IoT. We design a software-defined controller, which includes two management policies — device management and network management. Device management facilitates users to control their devices in the network. To enable device control mechanisms, we investigate three scheduling issues in a sensor node — sensing task, sensing delay, and active-sleep. On the other hand, topology of the network is controlled by the network management policies, which can be modified in run-time to deal with dynamic requirements of IoT. Furthermore, the proposed scheme is implemented in a real hardware platform without changing the underlying sensor networking concepts, so that existing sensor devices can be seamlessly integrated. Therefore, in contrast to the existing SDN solutions for wireless sensor networks (WSNs), the proposed system, *Soft-WSN*, focuses on both device management and topology management to meet run-time application-specific requirements of IoT, while enhancing flexibility and simplicity of WSN management. Experimental results on a real hardware based test-bed indicate that the proposed scheme is beneficial to meet real-time application-specific requirements of IoT, while ensuring significant improvements on network performance, over the traditional approaches.**

***Index Terms*—Wireless Sensor Networks, Software-Defined Networking, Internet of Things, Device Management, Topology Management, Network Performance**

I. Introduction

A WSN comprises of several sensor nodes, which can be used in different application scenarios such as agriculture, militant, health-care, and energy. A WSN may consist of several heterogeneous sensor nodes to perform dedicated tasks deployed in an area-of-interest. There exists many proprietary and non-proprietary existing solutions to fetch sensed data from the sensor nodes [1]. However, the current trend of using IP-based sensor networking solutions (such as 6LoWPAN and IPv6) enable the WSN to be connected to the Internet. Thus, WSNs can be used for monitoring/controlling different applications through the Internet connectivity, which, in turn, establishes the concept of *internet of things* (IoT). The concept of internet of things (IoT) enables physical objects, which are embedded with sensors, actuators, and network connectivity, to collect and exchange data among themselves in a collaborative manner [2], [3]. Consequently, it is evident that WSNs are expected to play a key role for monitoring different objects in order to build smart things (such as smart home, intelligent transportation systems, and smart health-care). Concurrently, software-defined networking (SDN) leverages a centralized network control paradigm, while decoupling control logic from physical devices in a real network platform [4]. Thus, traditional forwarding devices are logically separated into two planes — *control plane* and *data plane*. Different network control operations are performed through the control plane, whereas, the data plane refers to the physical devices in the network. Therefore, the physical devices are capable of adapting adequate forwarding rules defined by centralized controller in order to improve network performance.

A. Motivation

The business requirements of modern WSN implementation require them to change the in-built policy in order to meet these requirements [5]. The changes are dynamic in nature, which need to be carried out in real-time. Additionally, due to the presence of heterogeneous sensor nodes in a WSN, it is necessary to control adequately the data of such nature, in order to take optimal decisions. Therefore, it is required to program the sensor nodes to deploy desired policy and to control the data forwarding logic in sensor nodes in an adequate manner. SDN-enabled WSN implementations can help in addressing this problem.

Most of the existing solution approaches related to programmable WSN concentrated on FPGA-based programmable architecture, which suffers from functional complexity and resource-constrained nature of sensor nodes. As evident from the work of Sood et al. [4], the functionality of the physical devices/networks can be changed in real-time using SDN, to support the application-specific requirements of IoT, while changing the control logic of the devices/network. Thus, SDN-based technologies are expected to play a major role in establishing an efficient IoT environment. However, existing SDN-based solution approaches focus on low-level configuration and completely based on operating system. Recently, researchers proposed different SDN-based approaches for efficient data-center and high-speed backbone networking. In such approaches, OpenFlow [6] is considered as the main communication protocol between the controller and the physical

S. Bera, S. Misra and S. K. Roy are with the Computer Science and Engineering Department, Indian Institute of Technology, Kharagpur, 721302, India, Email: s.bera.1989@ieee.org, smisra@sit.iitkgp.ernet.in, sankukumar-roy@gmail.com

M. S. Obaidat is the Chair and Professor of the Dept. of Computer and Information Science, Fordham University, New York, 10458, USA, Email: mobaidat@fordham.edu

devices. However, due to the large number of rule-matching fields and energy expensive nature of OpenFlow, it may not be adequate to use it in low-power networks (such as sensor networks) in IoT. To address such issue, Luo et al. [5] proposed Sensor-OpenFlow, which defines flow-table implementation rules in sensor networks. However, issues related to real-time device and topology management in sensor networks, which are crucial factors to fulfill the requirements of IoT, are yet to be adequately addressed.

B. Contributions

To address the above mentioned issues, we propose a software-defined wireless sensor network architecture, which enables the sensor devices/networks to be configured in real-time for application-aware service provisioning in IoT, named as *Soft-WSN*. We design a controller having two management entities — *device manager* and *topology manager*. The *device manager* is responsible for controlling device-specific tasks. On the other hand, *topology manager* maintains the topology of the network so that the performance of the network is optimized. Therefore, the *device manager* and *network manager* define adequate control logic based on the application-specific requirements, and it is sent to the devices in the network. Accordingly, the devices in the network adapt the control logic defined by the controller in real-time, and change their activities. We use the IEEE 802.15.4 [7] and IEEE 802.11 [8] protocols in the proposed architecture, so that the existing sensor networking devices can be used without changing the hardware platform, while modifying the control logic in real-time. Therefore, in contrast to the existing SDN solutions for WSNs, *Soft-WSN* focuses on both device management and topology management to meet application-specific requirements of IoT in real-time, while enhancing flexibility and simplicity of WSN management. Experimental results on a real hardware platform show that the proposed approach outperforms the traditional WSN technologies in terms of energy consumption, message overhead, and packet delivery ratio. In brief, the contributions in this paper are as follows:

- We propose a software-defined wireless sensor network architecture to support application-specific requirements of IoT, named as *Soft-WSN*.
- In *Soft-WSN*, we design a software-defined controller with two management facilities — device and topology management. The former is responsible for controlling devices in the network, whereas the latter is responsible for topology management in the network.
- To change the functions of devices and network-topology in real-time, we present different rule management policies, while describing packet formats which are required to be exchanged.
- We evaluate the performance in hardware platform to show the effectiveness of the proposed scheme, *Soft-WSN*. From the results, it is evident that the proposed scheme is capable of enhancing the network performance in terms of energy consumption, message overhead, and packet delivery ratio.

The rest of the paper is organized as follows. Section II discusses the current state-of-the-art of programmable and software-defined networking aspects in WSN, while presenting their limitations. Section III presents a detailed architecture of the proposed system and its components. Section IV describes device-specific control mechanisms, whereas in Section V, network topology management is presented. Section VI presents the experimental setup and results to show the effectiveness of the proposed scheme. Finally, we conclude the paper in Section VII, while mentioning some future research directions.

II. Related Work

In this Section, we discuss the existing works from the programmable and software-defined wireless sensor networking aspects. Several works are proposed in the literature to enable real-time programmable features in wireless sensor networks. We discuss the existing works in two folds — programmable and software-defined WSNs.

A. Programmable WSNs

In recent years, researchers proposed different schemes for WSN, so that the latter can be programmed in real-time based on requirements [9]–[12]. Krasteva et al. [9] proposed a run-time re-configurable sensor node architecture, while incorporating field programmable gate array (FPGA) into the traditional sensor devices. In the proposed scheme, sensor nodes change their activities in run-time based on the control mechanisms decided by 'reconfiguration control' unit. The authors showed that deployment cost can be minimized significantly with the proposed scheme compared to pre-programmed sensor networks. Similarly, Hsieh et al. [10] proposed FPGA-based re-configurable sensor nodes to minimize the energy consumption in the network. The authors focused on hardware accelerated information compression to minimize energy consumption, while aggregating the sensed information at the cluster-head nodes. Vera et al. [11] proposed a programmable interface in WSN which supports the plug-and-play concept use in sensor nodes. In such a system, a sensor node can reconfigure itself according to the properties of the network.

Angove et al. [12] designed a platform for remote interaction with on-field sensor nodes, while deploying a programmable gateway device in the network. The gateway node is reconfigurable and mobile in nature, and thus, it interacts with the on-field sensor nodes. However, the proposed scheme only facilitates the gateway device to be reconfigured. Therefore, activities of the sensor node and network cannot be configured in real-time.

In sum, in the proposed programmable sensor network platforms, the *control mechanism is situated within the sensor node*, which increases the functional complexity of the system. Moreover, very specific things can be reconfigured due to the resource-constrained nature of the sensor nodes, which, in turn, may not be adequate to support different application-specific requirements of IoT.

B. Software-Defined WSNs

Several schemes are also proposed in the context of software-defined WSNs, which address different challenges involved in programmable sensor nodes [5], [13]–[16]. Ferrari et al. [13] proposed a new software-defined WSN architecture, where each of the sensor nodes are enabled with a software programmable transceiver. Depending on the environmental conditions, the transceiver switches to adequate radio communication channels and standards in order to optimize the network performance. Therefore, the proposed approach is capable of addressing different communication issues in WSN in the presence of adverse environmental affects.

Luo et al. [5] proposed a flow-table implementation mechanism in sensor networks (*Sensor-OpenFlow*), in which the forwarding rule is defined by a centralized controller. Thus, the software-defined networking concepts are applied in WSN in order to improve the network performance. The forwarding rules are implemented based on two aspects — *compact network-unique address* and *concatenated attribute-value pairs*. In the former one, node-ID is compared with the flow-table before taking any action (similar to *OpenFlow*), whereas sensed-values are compared in the latter one before taking decisions. Thus, based on the application-specific requirements, forwarding rules can be deployed in the network.

Galluccio et al. [14] proposed a solution concept for software-defined WSN in order to reduce information exchange between controller and the sensor nodes. Therefore, stateful operations are executed at the sensor nodes, so that they are able to adopt adequate policies (previously defined by controller) without requesting to the controller every time. Consequently, the proposed scheme minimizes the message overhead and energy consumption in the network.

Zeng et al. [15] proposed a software-defined sensor network platform, in which each node is embedded with multiple sensors. Three different issues are investigated in the proposed scheme — *sensor activation*, *sensor mapping*, and *sensor scheduling*. The authors showed that energy consumption of the sensor nodes can be minimized significantly, while the sensor nodes are embedded with multiple sensors and controlled in a centralized manner. Similarly, Miyazaki et al. [16] proposed a software-defined networking architecture for sensor networks. However, the proposed software-defined networking aspects in WSN are limited to individual sensor node management.

Synthesis: Critical analysis of the existing works reveals that there exists a research lacuna on software-defined wireless sensor networks to meet the application-specific requirements of IoT. The existing works focused on the issues related to either sensor node or flow-table implementation. However, both the device-specific and network-specific issues must be handled in an efficient manner for better monitoring of physical objects which is the main objective of IoT. In this paper, we propose a software-defined WSN architecture for application-aware service provisioning, so that both the devices and the network can be managed together to improve quality-of-service (QoS) to the IoT users.

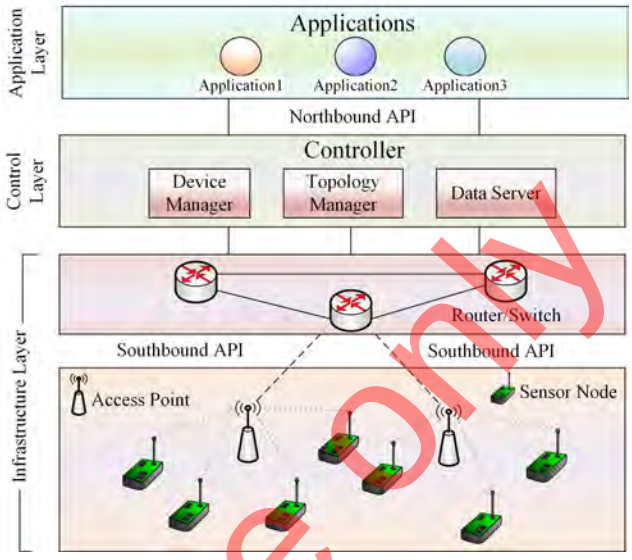


Fig. 1: Proposed software-defined wireless sensor network architecture

III. System Model

In this Section, we present the proposed software-defined WSN architecture, while describing different components of the controller and sensor node in detail.

A. Proposed Architecture

We follow the traditional layered architecture of SDN to design the proposed system [4]. Figure 1 presents an overall architecture of the proposed system with *infrastructure layer*, *control layer*, and *application layer*.

Assumption 1. *We assume that the sensor nodes are deployed in a uniform random fashion, and initially, the network is connected, i.e., each node is within the communication range of atleast one node in the network.*

All physical devices (such as sensor nodes and access points) exist in the infrastructure layer. The sensed data from the sensor node is forwarded to the access point (AP), and finally, it is forwarded to the base station (BS). In the proposed system, both the sensor nodes and AP can be reconfigured in real-time to support application-specific requirements. The detailed architecture of a sensor node is presented in Figure 2. We follow the generalized architecture of a sensor node with different modules — *communication*, *sensor*, *power*, and *micro-controller* [15].

In the control layer of the proposed architecture, adequate decisions are taken by the controller depending on the requirements and topology of the network in order to improve the QoS of the network. Therefore, we design two managers at the control layer — *device manager* and *topology manager*. Detailed architecture of the device manager and topology manager are presented in Figure 3. The device manager is responsible for controlling device-specific tasks such as *sensing delay*, *sensing task*, and *active-sleep* maintenance. On the other hand, the topology manager is responsible for

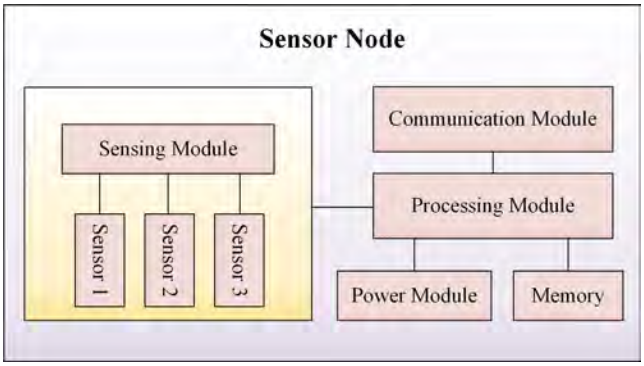


Fig. 2: Architecture of the sensor node

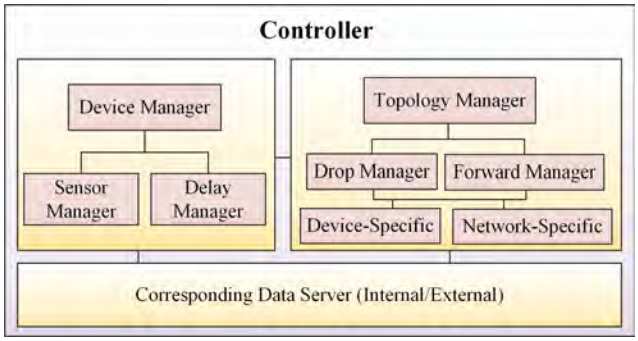


Fig. 3: Proposed architecture of the controller

managing different tasks such as *forwarding rule* management and *network-connectivity* management. As shown in Figure 3, the topology manager can instruct all the nodes within the network to drop or forward any message depending on requirements. For example, if there is any malicious node in the network, the topology manager instructs all nodes to drop the malicious node's message. On the other hand, the topology manager can instruct all the nodes to forward a node's message with higher priority. Therefore, the topology manager handles the topology management issues in the network.

Finally, in the application layer, application-specific requests are generated, and eventually, those are sent to the controller for execution at the control layer.

Thus, we design a complete software-defined sensor network architecture having application, control, and infrastructure layer, as maintained in the traditional SDN architecture for application-specific service provisioning in IoT. Consequently, any sensor node can be incorporated within the proposed system, and can be controlled in real-time without changing the hardware platform of the sensor node.

B. Protocol Architecture

In this Section, we discuss about the protocols used in the proposed system. The sensor nodes communicate among themselves and APs, and the APs communicate with the controller and server, which store the sensed data. Therefore, we discuss the *sensor-AP* and *AP-controller* communications, in brief, in the subsequent Sections III-B1 and III-B2.

1) Sensor-AP Communication: We use the Zig-bee (IEEE 802.15.4) [7] protocol architecture to enable communication among sensor nodes in the network, as it is widely used in low-power networks. Moreover, most of the sensor nodes, which are available in market, use Zig-bee protocol for communication. Therefore, we assume that the sensor nodes support Zig-bee for communication with the APs in the network. However, other radio technologies can also be integrated.

2) AP-Controller Communication: We use the IEEE 802.11 [8] protocol architecture to enable communication among APs, server and controller. Therefore, AP sends sensor data to the server through the WLAN technology, in which server is running in high-speed wired network. Similarly, the controller communicates with the APs through the traditional networking technologies. The decisions taken by the controller are forwarded to the corresponding nodes by APs.

Moreover, we use the IEEE 802.15.4 and the IEEE 802.11 protocols in the proposed system as they are well-established protocols used in IoT. We limit our discussions to both the sensor-AP and AP-controller communication architecture, as our main objectives in this work are device management and topology management in software-defined WSN.

C. Network Controlling

The main objective of the proposed software-defined WSN architecture is to manage the wireless sensor nodes and network in order to improve the network performance. Therefore, we propose two management policies — *device management* and *topology management*. In the device management, device-specific control tasks are included. On the other hand, network topology is controlled by topology management. We describe the functions of device management and topology management in Sections IV and V, respectively.

IV. Device Management

Device management policies are taken by device manager. The device manager manages individual sensor nodes in the network, and it is responsible for scheduling the *sensing tasks*, *sensing delay*, and *active-sleep* maintenance. We discuss sensor management, delay management, and active-sleep management policies in Sections IV-A, IV-B and IV-C. We primarily focus on the packet formats which need to be exchanged in real-time to change different tasks. Accordingly, sensor nodes change their activities to meet the application-specific requirements.

A. Sensor Management

As discussed in Section III, the sensor nodes have multiple sensors, which are required to be managed to meet application-specific requirements. Figure 4 presents the packet format to control the *sensor-IDs* within a sensor node. The packet includes different fields — *Header*, *Length*, *Type*, *Node-ID*, *Sensor-ID*, *Action*, *TTL*, *Checksum*, and *Options*. The *Header* defines the header format to enable application programming interface (API) mode on the Zig-bee modules. *Length* represents the total length of the packet, which is used to calculate the *Checksum*. We define three types of information exchange — *unicast*, *multicast* and *broadcast*. To enable/disable specific

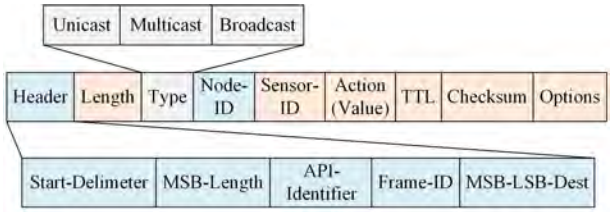


Fig. 4: Packet format for sensor management

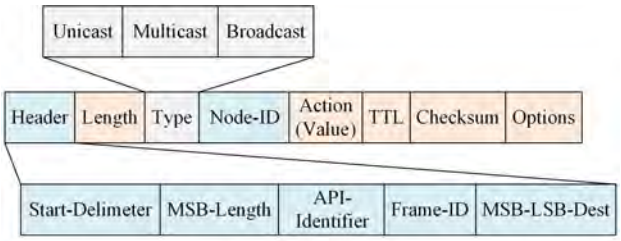


Fig. 5: Packet format for delay management

Sensor-ID in individual sensor node, the *unicast* mode is used. On the other hand, the *broadcast* mode is used to enable/disable specific *Sensor-ID* in all sensor nodes in the network. *Node-ID* and *Sensor-ID* represent the IDs of sensor node and sensor, respectively. In case of *Node-ID*, 16-bit address of the Zig-bee module is used. *Action* defines the required sensor to be activated. *TTL* defines the duration of time-to-live of a packet. *Checksum* is used for error checking. Finally, the *Options* field is used for any specific instruction to be given to sensor nodes.

B. Delay Management

The sensing delay of the WSN nodes is managed by the delay management policy. Similar to the sensing task, sensing delay can also be changed in real-time. Figure 5 presents the packet format used for delay management. In the packet format, *Header*, *Length*, *Type*, *Node-ID*, *TTL*, *Checksum* and *Options* represent similar things, as described in Section IV-A. *Action* defines the desired sensing-delay of the sensor nodes.

C. Active-Sleep Management

The active-sleep state of a sensor node is managed by the active-sleep management. Packet format for active-sleep management is presented in Figure 6. In the packet format,

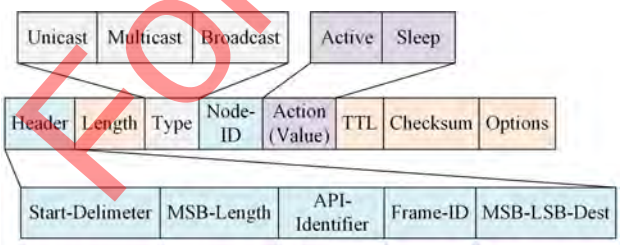


Fig. 6: Packet format for active-sleep management

Header, *Length*, *Type*, *Node-ID*, *TTL*, *Checksum* and *Options* represent similar things as described earlier. *Action* defines the desired state, i.e., active or sleep, of the sensor node.

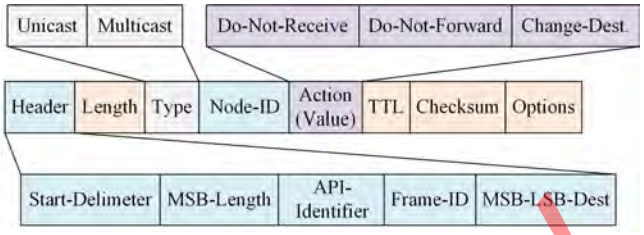


Fig. 7: Packet format for node-specific forwarding rule management

V. Topology Management

In this Section, we describe the topology management module, while focusing on the forwarding rule and network-connectivity management in the network. We categorize the forwarding rule management in two aspects — node-specific and network-specific — rather than considering a uniform forwarding rule management policy. Consequently, depending on the application-specific requirements, rules can be implemented, while preserving QoS of the network. We discuss both the forwarding rule management policies in the subsequent Sections.

A. Node-Specific Management

In node-specific rule management, we define the forwarding rules which are applicable for a node or group of nodes. Figure 7 presents the packet format used for node-specific forwarding rule management. The packet includes *Header*, *Length*, *Type*, *Node-ID*, *Action*, *TTL*, *Checksum*, and *Options*, which are already described above. However, in node-specific forwarding rule management, the *Type* field defines either unicast or multicast. The *Action* field includes different actions — *Change-Dest*, *Do-Not-Send* and *Do-Not-Receive*. The *Change-Dest* defines the change in the forwarding device ID to whom a node sends the packet to forward. Thus, the network remains connected in absence of any other nodes. A node is instructed not to send its data to a specific forwarding device. Finally, *Do-Not-Receive* signifies that a node should not receive from a specific node. Therefore, node-specific forwarding rules management policies are ensured with the above mentioned rules.

B. Network-Specific Management

In contrast to the node-specific forwarding rules management, network-specific management defines a uniform forwarding rules for all nodes in the network. The packet format used in network-specific management is shown in Figure 8. It also includes *Header*, *Length*, *Type*, *Action*, *TTL*, *Checksum*, and *Options*. Here, the rule is broadcasted to every node in the network, which is defined by the field *Type*. Different actions in the network-specific management are as follows — *Drop-From* and *Do-Not-Forward*. *Drop-From* defines that all the nodes in the network are instructed to drop a particular node's message. For example, if a malicious node is present in the network, the controller instructs all other nodes to drop the malicious node's information. Similarly, *Do-Not-Forward*

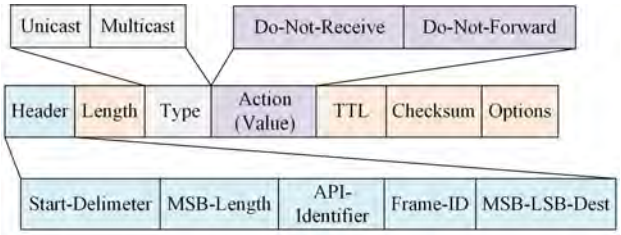


Fig. 8: Packet format for network-specific forwarding rule management

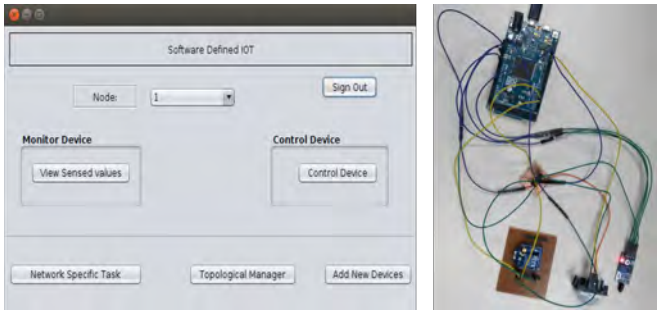


Fig. 9: Developed Soft-WSN hardware platform

defines that all the nodes in the network are instructed not to forward to a particular node. For example, this situation occurs when there is a *selfish* node in the network, which does not forward others' messages. Therefore, using the proposed scheme, the 'selfish' nodes can be avoided from data forwarding route. We limit our discussion on detecting the 'selfish' nodes in the network, as our main objective in this paper is to control the network topology. However, we ensure that using the proposed *Do-Not-Forward* concept, it is possible to avoid the selfish nodes in the network in data forwarding process. Therefore, we ensure that all the network-specific forwarding rule management can be done using the proposed method.

VI. Performance Evaluation

In this Section, we analyze the performance of the proposed system in terms of different performance metrics, which are the key-factors in a WSN. Figure 9 shows developed platform as discussed in Sections III, IV and V. The proposed system can be used for several IoT applications such as building a smart home, environment monitoring, and traffic monitoring, in which the sensors can be controlled in a centralized manner from both the aspects — device and topology management. It is noteworthy that we implement the flow-table rules at the devices as mentioned in Sections IV and V. Upon receiving requests, the devices take decisions based on the rules defined in the flow-tables. For simplicity, we limit our discussion on flow-table implementation as part of the performance evaluation, as we discuss it earlier in detail. Moreover, we believe that it is easy to implement the flow-table rules at the devices, while following the procedure presented in this paper.

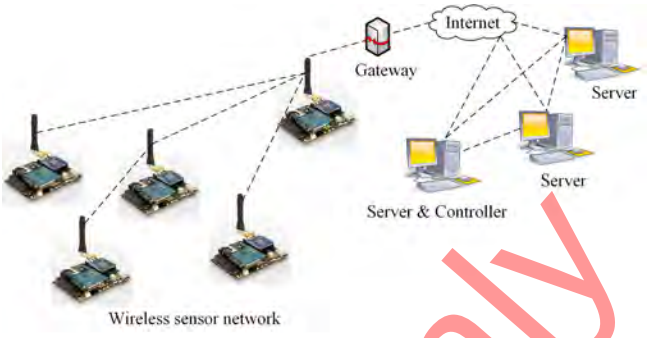


Fig. 10: A schematic view of the experimental setup

A. Experimental Setup

To evaluate the performance of the proposed scheme, we setup a hardware platform as mentioned in Section III. The schematic view of the hardware platform is shown in Figure 10. Table I shows the list of parameters and the corresponding values used for the experiment [17].

TABLE I: Experimental Setup

Parameter	Value
Transmission Range	15 m
Transmit Power	24.75 mW
Receive Power	13.5 mW
Idle Power	13.5 mW
Transmission bit rate	40 kbps
Protocol	IEEE 802.15.4 & IEEE 802.11

B. Performance Metrics

We use different performance metrics — *packet delivery ratio*, *energy consumption*, and *message overhead* for characterizing the performance of the proposed system. Detailed calculation of the performance metrics is discussed in the subsequent Sections.

1) Packet Delivery Ratio (ρ): The packet delivery ratio (ρ) is calculated as the ratio between total number of packets received and total number of packets transmitted, mathematically:

$$\rho = \frac{\sum \text{Total packet received}}{\sum \text{Total packet transmitted}} \tag{1}$$

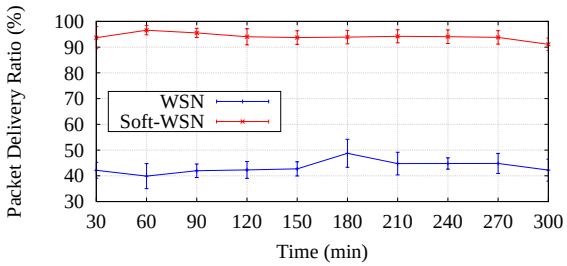
2) Energy Consumption: To calculate total energy consumption in the network, we calculate the energy spent for transmission, reception, and idle condition. The energy required for transmission (E_{tran}), reception (E_{rec}), and idle condition (E_{idle}) are calculated as follows:

$$E_{tran} = P_{tran} T_{tran} \tag{2}$$

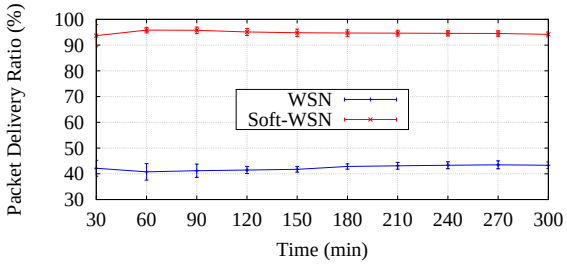
$$E_{rec} = P_{rec} T_{rec} \tag{3}$$

$$E_{idle} = P_{idle} T_{idle} \tag{4}$$

where P_{tran}, P_{rec} and P_{idle} denote energy required for transmission, reception and idle condition in a unit time,

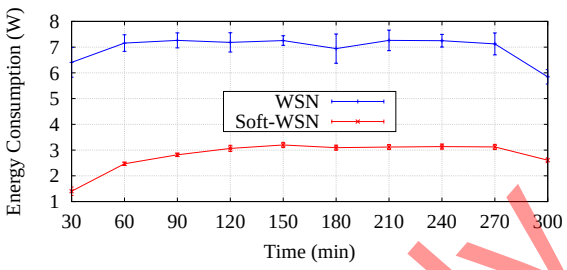


(a) Packet delivery ratio at different time periods

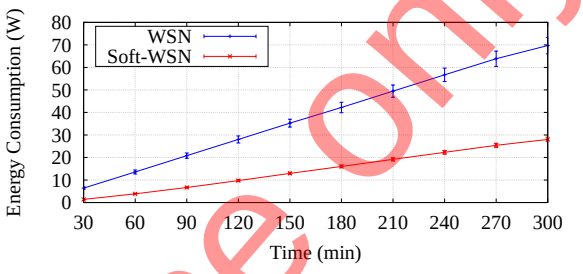


(b) Total packet delivery ratio

Fig. 11: Packet delivery ratio in the network



(a) Energy consumption in the network at different time periods



(b) Total energy consumption in the network

Fig. 12: Energy consumption in the network

respectively. T_{tran} and T_{rec} represent time required for transmission and reception, respectively. T_{idle} represents idle time of the sensor nodes.

3) Message Overhead: Message overhead is calculated as the number of messages transferred among the sensor nodes. We calculate the number of messages as the combination of both data packet and control message.

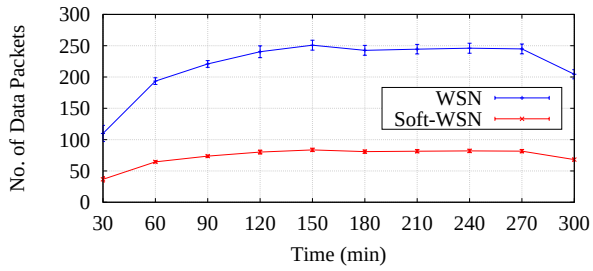
C. Results and Discussion

Using the performance metrics discussed above, we observe the performance of the proposed scheme, *Soft-WSN*, over traditional WSN to show the effectiveness of the proposed scheme. As discussed in Sections III, IV and V, the controller manages all the sensor nodes in the network in a centralized manner depending on the application-specific requirements. On the other hand, in traditional WSN, we use the distributed architecture in which sensor nodes act in a distributed manner. We discuss the performance of the proposed scheme, *Soft-WSN*, compared to the traditional WSN, *WSN*, in the following Sections with *confidence interval*. The *confidence interval* is considered to show the variation of obtained results in multiple runs. We adopt the use of 95% confidence rule [18], i.e., in 95% cases, we are confident that the obtained results lie between the specified range.

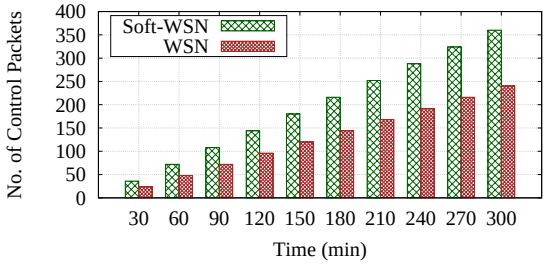
1) Packet Delivery Ratio: Figure 11 shows the packet delivery ratio in the network obtained by the proposed scheme, *Soft-WSN* and traditional scheme, *WSN*. In Figure 11(a), we present the packet delivery ratio at each time-period. On the contrary, Figure 11(b) shows the packet delivery ratio in the network in terms of cumulative distribution function. In both the cases, it is evident that the proposed scheme outperforms the traditional WSN schemes in terms of packet delivery ratio. In the proposed scheme, the controller defines the forwarding rules for each sensor node in the network. Therefore, the routing path is maintained by the controller. Additionally, network connectivity is also ensured by the controller, while leveraging global view of the network. Consequently, more data packets reach to the destination compared to the traditional WSN schemes in which routing is done in a distributed manner. Eventually, the packet delivery ratio using *Soft-WSN* is higher than the traditional *WSN* schemes.

2) Energy Consumption: We evaluate the energy consumption in the network incurred by sensor nodes due to transmission, reception, and idle condition. Figure 12(a) shows the energy consumption at different time periods in the network. On the other hand, Figure 12(b) presents the total energy consumption in the network. From the Figures, it is also evident that the proposed scheme, *Soft-WSN*, minimizes the energy consumption in the network significantly compared to the traditional *WSN* schemes. Similar to the information routing, transmission, reception and idle-condition of sensor nodes are controlled by the centralized controller. Thus, *Soft-WSN* optimizes the scheduling tasks of sensor nodes to minimize the energy consumption, while leveraging global view of the network. On the other hand, the sensor nodes schedule different activities (such as transmission, reception and idle-condition) in a distributed manner, in which central coordination is absent. Consequently, the proposed scheme outperforms the without SDN-based schemes in terms of energy consumption in the network. Thus, the proposed scheme is also beneficial to improve the lifetime of the network.

3) Message Overhead: Finally, we present the message overhead in the network in Figure 13 from two aspects — overhead for data packets and control packets. Figure 13(a) shows the message overhead at different time periods for data packets in the network. On the contrary, Figure 13(b) presents the cumulative overhead for control packets in the network. These figures signify that the proposed scheme is capable



(a) Overhead for data packets at different time periods



(b) Cumulative overhead for control packets

Fig. 13: Message overhead in the network

of minimizing the total message overhead in the network to a large extend compared to traditional *WSN* schemes. In the proposed scheme, *Soft-WSN*, the sensor nodes unicast (or multicast) its information to its neighbor(s) depending on the forwarding rule defined by the controller. Therefore, creation of data packet-replicas is controlled using the proposed scheme. On the other hand, in case of traditional *WSN*, the sensor nodes broadcast the data packets in the network. As a result, message overhead for data packets in traditional *WSN* is higher than the proposed scheme, *Soft-WSN*. On the other hand, overhead for control packets in the network is higher using *Soft-WSN* compared to that of using the traditional *WSN* schemes, as the nodes communicate with the controller on receiving a new packet from other nodes.

VII. Conclusion

In this paper, we proposed a software-defined WSN architecture to deal with the application-specific requirements of IoT, which is dynamic in nature. We presented controller and sensor node architecture to enable SDN support in WSN. We proposed two components of the controller — device manager and topology manager. The former is responsible for handling device-specific tasks (such as sensing task, sensing delay, and active/sleep scheduling), and the latter is responsible for managing network topology to ensure QoS of the network. Therefore, in contrast to the existing SDN solutions for wireless sensor networks (WSNs), the proposed system, *Soft-WSN*, focused on both device management and topology management in the network. From the experimental results, it is evident that the proposed scheme is beneficial for application-aware service provisioning in IoT, while improving network performances over the traditional sensor networking approaches.

In this work, we considered that the sensor devices use similar radio technology for communication, which may not be adequate to include all types of sensor devices in the network. Therefore, we plan to incorporate other radio technologies (such as Bluetooth) within the sensor devices as a future extension of this work. Additionally, in this work, we place the controller at the server end to control the network activities. However, minimization of network delay and control message overhead may be explored, while focusing on *optimal* controller placement problem in SDWSN. Additionally, we plan to analyze the complexity involved in the flow-table implementation at the devices using the proposed scheme.

References

[1] L. Mainetti, L. Patrono, and A. Vilei, "Evolution of wireless sensor networks towards the Internet of Things: A survey," in *Proc. of Intl .Conf. on Software, Telecomm. and Computer Networks*, Split, Sept. 2011, pp. 1–6.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks (Elsevier)*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] O. Bello and S. Zeadally, "Intelligent Device-to-Device Communication in the Internet of Things," *IEEE Systems Journal*, 2014, DOI: 10.1109/JSYST.2014.2298837.

[4] K. Sood, S. Yu, and Y. Xiang, "Software Defined Wireless Networking Opportunities and Challenges for Internet of Things: A Review," *IEEE Internet of Things Journal*, 2015, DOI: 10.1109/JIOT.2015.2480421.

[5] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. R. andScott Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, Apr. 2008, pp. 69–74.

[7] J. A. Gutierrez, E. H. Callaway, and R. Barrett, *IEEE 802.15.4 Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensor Networks*. IEEE Standards Office, NY, USA: IEEE, 2003.

[8] *IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std.

[9] Y. Krasteva, J. Portilla, E. de la Torre, and T. Riesgo, "Embedded Runtime Reconfigurable Nodes for Wireless Sensor Networks Applications," *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1800–1810, 2011.

[10] C.-M. Hsieh, Z. Wang, and J. Henkel, "A Reconfigurable Hardware Accelerated Platform for Clustered Wireless Sensor Networks," in *Proc. of the IEEE Intl. Conf. on Parallel and Distributed Systems (ICPADS)*, Singapore, Dec. 2012, pp. 498–505.

[11] S. Vera, A. Bayo, N. Medrano, B. Calvo, and S. Celma, "A programmable plug&play interface for WSN applications," in *Proc. of the IEEE Sensors*, Limerick, Oct. 2011, pp. 1808–1811.

[12] P. Angove, M. O'Grady, J. Hayes, B. O'Flynn, G. O'Hare, and D. Diamond, "A Mobile Gateway for Remote Interaction With Wireless Sensor Networks," *IEEE Sensors Journal*, vol. 11, no. 12, pp. 3309–3310, 2011.

[13] P. Ferrari, A. Flammini, and E. Sisinni, "New Architecture for a Wireless Smart Sensor Based on a Software-Defined Radio," *IEEE Trans. on Instrumentation and Measurement*, vol. 60, no. 6, pp. 2133–2141, 2011.

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

[15] D. Zeng, P. Li, S. Guo, T. Miyazaki, J. Hu, and Y. Xiang, "Energy Minimization in Multi-Task Software-Defined Sensor Networks," *IEEE Trans. on Computers*, vol. 64, no. 11, pp. 3128–3139, 2015.

[16] T. Miyazaki, S. Yamaguchi, K. Kobayashi, J. Kitamichi, S. Guo, T. Tsukahara, and T. Hayashi, "A software defined wireless sensor network," in *Proc. of Intl. Conf. on Computing, Networking and Communications (ICNC)*, Honolulu, HI, Feb. 2014, pp. 847–852.

[17] F. Bouabdallah, N. Bouabdallah, and R. Boutaba, "On Balancing Energy Consumption in Wireless Sensor Networks," *IEEE Trans. on Vehicular Technology*, vol. 58, no. 6, pp. 2909–2924, 2009.

[18] A. Hackshaw, *A Concise Guide to Clinical Trials*. Oxford, UK: BMJ, 2009, ch. Statistical formulae for calculating some 95% confidence intervals.