SDNMM- A Generic SDN-based Modular Management System for Wireless Sensor Networks

Musa Ndiaye, Student Member, IEEE, Adnan M. Abu-Mahfouz, Senior Member IEEE, and Gerhard P. Hancke, Senior Member IEEE

***Abstract*—Software Defined Networking (SDN) promises a wide range of application benefits to the Internet of Things (IoT) including the flexible management of Wireless Sensor Networks (WSNs). While the integration of SDN techniques in WSNs is being extensively investigated, there remains a need for a general SDN-based management system for WSNs. A system that should provide an opportunity for rapid testing and implementation of management modules to the IoT developer community working on taking advantage of the SDN benefits. Therefore, this paper proposes SDNMM, a generic and modular WSN management system based on SDN. SDNMM introduces the concept of management modularity using a Management Service Interface (MSI) that enables management entities to be added as modules. The system leverages the use of SDN in WSNs and by being modular it also allows for rapid development and implementation of IoT applications. The system has been built on an open source platform to support its generic aspect and a sample resource management module implemented and evaluated to support the proposed modular management approach. Results show how adding a resource management module via the MSI improved packet delivery, delay, control traffic and energy consumption over comparable frameworks.**

***Index Terms*—Wireless Sensor Network, Software Defined Networking, Management Service Interface, Modularity**

I. Introduction

In recent years applications of Wireless Sensor Networks (WSNs) are ever on the rise especially with the boom in Internet of Things (IoT) technology [1]. This means large-scale implementations of WSNs resulting in direct complexity in network management [2]. It is therefore imperative that an effective Network Management System (NMS) be put in place. By definition, an NMS should be able to monitor and control the network in either a centralized, distributed or hierarchical manner. The system must ideally support the following key functional areas: network configuration management, network monitoring, topology management, Quality of Service (QoS) Management, resource allocation, energy and security management [3], [4]. Meeting the above criteria becomes challenging considering the heterogeneous nature of WSNs made up of nodes that are resource constrained and susceptible to failure especially when deployed in harsh environments. To solve this issue WSN management systems have been proposed in the past. Several of these contributions solve only portions of the network management architecture such as RRP, Agilla, SNMS, SNMP, WinMS [5] while other contributions such as lightweight [6] and group mobility support [7] focus on network management protocols. However, there has been characteristic examples of systems that are based on overall traditional management of WSNs such as MANNA [8], BOSS [9] and DISON [10]. MANNA provides a general overview of managing a WSN based on multidimensional planes referred to as abstractions. The abstractions handled in MANNA provide support for the informational, functional and physical architectures of the network. It is more of a policy-based management system that does not have a direct application to managing critical WSN design criteria such as energy efficiency, adaptability, scalability, and robustness. The BOSS architecture is also an overall management system however it is specific to creating a bridge between resource-constrained WSN nodes and resource hungry Universal Plug and Play devices (UPnP). The DISON management framework is based on a multilevel mechanism where each sensor node depending on its resources can participate at various levels in managing the WSN. The sensor nodes are also able to adapt to various conditions based on a context and policy model. While these techniques provide some form of overall traditional WSN management based on context, policies, and protocols; meeting modern network management that requires the need for rapid prototyping and implementation of application specific services in a resource-constrained and heterogeneous environment still remains an open issue. Software-defined networking (SDN) provides a more promising solution in providing effective and flexible WSN management. SDN allows for the separation of the network into three planes namely the application, control, and data planes [11], [12]. This introduces a logically centralized view to network management, an aspect that results in efficiency and flexibility in managing the network [13], [14].

We present SDNMM, a generic SDN-based Modular Management system for WSNs that is implemented on the IT-SDN [15] architecture due to its open source availability and generic nature. However, with SDNMM we make the following novel contributions:

- We propose for the first time in SDN-based WSN management, a state machine based management service interface (MSI) that implements management applica-

Corresponding author: M. Ndiaye is with the Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria, 0028, South Africa, e-mail: mndiaye@ieee.org.

A.M. Abu-Mahfouz is with the Electrical, Electronic and Computer Engineering Department, University of Pretoria and the Council for Scientific and Industrial Research, Pretoria, South Africa, e-mail: a.abumahfouz@ieee.org.

G. P. Hancke is with the Computer Science Department, City University of Hong Kong, China. He is also with the Electrical, Electronic and Computer Engineering Department, University of Pretoria, South Africa., e-mail: ghancke@ieee.org.

tions as modules existing in each state. We implement a resource-aware task allocation management module to evaluate the feasibility of this proposal. The ON/OFF states of nodes are also considered by introducing an active-sleep module state in the MSI.
- A system is proposed to make use of a context data knowledge-base to act as a common reference for interfacing of management modules. This technique allows context data to be provided as input to various management applications with the output being management policies based on the input data. To achieve this, a context collect mechanism is implemented using context data packets that are stored in the controller via dynamic linked lists.
- We make performance improvements over baseline IT-SDN [15] in terms of packet delivery, packet delay, control overhead and energy consumption for grid topology. We also analyze and present the effects of varying topologies such as mesh and ring on the performance.

This paper is organized as follows: Section II discusses some related work, Section III introduces the SDNMM architecture, in Section IV we discuss the functions and features of the Management Service Interface (MSI). Section V discusses how the SDNMM system framework has been implemented and in Section VI we evaluate the performance of the implemented system. In Section VII we discuss the performance evaluation related to the SDNMM system and in Section VIII we set our conclusion and discuss future work.

II. Related Work

Several works have been done in enabling SDN for WSNs to develop Software Defined Wireless Sensor Network (SD-WSN), notable contributions include Sensor OpenFlow [16], TinySDN [17], SDN-WISE [18] , Coral-SDN [19] and IT-SDN [15]. These techniques play an important role in providing an operating environment for SDN-based management systems. However, the development of generic SDN-based management systems for WSNs with high-level abstractions still remains largely open for research. Smart [20], Soft-WSN [21] and more recently a 6LowPAN SDN-based IoT framework [22] have been proposed to provide a more generic management abstraction.

Smart offers some form of topology management through localization and tracking algorithms that work in tandem with the controller. In Smart, a general framework is proposed with the SDN controller at a base station. Gante *et al.* argue that SDN would greatly ease the complexity of sensor network management and raise important questions regarding the integration of a distributed controller in terms of general network performance and energy consumption of the WSN. They also discuss the effect of the OpenFlow protocol in constrained sensor networks and also synchronization issues regarding maintaining the controller global view. However, the authors in Smart mention that their proposal was a preliminary stage with no implementation or framework evaluation presented. Their proposed Smart framework did not address the management of application-specific requirements or integration of specific management entities. Furthermore, to allow for a global network view that SDN provides, Smart uses a single base station controller which has reliability and security challenges upon failure or compromise of the centralized controller respectively. Another challenge associated with such centralized management is the accumulation of overhead traffic data to the controller in an already resource-constrained WSN.

Soft-WSN addresses the challenge of meeting application-specific needs for IoT by use of SDN. Two management techniques are developed to meet device and network management. Bera *et al.* [21] investigate sensing tasks and delay including active sleep scheduling to implement device management. They also modify network policies at run time to meet topology requirements. Results from a hardware test-bed show improved packet delivery ratio and energy consumption compared to an ordinary WSN. However, the discussion and implementation in Soft-WSN are limited to providing only device and topology management services. Like Smart, it is also based on a single global controller resulting in a centralized management scheme and thus prone to the associated challenges. Furthermore, both Smart and Soft-WSN also do not address issues of modularity to allow for widespread WSN applications or upgrades of management components.

Lasso *et al.* [22] in an SDN-based IoT framework for 6LowPAN focus on SDN-based energy management using a transmit power control mechanism from the controller specifically for a 6LowPAN. The authors implement the framework on the open source Contiki [23] cooja platform and results show improved energy performance in the SDN-enabled nodes. However, most of the work focuses on improving the energy consumption, issues related to the management framework operations including the modular aspect of the effects of the framework on traffic overhead and packet delay are not addressed in detail.

Having looked at the related work, there is still a need for a more open source and general management framework that allows some form of modularity for easy implementation of management methods. Another, bottleneck that could be observed is the increased control traffic coupled with the use of a centralized controller, an aspect that can be mitigated by using a distributed controller mechanism [24], [25]. There is a need for a framework that would allow the management methods such as that those implemented in [21] and [22] to be packaged as a module for quick testing and implementation. The novel SDNMM framework being proposed in this paper provides a promising solution to addressing the above-named issues. The finding in SDNMM also answer some of the questions raised in Smart [20] regarding performance and energy consumption of an SDN-based WSN management framework.

III. SDNMM System Overview

A. SDNMM Architecture

The proposed SDNMM management system framework shown in Fig. 1 is based on a hierarchical SDN distributed controller architecture efficient in handling scalability and traffic overhead [24], [25]. The overall SDNMM framework is subdivided into SDN abstraction planes as follows:

1) Application Plane: The application plane is composed of multiple third-party user applications that communicates with the network using APIs provided by the controller. The applications can be used to handle the following tasks:

- Monitoring of network state including fault detection or isolation and energy level monitoring.
- Task configuration of nodes to meet application demand.
- Policy issuance for network and device-specific tasks based on context.

2) Control Plane: The control plane in the SDNMM framework is a two-tier configuration of a global controller and physically distributed local controllers (cluster managers). Both the global and local controllers are a higher resource capacity compared to sensor nodes. The Management Service Interface (MSI) and the various management modules are contained in the global controller while cluster managers handling sub-management tasks are in the local controller tier of the control lane. Each cluster manager is responsible for events and tasks occurring in the respectively assigned cluster. Cluster managers are able to communicate with each other via East-West APIs. Generally, the following functions should be executed in the control plane:

- Integration of core WSN management modules such as configuration, topology, energy, security and QoS.
- Provision of APIs to the application and data planes
- Collection of context information and handling of policies including issuing policies based on context and flow commands for application requests.

3) Data Plane: The data plane is made up of a network of sensor nodes configured in clusters to allow for improved scalability. In each cluster, SDN-enabled nodes which are a hardware combination of an SDN-enabled switch and an end-device [17], generate data packets which are sent to the sink node. The sink node allows for a data link to the neighboring cluster manager (controller) and cluster. Network APIs are made available by the controller to allow relaying of sensor data including context/policy information to and from the control/ application planes.

B. SDNMM Features and Functions

The generic SDNMM framework introduces the following features in WSN management:

- Modularity: Allows each management entity to be a well defined functional component with APIs available to communicate commands, contexts, and policies via the Management Service Interface (MSI). The management components/modules provide a higher level management abstraction thus introducing flexibility in the use of existing or new management techniques.
- Generic: The use of modular components allows the use of the system in various applications and heterogeneous environments as components can be designed and added to the system framework to meet the application specific needs.
- Scalability: The distributed hierarchical architecture on which the framework is based allows for cluster managers to be assigned to each node cluster limiting frequency of overhead traffic that needs to be sent to the global controller. This configuration gives room for efficient network expansion.
- Adaptability: Sensor tasks can be issued on demand to meet application-specific needs. The possibility of function alternation upon neighboring node failure is introduced.
- Robustness: Multiple context-aware cluster managers working in tandem with the global controller ensures effective detection and isolation of faults. Functionality can be introduced for a cluster manager to temporary manage a neighboring cluster that has a cluster manager that is down.
- Energy efficiency: Apart from the ability to use energy management modules for active-sleep management, SD-NMM leverages SDN to move resource-heavy tasks to more resource capable hardware in the network such as the cluster managers and the global controller. This leaves nodes with the simple task of forwarding event and context data.

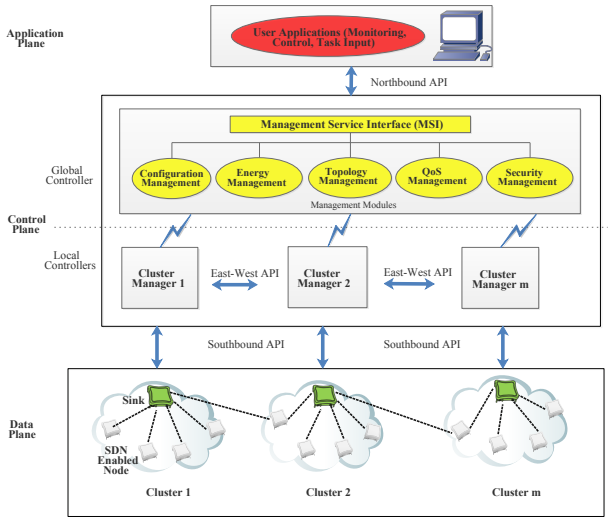


Fig. 1. SDNMM System Architectural Framework.

IV. Management Service Interface (MSI)

The Management Service Interface (MSI) provides an interface to enable this novel SDN-based approach of adding management services as modules or components. The interaction between the MSI and the modules can be through coupling APIs or through classification criteria based on management task type as has been implemented in this paper. The MSI is generally middle-ware between management services and user applications/ network infrastructure. Core features and functions of the MSI include:

- Enabling a modular approach to SDN-based management for easy and rapid addition or removal of management services to the WSN necessary for future upgrades or expansion.

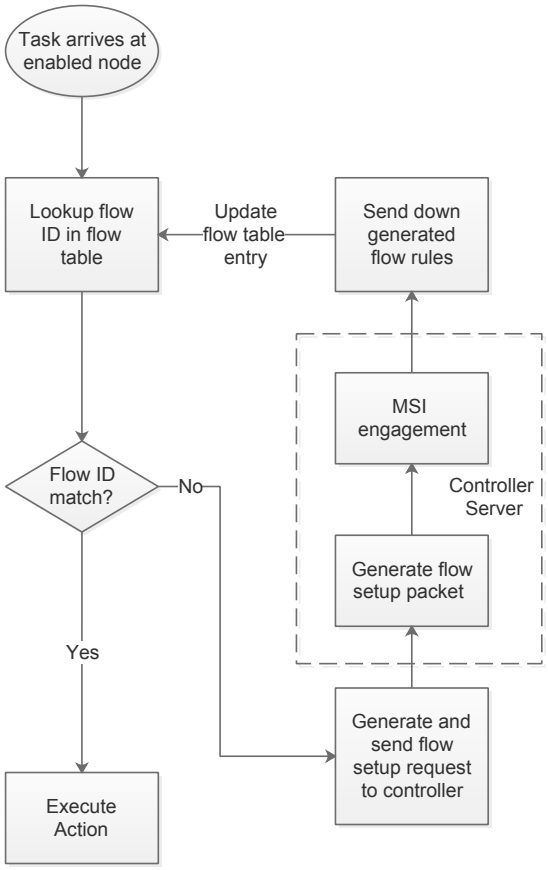


Fig. 2. MSI engagement in SDWSN task sequence

- Provision of loose coupling through APIs or event classifiers to management modules, application and data planes enabling the integration and use of third-party management services in an orderly manner.
- Provide an interface for exchange of context information between management modules, for example, the energy management module can give insights on the status of energy resources in the network to the QoS management module which can then make decisions based on available resources.

APIs, contexts, and policies play a crucial role in ensuring this flexible coupling and feature function between management components in the SDNMM framework. In this paper, the MSI has been built and implemented as a function that can be called at any stage in the processing of a task in the SDN architecture. Consider the sequence of processing a task in a typical software defined wireless sensor network shown in Fig. 2. The MSI can be engaged during the flow setup process in the controller as part of the data management process which may include altering task action based on destination node resource capability or addition of management data to the flow setup packet. We implement and evaluate a resource allocation management module in this paper based on this approach.

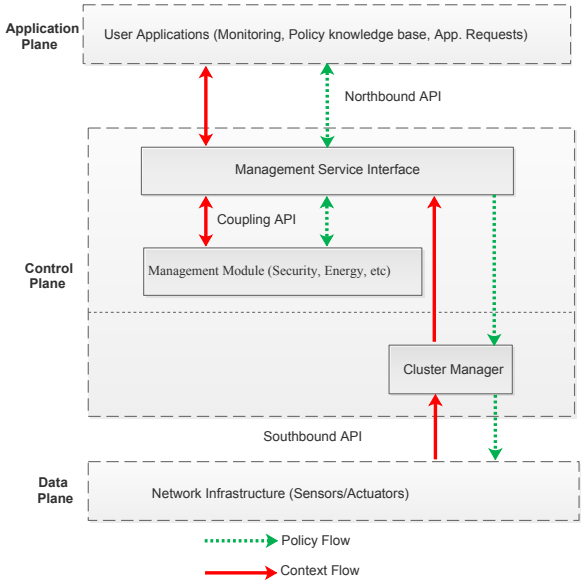


Fig. 3. SDNMM Context and Policy Flow.

A. Context and Policy Handling

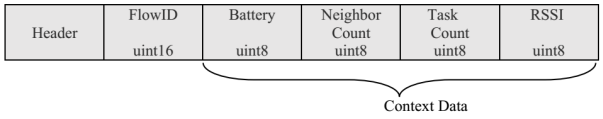
Context awareness and the issuance of the corresponding policies are an important aspect of WSN management. Context is basically information or an event that often results in an action policy being issued for reconfiguration purposes. WSN Context can be classified into three categories [10]:

- Node Resources: Information here mainly includes the status of node residual energy, memory usage, and sensing capabilities. In the SDNMM framework, node resource context information is obtained from the data plane. Resulting policies can be made from the application and control planes.
- Network State: Context in this category provides information on the running status of the network in terms of communication link status, availability of bandwidth and other network topology related aspects. The source of this information in SDNMM is both from the data plane and aggregate data from cluster managers. Aggregate cluster manager data provides network-wide context information. User applications and management modules can input policies to respond to this kind of context.
- Application requirements: Users can set parameters for a task with certain requirements which provides context data for the MSI and management modules upon which policies can be generated. Examples include requests to change sensing task, add or drop packets and requests to encrypt data. In the proposed SDNMM system, this context category originates from the application plane.

Fig. 3 shows the flow of contexts and policies in the SDNMM system.

B. API Functions

APIs allow the interaction and coupling of the components in the framework shown in Fig. 1. Northbound APIs allow the MSI in the control plane to interact with user applications in the application plane. Information such as task definition parameters, context, and policy data are transferred via this API. In the southbound direction, cluster managers interact with sensor nodes using southbound APIs. Cluster managers can request for sensor and network running statuses, provided flow commands, issue required task parameters and other context-related policies while being coupled by these APIs. East-West APIs facilitate the interaction of cluster managers. Information related to scheduling of network-wide tasks or policies can be exchanged over these APIs. Restful APIs are necessary to allow interaction of the MSI with management modules.



NAME		LENGTH (bytes)	FUNCTION
Header		6	Contains SDN header information such as packet type, source address and reserved slots.
FlowID		4	Used to contain source and destination node flowId information.
Context data	Battery	1	Battery shows the percentage battery level in the source node.
	Neighbor Count	1	Shows the number of neighbors the source node has.
	Task Count	1	Data showing the number of pending tasks of the source node.
	RSSI	1	Shows the last registered received signal strength.

Fig. 4. Context Report Packet Format

V. SDNMM Implementation

Notable platforms that can be used to implement an SDN-based management system framework include SDN-WISE [18], Coral-SDN [19] and IT-SDN [15]. All three platforms provide an environment controller and neighbor discovery suitable for SDWSNs. However, both SDN-WISE and Coral-SDN are limited in terms of developer support and source code availability making it a difficult choice for our generic framework to be implemented on. On the other hand, IT-SDN provides the required open source availability and hence SDNMM is built on it to allow for a more generic use case. IT-SDN is based on Qt which is available for all major computer operating systems and the Contiki [23] cooja tool which is open source and easily accessible. As far as the scope of this paper, we implement and evaluate SDNMM based on a single centralized controller with features demonstrating the integration of a resource allocation management module and the ability to monitor network parameters and performance from a graphic user interface in Qt.

A. Monitoring Management

The underlying IT-SDN platform uses the Cooja simulator GUI to set up and monitor various aspects of the WSN and also extends monitoring to a computer GUI developed in Qt taking advantage of a debug window to monitor network messages and configuration. With SDNMM, we add features to monitor data on network performance metrics as well as the ability to visualize these metrics graphically in a Qt-based GUI.

B. Context data approach to Management Modularity: A resource allocation module use case

1) Context data pooling: We introduce an aspect of management modularity based on a context data knowledge-base stored in the controller memory using dynamic linked lists. A context collect process has been developed to collect and send context report to the controller on event change and at various sampling intervals depending on battery level. To integrate energy efficiency in the SDNMM design, data collection outside event changes are based on a sampling interval (SI) that is determined from the granularity settings [26]. Equation (1) shows how this interval is calculated. As a measure, we set the lower bound (lb) to collect data every 120 seconds and the upper bound (ub) to collect data every 600 seconds. We also consider a battery level of 40 percent (%) as the critical battery threshold ($batt\_crit\_th$), below which the sampling interval is recalculated at the device level and sent to the SDN-core which is middleware that manages core events between the data and control planes of the framework.

$$SI = ub - battery\ (\frac{ub - lb}{batt\_crit\_th}) \tag{1}$$

where SI is the Sampling Interval, ub is the upper bound, lb is the lower bound, $battery$ is the percentage of battery remaining and $batt\_crit\_th$ is the critical battery threshold.

Context data collected includes battery remaining in percentage, Received Signal Strength Indicator (RSSI) in dBm, the number of tasks currently being processed by the node and the number of neighbors the node has in its locality of reference. We use powertraceK [27] which is an extension of powertrace in Contiki based on a near realistic kinetic battery model (KiBam) to monitor the battery usage and related energy parameters. Fig. 4 shows the structure of the context report packet.

The context data is stored in a context table in the controller which forms a unified knowledge-base and is accessible to the MSI and associated management modules.

2) The MSI function: As a modularity enabler, the MSI function is shown by the state diagram in Fig. 5 uses classification criteria to determine which management module should handle the task. Once the MSI receives the management task, it goes into the classification state which determines which module to handle the task. Based on set parameters the right management module is selected and the task sent to it otherwise the classification state flags an unknown and exits. With the case of the resource allocation module implemented in SDNMM, tasks such as data flow requests which occur when a node requests action on a received task and source routed data flow setups flag the resource allocation module in

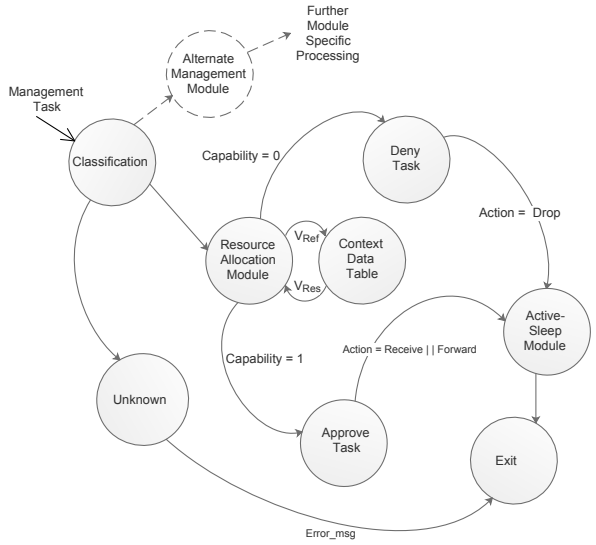


Fig. 5. MSI state diagram.

the classification state. In both tasks, it is necessary to check whether the destination node address is capable of handling the set task based on the available resources. The allocation module sets a verification reference (V_{ref}) to the context data knowledge-base and awaits response (V_{res}) which is used to determine capability. If the node is capable the task action is approved otherwise it is denied and the task is dropped. The active-sleep module state has been included to cater for the nodes ON and OFF state during sending down of management policies. This module can further be modified to work with a sleep-scheduling algorithm. A code snippet of this MSI function is shown in Table I and the pseudo code for the resource allocation module function in Table II.

3) Management system implementation: The overall program structure of our SDNMM implementation and operation can be summarized by the block diagram shown in Fig. 6. The block diagram shows all the core components of the SDNMM framework namely the SDN-enabled node (also referred to as an enabled node), SDN-core, the controller node, and the controller-PC. The enabled nodes and controller node communicate via a radio link while a serial link maintains data communication between the controller-PC. The controller-PC is a Qt-based front end handling application plane protocols including monitoring, routing, and processing of data and control flow setup and or request packets. The controller node handles events between the controller-PC and the data plane, for instance, the storage of context data upon reception of a context report packet. The SDN-core plays an important role in initiating neighbor/ controller discovery and context collect processes. It also handles events to send neighbor and context reports whenever an event request is posted to it. The process of engaging protocols for sending, receiving and enqueuing of packet data in the southbound region of the SDN architecture is also handled by the SDN-core. Each

TABLE I
MSI CODE IMPLEMENTATION

```
uint8_t msi( uint8_t * packet_ptr ) {
 uint8_t SI;//Collect rate
 management_state=0;
 nxtstate=0;
 exit_flag=0;
 mgt_action=0;
 management_state = CLASSFICATION;
 while(exit_flag == 0){
    switch(management_state){
     case CLASSFICATION :
      nxtstate = classVerify(packet_ptr);
      management_state=nxtstate;
      break;
     case DENY :
      mgt_action = drop();
      nxtstate = ACTIVE_SLEEP_MODULE;
      management_state=nxtstate;
      break;
     case APPROVAL :
      mgt_action = approve();
      nxtstate = ACTIVE_SLEEP_MODULE;
      management_state=nxtstate;
      break;
     case RES_MANAGEMENT_MODULE :
      printf("resources management state \n");
      nxtstate = Res_mgtModule(packet_ptr);
      management_state=nxtstate;
      break;
     case ACTIVE_SLEEP_MODULE:
      printf("checking on/off state of dest node");
      //refer to on/off state context data
      if (state==ON) send_down()
      else wake();
      exit_flag =1;
      break;
     case UNKNOWN :
      exit_flag = 1;
      break;
     default:
      printf ("Unknown management event.\n");
      exit_flag = 1;
      break;
        } //case
    }  // while
 return mgt_action;
}//msi
```

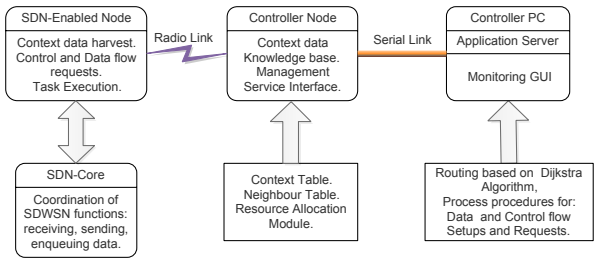


Fig. 6. SDNMM implementation program structure.

enabled node is responsible for the provision of context data and execution of sensing tasks and handling of messages based on flow table rules.

VI. SDNMM EVALUATION

We set up a series of simulations to evaluate the performance of the resource allocation module on top of the IT-SDN platform in the SDNMM framework with several iterations to

TABLE II
RESOURCE MANAGEMENT MODULE PSEUDO CODE

```
input: pointer to task destination sensor node
output: next state based on capability and
        collect rate (SI)
//get destination node address and set as
verification reference (Vref)
When a node is registered on a particular task
in a flow setup configuration, check the node
resource capability by referring to context
table information using node address as
search criteria.
   node address = SDN_HEADER(pointer)->source
   Vres = context_table_get(node address)
   //function to
   access context knowledge base.
//retrieve verification response with node
//resource information.
Get the battery level, given the critical
battery threshold, the upper and lower
sampling interval (SI) calculate the
suitable sampling interval using equation (1)
if (battery_level<=crit_batt_thresh)
    calculate new SI
//endif
//Calculate node capability based
//on pending tasks and neighbors available.
if (resources >= set requirements)
   capability = True   else
   capability = False
//set management action (Policy) based on node
//capability to handle task
if (capability==True)
    next state = approval else if (capability==False)
    next state = deny
    recalculate capable route.
//endif
```

TABLE III
SUMMARY OF SIMULATION SETUP SPECIFICATIONS

Parameter	Value Specification
Controller	Sky mote
SDN-enabled nodes	Sky mote
Transceiver	CC2420
Transmit power	0 dBm (1 mW)
Node to node distance	25 m
Node transmit range	60 m
Node alignment	linear
Cooja simulation speed	200 %
Protocol	IEEE 802.15.4 and IEEE 802.11

observe and ensure that outliers contributed to less than 5% of the results. The first set included three (3) ten (10) minute simulations running at 200% speed, each with 16, 25 and 36 nodes respectively in a grid topology. In this first setup we do not compare with SDN-WISE [18] as it exhibits scalability problems due to the high control traffic associated with it [28].

This first simulation set yielded the results necessary to evaluate packet delivery rate (PDR), packet delay in milliseconds and control traffic while having each node send a test message to the controller. In IT-SDN this message is sent periodically every 120 seconds while in SDNMM this message is sent aperiodically based on the Sampling Interval (SI) shown in equation (1). Table III shows the overall simulation specification.

Results were obtained while running baseline IT-SDN (original SDWSN platform) firmware and also while running SDNMM firmware (IT-SDN + SDNMM) then evaluated by performing a comparison to analyze the effect of building SDNMM on top of the IT-SDN platform. The results obtained

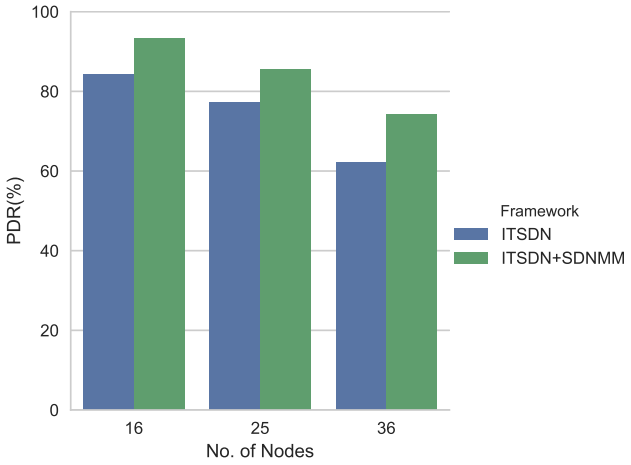


Fig. 7. Packet Delivery Rate

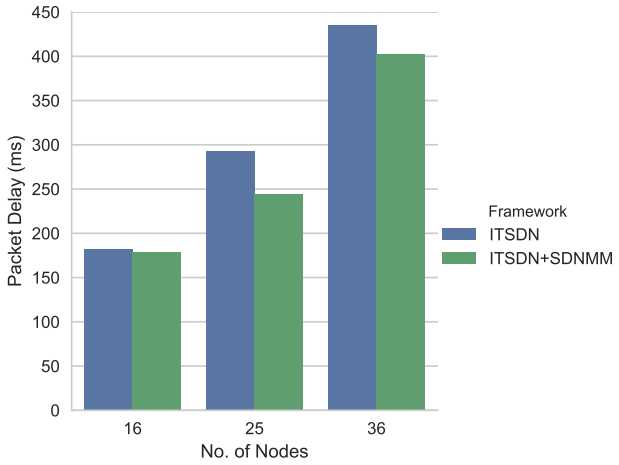


Fig. 8. Data Packet Delay

are presented in Figs. 7, 8 and 9 respectively.

The second simulation set was used to evaluate the impact on energy consumption of implementing SDNMM compared to SDN-WISE and baseline IT-SDN in a 16 nodes grid setup using powertraceK [27] as an energy measurement tool.

The energy model used is based on four power modes: transmit (E_{tx}), listen (E_{rx}), CPU (E_{cpu}) and LPM (E_{lpm}). E_{tx} represents the energest type time in ticks when the radio is transmitting, E_{rx} the energest when the radio is listening, E_{cpu} represents the active computational power consumption also in ticks and E_{lpm} represents the energest type time for low power mode operation. Given the energy mode current values and voltage rating for the transceiver in the specification sheet, the individual state energy consumption in Joules is shown in equations (2), (3), (4) and (5). Dimensional analysis of the equations results in the unit watt-second which is a derived unit of energy equivalent to a joule. Equation (6) shows the resulting total energy consumption.

$$W_{tx} = \frac{E_{tx} I_{tx} V}{RTIMER\_sec} \tag{2}$$

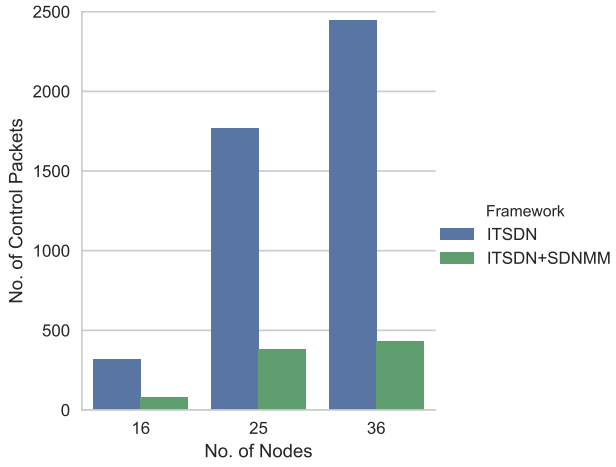


Fig. 9. Total control packets generated

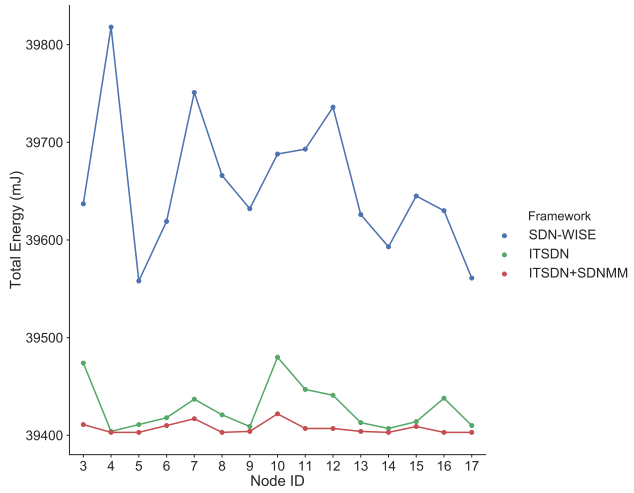


Fig. 10. Energy consumption analysis

$$W_{rx} = \frac{E_{rx} I_{rx} V}{RTIMER\_sec} \tag{3}$$

$$W_{cpu} = \frac{E_{cpu} I_{cpu} V}{RTIMER\_sec} \tag{4}$$

$$W_{lpm} = \frac{E_{lpm} I_{lpm} V}{RTIMER\_sec} \tag{5}$$

$$W_{total} = W_{tx} + W_{rx} + W_{cpu} + W_{lpm} \tag{6}$$

where:

$W_{tx}, W_{rx}, W_{cpu}, W_{lpm}$ represent energy consumption in transmit, listen, active cpu and low power mode states in joules respectively and W_{total} is the total energy in joules

V is the rated voltage of the transceiver in volts.

$I_{tx}, I_{rx}, I_{cpu}, I_{lpm}$ are the rated currents of the transceiver in amperes during transmit, listen, active cpu and low power mode operations respectively.

$RTIMER\_sec$ is the energest type tick frequency in ticks/second.

Fig. 10 shows the plot of energy consumption in that period.

The resulting computational overhead of SDNMM was evaluated by analyzing the computation cost as a percentage of the total energy cost in that time period. As depicted in Fig. 11, SDNMM shows a reduced computation cost compared to both SDN-WISE and IT-SDN furthermore, it also contributes to less than 1% of the total energy cost which is mostly due to radio communications.

The last set of simulations were used to perform an analysis on the effect of varying network topologies on the performance of SDNMM compared to SDN-WISE and ITSDN. 16 nodes were placed in a grid, mesh and ring topology in this evaluation and the packet delivery, delay and number of control packets measured in each case. Figs. 12, 13 and 14 show these results respectively.

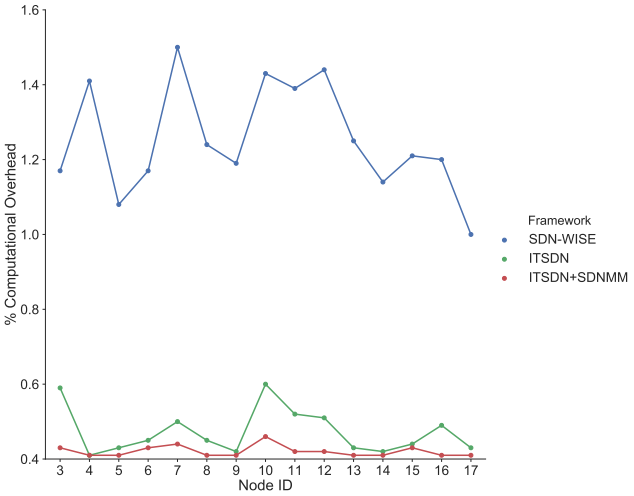


Fig. 11. Computational overhead analysis

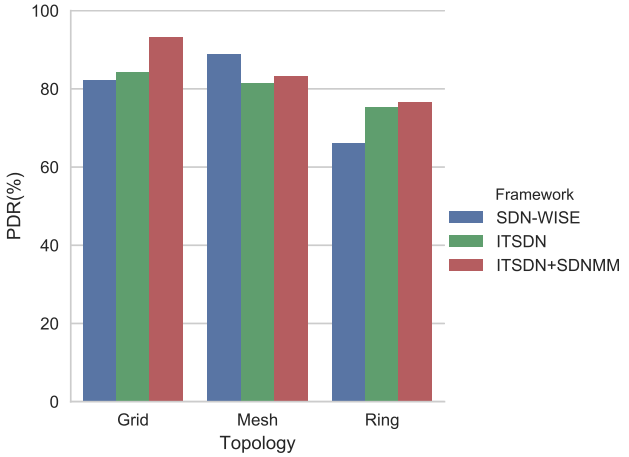


Fig. 12. Topology packet delivery rate

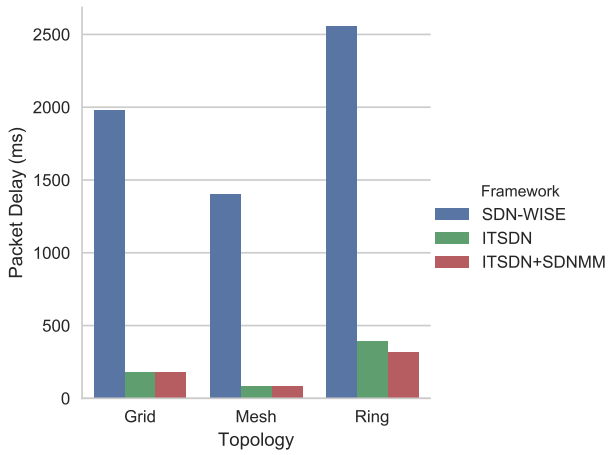


Fig. 13. Topology packet delay

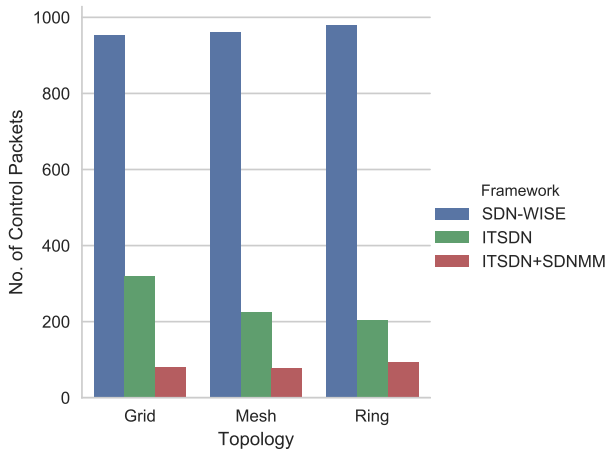


Fig. 14. Topology control traffic

VII. Discussion

The results obtained show that implementing SDNMM improves the performance of the IT-SDN platform it is developed on in several key aspects. We also observe significant performance improvement over a comparable SDN-WISE framework setup. The modular management being proposed allows the implementation of a resource allocation module capable of adapting the packet rate based on resources available and also improve task allocation based on node capability. The direct result of which is reduced congestion which is reflected by the improvements in delay and control traffic levels. These ultimately result in improved packet delivery rates. Firstly, in Fig. 7 SDNMM shows a notable improvement in packet delivery rate even as nodes are increased from 16 nodes to 25 nodes and then to 36 nodes mainly due to reduced traffic congestion hence fewer packet losses due to collisions. Figs. 8 and 9 respectively show a significant reduction in packet delay and control traffic compared to IT-SDN in each case. Predetermined actions based on context in the resource allocation minimize delay and resulting control overhead. In comparison to SDN-WISE, the high control traffic associated with the framework hindered the simulation of 25 and 36 nodes for the set time period as the network activity would exceed allowed memory usage at those network sizes.

Figs. 10 and 11 depict the energy consumed by each SDN-enabled node in the 16-node grid setup and the resulting computational overhead respectively. The graphs show that SDNMM results in a decreased energy consumption compared to IT-SDN as all points reflecting SDNMM energy consumption fell below that of IT-SDN only. The data sampling enabled in the MSI was mainly responsible for this increase in energy efficiency. We also notice that this is a promising phenomenon of the resource allocation module considering no dedicated energy management module was integrated and the extra energy consumed from context report generation did not impact the overall energy consumed extensively. Compared to SDN-WISE in the same 16 node grid setup, SDNMM consumed less energy in the same time period showing an improvement in energy efficiency. This improvement is also mainly due to the decreased traffic generation and transmission rates leading to a lesser period of active cpu, transmit and receive time which consume most of the node energy resources.

The SDNMM resource allocation module also showed improved performance in different topology setups. In Figs. 12, 13 and 14 the adaptive data rate based on battery level introduced minimized congestion leading to improvement in packet delivery rate with delay and control traffic being kept at a minimum. This resulting improvement in delay and control traffic can also be alluded to the high correlation that exists between the amount of data traffic to be processed for sending to the sink or controller and the associated control overhead and delay.

In Fig. 12 however, SDN-WISE showed around 6% improvement in packet delivery over SDNMM in the mesh topology due to the improved neighbor discovery in SDN-WISE which comes at the expense of a higher control traffic and packet delay. SDNMM still showed a considerable 83% while reducing the delay and control traffic by over 90% in the same mesh topology.

In terms of other SDN-based management systems, we can analyze and compare features as shown in Table IV. Usability as a management system is dependent on features such as the application use case, modularity, scalability, and availability of the development platform. Compared to other available SDN-based management systems, SDNMM has the advantage of providing a generic management approach based on its modularity feature that is developed on an open-source and widely available platform. SDNMM also provides an opportunity for large-scale implementation as the system framework allows for network scalability.

VIII. Conclusion and Future Work

In this paper, we have discussed in general the benefits SDN-based management of wireless sensor networks. We mentioned how SDN can solve inherent WSN management challenges due to resource constraints and also how the global network view maintained by SDN can improve network management. In this regard, a novel generic SDN-based

TABLE IV
FEATURE COMPARISON OF SDN-BASED MANAGEMENT SYSTEMS

System	Application	Modularity	Scalability	Platform
Smart [20]	Generic	NO	NO	-
Soft-WSN [21]	Device and topology	NO	NO	Proprietary
6LowPAN [22]	Energy	NO	NO	Open-source
Shsec [29]	Smart home security	YES	-	Proprietary
SDNMM	Generic	YES	YES	Open-source

modular management system to yield the benefits of SDN and modularity in managing wireless sensor networks was presented, implemented and discussed in detail. The benefits in terms of packet delivery, reduced packet delay, and energy consumption introduced by a sample resource allocation management module in the system framework have been shown.

Future work will involve the implementation of SDNMM on distributed controllers together with the use of information fusion techniques [30] specifically flow-rule aggregation [31] in a bid to further reduce the resulting control overhead. We also intend to add a dedicated energy management module with power management functions and an energy efficient routing mechanism to further reduce the energy consumption from generating context data. In addition, we expect machine learning methods to be used in the collection of context data more intelligently and efficiently [32] based on management module requirement. Context data could take any form beyond that collected in this paper, for example, a security management module could request the encryption status of data in the data plane as context and make decisions based on it. It is worthwhile to mention here that with management modularity in place, concepts based on adjusting transmission power [22] and transmission ranges to improve network lifetime can be built as flexible modules based on the context data. As other SDWSN development platforms become more available, the SDNMM system framework will be implemented on them and the resulting effects evaluated. In general, the SDNMM system has introduced a new niche in IoT management requiring the development of various modules for security, topology and QoS management for instance, all with the end goal of improving the efficiency of network performance and reliability.

REFERENCES

[1] G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz, "A survey on 5g networks for the internet of things: communication technologies and challenges," *IEEE Access*, vol. 6, pp. 3619–3647, 2018.

[2] J.-H. Shin and D. Park, "A virtual infrastructure for large-scale wireless sensor networks," *Computer Communications*, vol. 30, no. 14-15, pp. 2853–2866, 2007.

[3] I. T. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: a survey and taxonomy," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2016.

[4] M. Ndiaye, G. P. Hancke, and A. M. Abu-Mahfouz, "Software defined networking for improved wireless sensor network management: A survey," *Sensors*, vol. 17, no. 5, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/5/1031

[5] W. L. Lee, A. Datta, and R. Cardell-Oliver, "Network management in wireless sensor networks," *Handbook of Mobile Ad Hoc and Pervasive Communications*, pp. 1–20, 2006.

[6] Z. Sheng, H. Wang, C. Yin, X. Hu, S. Yang, and V. C. Leung, "Lightweight management of resource-constrained sensor devices in internet of things," *IEEE internet of things journal*, vol. 2, no. 5, pp. 402–411, 2015.

[7] Y. Qiu and M. Ma, "Secure group mobility support for 6lowpan networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1131–1141, 2018.

[8] L. B. Ruiz, J. M. Nogueira, and A. A. Loureiro, "Manna: A management architecture for wireless sensor networks," *IEEE communications Magazine*, vol. 41, no. 2, pp. 116–125, 2003.

[9] H. Song, D. Kim, K. Lee, and J. Sung, "Upnp-based sensor network management architecture," in *Proc. ICMU Conf*, 2005.

[10] T. M. Cao, B. Bellata, and M. Oliver, "Design of a generic management system for wireless sensor networks," *Ad Hoc Networks*, vol. 20, pp. 16–35, 2014.

[11] S. W. Pritchard, G. P. Hancke, and A. M. Abu-Mahfouz, "Security in software-defined wireless sensor networks: Threats, challenges and potential solutions," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 2017, pp. 168–173.

[12] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017.

[13] K. M. Modieginyane, R. Malekian, and B. B. Letswamotse, "Flexible network management and application service adaptability in software defined wireless sensor networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 4, pp. 1621–1630, 2019.

[14] H. I. Kobo, G. P. Hancke, and A. M. Abu-Mahfouz, "Towards a distributed control system for software defined wireless sensor networks," in *Industrial Electronics Society, IECON 2017-43rd Annual Conference of the IEEE*. IEEE, 2017, pp. 6125–6130.

[15] R. Alves, D. Oliveira, G. Nez, and C. B. Margi, "It-sdn: Improved architecture for sdwsn," in *Proceedings of the XXXV Brazilian Symposium on Computer Networks and Distributed Systems, Belem, Brazil*, 2017, pp. 15–19.

[16] T. Luo, H.-P. Tan, and T. Q. Quek, "Sensor openflow: Enabling software-defined wireless sensor networks," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[17] B. T. De Oliveira, L. B. Gabriel, and C. B. Margi, "Tinysdn: Enabling multiple controllers for software-defined wireless sensor networks," *IEEE Latin America Transactions*, vol. 13, no. 11, pp. 3690–3696, 2015.

[18] 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 *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 513–521.

[19] T. Theodorou and L. Mamatas, "Coral-sdn: A software-defined networking solution for the internet of things," in *Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017 IEEE Conference on*. IEEE, 2017, pp. 1–2.

[20] A. De Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," in *Communications (QBSC), 2014 27th Biennial Symposium on*. IEEE, 2014, pp. 71–75.

[21] S. Bera, S. Misra, S. K. Roy, and M. S. Obaidat, "Soft-wsn: Software-defined wsn management system for iot applications," *IEEE Systems Journal*, 2016.

[22] F. F. J. Lasso, K. Clarke, and A. Nirmalathas, "A software-defined networking framework for iot based on 6lowpan," in *Wireless Telecommunications Symposium (WTS), 2018*. IEEE, 2018, pp. 1–7.

[23] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *29th annual IEEE international conference on local computer networks*. IEEE, 2004, pp. 455–462.

[24] B. T. de Oliveira and C. B. Margi, "Distributed control plane architecture for software-defined wireless sensor networks," in *Consumer Electronics (ISCE), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 85–86.

[25] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "Fragmentation-based distributed control system for software defined wireless sensor networks," *IEEE Transactions on Industrial Informatics*, 2018.

[26] N. D. Phung, M. M. Gaber, and U. Rohm, "Resource-aware online data mining in wireless sensor networks," in *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*. IEEE, 2007, pp. 139–146.

[27] A. Riker, M. Curado, and E. Monteiro, "Neutral operation of the minimum energy node in energy-harvesting environments," in *2017*

IEEE Symposium on Computers and Communication (ISCC), July 2017, pp. 1–6.
[28] N. Q. Hieu, N. H. Thanh, T. T. Huong, N. Q. Thu, and H. Van Quang, "Integrating trickle timing in software defined wsns for energy efficiency," in *2018 IEEE Seventh International Conference on Communications and Electronics (ICCE)*. IEEE, 2018, pp. 75–80.
[29] P. K. Sharma, J. H. Park, Y.-S. Jeong, and J. H. Park, "Shsec: sdn based secure smart home network architecture for internet of things," *Mobile Networks and Applications*, vol. 24, no. 3, pp. 913–924, 2019.
[30] A. M. Abu-Mahfouz and G. P. Hancke, "Localised information fusion techniques for location discovery in wireless sensor networks," *International Journal of Sensor Networks*, vol. 26, no. 1, pp. 12–25, 2018.
[31] I. Maity, A. Mondal, S. Misra, and C. Mandal, "Tensor-based rule-space management system in sdn," *IEEE Systems Journal*, 2018.
[32] B. Cheng, J. Zhang, G. P. Hancke, S. Karnouskos, and A. W. Colombo, "Industrial cyberphysical systems: Realizing cloud-based big data infrastructures," *IEEE Industrial Electronics Magazine*, vol. 12, no. 1, pp. 25–35, 2018.

Musa Ndiaye (S'17) received his BEng in Electrical/Electronics Engineering from the Copperbelt University (Zambia) in 2011. He then obtained his MSc in Microelectronic and Communications Engineering from University of Northumbria at Newcastle (United Kingdom) in 2013. He is currently a PhD student with the Advanced Sensor Networks group at the University of Pretoria (South Africa). His research interests include software defined wireless sensor networks, network management, sensor node hardware development and a wide range of Internet of Things technologies.

Adnan M. Abu-Mahfouz (M'12-SM'17) received his MEng and PhD degrees in computer engineering from the University of Pretoria, Pretoria, South Africa, in 2005 and 2011, respectively.

He is currently a Principal researcher at the Council for Scientific and Industrial Research (CSIR), a Research and Innovation Associate at Tshwane University of Technology and Extraordinary faculty member at University of Pretoria. His research interests are wireless sensor and actuator network, low power wide area networks, software defined wireless sensor network, cognitive radio, network security, network management, sensor/actuator node development, smart grid and smart water systems.

Dr Abu-Mahfouz is an associate editor at IEEE ACCESS, IEEE INTERNET OF THINGS and IEEE TRANSACTION ON INDUSTRIAL INFORMATICS. He is a member of many IEEE technical communities.

Gerhard P. Hancke (S'99-M'07-SM'11) obtained B.Eng. and M. Eng. degrees from the University of Pretoria (South Africa) in 2002 and 2003, and a PhD in Computer Science with the Security Group at the University of Cambridge's Computer Laboratory in 2008. He is an Associate Professor with the City University of Hong Kong (Hong Kong SAR). Dr. Hancke's research interests are system security, embedded platforms and distributed sensing applications.