Towards A Distributed Control System For Software Defined Wireless Sensor Networks

Hlabishi I. Kobo[1,2], Gerhard P. Hancke[3, 1] and Adnan M. Abu-Mahfouz[2]
[1]Dept. of Electrical, Electronic and Computer Engineering, University of Pretoria, South Africa
[2]Meraka Institute, Council for Scientific and Industrial Research (CSIR), Pretoria, South Africa
[3] Department of Computer Science, City University of Hong Kong, China
HKobo@csir.co.za

***Abstract*—Software Defined Networking (SDN) is a developing networking paradigm that advocates a complete overhaul of the conventional networking. SDN decouples the control logic from the data forwarding functionality; which traditionally are coupled on the network device. The coupling stifles innovation and evolution because the network often becomes rigid. Software Defined Wireless Sensor Networks (SDWSN) is also an emerging network paradigm that infuses the SDN model into Wireless Sensor Networks (WSNs). WSNs have inherent constraints such as energy, memory etc. which have been a major hindrance of their progress. The application of SDN model in WSN is set to cultivate the potential of WSNs in modern communication and to bring about the efficiency that the WSNs have not yet achieved due to their inherent constraints. SDN based networks are anchored on the central controller for functionality. As the network scale up, issues of scalability, reliability and congestion arises and for that a distributed controllers are proposed. This paper investigates the viability of a distributed control system for SDWSN. The test results conducted show that it is viable to deploy a distributed control system for SDWSN; however an improvement is needed on the efficiency.**

Keywords—SDN;WSN;SDWSN;SDN Controllers; distributed controller

I. INTRODUCTION

Wireless Sensor Networks (WSNs) have been developing rapidly in recent years as a result of the advancements in Micro Electrical Mechanical Systems. This has led to a development of smart sensors which are small, inexpensive and intelligent devices that can be utilized in various smart systems [1]–[6]. However WSNs are still plagued by numerous challenges emanating from their inherent constraints such as limited energy, processing power, bandwidth and security [7]–[12]. These challenges continue to constrain the progression of WSNs in reaching their optimal efficiency; despite the legion of active research. Software defined wireless sensor networking (SDWSN) is a new networking paradigm that emerges as a result of applying Software Defined Networking (SDN) into WSNs. The SDN approach seeks to remove the control intelligence from the sensor node to a centralized controller, leaving the sensor node with only the packet forwarding functionality. This SDN premise relieves the sensor node of the energy and processor intensive functions [13].

Most research has been focusing on reducing energy usage on the sensor nodes so to prolong the lifespan of the nodes [14], [15]. The SDN model could significantly reduce energy usage on the node. SDN have a potential to address most of the WSN inherent challenges as shown in [16]–[18]. Most of the energy intensive functions such as routing, processing and management are removed from the physical node to a central controller or to the application plane. The sensor nodes become sheer devices that are controlled from the central controller. The SDN model is envisaged to bring flexible configuration, easier implementation of new policies, better performance, innovation, automation, vendor independence, security, ad hoc management, seamless protocol upgrades and evolution [19]–[21].

The controller is very central in SDWSN as it holds the intelligence of the entire network. Some of its fundamental functionalities are flow rule generation, mapping functions and programming interfaces. The flexibility of the SDN model enables dynamic addition of functionalities. Most of the research work in SDWSN thus far employs a centralised controller. However, there is work on distributed controllers albeit for SDN particularly targeted for enterprise networks.

The centralisation of the control logic introduces potential drawbacks, detrimental to the efficiency of the network. A central controller connotes a single point of failure and a potential target from adversaries. The controller could also be overwhelmed with rule setup and other requests from the sensor nodes resulting in an excessive overhead. Another drawback with a centralised controller is with regards to access; the distance between the network devices and the controller can negatively affect the performance of the network if not managed properly. Therefore a distributed control system for the SDWSN model could alleviate the highlighted challenges as well as some of the WSN inherent challenges.

This paper investigates the viability and efficiency of employing a distributed control system on SDWSN. The rest of the paper is organised as follows. Section II provides an overview of the SDWSN model. The rationale behind distributed control system is discussed in section III. Section IV reviews the literature pertaining to distributed controllers in SDWSN. The evaluation methodology is discussed in section V. Results and discussion follows in sections VI and VII respectively. Section VIII concludes the paper and also discusses future work.

II. SDWSN Overview

The SDN model consists of an application plane, a control plane and a data plane. The application plane host various applications such as routing, security, load balancing etc. The control plane consists of a controller (or controllers) responsible for the control and management of the network [22], [23]. The data plane consists of network elements such as routers/switches or sensor nodes. The three planes are connected via interfaces (APIs); Northbound interface between application and control planes and Southbound between control and data planes. An additional West/Eastbound interface is used in a distributed control framework to enable the interaction between the controllers.

The application of the SDN model in WSNs involves reorganising the different functionalities along the SDN planes. Fig. 1 [24] depicts the basic functionalities of SDWSN as applied by various research works. Policies are defined on the application plane and passed onto the controller which evaluates them and converts them into rules. The controller also ensures the consistency of the rules to avoid conflicts. The rules are passed onto the data plane for implementation. Packets that do not match any rule on the sensor node's flow tables are passed to the controller which makes a new rule or discards the packet.

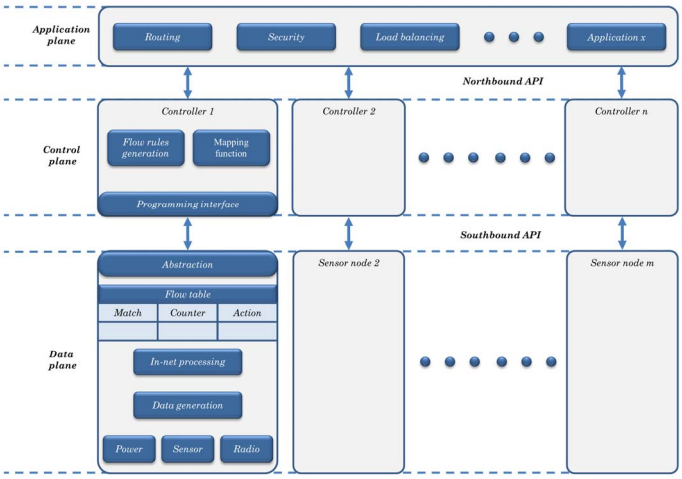

Fig. 1. Basic SDWSN architecture [24].

The sensor nodes consist of a hardware and software components. The hardware consists of a power unit, sensing unit and radio [25]. The hardware performs IEEE 802.15.4 functionalities as specified for LR-WPAN [26] and is made of the Physical layer (PHY) and MAC layer. The software component consists of a flow table, sensor, in-network processor and an abstraction layer. The sensing unit is used for sensing purposes e.g. temperature. The flow tables store rules set by the controller and the abstraction layer provides an interface for communication [24].

The SDWN architecture proposed by Costanzo et al. [18] comprises of a generic node and a sink node. The generic node consists of PHY, MAC and Network Operating System (NOS) while the sink node consists of a generic node and a controller. To enable flexible configuration and altering of parameters, a reconfigurable PHY layer is proposed [27]. The software component of the sensor node plays a huge role in the processing of the sensed data and the packet routing functionalities. The in-network processing is carried out differently by various researchers; Several works propose data aggregation [16], [18], [28] and information fusion [29]. A comprehensive survey on SDWSN is discussed in details in [24].

III. Rationale of Distributed Control System in SDWSN

Distributed control systems have been around for some time; they only change in form, scope and application. Thus the rationale behind a distributed application/computing is commonly applicable across the spectra of applications. However there are traits that are unique and exclusive to a particular form or application of distribution.

The SDN's centralisation of the control logic necessitates the need for distribution. The fundamental reasons for distributing the control logic are mainly to avert central point of failure (addressing reliability and security), congestion, delay, scalability and load balancing. The centralisation essentially means the whole network relies on the controller for functionality and if compromised the whole network will also cease to function. Also, it becomes a potential target for malicious attacks; to the detriment of the whole network.

The rationale behind the distributed control for SDWSN follows upon these facts and furthermore, also addresses WSN's unique challenges. Unlike enterprise SDN networks, SDWSN have inherent limitations in energy, memory, data rate etc. Thus the distribution criterion of the SDWSN has to take all these factors into consideration. Another distinctive factor with SDWSNs is the fact that they are used to capture real time events which changes rapidly and are thus delay sensitive. The sensed data could be rendered redundant if it does not arrive at the controller on time. The delay could be as a result of a variety of factors such as the distance between the sensor nodes and the controller, the limited data rate and the frequency of the data transmission. The distributed control is ideally suited for such scenarios as it shortens the distance to the controller, shares the load and ensures faster response time.

IV. Related Work

The SDWSN is a new and emerging network paradigm arising by infusing SDN model into WSN. This paradigm is new hence most work focuses on the single controller. However there exists few research works which have considered distributing the control logic of the SDWSN.

TinySDN [30] is one of the earliest distributed control solutions for SDWSN where multiple controllers are used with the aim of reducing control traffic. It consists of a sensor node and a sink node which is connected to a controller on a serial port. The SDN sensor node must first find a controller to belong using Collection Tree Protocol (CTP). This solution was tested using two controllers. The authors extended this work in [31] where they considered the use of a spotled architecture for the controllers. This architecture is both distributed and hierarchical; it consists of local nodes in a

distributed fashion as well as a global controller overseeing all the local controllers. All these solutions used simulation for validation testing.

SDN-WISE [28] presents a thorough solution for SDWSN. The SDN-WISE platform is stateful, meaning changes in the network affects the state of the nodes and subsequently decision are made based on the current state thereby minimising the traffic towards the controller [32]. The SDN-WISE platform allows developers to use their own controllers implemented in languages of their choice. This is because the architecture includes a virtualisation layer, WISE-visor. The WISE-visor is used as an abstraction which can handle different controller types. The solution was initially tested using a java controller implementing Dijkstra's algorithm. They also used ONOS, which is a popular SDN controller for enterprise networks. In this case an enterprise network and a WSN were simulated for heterogeneity using the ONOS controller [33].

V. Evaluation Methodology

The purpose of this paper is to evaluate the viability of using distributed controllers for SDWSN. The generic hypothesis of distribution, coupled with the rationale behind distributing the SDWSN's control logic is very profound; however its efficiency needs to be qualified. The paper uses the current solutions for this evaluation and thus the SDN-WISE with ONOS was used [33]. ONOS is a cluster-based distributed SDN controller. SDN-WISE authors customised it to work with SDN-based sensor nodes. An adaptation was also made into COOJA simulation tool.

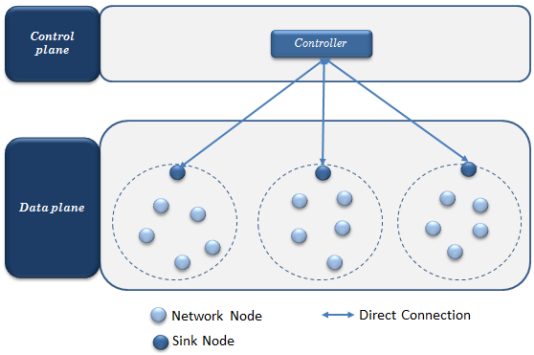

Fig. 2. Simulation experiment with one controller.

In ONOS, clusters can be formed out of one or more ONOS instances. Each ONOS instance is a controller. The ONOS instance (controller) can also operate alone; this uses a module called ONOS-trivial. When more than one instance are utilised in a clustered form, ONOS-core is used instead. ONOS is modular and based on the OSGi Java framework. The ONOS subsystem is built using OSGi's Apache Karaf container. To create a cluster, ONOS make use of two scripts in *onos-form-cluster* and/or using *test cells*. The onos-form-cluster script merges two or more ONOS instances into a cluster. A test cell is a controller cluster environment for testing purposes. Fig. 2 and Fig. 3 depict the architecture of the experiment. The first experiment uses one controller (ONOS-trivial) whereas the second experiment uses the test cell method to create a cluster of three controllers. The infrastructure elements are the same in both experiments: three sink nodes with fifteen (15) sensor nodes. On experiment one, the simulation and the controller are run from the same Virtual Machine (VM1). The VM consists of 2GHz CPU and 2G RAM. In the second experiment, three VMs (VM2-4) were used to create a cluster of three controllers; the first VM (VM1) was also used to run the WSN simulation. The simulation VM consists 2GHz CPU and 2G RAM while the other two VMs consist of 2GHz CPU and 1G RAM.

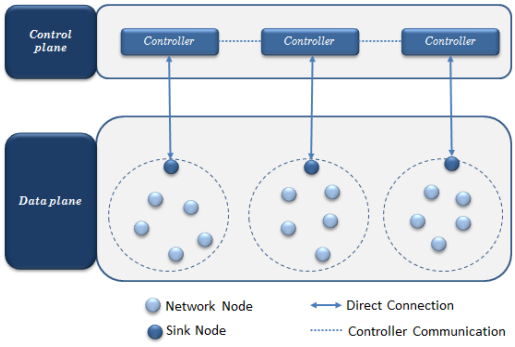

Fig. 3. Simulation experiment with distributed controllers.

The experiments are further categorised into different variations (scenarios: A, B, C and D) as follows.

- **Scenario A** is based on experiment one, where the WSN simulation and the single controller are hosted locally in one VM (VM1) as per Fig. 2.
- **Scenario B** is based on experiment two, where the WSN simulation is hosted in one VM (VM1) while the three distributed controllers are hosted remotely in three VMs (VM1-3).
- **Scenario C** is also based on experiment two. However, the WSN simulation and one of the controllers is hosted in one VM (VM1) while the other two controllers are hosted in VM3 and VM4.
- **Scenario D** is based on experiment one, using a single controller. However, the WSN simulation is hosted in VM1 and the single controller is hosted remotely in VM2.

TABLE I. THE VARIATIONS OF THE FOUR SCENARIOS.

Scenario	WSN VM	Controllers			
		Con1	Con2	Con3	Placement
A	VM1	VM1	-	-	local
B	VM1	VM2	VM3	VM4	Fully Remotely
C	VM1	VM1	VM3	VM4	Partially Remotely
D	VM1	VM2	-	-	Fully Remotely

The essence of this cross referencing is to evaluate using different physical space which is ideally close to real environment. Table I highlights the scenarios in a tabular form which shows which VM hosts the WSN simulation and the controller across the each scenario. The controller placement

column distinguishes whether a controller is local (coupled in one VM with WSN simulation) or remotely (WSN simulation and controller hosted in different VMs).

Fig. 4 shows the details of the simulation in Cooja; the three sink nodes and the 15 sensor nodes are displayed on the Network window. The Simulation control window is used to control the simulation and also time the simulation using a simulation script. The radio communication window displays the communication flow amongst the sensor nodes, most notably from the sinks to the sensor. The PowerTracker window shows the radio duty cycles of the nodes.

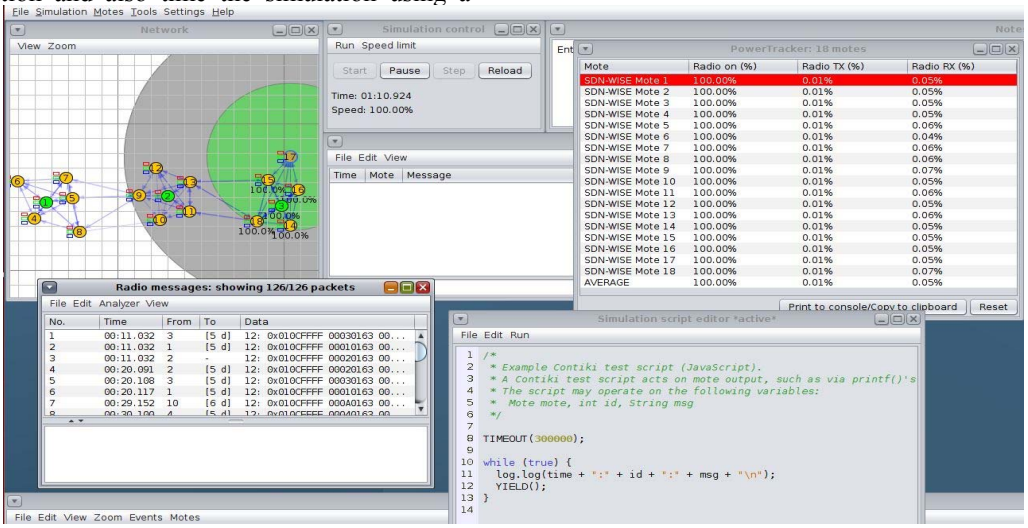

Fig. 4. Simulation in Cooja.

VI. Results

The sink nodes communicate with the other sensor nodes using the IEEE 802.15.4 medium and on the other hand communicate with the controller using the Contiki IP stack whilst they also act as gateways. The IP stack is based on IPv6's 6LowPAN. The sink nodes connect with the controller on their IPv4 interface; thus there is an internal IP tunnelling. As a result each sink node assumes the IP address of the host VM.

T-shark is used to capture the packets between the sink node and the controller. The sink node communicates with the controller on port 6633 or 9999. This is clearly distinguishable with the cross controller communication packets which use port 9876 as per Hazelcast specification [34].

A. Controller setup time

We gather the controller setup time (in nanoseconds) by observing the SYNC packet, this is the first packet that is sent to the controller for handshaking, we take the difference between the time the packet was sent and the time it was returned. Since each sink node connects to the controller, we observe the packets transmission on all sink nodes.

Scenario A entailed a VM with the Cooja simulation and also hosting the controller (ONOS instance). The infrastructure elements consists of 3 sink nodes compassed by (or controlling) 5 sensor nodes each. In this case the sink connects to the controller via the localhost or the IP address.

Scenario A took about 2.50E+01 ns for the first sink to establish a connection with the controller. The second and third sink nodes took 1.15E+01ns and 1.28E+01 ns to establish a communication respectively. There was about 1 second disparity between the SYNC packets of the two sink nodes; mainly because the process of connecting to the controller involves manual entering of the IP, not automated. On scenario B, the simulation was tested with the distributed cluster of controllers instead of a single controller as used in the scenario A. The controllers were hosted remotely instead. In this case, each sink was connected to one of the controllers in the cluster on a 1-1 mapping. The first sink took 1.02E+03 ns to establish a connection while the second and the third took 5.30E+02 ns and 7.80E+02 ns respectively. The first connection took longer in this scenario than in the previous one (scenario A). The average connection time amongst the three sink nodes is also higher in this scenario.

Scenario C is almost similar to the scenario B. The Cooja simulation is ran from a VM containing one of the controller nodes making up the distributed controller cluster. The three sinks in this case took 5.21E+02 ns, 5.09E+02 ns and 5.10E+02 ns to establish a connection with the controller respectively.

The Round-Trip Time (RTT) recorded in this scenario is relative and they are also almost similar to the RTT of the second sink node in scenario B. The major difference is the fully remote connection in B and the partially remote connection in C.

The fourth scenario, D is a variation of scenario C, with the simulator sitting in one of the VMs containing the cluster nodes as in scenario C but connecting to a single controller sitting in a VM independent of the cluster. This case is a cross inverse of scenario A, difference being scenario A was ran locally. The

first sink in this case took 5.00E+02 ns while the second and the third took 5.58E+02 ns and 5.12E+02 ns respectively. The results are relatively similar across the three sink nodes. As compared to its inverse scenario A; this setting has a higher response time.

The deduction thus far is that the single controller setting has the fastest response time compared to the cluster setting. Also noticeable is the fact that the first connection of the sinks takes longer compared to the others. Scenario A and C had fastest response time partly because the simulation was ran locally.

B. Number of packets

The simulations were run for five minutes (300 seconds) using a testing script in Cooja. The timing was consistent in all the simulations.

Scenario A produced 505 packets in total for the 300 seconds, meaning a total packet exchange of 253. Scenario B produced 508 packets, with 254 packet exchanges. Scenario C produced 341 packets with 171 packet exchanges. Scenario D produced 505 packets, making up 253 packet exchanges, ran with one controller.

C. RTT and Standard deviation

The RTT was calculated by taking the difference of times between a sent packet and the returned packet. The average RTT for scenario A was 1.05E+01 ns. On scenario B, C and D, the average RTT was 6.37E+02 ns, 4.15E+02 ns and 1.01E+03 ns. Scenario A has the fastest response time. In this case the simulation and the controller were run locally in one machine. The RTTs of the scenario B and C were relatively similar, with C slightly faster because one of the cluster nodes contains the simulation, partially remote.

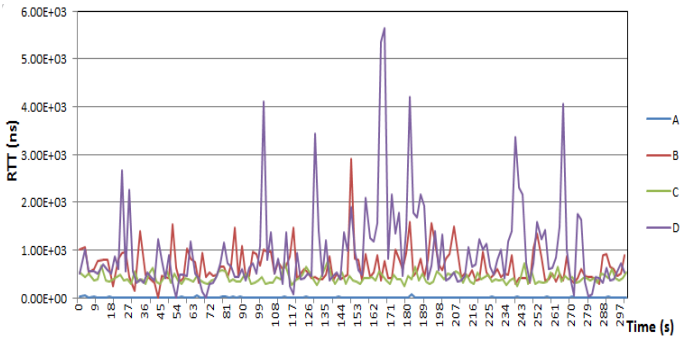

Fig. 5. RTT over 300 seconds of the four scenario A, B, C and D.

The standard deviation is defined as a measure to quantify a variation of data. It is used herein to measure the variation of the RTTs over the prescribed testing period of 300 seconds. The standard deviation measured on scenario A, B, C and D was 8.38E+00 ns, 3.13E+02 ns, 1.02E+02 ns and 9.43E+02 ns respectively. The RTT and the standard deviation assist the evaluation in determining the delay of packets round trip and the variation of different RTT's over time. Fig. 5 depicts a graphical representation of the RTTs for all scenarios over time. In Fig. 6, the average RTT and the standard deviation of all tests are represented.

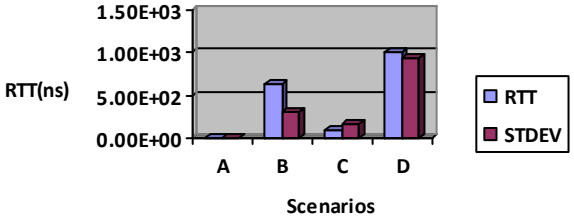

Fig. 6. Average RTT and Standard deviation for the experiments.

VII. Discussion

The main purpose of these experiments was to evaluate the relationship between the controller and the sink nodes and therefore the results are based on the control packets. There are two major distinctive factors on the experiments; the single or distributed control and the remote or local control. Remote controller is most ideal as it is not far from a real employment, however the tests above showed that it fell short compared to a locally run controller but nonetheless the delay was still within an acceptable standard. The main reason behind this disparity is solely on the basis of proximity. The RTT and the standard deviation can assist to determine the most acceptable delay threshold. The tests also showed over the two experiments that the single controller was a better performer over the scenarios tested. There are numerous contributing factors to this. If the number of nodes and the amount and size of the packets increases, a different outcome could be reached. These factors could not be tested adequately. These factors require an improvement of the sensor nodes capacity to handle such conditions.

VIII. Conclusion and Future Work

This paper evaluated the viability of a distributed control system for Software Defined Wireless Sensor Networks (SDWSN). SDWSN is a new networking paradigm that seeks to improve the efficiency of WSNs by leveraging the SDN model. Distributed control solutions have been proposed for SDN enterprise networks but few for SDWSN. The SDN model centralises the control intelligence of the whole network; although this abstraction brings lots of positive benefits, it also introduces several challenges. The entire network could be at risk if the central controller gets compromised. Failure of the controller could also negate the availability of the network thereby rendering it unreliable. A centralised controller is not ideal for a wireless sensor network, more so considering the inherent challenges such as unreliable links, low bandwidth etc. Performance and efficiency will also suffer as the network grows. This paper evaluated this viability using SDN-WISE [28], [33] solution, which also uses ONOS controller.

The experiments carried show that it is viable to distribute the control logic of the SDWSN. However as per the results, the question of efficiency is still lacking as far as comparison with a single and central controller is the benchmark. Thus the

results obtained are relative and would suite large networks. There is a need of improvement both on the ability of the infrastructure devices to handle large amount of data as well as the structure of the distributed control system. Thus in order to realise scalability, reliability and performance in SDWSN, an efficient distributed control system is needed.

REFERENCES

[1] A. M. Abu-Mahfouz, Y. Hamam, P. R. Page, K. Djouani, and A. Kurien, "Real-time Dynamic Hydraulic Model for Potable Water Loss Reduction," *Procedia Eng.*, vol. 154, no. 7, pp. 99–106, 2016.

[2] A. M. Abu-Mahfouz, T. Olwal, A. Kurien, J. L. Munda, and K. Djouani, "Toward developing a distributed autonomous energy management system (DAEMS)," in *Proc. of the IEEE AFRICON 2015 Conference on Green Innovation for African Renaissancce*, 2015, pp. 1–6.

[3] M. J. Mudumbe and A. M. Abu-Mahfouz, "Smart water meter system for user-centric consumption measurement," in *IEEE 13th International Conference on Industrial Informatics* , 2015, pp. 993–998.

[4] T. M. Chiwewe, C. F. Mbuya, and G. P. Hancke, "Using Cognitive Radio for Interference-Resistant Industrial Wireless Sensor Networks: An Overview," *IEEE Trans. Ind. Informatics*, vol. 11, no. 6, pp. 1466–1481, Dec. 2015.

[5] B. Cheng, L. Cui, W. Jia, W. Zhao, and P. H. Gerhard, "Multiple Region of Interest Coverage in Camera Sensor Networks for Tele-Intensive Care Units," *IEEE Trans. Ind. Informatics*, vol. 12, no. 6, pp. 2331–2341, Dec. 2016.

[6] K. S. E. Phala, A. Kumar, and G. P. Hancke, "Air Quality Monitoring System Based on ISO/IEC/IEEE 21451 Standards," *IEEE Sens. J.*, vol. 16, no. 12, pp. 5037–5045, Jun. 2016.

[7] J. Louw, G. Niezen, T. D. Ramotsoela, and A. M. Abu-Mahfouz, "A key distribution scheme using elliptic curve cryptography in wireless sensor networks," in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, 2016, pp. 1166–1170.

[8] N. Ntuli and A. M. Abu-Mahfouz, "A Simple Security Architecture for Smart Water Management System," *Procedia Comput. Sci.*, vol. 83, no. 4, pp. 1164–1169, 2016.

[9] A. M. Abu-Mahfouz and G. P. Hancke, "Evaluating ALWadHA for providing secure localisation for wireless sensor networks," in *IEEE AFRICON Conference*, 2013, pp. 1–5.

[10] A. G. Dludla, A. M. Abu-Mahfouz, C. P. Kruger, and J. S. Isaac, "Wireless sensor networks testbed: ASNTbed," in *Proc. of the IST-Africa 2013 Conference*, 2013, pp. 1–10.

[11] A. M. Abu-mahfouz, L. P. Steyn, S. J. Isaac, and G. P. Hancke, "Multi-level Infrastructure of Interconnected Testbeds of Large-scale Wireless Sensor Networks (MI T-WSN)," *in Proc. of the International Conference on Wireless Networks— ICWN '12*, 2012, pp. 445–450.

[12] B. Silva and G. P. Hancke, "IR-UWB-Based Non-Line-of-Sight Identification in Harsh Environments: Principles and Challenges," *IEEE Trans. Ind. Informatics*, vol. 12, no. 3, pp. 1188–1195, Jun. 2016.

[13] K. M. Modieginyane, B. B. Letswamotse, R. Malekian, and A. M. Abu-Mahfouz, "Software defined wireless sensor networks application opportunities for efficient network management: A survey," *Comput. Electr. Eng.*, 2017.

[14] A. M. Abu-Mahfouz and G. P. Hancke, "ALWadHA Localisation Algorithm: Yet More Energy Efficient," *IEEE Access*, 2017.

[15] A. M. Abu-Mahfouz and G. P. Hancke, "Localised Information Fusion Techniques for Location Discovery in Wireless Sensor Networks," *Int. J. Sens. Networks*, 2017.

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

[17] F. Olivier, G. Carlos, and N. Florent, "SDN Based Architecture for Clustered WSN," in *9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2015, pp. 342–347.

[18] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks: Unbridling SDNs," *Proc. - Eur. Work. Softw. Defin. Networks, EWSDN 2012*, pp. 1–6, 2012.

[19] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[20] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[21] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and Haiyong Xie, "A Survey on Software-Defined Networking," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 1, pp. 1–1, 2014.

[22] A. De Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," *2014 27th Bienn. Symp. Commun.*, pp. 71–75, 2014.

[23] 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. 5:1031, pp. 1–32, 2017.

[24] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements," *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

[25] B. Silva, R. M. Fisher, A. Kumar, and G. P. Hancke, "Experimental Link Quality Characterization of Wireless Sensor Networks for Underground Monitoring," *IEEE Trans. Ind. Informatics*, vol. 11, no. 5, pp. 1099–1110, Oct. 2015.

[26] I. Howitt and J. Gutierrez, "IEEE 802.15.4 low rate - wireless personal area network coexistence issues," *2003 IEEE Wirel. Commun. Networking*, vol. 3, no. C, pp. 1481–1486, 2003.

[27] M. Jacobsson and C. Orfanidis, "Using Software-defined Networking Principles for Wireless Sensor Networks," in *In: Proc. 11th Swedish National Computer Networking Workshop Karlstad, May 28-29, 2015*, 2015, pp. 1–5.

[28] 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 *IEEE Conference on Computer Communications* , 2015, pp. 513–521.

[29] R. Huang, X. Chu, J. Zhang, and Y. H. Hu, "Energy-Efficient Monitoring in Software Defined Wireless Sensor Networks Using Reinforcement Learning : A Prototype," *Int. J. Distrib. Sens. Networks*, vol. 11, no. 10, 2015.

[30] B. T. de Oliveira, C. B. Margi, and L. B. Gabriel, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," in *IEEE Latin-America Conference on Communications (LATINCOM)*, 2014, pp. 1–6.

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

[32] P. Di Dio, S. Faraci, L. Galluccio, S. Milardo, G. Morabito, S. Palazzo, and P. Livreri, "Exploiting state information to support QoS in Software-Defined WSNs," in *Mediterranean Ad Hoc Networking Workshop* , 2016, pp. 1–7.

[33] SDN-WISE, "Controlling heterogeneous networks using SDN-WISE and ONOS." [Online]. Available: http://sdn-wise.dieei.unict.it/docs/guides/GetStartedONOS.html. [Accessed: 26-Apr-2017].

[34] Hazelcast, "Hazelcast the Leading In-Memory Data Grid - Hazelcast.com." [Online]. Available: https://hazelcast.com/. [Accessed: 02-May-2017].