IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 12, 2017, accepted February 6, 2017, date of publication February 8, 2017, date of current version March 13, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2666200

A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements

HLABISHI I. KOBO[1,2], ADNAN M. ABU-MAHFOUZ[2], AND GERHARD P. HANCKE[1,3]

[1]Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa
[2]Meraka Institute, Council for Scientific and Industrial Research, Pretoria 0081, South Africa
[3]Department of Computer Science, City University of Hong Kong, Hong Kong

Corresponding author: H. I. Kobo (hlabishik@gmail.com)

ABSTRACT Software defined networking (SDN) brings about innovation, simplicity in network management, and configuration in network computing. Traditional networks often lack the flexibility to bring into effect instant changes because of the rigidity of the network and also the over dependence on proprietary services. SDN decouples the control plane from the data plane, thus moving the control logic from the node to a central controller. A wireless sensor network (WSN) is a great platform for low-rate wireless personal area networks with little resources and short communication ranges. However, as the scale of WSN expands, it faces several challenges, such as network management and heterogeneous-node networks. The SDN approach to WSNs seeks to alleviate most of the challenges and ultimately foster efficiency and sustainability in WSNs. The fusion of these two models gives rise to a new paradigm: Software defined wireless sensor networks (SDWSN). The SDWSN model is also envisioned to play a critical role in the looming Internet of Things paradigm. This paper presents a comprehensive review of the SDWSN literature. Moreover, it delves into some of the challenges facing this paradigm, as well as the major SDWSN design requirements that need to be considered to address these challenges.

INDEX TERMS Software defined wireless sensor networks, software defined networking, wireless sensor networks.

I. INTRODUCTION

The development of smart sensors in recent years has given traction to the advancements of wireless sensor networks. Wireless sensor networks (WSN) consist of micro-sensors capable of monitoring physical and environmental factors such as temperature, humidity, vibrations, motions, seismic events, etc. The sensor nodes are small, inexpensive, and intelligent [1] owing to the drastic improvement in Micro Electrical Mechanical Systems (MEMS) development. The emergence of the Internet of Things (IoT) paradigm has augmented the scope of WSNs demand, further cultivating the ongoing research in this field. IoT is a network of smart objects interconnected through a communication medium [2].

WSNs are expected to play a significant role in IoT, since the sensor nodes are the main building blocks of this concept [3]. An estimated 50 billion devices are envisioned to be connected to the network by 2020 [4], and most of them will be equipped with sensors and actuators. Thus WSNs will be pivotal to the efficacy of IoT.

WSN consists of sensor nodes deployed in a structured or unstructured manner over a chosen area of interest. A typical sensor node consists of a power unit, radio, sensing unit, and a processing unit [5]. Networking these nodes presents several challenges due to device constraints, e.g. limited computational capability, energy, data storage and communication bandwidth. Thus WSNs in their current state would not be able to meet the demands of the IoT unless appropriate solutions to these challenges are found. Many studies have identified SDN as a potential solution to the WSN challenges, as well as a model for heterogeneous integration.

Software defined networking (SDN) is an emerging network paradigm that separates the control logic from the network device (switch or sensor node), leaving the device with only data forwarding functionality. The SDN model addresses most of the challenges besetting WSNs, especially the energy constraint, which is a determinant of the network lifespan. In SDN, most of the energy intensive functions are removed from the physical node to a logically centralised controller. The nodes become devices with no intelligence as functions such as routing, major processing and management are handled at the controller or application level. SDN also enables flexible network management, which is also a key

element in IoT. SDN is also regarded as a way of introducing formal methods into networking [6]. Formal methods are mathematical techniques used to manage complex computer systems through abstraction and modularity. The application of SDN in WSN gives rise to Software Defined Wireless Sensor Network (SDWSN).

A Software Defined Wireless Sensor Network (SDWSN) is a new emerging paradigm for Low-Rate Wireless Personal Area Networks (LR-WPAN). It is realised by infusing the SDN model into a WSN. The SDN model has been applied in a variety of enterprise solutions i.e. data centres, network function virtualization (NFV) and enterprise networks. NFV is another concept closely related to SDN which virtualises network functionalities for flexible provisioning, deployment and management.

The application of SDN principles in WSN is envisaged to cultivate a latent efficiency of WSN, which despite the hype that came with it about a decade ago is yet to be realised. Most of the challenges are attributed to the limited resources that WSNs possess. Also, WSNs have always been considered to be application specific [7]. This results in WSN being prone to resource underutilization. For example, two or more WSNs are deployed for specific applications in an overlapping physical space, while a single WSN could have achieved the same objectives. SDWSN is a good approach to improve efficiency and sustainability of WSNs and to foster interoperability with other networks; it is also envisaged to play a critical role in the looming internet of things.

The application of SDN has intensified over the years as different network and computing platforms seek to seize its benefits. SDN has received significant attention in networks such as mobile networks, enterprise networks, wireless networks, optic networks; and computing models such as cloud computing, fog computing, mobile cloud computing, mobile edge computing in an effort to enable the Internet of Things. There are several survey papers for both the WSN and SDN respectively, but not the combination. WSN surveys dates a while back and have been improved with time and can be found in [1] and [8]–[10]; more recently Khan *et al.* [11] reviewed the prospect of WSN virtualization. Surveys in [12]–[18] provides a comprehensive review of SDN in enterprise and wireless networks, whereas Li and Chen [19] research pertains to SDN and NFV. Surveys in [20]–[26] review the application of SDN in cloud, fog, mobile cloud, and mobile edge computing. The surveys in [22] and [27] review SDN application in IoT, furthermore Bizanis and Kuipers [28] review the conjunction of SDN and NFV for IoT. These surveys highlight the state of the art work pertaining to SDN in a holistic view of the IoT narrative; however, a review of the current state of SDN in WSN (SDWSN) does not exist yet. To the best of our knowledge, currently there is no survey which deals comprehensively with the concept of SDWSN. However, there are few papers, such as [3], [7], and [29]–[31], which specifically deal with some individual components of SDN that could be applied to an SDWSN composition. Haque and Abu-Ghazaleh [32] conduct general review of SDN based wireless networks including Cellular Network, WSN, Wireless Mesh Network and Home Networks. In contrast, this paper discusses the imperative role of SDN in WSN; other related concepts are also analysed conjunctively with SDWSN in respect of IoT for future inter-networking. A comprehensive analysis of different aspects of SDWSN is undertaken where design requirements are extracted. These design requirements are amended with some of the research challenges identified and presented for future consideration; to our knowledge, this is the first review paper to do so.

In this paper, we present a comprehensive survey on the emerging Software Defined Wireless Sensor Networks. We start by providing an overview of the SDN model in section II. This is followed by current challenges in WSNs in section III, and the potential importance of SDN in addressing WSN's inherent challenges in section IV. In section V, we discuss other related networking concepts. Section VI presents a review of the five major aspects of SDWSN: architecture, routing, network management, security and standardisation. Different kinds of controller implementations are reviewed in section VII. Future research challenges and major design requirements for SDWSN are highlighted in section VIII and section IX respectively. Section X concludes the paper.

II. OVERVIEW OF SOFTWARE DEFINED NETWORKS

Software defined networking (SDN) is a new networking paradigm that aims to simplify network management and configuration. SDN offers a complete paradigm shift from traditional networking. It seeks to greatly improve network efficiency through high level novel abstractions. SDN decouples the network intelligence, the control plane, from the packet forwarding engine, the data plane. The separation enables a provision of centralized network intelligence at the controller, which has a global view of the network [33]. SDN introduces benefits such as vendor independence, heterogeneous network management, reliability and security not possible in traditional networks [12], [13], [16], [34], [35]. While SDN was initially earmarked for large-scale enterprise networks, it has the potential to impact on any networked system. The rest of this section briefly discusses some of the major aspects of SDN, namely the architecture, protocols, standards, applications and security.

A. ARCHITECTURE

Traditional networks, which typically consist of routers and switches as network devices; become difficult to monitor and upgrade as the network grows, thus stifling growth. Large networks also become heterogeneous due to the use of different proprietary protocols, which fundamentally means they consist of different network islands that only cooperate at lower levels of communication [13], [33], [36]. This makes it difficult to implement any policy changes, upgrades, and patches. Traditional networks are also mostly hierarchical, tree based and static, which leads to what most have termed

TABLE 1. Common SDN protocols.

Protocol	Standard Body	Communication Protocol	Security	Determinants	API Interface	IP	
Forces	IETF	SCTP	Secure channel: TLS	LFB components	Southbound	IPv4	
OpenFlow	ONF	TCP	Secure channel: IPsec	Match fields/Actions	Southbound East/Westbound	IPv4	IPv6

"ossification" [13]. Ossification refers to a phenomenon of conforming to the conventional way of networking where everything is coupled on the network device.

A SDN network typically consists of a centralised control plane and highly dispersed data plane (depending on the deployment). The control plane houses the decision making intelligence of the network, responsible for control and management [33]. After the process of decoupling the routers become more like data forwarding switches, with routing decisions made by the controller within the control plane. The controller enables ad hoc management, easier implementation of new policies, seamless protocol upgrades or changes, global visualization and the avoidance of middle-boxes such as firewalls, load balancers, intrusion detection systems, etc. Recent research shows that the deployment of middle-boxes is growing on par with network routers [37]. The control plane also offers a high level of abstraction and provides an interface to the application plane.

The data plane on the other hand is responsible for packet forwarding. The data plane devices store the rules that guide them on how to handle the packets when they receive them. The rules are implemented and enforced at the control plane.

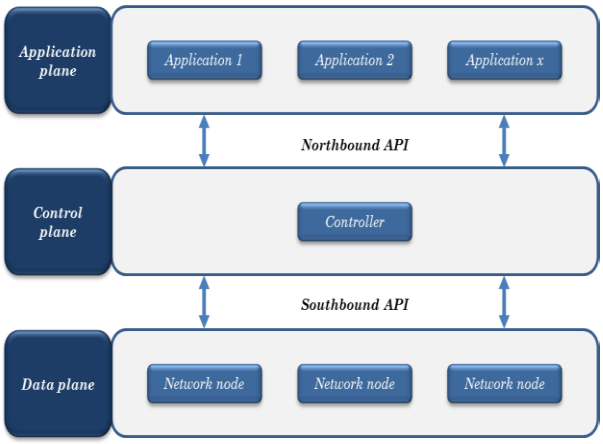

FIGURE 1. The basic SDN framework with the three planes and a central controller.

The SDN architectural framework consists of three layered components, as shown in Fig. 1. These components are interconnected by various APIs. The first component is the application plane. The application plane interfaces with the various network applications. The second component is the control plane, which houses the control software. The last component is the data plane (infrastructure plane). This consists of the network devices. The communication between the control and the application planes is defined by APIs referred herein as the northbound (NB) interface, while the southbound (SB) interface refers to the communication between the control and the data plane.

B. PROTOCOLS AND STANDARDS

To achieve a high integration of heterogeneous networks, the protocols in between the architectural planes need to be standardised. Recent research has focused on the development of both the southbound and northbound API standardisation, with more focus on the southbound API [13]. The standard development organisations are detailed in [12]. The two most popular southbound interface specifications are Forwarding and Control element separation (ForCes) [38], [39] and OpenFlow [40]–[42]. They both conform to the principle of decoupling the control and the data plane, but are fundamentally different [13]. Table 1 below compares the two protocols. ForCes, which is developed by the Internet Engineering Task Force (IETF) working group, consists of two components the Forwarding Element (FE) and the Control Element (CE). The FE handles the packets while the CE executes control and signal functions and also sends instructions to the FE on how to handle the packet. ForCes use the concept of a Logical Function Block (LFB), which resides inside the FEs. The LFB has a specific function (such as routing) to process the packets [43], and enables the CE to control the FE [13].

OpenFlow, which is developed by the Open Network Foundation (ONF), is by far the most common southbound interface. Although ForCes is deemed more powerful and dynamic than OpenFlow, the latter's penetration and adoption in the industry has upstaged ForCes on many fronts [44]. Some literature even deem OpenFlow the principal [14] and de facto [13] protocol for SDN networks. This paper focus more on OpenFlow due to its prevalence and influence on SDN based developments. Before the standardisation of ForCes, OpenFlow was the only standardized protocol that allowed direct manipulation of the data plane by the controller [43].

The OpenFlow protocol is flow based, thus each switch maintains a flow table (which can be altered dynamically by the controller), which consists of flow rules (entries) that determine the handling of packets [30], [38], [41]. Flow entries mainly consist of match fields, counters and instructions/actions. The match field entry is used to match the incoming packets. The match determinants are the packet header, ingress port and metadata [40]. Counters collect flow

statistics such as the number of received packets, number (size) of bytes and the duration of the flow. The instruction field determines what action should be taken upon a packet match [13]. When a packet is received, the header is extracted upon which the relevant fields are matched against the flow table entries. If they match, appropriate actions are applied and if more than one entry match is found, prioritization, based on the highest degree of match, applies [13], [40].

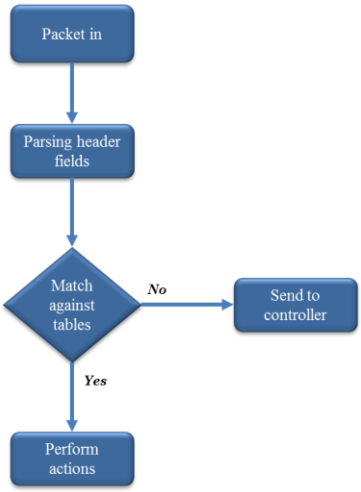

FIGURE 2. Basic packet forwarding flow in OpenFlow [40].

If no match is found, then appropriate rules are actioned, i.e. drop the packet, pass it to the next flow table or send it to the controller for new rules to be made [13]. Fig. 2 depicts the basic flow of a packet in OpenFlow highlighted by [45] and [46]. OpenFlow consists of three classes of communication: controller-to-switch, asynchronous and symmetric [13]. Controller-to-switch is used for configuration, programming and information retrieval and is from the controller to the switches. Asynchronous communication is initiated by the switch to the controller and is about packet arrivals, changes, errors, etc. Symmetric communication is send without the initiation (solicitation) from either the controller or the switch, examples of which are echo packets.

C. APPLICATIONS

SDN has been applied in varied environments such as enterprise campus networks, data centres and cloud computing services.

Enterprise networks are traditionally large and ever expanding at a rapid pace. As the networks grow, the demand for security and better performance becomes a necessity. Campus networks in particular are very dynamic and frequently require a change in policies. SDN simplifies this complex task of making changes by enabling network wide defined policies to be mapped and programmed onto underlying network devices. Another advantage in SDN networks is the elimination of middle-boxes, which can now be implemented inside the controller [13] or in the application plane, thus cutting costs of deployment and maintenance [47]. In data centres, many big companies have already implemented SDN architecture to simplify their service provision. SDN has also been used to build private wide-area networks (WAN) connecting data centres successfully, e.g. B4 by Google [48]. The issue of virtualisation and cloud computing services has also necessitated a shift from conventional data storage towards SDN-oriented data storage and management [14].

Other applications of SDN are in software defined radios, cellular networks, and wireless networks. In SDN radios, OpenRadio [49] has been proposed, which aims to enable programmability on the PHY and MAC layer [34]. Odin [50] tackles the issue of Authentication, Authorisation and Accounting in the enterprise Wireless Local Area Network (WLAN) services. SDN is also applied in cognitive radios for dynamic spectrum access [51], [52]. In cellular networks, OpenRoads [53] presents an approach of introducing SDN based heterogeneity in wireless networks for operators. In wireless networks, SDN has been proposed in wireless mesh networks [54], [55]. On campus networks SDN oriented OpenFlow networks have been implemented [54], [56], [57]. Another area which is the crux of this paper, is wireless sensor networks [14], [32], [56], [58].

D. SECURITY

The SDN paradigm presents new security challenges. By virtue of a centralised control, security concerns are introduced [59]. Two areas are potential targets [60], the centralisation of the network intelligence and the risk of controlling the network through a software amid their susceptibility to bugs and other vulnerabilities. The authors in [60] further identify seven potential security attacks on SDNs:

1) Forged or faked traffic flows.
2) Vulnerabilities in switches.
3) Control plane communication.
4) Vulnerabilities on the controller.
5) Lack of mechanisms ensuring trust between controller and the switch.
6) Vulnerabilities on the administration station.
7) Lack of trusted resources for forensic and remediation.

Ali *et al.* [36] propose a multi-pronged security response in SDN networks of threat detection, remediation and correctness to enhance security in SDNs, as well as security as a service by which anonymity methods are implemented.

Most SDN security measures consist of various verification and validation models. Fresco [61] defines an application layer and a security kernel on top of the network operating system NOX [62] controller, which interface with the security applications and ensures that their policies are implemented. NOX is the OpenFlow controller that allows management and network control applications to be written as centralized programs through a high level programmatic interface. As multiple applications get implemented, FortNox [63] was developed to alleviate potential conflicts posed by varying applications. It achieves this by setting priorities according

to different roles of authorization. Human administrators are assigned highest priority, followed by security applications, and then followed by non-security applications [63]. Accordingly, a non-security application would not be able to alter a policy or rule implemented by a human administrator.

PermOF [64] is a permission control system used to grant different permission privileges to SDN applications. The changing of network conditions affects the network verification measures. Beckett *et al.* [65] propose an assertion language based on the VeriFlow [66] algorithm to enable verification and debugging of the SDN applications with dynamic verification conditions. VeriFlow debugs faulty rules inserted by SDN applications before they harm the network. Flover [67] is another verification system that ensures that new security policies do not violate the existing security policies within the OpenFlow network. FlowChecker [68] is a configuration verification model which ensures that OpenFlow rules are consistent within flow tables as well as with other flow rules residing in other switches in a federated network. The SDN security verification solutions are explained in detail in [36] and [59].

III. WSN CHALLENGES

It is worth noting that WSNs, albeit their great potential, are yet to reach their optimal effectiveness. This is largely due to the inherent challenges that they exhibit and the ever growing scope of demand from applications. These challenges have ignited much research interest in the past decade. Some of the main challenges are listed below.

A. ENERGY

Energy conservation is central to the development of WSNs. Since sensor nodes run on a limited battery [1]; it is vitally important to use energy wisely and efficiently to significantly prolong the lifespan of the network. In other instances, the energy source could be replenished through solar and other means. However as WSNs grow, it could get difficult to replenish the power source which could lead to the complete disposition of the sensor nodes as envisioned in [10]. This would however depend on the terrain of deployment and this remains an open research and design consideration.

The main cause of power consumption in sensor nodes is largely attributed to sensing, communication and data processing. There have been several attempts in dealing with the energy question in WSNs and this section provides an overview of some of the main research directions thus far.

There are many type of applications in WSNs and each application has its own power needs [5]. The most common approach in reducing high sensing energy consumption is through sporadic sensing [5]. The sensing unit is used only on demand and is put in idle mode when not in use (inactive mode). Radio communication also consumes a huge chunk of power [8], it involves data transmission and reception. Putting the radio communication in a sleep mode when there is no data exchange efficiently saves considerable amount of energy. This is called ''duty-cycling'' [5]. The internal computation of data is another area where energy saving measures has been applied. Anastasi *et al.* [5] propose that communication should be traded off with computation, i.e. reducing the communication overhead by performing more computation. Different solutions have thus been suggested to intelligently deal with local data processing to minimise energy usage. The most common is data aggregation [1], [4] where data is internally compressed before it is sent to the controller. Another approach is to disregard redundant data from neighbouring sensor nodes [4]. Mobility-driven approaches have also been suggested where a specific mobile sensor node collects the data from static sensor nodes and sends it to the controller, thereby saving energy on the static nodes. The topology of the network also plays a crucial role in energy. Sparse placement of nodes uses a lot of energy because the communication range between nodes is long [1].

B. COMMUNICATION

Communication in WSN is through a wireless medium guided by different IEEE specifications operating under the unlicensed industrial, scientific and medical (ISM) frequency bands. IEEE defines the PHY and MAC layer for Low-Rate Wireless Personal Area Networks (LR-WPAN). IEEE 802.15.1 (Bluetooth) and 802.15.4 are the two most viable protocols for WSNs. These protocols have to coexist mutually with other wireless protocols operating on the same ISM band such as IEEE 802.11a/b/g (WLAN) and IEEE 802.15.3 (ultra-wideband: UWB). WLAN and UWB are not ideally suited for resource constraint wireless sensors. WLAN and UWB are high bandwidth wireless communication technologies for devices with high processing power and consistent or easily rechargeable power sources.

Bluetooth is a short-range wireless communication technology based on the IEEE 802.15.1 specification [69]. The earlier version of Bluetooth (Classic) had high power consumption and wasn't entirely suited for LR-WPAN devices. Bluetooth Low Energy (Bluetooth Smart) is an ultra-low power consumption protocol enhancement of the Bluetooth technology, which also increases the communication range [70]. Bluetooth uses two topologies: Piconet and Scatternet. Piconet is formed by one or more Bluetooth devices, referred as slaves connected to another Bluetooth device serving as a master. A Scatternet is a cluster of several Piconets. The only drawback with Bluetooth is with regards to its scalability.

IEEE 802.15.4 [71] was developed to address the requirements of the LR-WPAN , particularly ad-hoc wireless sensor networks. This standard was proposed specifically for networks with low power consumption, low deployment cost, less complexity and short range communication, while maintaining a simple protocol stack [1], [10]. The physical layer supports three frequencies, i.e. 2450 MHz, 915 MHz and 868 MHz [10]. The MAC layer defines two types of nodes that participate in WSN: Reduced Functional Nodes, which only act as a sensor end device, and Full Functional Nodes, which can act as both the network coordinator and network

end device. Network coordinators provide synchronisation, communication and network joining services, while end user devices are the actual sensor nodes. Table 2 below enlists the IEEE 802.15.1 and 802.15.4 specifications, which forms the basis of many other protocol standards.

TABLE 2. IEEE 802.15.4 specification [71].

	IEEE 802.15.4	IEEE 802.15.1 (BLE)
Layers	Physical	Physical
	MAC	MAC
Frequency	868/915 MHz 2.4GHz	2.4GHz
Range	10 to 20m	10m to 100m
Data rate	0.25Mbps	1Mbps
Addressing	8bit or 16bit	16bit
Devices	100+	7+

The communication range between the WSN nodes is very short, up to 10 to 20 meters [72]. The data rate is also very low, at around 250kbps. The low data rate could cause congestion problems especially in large and highly active deployments, which could affect the overall throughput and latency of the network. Although WSNs are traditionally delay-tolerant, this is likely to change with the introduction of IoT.

The ZigBee standard is built on top of IEEE 802.15.4 and defines the communication of the higher layer protocols: Network, Transport and Application. There has been a lot of work on communication with some researchers even proposing cross layer communication to save energy [1]. Despite all the great efforts, most communication protocols are yet to propel WSNs to optimal efficacy levels. The TCP/IP protocol in particular was considered too heavy for sensor nodes to handle [1], [7], which resulted in addressing challenges. The lack of addressing led to an identity problem particularly on large scale deployment until the introduction of the new IPv6 addressing for Low-Rate Wireless Personal Networks (6LoWPAN). Although that could potentially resolve the intra WSNs communication, inter communication remains a challenge in view of the heterogeneity in IoT framework.

WirelessHart is another WSN standard based on IEEE 802.15.4 for process automation and control [1], [73], [74]. ISA 100.11a is also an IEEE 802.15.4 based standard designed for low data rate wireless monitoring and process automation networks [1], [73]. 6LoWPAN standard enables the IEEE 802.15.4 based devices to communicate using IPv6 [1], [75]. Table 3 highlights some of the current IEEE 802.15.4 based standards.

C. ROUTING

WSN topologies are unstructured and therefore many traditional routing protocols are not suitable. Also, the fact that they are not IP-based makes routing a very challenging yet interesting aspect. The routing is based on the network layer as defined by IEEE 802.15.4. WSN routing protocols should be lightweight owing to the limited resources that these networks exhibit. WSN routing protocols are classified in greedy forwarding, data centric, energy oriented, localisation and flood based. Greedy forwarding forward packets to neighbours close to the destination [1]. These kind of protocols are more effective in dense deployments as opposed to sporadic/intermittent deployments [1]. Data centric protocols are attribute based, i.e. they are based on a particular attribute such as temperature, and these help in removing redundant data [76]. They normally use compression and aggregation to route packets. Flooding is a common technique in wireless networks where a node reactively broadcasts hello/control packets to its neighbours for possible route determination. It is argued that this method suffers 'implosion'', ''overlap'' and ''blindness'' [76]. Implosion is when redundant messages are received from different nodes. Overlap is when neighbouring sensor nodes, which observe the same attributes, send similar information. Lastly, they are resource blind because their routing is incognisant of the resource constraints. Energy efficient algorithm protocols will choose neighbours with high energy levels to route the packets [76]. Localization based algorithms use GPS or any other localization models [77] to localize neighbours in the network and base their routing on that.

TABLE 3. WSN standards based on IEEE 802.15.4

Protocol	Layer	Security	Standard Body
ZigBee	Network Application Transport	Link Keys	ZigBee Alliance
Wireless Hart	Network Application Transport	Payload Encryption Message Authentication	Hart Communication Foundation
ISA100a	Network Application Transport	Payload Encryption Message Authentication	ISA
6LoWPAN	Network	Access Control List (ACL) Secure mode	IETF

The transport layer protocol handles issues such as congestion, packet loss and memory capacity. All these factors, if uncontrolled, waste energy to the detriment of the network. The main goal of the transport layer is thus to minimize congestion and achieve high reliability [1].

D. SECURITY

WSNs, like other wireless networks, are susceptible to security threats. Some of the security measures and suitable cryptographic algorithms are discussed in [1]. Furthermore [72], [78], and [79] have identified the fundamental security requirements that must be reached in WSNs: *Data Authentication, Data Confidentiality, Data Integrity, Availability and Redundancy.*

To attain the above security goals, WSNs have to deal with different threats that they are susceptible to [80]. WSN attacks can be classified into three categories: goal-oriented, performer-oriented and layer-oriented [81], [82].

Goal-oriented attacks comprise of passive and active attacks. A passive attacker monitors and listens to the communication channel and gather sensitive information,

but does not interfere nor interrupt the network operation. An active attacker, in contrary, monitors, listens and modifies the data, thus interrupts the functioning of the network. Some of the active attacks are DoS, Blackhole/Sinkhole, Wormhole, Hello Flood, Sybil, Modification of data, Node Subversion, Node Malfunction, Message Corruption, False Node, Node Replication, Selective Forwarding, Spoofing and Fabrication [79], [81]–[83].

Performer-oriented attacks comprise of outside and inside attacks. Outside attacks occur when the adversary exhaust the resources of the node by injecting bogus/unnecessary packets causing a DoS. Inside attacks occur when a malicious node acts legitimate and slowly wreak havoc of the network.

Layer-oriented attacks target the different layers of the network stack [79], [82]. Physical layer attacks target the radio operation of the node either through jamming or tampering. On the Data Link layer (MAC), the attacker deliberately violates the predefined communication protocol. Network layer attacks target the operation of the routing protocol, i.e. attacks such as Sinkhole diverts all traffic towards an already compromised node. On the Transport layer, the adversary floods connection requests to a particular node to consume its resources. Attacks on the application layer include data corruption and malicious code.

Resource constraints are a major obstacle in implementing optimal security measures in WSNs [1], [82], [84], [85]. There has been a lot of research activity on security counter measures in recent times, with 'low-resource' cryptographic measures at the forefront. Symmetric key cryptography solutions are currently the most preferred, due to their lower implementation cost and time efficiency. However, there are major drawbacks in that keys are difficult to manage in large networks, with each device requiring a shared key with every other device it wishes to communicate with. Thus, if a single key is used and one node gets compromised, the entire network would be at risk. Although there have been several attempts at improving symmetric key cryptography for WSNs, as highlighted in [59], all the attempts come at the cost of processing resources. Other challenges are their resistance to scalability and difficulty to implement in software [79], [82].

Asymmetric cryptography addresses some of the symmetric cryptography drawbacks, e.g. key management would be simplified, however they are considered too heavy and computationally too expensive for the resource constrained sensor nodes [79], [82], [85]. Recently more research has focused on toning down these algorithms for WSN nodes as in [82]. The symmetric methods requires less computation but are not robust enough and on the other hand, asymmetric methods offers a potential robustness but at a cost of computation. These are subject to active research in search for optimal solutions and thus remain an open challenge. With the future direction of the WSNs seemingly converging to IoT, Alcaraz *et al.* [86] note that it is important to have a global perspective of security which do not only focus on WSN but the entire IoT framework.

E. CONFIGURATION

Manual configuration of any network device is challenging and tedious especially when the network grows. Sensor nodes need to respond swiftly to any change in the network and thus need a dynamic configuration management. Christin *et al.* [78] state that the role of sensor nodes could extend to offer autonomous functions such as self-healing, self-discovery, thus a more subtle and energy cognisant approach is needed.

IV. IMPORTANCE OF SDN IN WSN

The advancement of WSNs is thwarted by the inherent problems that they exhibit. Although much work has been done in an attempt to minimize these problems, there is not yet a holistic solution, as each focuses on a particular problem in isolation. It is thus very unlikely that these challenges could be eradicated through the same approach of algorithms and optimisations coupled with the ever changing specifications and demands of interest. The SDN approach to WSNs is envisaged to potentially solve most of the inherent WSN challenges [7], [33], [87]. The most prevalent and critical WSN problems can potentially be addressed by SDN as follows.

A. ENERGY

Energy constraint is a challenge in the development of WSNs and it is by far the most important factor for consideration in this area. Almost all research work in WSNs, one way or the other, attempts to address the issue of energy. Costanzo *et al.* [29] state that SDN in WSN should support common energy conscious measures as currently being explored in traditional WSNs such as duty cycling, in-network data aggregation and cross layer optimization. The SDN paradigm is handy because by virtue of decoupling, the forwarding (switch) nodes are relieved from much of the energy-intensive computational functionalities [33], [88]. An energy efficient sleep scheduling algorithm based on SDN is proposed in [89]. Most of these energy consuming functions will now reside in the controller, which has enough power resources. This saves a considerable amount of energy and could potentially prolong the network life. Heuristically this is so but the degree of this assertion (prolonging) is yet to be quantified and remains open for future research.

B. NETWORK MANAGEMENT AND CONFIGURATION

Network Management in wireless sensor networks is very complicated and tedious. The network management challenges in WSNs are mostly inherent from traditional infrastructure networks, which include among others provisioning, configuration and maintenance [33]. The SDN approach simplifies network management considerably with its simplicity and ability to evolve [29], [33]. In WSN, the reconfiguration and maintenance of the sensor nodes tends to be a complicated and tedious process if management is not flexible; this is also exacerbated by the environments in which WSNs are deployed. These challenges are alleviated by removing the

control logic from the sensor nodes leaving them as mere forwarding elements. These forwarding elements would now be controlled and manipulated from the centralised controller thereby enabling programmability on the physical infrastructure nodes [33]. SDN also enables a dynamic mapping configuration between the sensor nodes and the controllers (if more than one controller is used) as illustrated with TinySDN [90].

C. SCALABILITY

The scalability of the WSNs is very important, especially in the advent of the IoT, where SDWSNs are sought to play a critical role. The WSNs become cumbersome as the network grows to the detriment of efficiency. An abstraction based model of SDN would aid in keeping the topological organisation and efficacy of the network intact, thus consigns the scalability oversight of the WSN to the SDN controller. Although the SDN model traditionally relies on a central controller, many efforts and strides have been made over the years to distribute the control plane; ONOS [91], Hyperflow [92], Difane [93], DevoFlow [94], Kandoo [95], Disco [96], Pratyaatsha [97] and Elasticon [98] are some of the distributed controllers.

D. ROUTING, MOBILITY AND LOCALISATION

Mobility and localisation are critical for better routing in wireless sensor networks. Depending on the nature of deployment, there could be device mobility, which the network should be able to deal with. Normally traditional routing protocols will periodically update the routing table (proactive) or otherwise source a route on request (reactive) in an event of change. This process is energy intensive and is not suitable for WSN networks. SDN simplifies this by managing the mobility from the central controller i.e. routing decisions and policies are managed at the controller [33]. Localisation algorithms, for example [99] can also be implemented at the controller or at the application plane instead of the resource constrained sensor nodes. This will aid with the network topology discovery and subsequently better decision making [33].

E. INTEROPERABILITY

WSNs have long been said to be application specific, which leads to resource underutilisation [7]. This shortfall can be resolved by using the SDN approach. SDN alleviates the dependency from vendors by allowing infrastructure elements to be controlled from one central point, thus running one protocol on the elements albeit from different manufacturers.

F. COMMUNICATION

Physical communication is largely managed by the device, but aspects such as media access and duty/sleep scheduling could still be system wide decisions by the controller. The duty cycling functionality could efficiently be managed by the controller. SDN also enhances better control of heterogeneous network infrastructures. Thus the communication between the SDWSN and other networks could adequately be managed centrally.

G. SECURITY

The WSN security solutions discussed in section III.D above are based on the coupled architecture of the sensor nodes. The SDN's decoupling renders them undesirable for the same purpose, however these security measures could still play a vital role being implemented on the control or application plane. The centralisation of the security management simplifies the implementation and configuration of security mechanisms. This global perspective also enables proactive monitoring and evaluation, which leads to quick counter response in a event of attack, i.e a malicious node. The traditional SDN security challenges and counter measures are discussed in section II.D. These measures apply to WSN to a certain extend because WSNs have unique traits compared to traditional networks.

A fundamental importance of SDN in WSN is that the sensor node become dump element which only understands controller messages or commands. This makes it difficult and improbable to be used as conduit of malice. Another advantage is on the basis that sensor nodes are the peripheral devices on the network; unlike traditional networks where the host computers, which are peripheral could also be security targets. The SDN model also enables flexible configuration moving away from the cumbersome and error prone manual process currently in place. As noted in [35], [100], and [101], configuration errors could lead to security vulnerability.

H. SUMMARY OF DISCUSSION

SDN possesses an immense potential to improve network computing and WSNs, without an exception, also stands to benefit. This section discussed the importance of SDN in WSN with a focal point on pertinent issues, such as energy, network management, security, configuration, mobility, routing, interoperability and scalability. SDN thus indeed alleviate most of the inherent WSN challenges and could bring WSN to efficacy. However, there are issues of security due to the centralisation of the control logic which could lead to a single point of failure, as well as congestion and overheads concerns, as all decision making are centralised. These concerns are however mitigated by the provision of distributed control platforms.

V. RELATED CONCEPTS

The future of Internet computing is currently a subject of many concerted research in both the academia and industry. At the centre of this subject is the concept of cloud computing (CC), which has been largely accepted as the next generation of computing infrastructure [25]. Cloud computing in a nutshell provides data, computational, and application services to end users in the form of Data Centres, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and Hardware as a Service (HaaS).

A. THE APPLICATION OF SDN IN THE RELATED CONCEPTS

The emergence of the IoT paradigm as a vital role player in the advent of the ubiquitous connectivity of various devices onto the cloud and the Internet added more complexities which necessitates structural reconsideration. A plethora of IoT devices are envisaged to dominate the Internet space in the years to come. Thus the research community and the industry are vigorously leading the course of finding architectures and models which would support the IoT upsurge, one such an enabling model being SDN.

There has been a plethora of studies (with industrial intervention) synergising SDN with CC and IoT such as in [21], [102], [103], and [104]–[107] respectively. The major characteristics of IoT are low latency, long lived connection, location-awareness, geographical distribution, wireless access, mobility and heterogeneity [108]. In contrast, cloud computing services inherently have challenges of fluctuating latency, lack of mobility and location-awareness [24], [108]. This ushered in the need to move some of the processing closer to the devices to improve network response efficiency. Fog Computing, conceptualised by Cisco in 2012 [4], extends the cloud computing service provisioning to the edge of the network. It is regarded as a "*highly virtualised platform that provides computation, storage and network services between end devices and traditional cloud servers*" [23], [24], [108].

Although Fog computing refers to "edge" in its definition; it should not, as is often the case, be confused with Edge Computing which in essence extends Fog computing further to the end devices. Thus, it devolves some processing, analytics and decision making to end devices and also allows them to communicate and share information. Fog computing could also be applied in conjunction with SDN; Truong *et al.* [109] applied SDN with Fog computing for a Vehicular Ad Hoc Network (VANET). Fog computing gives rise to other similar concepts in Mobile Cloud Computing (MCC) and Mobile Edge Computing (MEC). MCC "*refers to an infrastructure in which both the data storage and the data processing happen outside of the mobile devices. Mobile cloud applications move the computing power and data storage away from the mobile phones and into the cloud*" [24], [25]. Thus MCC combines the concept of cloud computing and SDN on mobile phones. MEC on the other hand refers to the concept bringing cloud computing capabilities to the edge of a mobile network closer to the mobile devices for better performance [110], [111]. There are glaring similarities between Fog computing and MEC as well as MCC and SDN. An architecture infusing SDN and MCC is discussed in [112].

MCC is also envisaged to play a key role in the next generation of mobile networks. Aissioui *et al.* [64] propose a SDN controller for 5G mobile cloud management systems. The 5G next generation mobile network is envisaged to be very dense (very close to the end devices) in order to meet the demand which has been exponentially growing of recent. Bhushan *et al.* [113] discuss two methods of densification expected to be dominant in the composition of 5G networks: Densification of space and frequency. The former refers to a process of densely deploying cells to a spatial locale, called an Ultra-Dense Network [114]. More ultra-dense networks are studied in [115], while Soret *et al.* [116] expound the interference coordination of these networks. The later refers to the use of large portions of the radio spectrum. This method uses the millimetre wave frequency spectrum [117]. Ali-Ahmad *et al.* [118] and Trivisonno *et al.* [119] describe different SDN based network architectures for a 5G mobile network, while Rost *et al.* [120] discuss cloud technologies expected to bring flexibility to the 5G network, including Network Function virtualization (NFV).

It is clear from the above evolution of wireless networks that heterogeneity will play a huge role in the future of computing and the Internet. Also, the different concepts of computing are not mutually exclusive. Therefore there is a need for an inclusively harmonic coordination and corporation to efficiently deal with the data from various sources. SDN and NFV have the potential to ease this integration and thereby bring efficiency in the Internet of Everything [121].

An SDN based Vehicular Ad Hoc Network (VANET) with Fog computing is explored and implemented in [109]. The architecture used a SDN controller, sitting above the Fog but connected to the cloud. The Fog is composed of a SDN Road-site-unit controller (RSUC), SDN RSU, SDN wireless nodes (vehicles) and a cellular Base Station (BS). The communication between the controller and the RSUC and between the RSUC and RSU is through a broadband link. The vehicle nodes use long range cellular network such as 3G, WiMAX, and 4G LTE to communicate with the BS whilst they use a wireless connection or Wave to communicate amongst themselves. Anadiotis *et al.* [122] define a SDN operating system for IoT that integrates SDN based WSN (SDN-WISE) and a SDN-based Ethernet network using an ONOS controller. This experiment shows how heterogeneity between different kinds of SDN networks can be achieved.

The studies above demonstrate that different kinds of networks and systems can co-exist. Systems such as smart grid [123], smart water system [124], smart cities [125], health care [126], Vanets [109], [127] etc. are expected to grow explosively with Cisco estimating a total of 50 billion devices by 2020 [4]. Most of these systems will be equipped with sensors (mostly wireless) which collects data from the ground, as Anadiotis *et al.* [122] note that they are "*fundamental ingredients to the IoT ecosystem*". It is for this reason that this study focus on the application of SDN directly from the bottom devices. This will ensure a bottom up approach and consistent architectural application of the SDN principles.

Fig. 3 depicts the technologies envisaged to feature predominantly in the future computing of IoT in an inverted pyramid form. The technologies above the "Internet of Things" perimeter are considered carriers of the IoT, whilst those below are the building blocks of the IoT network. The SDN model should ideally be applied throughout, i.e. from the bottom devices to the top technologies. Recent studies show

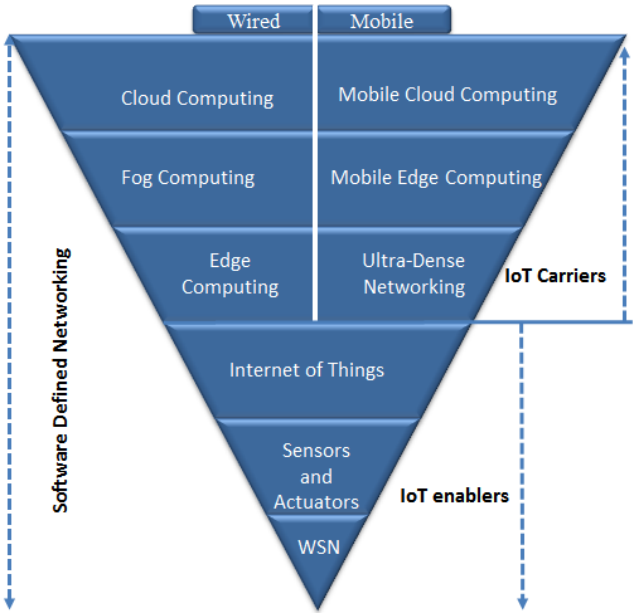

FIGURE 3. Current and future network computing technologies with SDN.

that the application of SDN has gained a lot of traction from the top pyramid technologies. In contrast, the application of SDN on IoT devices is still lacking behind. However, though progressive strides have been made; they are still very much in a development stage.

The survey study below provides a state of measure regarding the extent of SDN applications: SDN in traditional networks [12]–[16], [128], SDN in wireless networks [15], [18], [34], SDN in cloud computing [20]–[22], SDN in Fog computing [23], [24], SDN in mobile cloud computing [25], [26], SDN in IoT [22], [27]. At this stage there is no survey focusing on the bottom devices such as sensors. This paper fills this gap by surveying recent work in the application of SDN in WSN, and the set of design requirements which should be considered for the development of the SDWSN paradigm. Although some of the sensors from the IoT perspective will be wired, considerable literature show that WSNs have an extensive role to play. Vaquero and Rodero-Merino [129] concur, and further identify mobile devices and sensor/actuators as the major driving force behind the growth of IoT with the sensing devices estimated to surpass all. Also, WSNs exhibit a particular set of challenges that warrants urgent attention, particularly on the premise that SDN could potentially alleviate.

B. SUMMARY OF DISCUSSION

IoT is at the centre of future Internet computing as the connectivity of various distinct elements is intensified. As WSN are envisaged to play a pivotal role in this IoT upsurge, there are many other related concepts which are intrinsically imperative. These concepts can be summarised categorically as carriers of the IoT traffic: the wireless and wireless mobile networks (Wi-Fi, LTE, 5G, etc.); computing frameworks: cloud computing, fog computing, mobile edge computing, mobile cloud computing, as well as the building blocks: sensors and actuators. This section discussed the application of the SDN model on all these concepts, including the IoT. The relationship amongst them was also highlighted and elucidated through a diagram. Also highlighted is the need for a SDWSN review to provide an indication of the state of the art research in this field, which is lacking behind as compared to the other concepts. Thus a bottom-up approach application of SDN to the realisation of heterogeneous IoT is suggested.

VI. SOFTWARE DEFINED WIRELESS SENSOR NETWORKS

A. SDWSN ARCHITECTURES

The SDN approach to wireless sensor networks entails abstracting different functionalities and reorganising them along the three logical planes of the SDN model: application, control and data. The development of the SDWSN architecture is still in its infancy stage but valuable inroads has been made in the research fraternity. Although there are different implementations of the architectures, they all conform to the fundamentals of SDN: decoupling. Fig. 4 depicts the basic functionalities of SDWSN as applied by various researchers.

1) FORWARDING DEVICES

The sensor nodes are the basis of the infrastructure layer or the data plane. The sensor nodes comprise of hardware and software components. The hardware consists of a power unit, sensing unit and radio. The hardware components are on the Physical layer (PHY), which together with the MAC layer performs the IEEE 802.15.4 functionalities as specified for LR-WPAN [71].

a: MAC and PHY

The IEEE 802.15.4 functionalities are pertinent in the communication of the SDWSN wireless networking. The proposed architecture by Costanzo *et al.* [29], SDWN, consists of a generic node and a sink node. The generic node consists of three layers: PHY, MAC and Network Operating System (NOS). The sink node on the other hand has two parts, the generic node and the controller, which are serially connected. Jacobsson and Orfanidis [3] propose a reconfigurable PHY layer to enable flexible configuration and altering of parameters. The MAC layer handles node identification, which is inherently a major challenge in WSNs. Luo *et al.* [7] propose sensor OpenFlow, which aims to address the identification (addressing) problem which emanates from the fact that WSNs are data-centric and attribute based as opposed to the Ethernet based networks (address-centric) [1], [7]. To alleviate the identification challenge, the authors aim to redefine the addressing model by using a ZigBee 16-bit network address and using concatenated value pairs, which entail using the attribute descriptions e.g. all packets with a temperature of a certain threshold.

Another method proposed is to augment the WSN with IP addresses, which many deem inviable for sensor nodes.

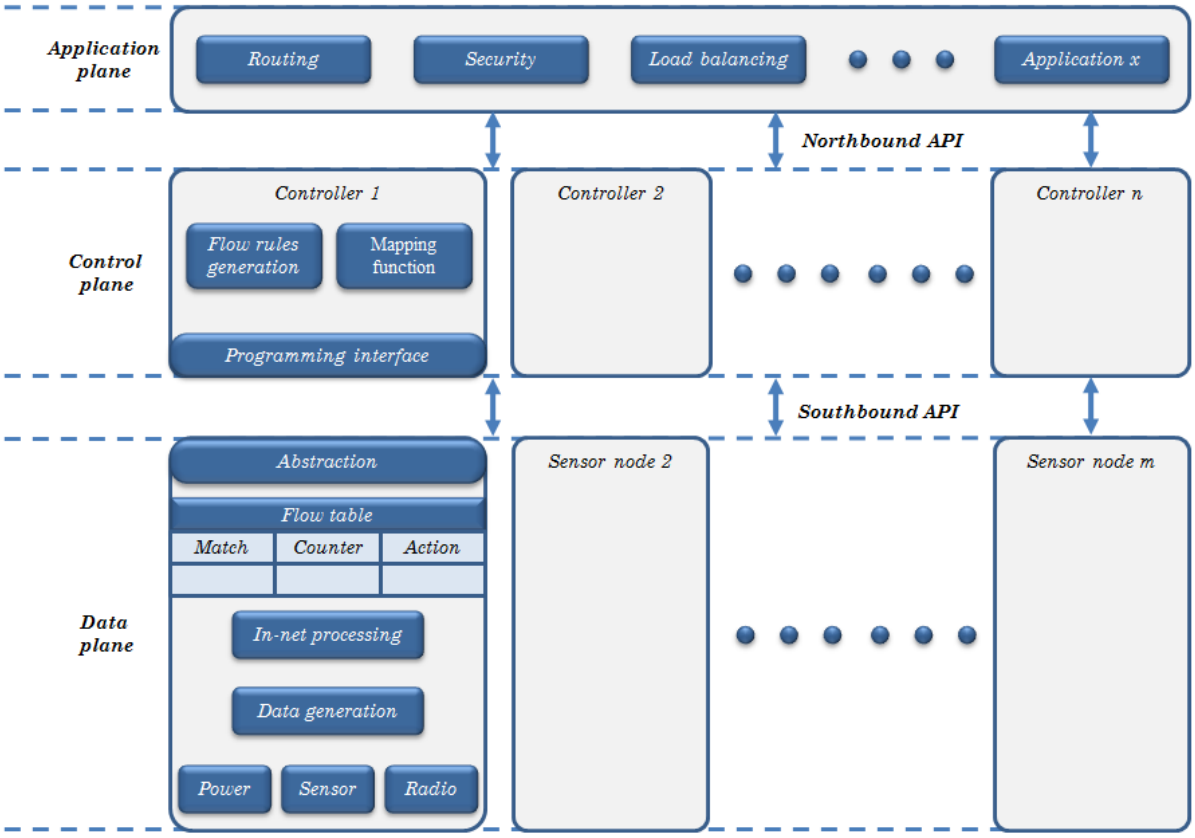

FIGURE 4. Basic SDWSN architecture as currently applied by various studies. Different functionalities are distributed along the three planes.

However several studies recently, for example [77], [130], have considered the deployment of an IP-based WSN using different technologies such as 6LoWPAN [131] (IPv6 over Low Power Wireless Area Network) protocol. Despite IPv4 being ruled of on WSNs, IPv6 based 6LoWPAN uses compression for the much constrained IEEE 802.15.4 devices. 6LoWPAN is earmarked as a pragmatic enabler of the Internet of Things [75], [130], [132]. An IP-based WSN can be considered as an alternative solution. Mahmud et al. propose Flow-Sensor [58] which aims to achieve reliability by separating the propagation of the control and data packets. The control packets use OpenFlow, while the data packets use the normal TCP/IP.

b: Data Processing

The software part of the sensor node typically consists of a flow table, sensing element, in-network processing and an abstraction layer. The sensing module generates traffic which is then passed to the processor. The in-network processing could either be data aggregation or data fusion. Ideally, aggregation would be at the sensor node while fusion takes place at the sink node. The flow tables store rules as prescribed by the controller, while the abstraction layer provides an interface for communicating with the controller. The software component of the sensor node plays a critical role in the processing of the sensed data and routing functionalities. The processing of the sensed data is carried out differently by various research works. Data aggregation is recommended for the in-network processing in [7], [29], and [133]. In the cognitive SDWSN architecture proposed by Huang *et al.* [134], information fusion is employed. They further use Over-The-Air-Programming (OTAP) [135] to distribute routing information.

2) CONTROLLER

The controller in SDWSN plays a very critical role as it holds the intelligence of the whole network. Its fundamental functionalities are flow rule generation, mapping functions and programming interfaces. The SDN model allows more functionality to be added. There are different implementations of the control plane: it could be centralised [62], [136]–[138] or distributed [91], [92], [96]. The use of centralised controllers in addition to local controllers is considered in [93]–[95]. Additionally, elastic solutions [64], [97], [139], [140] dynamically add or reduce controllers according to network load.

The SDWN architecture in [29] uses an embedded controller in a Linux system which is attached to a sink node through a serial connection. The sink node communicates with other sensor nodes on a wireless interface. The embedded system consists of an adaptation module, a virtualizer

and a controller. The adaptation module is used for message formatting. The virtualizer ensures that all information collected is collated to form a virtualisation of the network state to the controller [29], [141]. A major drawback with this approach is that the serial communication between the sink and the controller limits the controller and the sink to a one-to-one relationship. This also poses a problem of scalability. Such architectures will only be viable in a small controllable network.

In other instances, the sink is a base station (BS), which also houses the controller, as in [31] and [33]. Gante *et al.* [33] propose a base-station based SDWSN which consists of five layers: Physical, Medium Access, NOS, Middleware and Application. The middleware layer consists of a controller, mapping function, mapping information and a flow table's definition. The controller controls and manages the network and keeps the state and topology of the network up to date. It uses monitoring messages to acquire network information, such as sensor node energy levels, distance from the BS, node's neighbour list and link status parameters, such as link quality, response time, etc. The Mapping Function processes the monitored data from the sensor nodes and creates a network map (topology view). The acquired information is stored in the Mapping Information module. The purpose of the application layer is to locate the sensed area, thus consists of Location and Tracking algorithms to maintain accurate information about the node's position.

Olivier *et al.* [31] propose a cluster-based SDWSN architecture, which also has a base station. They apply the SDN model in a clustered WSN for the formation of what they refer to as a software defined cluster sensor network (SDCSN). The sensor network is organized in clusters (SDN domains) comprising of sensor nodes and a gateway or cluster head. The cluster head referred herein as SDN cluster head (SDNCH), controls and coordinate all sensor nodes in its domain. The SDNCH can implement its own security policies and thus securing the domain from outside attacks. The SDNCH has a partial view of the network and communicates with other SDNCHs through the gateway. It uses WE-Bridge [142] to exchange local data with other SDNCHs.

Huang *et al.* [134] propose a cognitive SDWSN framework to improve energy efficiency and adaptability of WSNs for environmental monitoring. The architecture employs a Reinforcement Learning method to perform value redundancy filtering and load balancing routing in order to realise energy efficiency and adaptability of WSN to a changing environment [134].

In TinySDN multiple controllers are utilised. The prototype consists of a sensor node and a sink node attached serially to a controller. The SDN sensor node first must find a controller to join using a Collection Tree Protocol (CTP). The CTP protocol is also used to send information, such as link quality to the controller. The link quality in TinySDN uses link estimator instead of RSSI (Received Signal Strength Indicator), even though they acknowledge that RSSI is more accurate, the link estimator is chosen as it is hardware independent. The framework was implemented and tested using a Cooja simulator [143].

A hybrid control model is employed in [3] where there is a main controller at the control plane and also a local controller on each sensor node. The purpose of the local controller is to setup, reconfigure, monitor and also execute instructions from the controller. The sensor nodes are also equipped with Virtual Machines (VM) which helps in installing new protocols or code patches when needed. Changes can be installed using native code and dynamic linking for homogenous networks and/or byte code with VMs for heterogeneous networks.

3) TOOLS

TinySDN is proposed in [90] and implemented as like SDWN [29]. TinySDN is based on TinyOS [144], [145], a sensor network based OS. TinySDN aims to reduce the control traffic. The TinySDN architecture consists of two nodes, an SDN sensor node and SDN controller. Sensor OpenFlow is proposed in [7], which is a communication protocol between the control plane and the data plane [6]. Sensor OpenFlow is based on OpenFlow [20], which until recently was earmarked for enterprise and carrier networks [64]. Sensor OpenFlow also aims to achieve compatibility with other OpenFlow-based sensors.

TinySDN was evaluated using a Cooja simulator. Cooja [143] is a simulator tool used to simulate sensor motes running Contiki OS. Mininet [146] is the most prevalent simulation tool used for SDN OpenFlow networks. However, Estinet [147] is purported to be better than Mininet in terms of performance and scalability though it remains proprietary [148]. Fs-SDN [149] is another SDN simulator which was developed to facilitate SDN controller application prototyping; POX [150] controller was used in development. Son *et al.* [151] developed CloudSimSDN [151] to simulate and test SDN cloud services, since Mininet and other simulators cannot simulate cloud specific features. Other simulators that support OpenFlow protocol are NS-3 [152] and Trema [153] which are implemented in C++ and C (and Ruby) respectively [154].

4) INTERNET OF THINGS ENABLING ARCHITECTURES

The SDWSN architecture proposed by Jacobsson et al. extend this novelty to the new Internet of Things (IoT) paradigm in which WSNs are envisaged to play a critical role [3], [105]. There has been more work in trying to integrate SDWSN aspects into the IoT framework. Mahmud and Rahmani [58] also extend their work to consider the use in IoT [155]. Perhaps a more complete IoT architecture is given in [106], where the authors apply SDN principles in IoT heterogeneous networks. Hakiri *et al.* [156] outline an IoT architecture that synergises SDN with Data Distribution Services (DDS) and also highlights some of the challenges pertaining to standardization, mobility, network gateway access and QoS support. Hu [107] introduces a new architectural paradigm in Industrial Internet of Things (IoT), which focus on industrial

TABLE 4. Current SDWSN architectures.

Architecture	Implementation	Controller	In-Processing	Focus
Sensor-OpenFlow [7]	Simulation Testbed	Central	√	Protocol (SB API) Identification
SDWN [29]	Proposed	Central	√	Generic
Flow-Sensor [58]	Simulation	Central		Reliability
SDWSN [33]	Proposed	Central		Management Mobility Localisation
TinySDN [90]	Simulation	Distributed		Framework Tool for multiple controllers
SDWSN[3]	Proposed	Central	√	Architecture/IOT In-network processing
Cognitive SDWSN [134]	Simulation	Central	√	Energy Efficiency Adaptability
SDCSN [31]	Proposed	Distributed	√	Controller placement
SDSN [30]	Proposed	Distributed		Sensing as a service
SDN-WISE [133]	Testbed Simulation	Distributed	√	Reducing overheads Finite state SDN-WISE

production systems. It hereby calls for a detailed feasibility study on the synergy of SDWSN with other wireless networks in the context of IoT.

Zeng *et al.* [30] propose an architecture model that combines SDWSN and cloud computing. In a cloud computing service provision, e.g. IaaS, sensing is offered as a service to different applications. A sensing request is sent to the controller, which will subsequently send the request to a suitable WSN(s) which offers that sensing service. Many WSNs exist in a physical space and SDN is used to normalise them into one integrated common network of SDWSN. The SDN model is used to enable flexible alterations to meet the dynamic sensing requests (by different applications). The sensed data can also be combined with other cloud service data for mashup services.

Comparison Table 4 highlights a compact comparison of the different architectural frameworks stated above. The implementation attribute (column) checks whether the proposed architecture was implemented (simulation or testbed) or not. The controller attribute checks whether the controller is distributed or centralised.

The in-network processing checks whether there is any form of processing on the nodes. Finally the main purpose of the architecture is listed in the focus attribute.

B. ROUTING

It is worth noting that in SDN's paradigm, the routing functionalities are logically centralised at the controller. Traditional WSN routing protocols runs on the nodes and consumes lot of energy. Such approaches in their current state are not viable for SDWSN; however their algorithmic concepts could be implemented at the controller level, thus requiring a transformation from an infrastructure-based to a software-based approach. The traditional WSN routing methods, as discussed and classified in [87] remain to be tailored for SDWSN based on SDN guidelines. The most related current work thus far is found in the routing model architecture proposed by Shanmugapriya and Shivakumar [157]. The authors combine context-aware and policy based routing modules in line with the SDN principles. Fig. 5 depicts the proposed architecture.

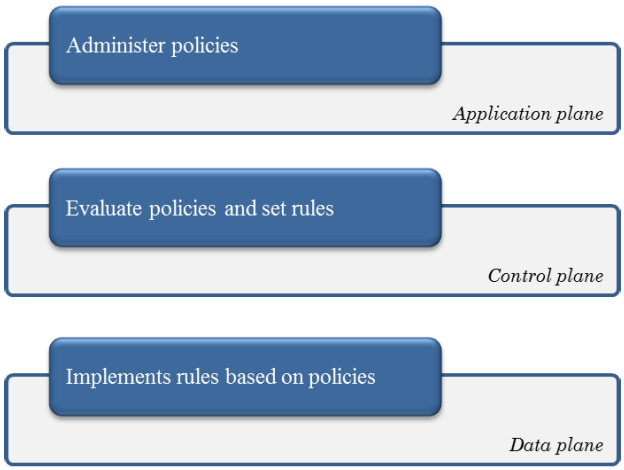

FIGURE 5. The context-aware and policy based routing model.

The model has six phases:

Initiation phase: sensor nodes connect to the controller using the Sensor OpenFlow protocol. The nodes also supply the controller with all relevant information such as node status, CPU load, etc.

Topology discovery phase: The node supplies the controller not only with its own information but its neighbours' as well. The controller forms a routing map table. The table has all the nodes and their next best hops. The route link has context information such as CPU load, service information and power levels of the next hop. Services are any policies defined for that route such as security, privacy, etc. [157].

Decision phase: The controller uses a recursive destination based lookup algorithm to look for a route in the mapping table.

Policy based route phase: The routes are chosen using defined policies. The algorithm lookup will determine all available routes towards the destination and a particular route will be chosen if it matches the policy criteria, i.e. ignore any route with a CPU usage of at least 90%. Should all routes not match the policy, packets could be dropped.

Enforcement phase: The controller enforces the routing onto the switch devices.

Han and Ren [87] propose a cluster-based routing protocol based on SDN. They set an OpenFlow oriented SDN network based on three types of nodes: master node, centre node, and a normal node, where the master node is a controller, the centre node is equivalent to a switch/sensor node (packet forwarding device) and the normal node is only for receiving data. The SDN network uses a NOX controller. They further highlight the use of FlowVisor [158], [159] for controller scalability purposes. FlowVisor is a SDN based virtualization application model, which enables multiple controllers to use or manage one switch concurrently. Each controller accesses the switch through a dedicated virtual portion called slice. Like all the OpenFlow based routing, the master node determines the route upon receiving a new request. The routing or the forwarding policy is based on the QoS information. The centre nodes maintain the flow tables. The network is arranged in clusters, with the centre nodes as cluster heads, which then communicates with the master node via a secure channel, as shown in Fig. 6. The location of the centre node is critical and thus uses a distance aware routing algorithm based on Content Addressable Network (CAN). The authors highlight limited energy on the nodes as a challenge.

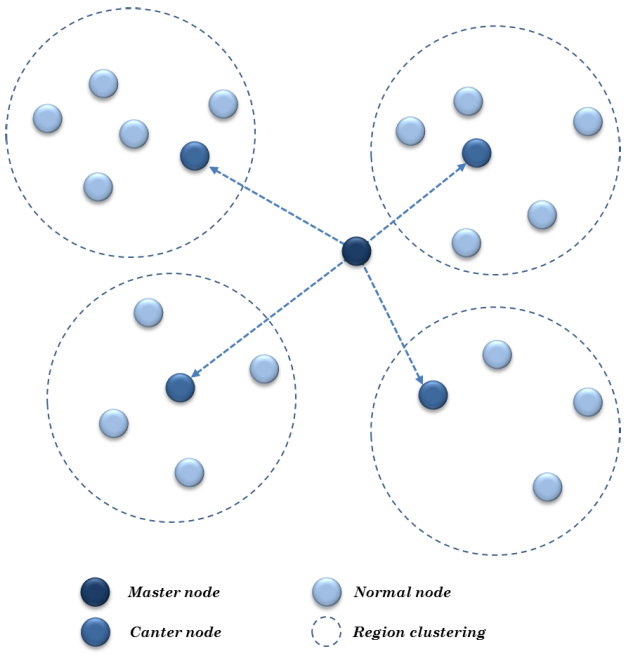

FIGURE 6. SDN cluster based routing [61].

Yuan *et al.* [160], propose a hybrid routing model which combines the ideals of an Ad Hoc On-Demand Distance Vector (AODV) [161] protocol with OpenFlow. AODV is a wireless and mobile ad hoc network routing protocol. They use an AODV daemon to implement the AODV algorithm as well as OpenvSwitch [162] as an OpenFlow agent which enables the communication between the devices and the controller [160]. Fig. 7 depicts the system architecture, which was implemented on a physical device (Raspberry Pi).

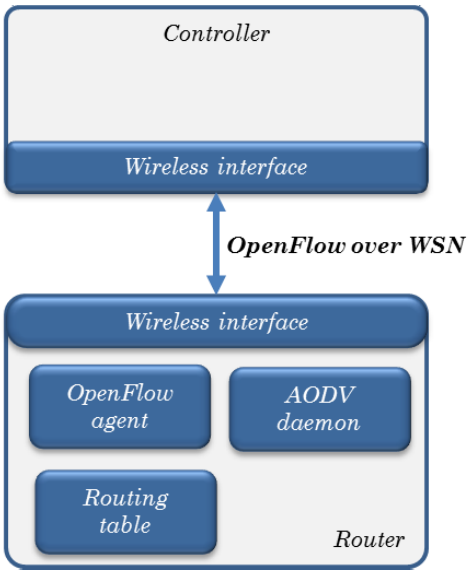

FIGURE 7. The hybrid routing model architecture [88].

TABLE 5. Current SDWSN routing protocols.

Name	Algorithm	SDN layer	Testing
Shanmugapriya et al. [157]	Context aware and policy based	Application	proposed
Han et al. [87]	Cluster based (CAN)	Controller	Matlab
Yuan et al. [160]	Link State (AODV)	Controller	tshark

Most of the current efforts lack qualitative and quantitative evidence of their efficiency in a real network environment. Their performance ought to be tested against common network factors such as latency, QoS, packet delivery, congestion etc. The current SDWSN routing protocols are listed in Table 5.

C. NETWORK MANAGEMENT

Network management creates a platform for service management applications to run and configure various services with ease, i.e. new policies, patches and new code. It also defines the interface between the controller and the applications, thus northbound in accordance with the SDN model. As Akyildiz *et al.* [8] and Qin *et al.* [106] allude, there has been less focus on the NB interface, whereas much attention has been dedicated to SB interfaces. The northbound interfaces are mostly built on top of the network SDN controllers. There are several SDN controllers which support OpenFlow

as highlighted in section VII. These controllers were built for traditional SDN networks. Though some of these controllers have been successfully used in wireless networks, they remain to be implemented and tested for SDWSN. While the controllers provide low-level control over the underlying network devices, they are supported by network programming languages which translate high level network policies into forwarding rules. Procera [163], [164] is a declarative language which also couples as a network control framework. More SDN based network declarative languages can be found in [165]–[168].

OpenRoads [53], [57] is the most common northbound API that allow network applications to define and implement any policy framework on the switch through the controller. OpenRoads, also called wireless OpenFlow, is a network management interface that allows management of various wireless networks using virtualization [169]. OpenRoads uses FlowVisor to slice the OpenFlow network and provides a proper isolation amongst the slices, thus enabling multiple controllers as envisaged in [158]. Each controller is responsible for its slice. This also enables multiple experiments to run concurrently on the network [57], [169]. OpenRoads uses OpenFlow to control and manipulate the OpenFlow based switches and also uses SNMPVisor (based on SNMP protocol) to configure the devices [169]. However the newer releases of OpenFlow comes with OF-CONFIG, which does the configuration and management of the switches [40]. Fig. 8 shows the architectural stack of the OpenRoads deployment.

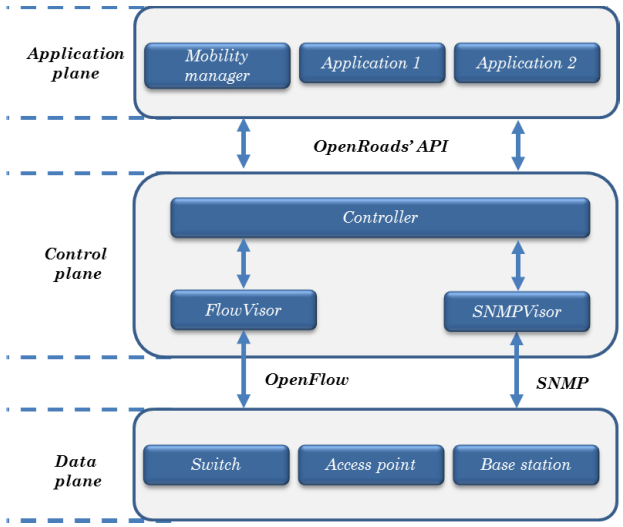

FIGURE 8. OpenRoads architecture [101].

The OpenRoads [57] platform was used in a heterogeneous wireless network consisting of Wi-Fi APs, Wi-Max base station and Ethernet switches, all of which run the OpenFlow protocol. The network was used to test various mobility management applications. The OpenRoads architecture is built on top of the NOX OpenFlow controller [57], [169] which controls the Wi-Fi APS, Wi-Max base station and Ethernet switches as well as the SNMP protocol to have control over the power, frequency, data rate, SSID, etc. of the device elements [169].

Huang *et al.* [170] propose an SDN based management framework for the IoT devices. As the 6LoWPAN protocol gains momentum, particularly in view of the IoT, new network management solutions are developed to cater for these devices. Feng *et al.* [171], LNMP [172] and 6LoWPAN-SNMP [173] are Simple Network Management Protocol (SNMP) based network management solutions for 6LoWPANs. SNMP is a well-known and commonly used network management protocol in enterprise networks and its ripple influence on the development of new management architectures is no surprise.

Although this model was used in different wireless networks, it provides the SDWSN with a proper model of network management, especially considering the potential of heterogeneity in SDWSN, and thus remains open in respect of SDWSN. The challenge with northbound interfaces is that they are not standardized and as such a plethora of different incompatible designs are probable [40].

D. SUMMARY OF DISCUSSION

The SDN model in WSN has been embraced with vigour and enthusiasm. This section discussed the current state-of-the art research in SDWSN with a particular focus on architecture design, network management and routing. There has been huge progress made to advance the SDWSN model albeit in its infancy. The architectures studied above vary slightly in implementation, affirming that a consensus on the design has yet to be reached. On the forwarding nodes, the functions of the MAC and PHY layers remain in WSNs, however there are disparities with how processing of data is implemented across various research studies. The implementation of the control logic also varies; some having it at the sink while some having it at a level higher. Ideally the end-to-end network management would include the two interfaces, southbound and northbound; but seemingly the NB is lacking behind as far as SDWSN is concerned. The OpenFlow protocol is most prevalent between the sensor nodes and the controller. On the other hand there is no commonality with the NB interface as different control platforms uses their own interfaces for this purpose. However, REST APIs are envisaged to play a pivotal role, although not thoroughly tried or SDWSN.

VII. CONTROLLER IMPLEMENTATIONS

There are different implementations of the controllers, such as centralised, distributed, logically centralized but physically distributed and data plane extensions. This section reviews the current control systems in SDN from the perspective of SDWSN for viability purposes.

A. A SINGLE CENTRALIZED CONTROLLER

A centralized controller is an epitome of the basic SDN implementation where a single controller controls the entire network. An example of a centralized controller is depicted in Fig. 9. SDN central controllers include

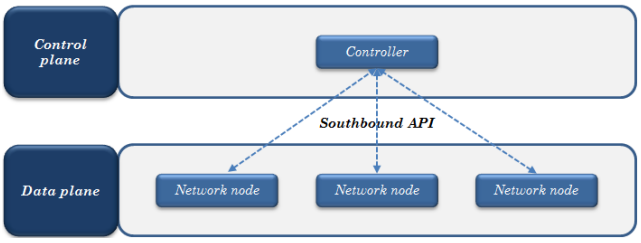

FIGURE 9. Central controller.

Floodlight [138], [174], Ryu [175], Maestro [136], NOX [62], Beacon [137]. A major concern with a single central controller is scalability, reliability and congestion.

B. DISTRIBUTED CONTROLLERS

In order to overcome the drawbacks of the single centralized controller mentioned above, several researchers used multiple distributed controllers. A distributed control system is depicted in Fig. 10. In TinySDN [90] multiple controllers are utilised. The prototype consists of a sensor node and a sink node attached serially to a controller. Olivier *et al.* [31] propose a cluster based SDWSN architecture which also has a base station. They apply the SDN model in a clustered WSN for the formation of what they refer to as a software defined cluster sensor network. The sensor network is organized in clusters (SDN domains) comprising of a simple node (sensor), gateway or cluster head. The SDN cluster head controls and coordinate all sensor nodes in its domain.

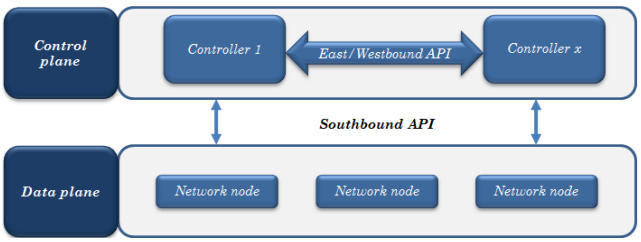

FIGURE 10. Distributed control system.

SDNi protocol [176] is an East/Westbound interface between controllers in a distributed control environment referred to as controller domains. A domain is a network cluster with a controller. SDNi's main purpose is to coordinate controller behaviours and to facilitate the exchange of control and application information across multiple domains. The protocol has three types of messages: reachability update, which exchange reachability information to facilitate inter-SDN domain routing; flow setup/tear-down/update request to coordinate the flow setup as well as capability update.

C. LOGICALLY CENTRALIZED BUT PHYSICALLY DISTRIBUTED

Logically centralised but physically distributed controllers operate as a central controller though physically apart, as shown in Fig. 11. Hyperflow [92] is an event based

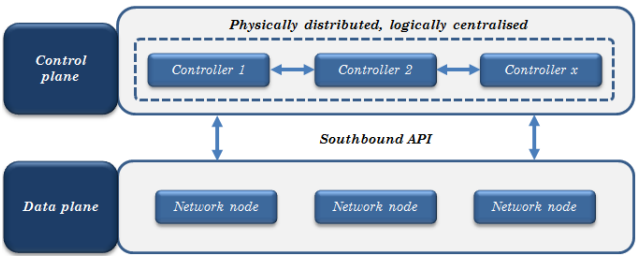

FIGURE 11. Logically centralized controller.

logically centralized but physically distributed controller for OpenFlow. It is based on NOX [62], a network operating system controller; built as an application. The Hyperflow application resides in each NOX controller on the network. The application propagates network events across the controllers thereby synchronizing their network views. Hyperflow uses publish/subscribe messaging to facilitate the lateral controller communication. The published events are stored using WheelFS [177], a distributed file system. Switches are connected to a controller close to them and upon controller failure, they must be reconfigured

ONOS [91] is a distributed control architecture based on Floodlight [138] which aims to ensure scalability, performance and availability. It is also installed into the physically distributed controllers which operate as a unit. Each node has a global view of the network. The ONOS instances control a subset of switches. The ONOS instances propagate state changes of the switches to the global network view. A switch connects to multiple ONOS instances but only one can be the master. Upon failure, the switches elect a new master. ONOS uses a distributed data store for the multiple clusters. OpenDaylight (ODL) [178], [179] is another SDN controller based on Beacon which, like ONOS employs a multiple cluster of controllers maintained through a distributed data store. ODL uses model driven software engineering (MDSE) for inter-model relationship and model driven network management for inter-protocol configuration management such as NETCONFIG and RESTCONFIG [179]. Both ODL and ONOS supports OSGI framework.

Disco [96] is a distributed control system for WAN and overlay networks. The authors aim to cater for heterogeneous and constrained networks as they maintain that current solutions are not adaptable. Disco is based on a Floodlight [138] controller, built as an application. It defines two types of communications: Inter-domain and Intra-domain. The intra-domain communication monitor network events within the domain cluster and use network state to do flow prioritization. The inter-domain communication allows SDN controllers to exchange aggregated network wide information amongst each other. The inter-domain communication employs two methods; Messenger, which ensures connectivity using the Advanced Message Queuing Protocol (AMQP) and Agents, which exchanges the real information. The communication supports diffusion, flooding, publish/subscribe messages.

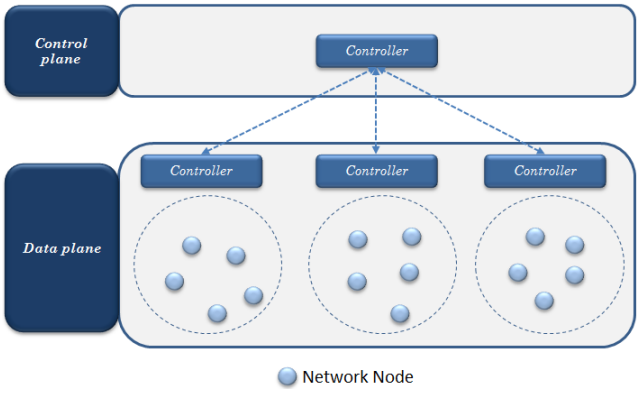

FIGURE 12. Data plane extension controller.

D. A DATA PLANE EXTENSION CONTROL

Other solutions, referred to as data plane extension control systems, enhance the data plane with more control functionality to reduce the overhead towards the controller; Fig. 12 depicts the architecture. Difane [93] is a data plane extension distributed control architecture which keeps most of the traffic in the data plane. Difane offloads some control functionalities to the data plane switches referred to as authority switches which have more power and processing resources. The controller generates flow rules, and then passes them to the authority switches which then install them to the rest of the switches. In DevoFlow [94], the authors aim to devolve the control of most flows to the switches as well. The controller would still control few significant flows referred to as elephant flows; these are the flows that carry large traffic. Some of these switches can even make routing decisions.

Kandoo [95] is another data plane extension solution which employs a rather different approach or method as compared to DevoFlow and Difane. Kandoo scale the control plane with local controllers at the data plane thereby creating two-level control architecture. Local controllers execute local applications but have no interconnection amongst themselves and no global network view. The root controller is logically centralized and has the global view of the entire network. It executes non-local applications, i.e. applications that need global network knowledge. The root controller in turn controls the local controllers. Local controllers could also be implemented at the switch. Frequent events are processed at the data plane while rare events at the central control. The root controller has to subscribe to events from the local controller; otherwise those events would not be propagated.

E. SWITCH TO CONTROLLER MAPPING CONFIGURATIONS

Most of the distributed solutions observed above use static configuration to map the switches to the controllers. Elastic solutions in contrast, offer more flexibility and dynamic assignment of switches to controllers [64], [97], [139], [140]. Elasticon [140] is a distributed control architecture in which switches are dynamically shifted across the controllers to balance the load; the controller pool is also dynamically shrunk or increased. The controllers have a synchronized coordinator providing a consistent control logic/state for the network. A switch connects to many controllers but only one can be the master while others are slaves. The load on the network is shared. There is a threshold that is used to migrate switches to other controllers and also, to reduce or increase the controller pool. It has a load adaptation algorithm and migration protocol. The model was tested using an enhanced Mininet simulator.

Pratyaatsha [97] is also an elastic control solution which, like Elasticon dynamically, assigns SDN switches. In addition, it also assigns SDN application partitions to the distributed controllers. It aims to address two issues: minimize flow setup latency and minimize costs through efficient resource allocation, which includes memory as opposed to only CPU load. Pratyaatsha use the Integer Linear Programming (ILP) algorithm to efficiently assign application state partitions and switches to controllers. Aissioui *et al.* [64] propose another elastic solution for 5G mobile cloud management. The authors aim to address issues of scalability and performance in the context of 5G networks. The solution is based on mobile cloud computing, particularly Follow-Me Cloud (FMC) concept which ensures that mobile users, subjected to lots of movement constraints and migrations, are always connected to an optimal data centre. The proposed elastic control solution has two levels: global FMC controller (G-FMCC) and local FMC controller (L-FMCC). The L-FMCC is deployed on demand using Network Function Virtualisation (NFV) depending on the network dynamics and traffic patterns. G-FMCCs are permanent and responsible for generating, managing and installing OpenFlow rules which ensure seamless migration of services on the cloud.

F. COMPARISON

The different types of controller implementations are summarised in Table 6 below, which includes most of the existing distributed system in addition to few centralized systems. The SDWSN controllers are mostly centralised since most of the work is still in a development stage. In contrast distributed controllers are yet to be implemented in SDWSN. However, traditional SDN distributed controllers can be used as a reference point to develop a tailored distributed controller for SDWSN.

All controller types, apart from data plane extension, keep the control functionality at the control plane, while the data plane extension devolve some of the control functionality to the data plane to reduce the overhead on the centralized controller and reduce traffic in the network. In the logically centralised but physically distributed types, an East/westbound API is used to enable interaction amongst the controllers. However, not all distribution system considers the use of the East/westbound API. Table 7 below summarises the controller type implementation details and further highlights the advantages and disadvantages of each type.

G. SUMMARY OF DISCUSSION

The SDN model centralises the control intelligence of the whole network. Although this abstraction brings lots of

TABLE 6. Distributed controllers in SDN.

Controller	Implementation				Switch mapping		SDWSN	SDN
	Centralised	*Distributed*	*Logically Distributed*	*Data Plane Extension*	*Static*	*Dynamic*		
TinySDN[90]		√				√	√	
SDWN[29]	√				√		√	
SDCSN [31]		√			√		√	
Cognitive SDWSN[134]	√				√		√	
Hyperflow[92]			√		√			√
ONOS[91]			√		√			√
Disco[96]			√		√			√
Difane[93]				√	√			√
DevoFlow[94]				√	√			√
Kandoo[95]				√	√			√
Elasticon[36],		√				√		√
Pratyaatsha[97]		√				√		√
Aissioui et al. [64]		√				√		√

TABLE 7. Detailed comparison of the controllers.

Controller Type	Plane	API	Advantage	Disadvantage
Centralised	Control	North/Southbound	Global network view, better and informed decisions	Central point of failure. Congestion. Not scalable Performance degradation due to distance.
Distributed	Control	North/Southbound and East/Westbound	Scalability Quick response in decisions Rapid ad hoc control access. Improves security as each controller implements its own security measures.	Costs Static configuration Unbalanced load distribution High memory cost of keeping knowledge of global network. Constant synchronisation of large amount of data.
Logically Distributed				
Data Plane Extension	Control and Data	North/Southbound	Reduce overheard by keeping most traffic in the data plane. Quick response time. Improves performance. Improves security	Increases cost of allocating more controllers. Still depends on a central controller which has the global knowledge.

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. The performance of the network will also suffer as a result of overhead. The distance between the gateway nodes and the controller would also affect the performance. Thus a centralised controller would not be viable 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.

VIII. FUTURE RESEARCH CHALLENGES

Most of the existing SDWSN architectures may differ in certain aspects. However, they share fundamental commonalities to which most are influenced by the OpenFlow protocol. While there seem to be a converging consensus with the application of OpenFlow, it is yet to be adequately proven for SDWSN. This section looks at some of the shortcomings of the reviewed SDWSN architectures, while aligning them for future research considerations.

A. WSN INHERENT CHALLENGES

Most WSN inherent challenges are also not yet adequately addressed. Although the SDN paradigm promises a huge reduction in energy consumption by the nodes, the extent of this assertion needs to be evaluated and quantified. The local in-node processing needs to be scrutinized in detail and its impact thereof. The amount of processing needed in proportion to energy usage should be determined. Processing is very critical in alleviating issues such as implosion, redundancy, etc. As far as processing is concerned, when coupled with energy considerations, what amount of processing should be left on the node? The trade-off between internal processing and controller processing need to be evaluated so to also address the issue of aggregation. The aggregation of the sensed data is also paramount and needs closer research attention. Another particular problem is processing the different data attributes that these sensor nodes represent. The aggregation problem in heterogeneous networks also needs to be explored to include the model of abstraction.

The transmission of the data is also a concern. It might not be pragmatic to have all sensor nodes transmitting their raw data to the controller as that will result in an excessive delay

and congestion on the network. As some architectural designs above have in-network processing on the sensor nodes; this has to be tested for its efficiency. Having a local controller on the sensor node has been suggested in [3] while others suggest having a sink node which houses the local controller.

Most of the switches used for SDN purposes use Ternary Content-Addressable Memory (TCAM). Memory is one of the scarce resources in WSNs and remains to be seen whether TCAM would work or even be affordable. The memory management techniques explored in [180] could be investigated from the SDWSN perspective. The other question that begs research attention is how the sensor nodes dynamically join the network and also what happens when a change is effected when a particular node is down.

B. NETWORK OPERATING SYSTEM

There seem to be a disparity on the use of the network operating system and a functional protocol. Some implementations above seem to be more inclined to an OS which includes some basic functionalities of a protocol and likewise the protocol based architectures seem to include some functionalities of an OS. This standoff needs proper evaluation, to also determine if this two should co-exist or operate independently.

C. PRACTICAL IMPLEMENTATION AND EVALUATION

Although there have been research efforts that made inroads in tailoring OpenFlow for SDWSN, there has been no practical application, because most are simulated or just a proposed general framework. The practicality of the SDWSN would provide a clear indication of the progress made thus far. That would also usher in an opportunity to evaluate common wireless network issues such as QoS, reliability, packet loss, bandwidth, stability, efficiency, scalability, etc. Although [7] and [90] propose an architecture for the physical sensor nodes, they were only tested through simulation and therefore there is a need for an actual prototype of the SDN based sensor node.

D. INTER AND INTRA-PLANE COMMUNICATION

The communication between the controller and the application layer is important for the overall structural security of the network. Hence any protocol considered needs to adequately address the security impasse. The communication between the controller and the infrastructure devices, the southbound interface, is also particularly paramount because it is the enabler of the transition from the resourced control plane to the less resourced data plane. This transition presents an open research problem to be explored. The communication amongst the sensor nodes also need to be defined properly amid TCP/IP (IPv4) being ruled out [58], thus the exploration of 6LoWPAN needs to be intensified.

E. STANDARDISATION

The architectures also differ fundamentally on the allotment of different functionalities along the two SDN planes. Most of these architectures focus on one or two entities, thus there is a need for a holistic architecture which covers all building blocks of the model. The lack of standards in SDWSN could derail the development and also further exacerbate the issue of dependant compatibility, which the SDN model seeks to avoid.

There is an urgent need for SDWSN standardisation. The lack of it would result in different incoherent and incompatible architectures, which goes against the SDN's principle of heterogeneity. Whilst there is yet a formal standard for SDWSN to emerge, the standardisation of its constituents, WSN and SDN have been developing with pace, IEEE 802.15.4 [72], ZigBee [71] and ONF [43], IETF [38] respectively. It is not yet known whether conformation to these two standard groups would satisfy the requirements of SDWSN, or perhaps a new standard will be necessary. The standardisation of the IoT framework is also imperative which likewise has seen standardisation of its constituent networks [181]. Thus there is a need for a holistic standard or specification, which will guide the compliance of future networks.

F. SECURITY

Security is a very critical architectural consideration especially in this cyberage, where everything is envisaged to be connected. However security in SDWSN is still a very open area that is yet to receive much attention. Most of the work in SDWSN is still very much on the architectural framework, partly due to the infancy of this field. But nonetheless, since SDWSN synergises two areas, we ought to draw reference from their respective work. The security structure of the model needs to take care of the SDWSN network itself, the controller, the sensor node device and the communication protocols among others.

Most research work addresses security in traditional SDN and WSN networks respectively. Some of these concepts can be adapted into SDWSN while others are very improbable. The network needs to be proactive and alert about any potential threats. Ali *et al.* [36] outline different network verification solutions on the traditional SDN networks, which deals with security configuration, threat detection, threat remediation and network verification (refer to section II.D). The programmability of the SDN makes it vulnerable to attacks [36], since applications can be installed with ease.

In reference to WSN, the security solutions are mostly implemented on the sensor nodes and such protocols tends to be energy intensive and hence not practical [36]. These protocols can be implemented at the controller or application level. Some of these solutions are outlined in detail in [1]. The implementation of these concepts in SDWSN remains open to the research community. Furthermore, most security solutions can also be implemented through Network Function virtualization (NFV), which virtualises network functionalities.

From an SDN perspective, the control layer is more likely to be targeted by adversaries and thus needs to be safeguarded accordingly [59]. The communication protocol also should

TABLE 8. Security threats on both WSN layers and SDN [36].

WSN OSI Layer	SDN Plane	Threat
Application	Application	Poor Authentication and Control
		Fraudulent flows rules insertion
		Poor access control and accountability
		Malicious Application
		DoS
		Northbound Interface (API) attack
Transport	Control	Threats from applications
		DoS
Network		Unauthorised access
		Scalability & Unavailability
		Faulty or Malicious controller
Data Link	Data	Unauthorised access
		Fraudulent rules
		Forged/False traffic flows
		Flooding, Spoofing
Physical		Southbound Interface (API) attack
		Jamming, Tampering
		Sybil
		Compromised/hi-jacked controller
		Malicious node

be protected against any form of interception. The OpenFlow protocol uses a secure TCP protocol on port 6633 and/or also a secure channel based on TLS [45]. However, as [36] maintains the impracticality of running SSL/TLS protocols on small devices, the question of securing network communication remains an interesting research question.

Another major challenge in traditional SDN networks is that the switches are connected to traffic generating hosts, which could be used as gateways for attacks. However, in contrast, sensor nodes are at the periphery of the network, generating traffic. As a result, security threats through the switches are considered to be of less concern as they are just sheer devices, which only understands few entries in the flow table, although further study is needed to confirm this opinion. Table 8 maps the WSN OSI layers along the three planes of SDN together with the security threats associated with those layers.

G. DISTRIBUTED CONTROL SYSTEM

In order to realise scalability, reliability and performance in SDWSN, an efficient distributed control system is needed. Distributed control solutions have been proposed for SDN enterprise networks, but few for SDWSN. This encourages the need to investigate a novel distributed control system for SDWSN without compromising any of the quality imperatives.

IX. DESIGN REQUIREMENTS

In order to address the WSN inherent challenges, this section highlights some of the requirements that need to be considered in the future design of SDWSN. The requirements are depicted in Fig. 13 which is an extension of the architecture presented in Fig. 4. The main differences between the two is that Fig. 4 represents the requirements as currently applied by different studies while Fig. 13 enhance the requirements in accordance with the lessons learned and future architectural considerations. The added functions are coloured in dark blue. The relationship mapping between the controller and the sensor nodes is 1:M, however, this figure represents a distributed control environment.

A. DUTY CYCLES

The SDWSN should support duty cycling, i.e. switching the radio communication to a sleep mode, when not in use [29]. That can be achieved in different ways, either reactively on demand or periodically through diligent synchronization [1]. Low duty cycle operations are preferred in WSNs because high duty cycling could be more detrimental to energy efficiency. Saraswai and Bhattacharya [182] evaluated this trade-off and concluded that energy consumption decreases if duty cycle is less than 0.01% and 0.02% for dense and less dense networks respectively, but otherwise duty increases energy consumption.

B. IN-NETWORK DATA AGGREGATION AND DECISION FUSION

In-network processing of data must be supported to avoid sending raw data to the sink or the controller [29]. Different data aggregation methods should be used to collate the data and transmit only the processed information. The aggregation of data could be based on the source, destination or the application attribute [1]. The protocol should allow a dynamic setting of these combinations through predefined permutations. The determination of what would guide the method of choosing these factors needs proper structural evaluation.

C. FLEXIBLE DEFINITION OF RULES

In line with the SDN premise of simple management, SDWSN should support flexible definition and application of rules and policies. There should also be a mechanism to prevent any advent clashes of rules or policies. FortNox [63] is a perfect example of such, albeit implemented for basic SDN, the inference is relative to note. The rules placement problem is expounded with a comprehensive review of different solutions in [180].

D. NODE MOBILITY

Mobility is a very important design consideration for SDWSN, because the deployment of the sensor nodes varies per application: some structural and others not. Thus nodes could move and change positions. The SDWSN should be able to deal with the inevitably rapid physical topology changes.

E. UNRELIABILITY OF WIRELESS LINKS

WSNs consists of various RF (Radio Frequency) wireless communication links [79]. The wireless links are unstable and therefore not reliable. The instabilities are as a result of common factors, such as a limited bandwidth, node failures, etc. The duty cycling also affects the availability of nodes. The design of the SDWSN should consider the rapid topological changes caused by temporary unavailability of nodes.

F. SELF-HEALING ABILITY

Node failures in WSN network are to be expected and thus the SDWSN network needs to be dynamic with swift reorganization to deal with such occurrences. The controller is expected to be logically centralized to avoid a single point of failure scenario, i.e. physically distributed but operating logically as one controller. CPRecovery [183] is a technique used to ensure resiliency in case of controller failure. It restores the state of the applications (components) to the backup computer. This method employs a central controller and will thus suffer scalability demands.

G. BACKWARD AND PEER COMPATIBILITY

The SDWSN network should be compatible with existing WSN networks, and SDN-based sensor nodes should be able to interface with normal sensor nodes. There should also be a peer compatibility with other networks i.e. other OpenFlow networks. The SDWSN should also integrate well within the IoT framework and its protocols.

H. DATA-CENTRIC AND ADDRESS-CENTRIC (MULTIPLE IDENTIFICATION)

WSNs have always been considered data centric [1], [7], but recent studies are leaning towards address-centric approaches using IP addresses, more so IPv6 (6LoWPAN [77]). Moreover authors in [78], [75], and [132] states that augmenting WSNs with IP is an adequate solution for integrating WSNs into IoT. A SDWSN therefore should support both scenarios.

I. SCALABILITY

The SDWSN framework should be scalable and one way to realize that is to ensure that as the sensor nodes scale up, there is equally sufficient controller service. Although studies in [137] and [139] state that one controller can handle millions of flows per second on traditional SDNs with switches, the same is yet to be ascertained for sensor nodes. Nunes *et al.* [13] state that to achieve a reliable and robust scalability, the logically distributed controllers should also be physically distributed as in [92].

J. EAST/WESTBOUND INTERFACE

To realise scalability, a distributed control plane should be considered. A distributed control plane requires consistent and synchronized communication between the controllers. The communication between the controllers is herein referred as East/Westbound API [12]. Like with the northbound API, there has been very little attention paid to this interface. However work such as SDNi [176], [142], which focus on flow setup coordination and reachability of exchange information, is a progressive step. The development of distributed controllers such as Onix [139], Hyperflow [92], ONOS [91], Kandoo [95], Elasticon [139], [140], Pratyaastha [97] and [64] will necessitate the development and standardization of a common interface.

K. OPTIMUM DEVICE PLACEMENT (CONTROLLERS AND NODES)

The placement of controllers and nodes is very critical and will have to be in line with the scalability considerations. This, however, varies from network to network. Heller *et al.* [184] thoroughly evaluated the controller placement problem.

L. NORTHBOUND AND SOUTHBOUND INTERFACES

The northbound and southbound interfaces are very important to SDWSN for the fluidity of vertical cross layer communication. However, there has been a lot of work focusing on the southbound interface and less on the northbound. As [13] and [163] note, there is yet to be a standardised API or an interface for northbound communication. However, the SDWSN framework should create a platform for interoperability. The northbound interface is very important as it connects to most of the functionalities that were removed from the node. This interface has to be rigid with security and conflict prevention measures, [43], [110]. As heterogeneity is expected to be high in IoT, this interface could be used in conjunction with a virtualization layer, same as OpenRoads [53] did with heterogeneous wireless networks. Sneps-Sneppe *et al.* [186] asserts the need to have a metadata-based interface for the NB API and further suggests that the NB API will be based on REST [187]. However, unlike its counterpart service-oriented architecture (SOA) [187], REST does not offer metadata. Although SOA services such as WSDL (Web Service Definition Language), XSD (XML Schema Definition) provides metadata, they define different interfaces for each service application. Thus, for the purpose of ubiquity and heterogeneity, REST would be feasible, but the metadata remains an open challenge.

M. SECURITY

Security is a very important consideration for SDWSN frameworks. As SDWSN is envisaged to play a critical role in the IoT framework, it is equally imperative that security be as stringent as possible. Most of the architectures do not have security as a build-in feature for SDN [59]; Kreutz *et al.* [60]

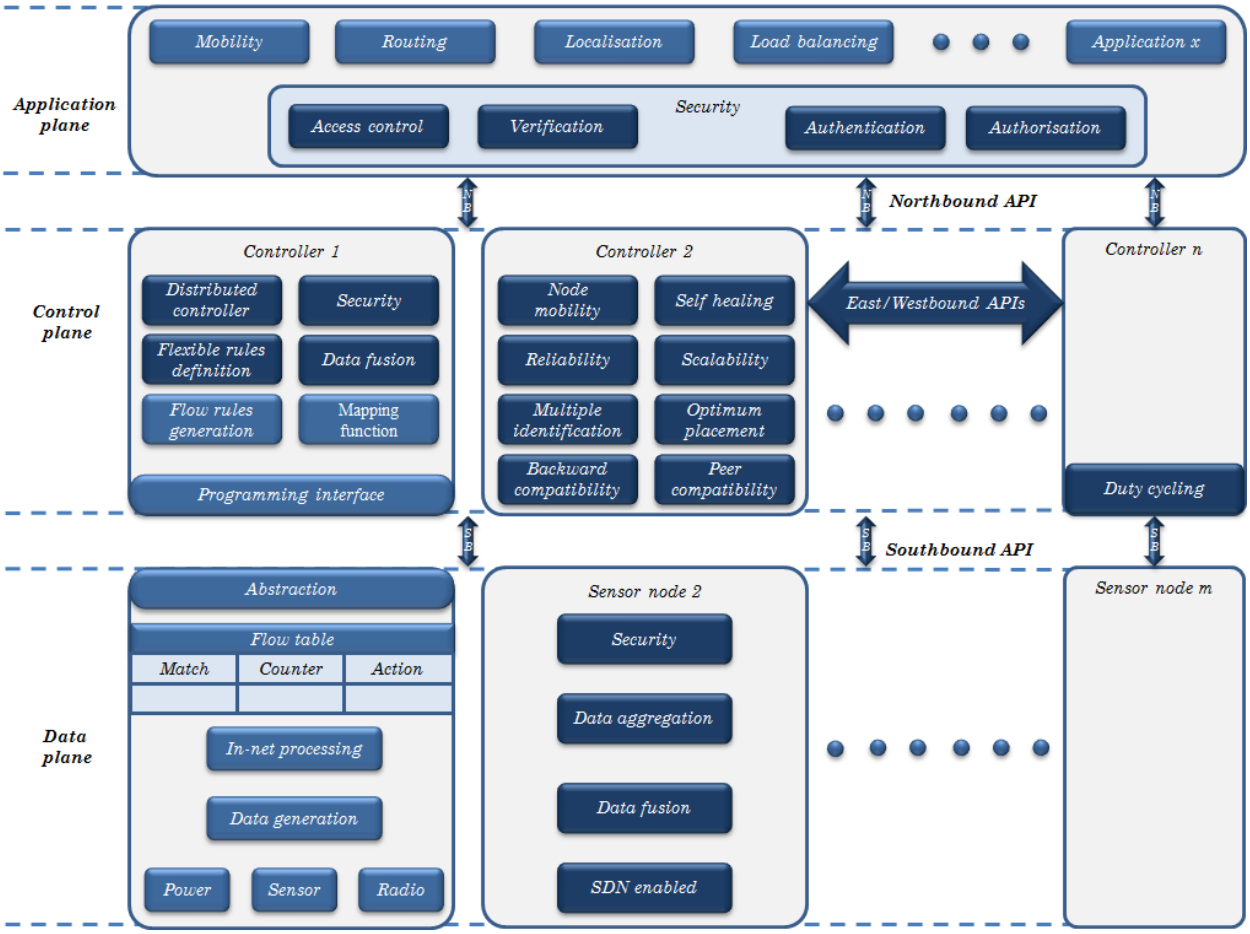

FIGURE 13. SDWSN design requirements. This captures the requirements as currently applied and also those that should be considered for future considerations.

rightfully state that security and dependability of SDN should be built-in from design. Most of the current SDN security measures are earmarked for enterprise networks, where the network devices are switches or routers. These need to be adapted to consider wireless sensor network. Security is discussed in detail at the beginning of section III.

X. LESSONS LEARNED

This paper reviewed the current state of the art application of SDN in WSNs, the SDWSN. The SDWSN falls within the broader context of the Internet of Things and as such, some of the related concepts were also highlighted. This exercise was also to locate the purpose and role of SDWSN within the IoT space. The IoT paradigm seeks to create a networking environment for all devices expected to partake. Most of the devices will be equipped with sensors and actuators. Data from these devices will be carried by various networks such as enterprise, mobile wireless and optical networks. Various computing platforms such as cloud, fog, mobile cloud and mobile edge computing will feature prominently for service provisions. The application of SDN on these networks and computing platforms has been receiving significant and devoted attention from both the academic and industrial research. This paper has highlighted the bottom-up approach on the application of SDN to the realisation of IoT. Sensors and actuators are located at the bottom as collectors of data, thus the evaluation of SDWSN.

WSNs are envisaged to be a significant building block of the IoT paradigm. However WSNs inherently exhibit major constraints in resources such as power, processing, memory, etc. Most of the attempted solutions thus far are not efficient, thereby exacerbating the challenges at cost. SDN comes as a potential to solve some of these challenges. The SDN premise of decoupling the control logic from the forwarding engine brings a substantial respite as most of the energy intensive functions are relocated to the controller.

The paper also delved into the importance of SDN in WSNs. Issues such as energy, network management, configuration, scalability, routing, mobility, localisation interoperability, communication and security are envisaged to improve. It is also noted that the centralization of the controller could pose a security threat, as it could be a potential target and a reliability threat resulting in a single point of failure. However, various research strides are been made to resolve this.

The fusion of SDN and WSN beget SDWSN, which is relatively a new research area. The paper looked at different components of SDWSN, such as the architecture, network

management, routing, etc. Due to the infancy of this field; most of the studies are still unriddling the architectural framework. As such, some architecture, such as [7] and [29] are more leaned towards the sensor node, while some such as Gante et al. [33] are leaned towards the controller. Some such as [29] and [134] have some processing on the node while Jacobsson and Orfanidis [3] have a local controller on the node. Others, such as [29] and [90], propose the use of a serial connection between the controller and the sink, which could stifle scalability. The lack of practicality is understandably common, with most of the work still on simulation, except a few.

The challenges besetting SDWSN were presented for future consideration. These challenges include among others; some of the WSN inherent challenges not wholly addressed by SDN, such as processing clarity, memory, etc. Other challenges identified include standardisation and security. Standardisation is very pertinent to the ideals of IoT heterogeneity, while security will also be central in IoT to ensure that future networks are secure and reliable. Finally, after the evaluation of the state of the art architectures and their current challenges, design requirements were identified.

XI. CONCLUSION

The SDWSN model is very challenging, as it comprises of two unfledged models which are still entangled in their own complexities. The WSN networks are resource constrained, which compels all research efforts to be energy conscious. Despite many efforts, this is yet to be fully realised and thus have yet to reach their optimal efficacy. The introduction of SDN in WSN presents a very novel and progressive step in leveraging the challenges of resources in WSN. However, the SDN model brings along its own challenges, especially the trade-off between functionalities that need to be retained on the sensor device and the impact on common network factors such as latency, congestion, etc.

This paper has reviewed the current work in SDWSN. Most of the architectures proposed are still in the development stage. The prevalence of the OpenFlow protocol in SDN applications seems to have inspired influence even towards the SDWSN model and thus, we envisage it to play a major role in the development of this model. Although there are still many challenges in this model, great strides have been made thus far. However, the lack of standardisation in SDWSN is still a concern and standards have to be developed to create an oversight for compatibility and sustainability.

The paper also discussed the SDWSN design requirements that need to be carefully evaluated and considered in designing and implementing a practical SDWSN framework. These design requirements would assist in overcoming the various challenges inherent to WSN as well as the other challenges associated with SDN. Finally the paper tried to highlight some of the open research challenges that require more attention by the research community.

APPENDIX

All the acronyms used in the paper are enlisted in Table 9.

TABLE 9. Definitions of all acronyms used in the paper.

Abbreviations	
6LoWPAN	IPv6 addressing for Low-Rate Personal Networks
AODV	Ad Hoc On-Demand Distance Vector
API	Application Programming Interface
BS	Base Station
CAN	Content Addressable Network
CC	Cloud Computing
CE	Controlling Element
CPU	Central Processing Unit
CTP	Collection Tree Protocol
DC	Data Centres
DDS	Data Distribution Service
FE	Forwarding Element
FMC	Follow Me Cloud
ForCes	Forwarding and Control Element Separation
HaaS	Hardware as a Service
IaaS	Infrastructure as a Service
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
IoT	Internet of Things
ISA	International Society of Automation
ISM	Industrial, Scientific and Medical
LFB	Logical Function Block
LNMP	LoWPAN Network Management Protocol
LR-WPAN	Low-Rate Wireless Personal Networks
MAC	Medium Access Control
MCC	Mobile Cloud Computing
MEC	Mobile Edge Computing
MEMS	Micro Electrical Mechanical Systems
NB	Northbound
NFV.	Network Functions Virtualization
NOX/S	Network Operating System
ODL	OpenDaylight
ONF	Open Network Foundation
ONOS	Open Network Operating System
OTAP	Over-The-Air-Programming
PaaS	Platform as a Service
PHY	Physical (layer)
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
RSU	Road-Site-Unit
RSUC	Road-Site-Unit Controller
SaaS	Software as a Service
SB	Southbound
SDCSN	Software-Defined Cluster Sensor Network
SDN	Software-Defined Networking
SDNCH	SDN Cluster Head
SDN-WISE	SDN-Wireless Sensor network
SDWN	Software-Defined Wireless Network
SDWSN	Software-Defined Wireless Sensor Networks
SNMP	Simple Network Management Protocol
SOA	Service-Oriented Architecture
SSL	Secure Sockets Layer
TCAM	Ternary Content-Addressable Memory
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
UWB	Ultra-Wideband
VANET	Vehicular Ad Hoc Network
VM	Virtual Machine
WAN	Wide Area Networks
WLAN	Wireless Local Area Network
WSDL	Web Service Definition Language
WSN	Wireless Sensor Networks
XML	Extensible Markup Language
XSD	XML Schema Definition

REFERENCES

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Comput. Netw.*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] M. Jacobsson and C. Orfanidis, "Using software-defined networking principles for wireless sensor networks," in *Proc. 11th Swedish Nat. Comput. Netw. Workshop*, Karlstad, Sweden, May 2015.

[4] *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*, Cisco Syst., San Jose, CA, USA, 2016.

[5] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Netw.*, vol. 7, no. 3, pp. 537–568, May 2009.

[6] J. Qadir and O. Hasan, "Applying formal methods to networking: Theory, techniques, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 256–291, 1st Quart., 2015.

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

[8] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Netw.*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[9] V. Potdar, A. Sharif, and E. Chang, "Wireless sensor networks: A survey," in *Proc. Int. Conf. Adv. Inf. Netw. Appl. Workshops*, May 2009, pp. 636–641.

[10] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards," *Comput. Commun.*, vol. 30, no. 7, pp. 1655–1695, 2007.

[11] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless sensor network virtualization: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 553–576, 1st Quart., 2016.

[12] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[13] 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. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, Third Quarter 2014.

[14] J. Qadir, N. Ahmed, and N. Ahad, "Building programmable wireless networks: An architectural survey," *EURASIP J. Wireless Commun. Netw.*, vol. 2014, no. 1, pp. 1–19, 2014.

[15] S. Sadiq, S. Fizza, and M. Tahira, "A survey on wireless software defined networks," *Int. J. Comput. Commun. Syst. Eng.*, vol. 2, no. 1, pp. 155–159, 2015.

[16] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 27–51, 1st Quart., 2014.

[17] N. A. Jagadeesan and B. Krishnamachari, "Software-defined networking paradigms in wireless networks: A survey," *ACM Comput. Surveys*, vol. 47, no. 2, pp. 27:1–27:11, Jan. 2015.

[18] A. Drescher, "A survey of software-defined wireless networks," Dept. Comput. Sci. Eng., Washington Univ. St. Louis, St. Louis, MO, USA, Tech. Rep., 2014, pp. 1–15.

[19] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[20] Q. Duan, Y. Yan, and A. V. Vasilakos, "A survey on service-oriented network virtualization toward convergence of networking and cloud computing," *IEEE Trans. Netw. Service Manage.*, vol. 9, no. 4, pp. 373–392, Dec. 2012.

[21] S. Azodolmolky, P. Wieder, and R. Yahyapour, "SDN-based cloud computing networking," in *Proc. 15th Int. Conf. Transparent Opt. Netw. (ICTON)*, Jun. 2013, pp. 1–4.

[22] J. Pan, S. Paul, and R. Jain, "A survey of the research on future Internet architectures," *IEEE Commun. Mag.*, vol. 49, no. 7, pp. 26–36, Jul. 2011.

[23] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *Proc. Austral. Telecommun. Netw. Appl. Conf. (ATNAC)*, Nov. 2014, pp. 117–122.

[24] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data (Mobidata)*, 2015, pp. 37–42.

[25] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

[26] M. Yang, Y. Li, D. Jin, L. Zeng, X. Wu, and A. V. Vasilakos, "Software-defined and virtualized future mobile and wireless networks: A survey," *Mobile Netw. Appl.*, vol. 20, no. 1, pp. 4–18, Feb. 2015.

[27] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.

[28] N. Bizanis and F. A. Kuipers, "SDN and virtualization solutions for the Internet of Things: A survey," *IEEE Access*, vol. 4, pp. 5591–5606, 2016.

[29] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks: Unbridling SDNs," *Proc. Eur. Workshop Softw. Defined Netw. (EWSDN)*, Oct. 2012, pp. 1–6.

[30] D. Zeng, T. Miyazaki, S. Guo, T. Tsukahara, J. Kitamichi, and T. Hayashi, "Evolution of software-defined sensor networks," in *Proc. IEEE 9th Int. Conf. Mobile Ad-Hoc Sensor Netw.*, Dec. 2013, pp. 410–413.

[31] F. Olivier, G. Carlos, and N. Florent, "SDN based architecture for clustered WSN," in *Proc. 9th Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput.*, Jul. 2015, pp. 342–347.

[32] I. T. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: A survey and taxonomy," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2713–2737, 4th Quart., 2016.

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

[34] N. A. Jagadeesan and B. Krishnamachari, "Software-defined networking paradigms in wireless networks: A survey," *ACM Comput. Surv.*, vol. 47, no. 2, Jan. 2014, Art. no. 2.

[35] S. Sezer *et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, Jul. 2013.

[36] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE Trans. Rel.*, vol. 64, no. 3, pp. 1086–1097, Sep. 2015.

[37] J. Sherry and S. Ratnasamy, "A survey of enterprise middlebox deployments," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2012-24, 2012.

[38] I. Kovacevic, "FoRCES protocol as a solution for interaction of control and forwarding planes in distributed routers," in *Proc. 17th Telecommun. Forum (TELFOR)*, 2009, pp. 529–532.

[39] E. Haleplidis *et al.*, "Network programmability with ForCES," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1423–1440, 3rd Quart., 2015.

[40] W. Braun and M. Menth, "Software-defined networking using OpenFlow: Protocols, applications and architectural design choices," *Future Internet*, vol. 6, no. 2, pp. 302–336, 2014.

[41] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and OpenFlow: From concept to implementation," *IEEE Commun. Surv. Tuts.*, vol. 16, no. 4, pp. 2181–2206, 4th Quart., 2014.

[42] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using OpenFlow: A survey" *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 493–512, 4th Quart., 2013.

[43] Open Networking Foundation, "Software-defined networking: The new norm for networks," Open Netw. Found., Palo Alto, CA, USA, White Paper 2, 2012, pp. 1–12.

[44] E. Haleplidis, S. Denazis, O. Koufopavlou, J. H. Salim, and J. Halpern, "Software-defined networking: Experimenting with the control to forwarding plane interface," in *Proc. Eur. Workshop Softw. Defined Netw. (EWSDN)*, Oct. 2012, pp. 91–96.

[45] M. P. Fernandez, "Comparing OpenFlow controller paradigms scalability: Reactive and proactive," in *Proc. Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2013, pp. 1009–1016.

[46] B. Heller, "OpenFlow switch specification," Open Netw. Found., 2009, pp. 1–36.

[47] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1955–1980, 4th Quart. 2014.

[48] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined wan," in *Proc. ACM SIGCOMM Conf. SIGCOMM (SIGCOMM)*, vol. 13. 2013, p. 12.

[49] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "OpenRadio: A programmable wireless dataplane," in *Proc. HotSDN*, 2012, pp. 109–114.
[50] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards programmable enterprise WLANS with Odin," *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 115–120.
[51] Y. Jararweh, M. Al Ayyoub, A. Doulat, A. A. A. Al Aziz, H. A. B. Salameh, and A. A. Khreishah, "SD-CRN: Software defined cognitive radio network framework," in *Proc. IEEE Int. Conf. Cloud Eng.*, Mar. 2014, pp. 592–597.
[52] G. Sun, G. Liu, and Y. Wang, "SDN architecture for cognitive radio networks," in *Proc. 1st Int. Workshop Cognit. Cellular Syst. (CCS)*, Sep. 2014, pp. 1–5.
[53] K. Yap *et al.*, "OpenRoads: Empowering research in mobile networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 125–126, Jan. 2010.
[54] V. Nascimento *et al.*, "Filling the gap between software defined networking and wireless mesh networks," in *Proc. 10th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2014, pp. 451–454.
[55] P. Dely, A. Kassler, and N. Bayer, "OpenFlow for wireless mesh networks," in *Proc. 20th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul./Aug. 2011, pp. 1–6.
[56] N. V. R. Gupta and M. V. Ramakrishna, "A road map for SDN-open flow networks," *Int. J. Modern Commun. Technol. Res.*, vol. 3, no. 6, pp. 38–46, 2015.
[57] K.-K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. McKeown, "The Stanford OpenRoads deployment," in *Proc. 4th ACM Int. Workshop Experim. Eval. Characterization (WINTECH)*, 2009, pp. 59–66.
[58] A. Mahmud and R. Rahmani, "Exploitation of OpenFlow in wireless sensor networks," in *Proc. Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT)*, Dec. 2012, pp. 594–600.
[59] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2317–2346, 4th Quart., 2015.
[60] D. Kreutz, F. M. V. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, vol. 13. 2013, pp. 55–60.
[61] S. Shin, P. Porras, V. Yegneswaran, and M. Fong, "FRESCO: Modular composable security services for software-defined networks," in *Proc. ISOC Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2013.
3.
[62] N. Gude *et al.*, "NOX: Towards an operating system for networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, 2008.
[63] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, vol. 12. 2012, pp. 121–126.
[64] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb, "Toward elastic distributed SDN/NFV controller for 5G mobile cloud management systems," *IEEE Access*, vol. 3, pp. 2055–2064, 2015.
[65] R. Beckett, X. K. Zou, S. Zhang, S. Malik, J. Rexford, and D. Walker, "An assertion language for debugging SDN applications," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2014, pp. 91–96.
[66] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey, "VeriFlow: Verifying network-wide invariants in real time," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2012, pp. 49–54.
[67] S. Son, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Model checking invariant security properties in OpenFlow," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2013, pp. 1974–1979.
[68] E. Al-Shaer and S. Al-Haj, "FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures," in *Proc. 3rd ACM Workshop Assurable Usable Secur. Configuration (SafeConfig)*, 2010, pp. 37–44.
[69] J.-S. Lee, Y.-W. Su, and C.-C. Shen, "A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi," in *Proc. 33rd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Nov. 2007, pp. 46–51.
[70] J. Hughes, J. Yan, and K. Soga, "Development of wireless sensor network using Bluetooth low energy (BLE) for construction noise monitoring," *Int. J. Smart Sens. Intell. Syst.*, vol. 8, no. 2, pp. 1379–1405, 2015.
[71] I. Howitt and J. A. Gutierrez, "IEEE 802.15.4 low rate—Wireless personal area network coexistence issues," in *Proc. IEEE Wireless Commun. Netw. (WCNC)*, vol. 3. Mar. 2003, pp. 1481–1486.
[72] *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, IEEE Standard 802.15.4-2006, 2006, pp. 1–305.
[73] S. Petersen and S. Carlsen, "WirelessHART versus ISA100.11a: The format war hits the factory floor," *IEEE Ind. Electron. Mag.*, vol. 5, no. 4, pp. 23–34, Dec. 2011.
[74] T. Lennvall, S. Svensson, and F. Hekland, "A comparison of WirelessHART and ZigBee for industrial applications," in *Proc. IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, May 2008, pp. 85–88.
[75] Y. Mazzer and B. Tourancheau, "Comparisons of 6LoWPAN implementations on wireless sensor networks," in *Proc. 3rd Int. Conf. Sensor Technol. Appl.*, Jun. 2009, pp. 689–692.
[76] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Netw.*, vol. 3, no. 3, pp. 325–349, May 2010.
[77] M. J. Mudumbe and A. M. Abu-Mahfouz, "Smart water meter system for user-centric consumption measurement," in *Proc. IEEE 13th Int. Conf. Ind. Inform. (INDIN)*, Jul. 2015, pp. 993–998.
[78] D. Christin, A. Reinhardt, P. S. Mogre, and R. Steinmetz, "Wireless sensor networks and the Internet of Things: Selected challenges," in *Proc. 8th GI/ITG KuVS Fachgespräch 'Drahtlose Sensornetze'*, Hamburg, Germany, 2009, pp. 31–34.
[79] H. Modares, R. Salleh, and A. Moravejosharieh, "Overview of security issues in wireless sensor networks," in *Proc. 3rd Int. Conf. Comput. Intell. Modelling Simulation*, Sep. 2011, pp. 308–311.
[80] N. Ntuli and A. M. Abu-Mahfouz, "A simple security architecture for smart water management system," *Proc. Comput. Sci.*, vol. 83, no. 4, pp. 1164–1169, 2016.
[81] A. S. K. Pathan, H.-W. Lee, and C. S. Hong, "Security in wireless sensor networks: Issues and challenges," in *Proc. 8th Int. Conf. Adv. Commun. Technol.*, vol. 2. May 2006, pp. 1042–1048.
[82] K. Chelli, "Security issues in wireless sensor networks: Attacks and countermeasures," in *Proc. World Congr. Eng.*, vol. 1. 2015, pp. 1–6.
[83] V. Kumar, A. Jain, and P. N. Barwal, "Wireless sensor networks: Security issues, challenges and solutions," *Int. J. Inf. Comput. Technol.*, vol. 4, no. 8, pp. 859–868, 2014.
[84] T. Zia and A. Zomaya, "Security issues in wireless sensor networks," in *Proc. Int. Conf. Syst. Netw. Commun. (ICSNC)*, Oct./Nov. 2006, p. 40.
[85] 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 *Proc. IEEE 14th Int. Conf. Ind. Inform. (INDIN)*, Jul. 2016, pp. 1166–1170.
[86] C. Alcaraz, P. Najera, J. Lopez, and R. Roman, "Wireless sensor networks and the Internet of Things: Do we need a complete integration," in *Proc. 1st Int. Workshop Secur. Internet Things*, 2010, pp. 1–8.
[87] Z. Han and W. Ren, "A novel wireless sensor networks structure based on the SDN," *Int. J. Distrib. Sensor Netw.*, vol. 10, no. 3, pp. 1–8, 2014.
[88] P. Jayashree and F. I. Princy, "Leveraging SDN to conserve energy in WSN—An analysis," in *Proc. 3rd Int. Conf. Signal Process., Commun. Netw. (ICSCN)*, Mar. 2015, pp. 1–6.
[89] Y. Wang, H. Chen, X. Wu, and L. Shu, "An energy-efficient SDN based sleep scheduling algorithm for WSNs," *J. Netw. Comput. Appl.*, vol. 59, pp. 39–45, Jan. 2015.
[90] B. T. de Oliveira, C. B. Margi, and L. B. Gabriel, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," in *Proc. IEEE Latin-Amer. Conf. Commun. (LATINCOM)*, Nov. 2014, pp. 1–6.
[91] P. Berde *et al.*, "ONOS: Towards an open, distributed SDN OS," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2014, pp. 1–6.
[92] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw. (INM/WREN)*, 2010, p. 3.
[93] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 351–362, Oct. 2011.
[94] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, Aug. 2011.
[95] S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," *Proc. 1st Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2012, pp. 19–24.

[96] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–4.

[97] A. Krishnamurthy, S. P. Chandrabose, and A. Gember-Jacobson, "Pratyaastha: An efficient elastic distributed SDN control plane," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2014, pp. 133–138.

[98] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 7–12, Sep. 2013.

[99] A. M. Abu-mahfouz and G. P. Hancke, "An efficient distributed localisation algorithm for wireless sensor networks: based on smart reference-selection method," *Int. J. Sensor Netw.*, vol. 13, no. 2, pp. 94–111, May 2013.

[100] Z. Shu, J. Wan, D. Li, J. Lin, A. V. Vasilakos, and M. Imran, "Security in software-defined networking: Threats and countermeasures," *Mobile Netw. Appl.*, vol. 21, no. 5, pp. 764–776, Jan. 2016.

[101] I. Alsmadi and D. Xu, "Security of software defined networks: A survey," *Comput. Secur.*, vol. 53, pp. 79–108, Sep. 2015.

[102] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, and G. Wang, "Meridian: An SDN platform for cloud network services," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 120–127, Feb. 2013.

[103] S. Azodolmolky, P. Wieder, and R. Yahyapour, "Cloud computing networking: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 54–62, Jul. 2013.

[104] Y. Jararweh, A. Mahmoud, A. Darabseh, E. Benkhelifa, M. Vouk, and A. Rindos, "SDIoT: A software defined based Internet of things framework," *J. Ambient Intell. Humanized Comput.*, vol. 6, no. 4, pp. 453–461, Aug. 2015.

[105] Á. L. V. Caraguay, A. B. Peral, L. I. B. López, and L. J. G. Villalba, "SDN: Evolution and opportunities in the development iot applications," *Int. J. Distrib. Sens. Netw.*, vol. 10, no. 5, pp. 1896–1899, May 2014.

[106] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the Internet-of-Things," in *Proc. IEEE/IFIP NOMS—IEEE/IFIP Netw. Oper. Manage. Symp. Manage. Softw. Defined World*, May 2014, pp. 1–9.

[107] P. Hu, "A system architecture for software-defined industrial Internet of Things," in *Proc. IEEE Int. Conf. Ubiquitous Wireless Broadband (ICUWB)*, Oct. 2015, pp. 1–5.

[108] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput. (MCC)*, 2012, pp. 13–16.

[109] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane, "Software defined networking-based vehicular adhoc network with fog computing," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 1202–1207.

[110] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.

[111] M. T. Beck, M. Werner, S. Feld, and T. Schimper, "Mobile edge computing: A taxonomy," in *Proc. 6th Int. Conf. Adv. Future Internet*, 2014, pp. 48–54.

[112] I. Ku, Y. Lu, and M. Gerla, "Software-defined mobile cloud: Architecture, services and use cases," in *Proc. Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Aug. 2014, pp. 1–6.

[113] N. Bhushan *et al.*, "Network densification: The dominant theme for wireless evolution into 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 82–89, Feb. 2014.

[114] X. Ge, S. Tu, G. Mao, and C. X. Wang, "5G ultra-dense cellular networks," *IEEE Trans. Wireless Commun.*, vol. 23, no. 1, pp. 72–79, Feb. 2016.

[115] A. Osseiran *et al.*, "Scenarios for 5G mobile and wireless communications: The vision of the METIS project," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 26–35, May 2014.

[116] B. Soret, K. I. Pedersen, N. T. K. Jørgensen, and V. Fernández-López, "Interference coordination for dense wireless networks," *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 102–109, Jan. 2015.

[117] T. S. Rappaport *et al.*, "Millimeter wave mobile communications for 5G cellular: It will work!" *IEEE Access*, vol. 1, pp. 335–349, May 2013.

[118] H. Ali-Ahmad *et al.*, "An SDN-based network architecture for extremely dense wireless networks," in *Proc. IEEE SDN Future Netw. Services (SDN4FNS)*, Nov. 2013, pp. 1–7.

[119] R. Trivisonno, R. Guerzoni, I. Vaishnavi, and D. Soldani, "SDN-based 5G mobile networks: Architecture, functions, procedures and backward compatibility," *Trans. Emerg. Telecommun. Technol.*, vol. 26, no. 1, pp. 82–92, Jan. 2015.

[120] P. Rost *et al.*, "Cloud technologies for flexible 5G radio access networks," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 68–76, May 2014.

[121] N. Omnes, M. Bouillon, G. Fromentoux, and O. L. Grand, "A programmable and virtualized network & IT infrastructure for the Internet of Things: How can NFV & SDN help for facing the upcoming challenges," in *Proc. 18th Int. Conf. Intell. Next Generat. Netw.*, Feb. 2015, pp. 64–69.

[122] A.-C. G. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "Towards a software-defined Network operating system for the IoT," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 579–584.

[123] A. M. Abu-Mahfouz, T. O. Olwal, A. M. Kurien, J. L. Munda, and K. Djouani, "Toward developing a distributed autonomous energy management system (DAEMS)," in *Proc. AFRICON*, 2015, pp. 1–6.

[124] A. M. Abu-Mahfouz, Y. Hamam, P. R. Page, K. Djouani, and A. Kurien, "Real-time dynamic hydraulic model for potable water loss reduction," *Proc. Eng.*, vol. 154, no. 7, pp. 99–106, 2016.

[125] L. B. Campos, C. E. Cugnasca, A. R. Hirakawa, and J. S. C. Martini, "Towards an IoT-based system for smart city," in *Proc. IEEE Int. Symp. Consum. Electron. (ISCE)*, Sep. 2016, pp. 129–130.

[126] S. Bhushan, B. Bohara, P. Kumar, and V. Sharma, "A new approach towards IoT by using health care-IoT and food distribution IoT," in *Proc. 2nd Int. Conf. Adv. Comput., Commun., Autom. (ICACCA Fall)*, Sep./Oct. 2016, pp. 1–7.

[127] A. Luckshetty, S. Dontal, S. Tangade, and S. S. Manvi, "A survey: Comparative study of applications, attacks, security and privacy in VANETs," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2016, pp. 1594–1598.

[128] A. Malik, J. Qadir, B. Ahmad, K.-L. A. Yau, and U. Ullah, "QoS in IEEE 802.11-based wireless networks: A contemporary review," *J. Netw. Comput. Appl.*, vol. 55, pp. 24–46, Sep. 2015.

[129] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.

[130] B. Pediredla, K. I.-K. Wang, Z. Salcic, and A. Ivoghlian, "A 6LoWPAN implementation for memory constrained and power efficient wireless sensor nodes," in *Proc. 39th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Nov. 2013, pp. 4432–4437.

[131] C. P. P. Schumacher, N. Kushalnagar, and G. Montenegro, "IPv6 over low-power wireless personal area networks (6LoWPANs): Overview, assumptions, problem statement, and goals," Network Working Group, Intel Corp, Microsoft Corp., Redmond, WA, USA, 2007.

[132] S. A. Catapang, Z. J. M. Roberts, K. I.-K. Wang, and Z. Salcic, "An infrastructure for integrating heterogeneous embedded 6LoWPAN networks for Internet of Things applications," in *Proc. 7th Int. Conf. Sens. Technol.*, Dec. 2013, pp. 741–746.

[133] 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. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2015, pp. 513–521.

[134] R. Huang, X. Chu, J. Zhang, and Y. H. Hu, "Energy-efficient monitoring in software defined wireless sensor networks using reinforcement learning: A prototype," in *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 10, pp. 1–13, 2015.

[135] Y.-F. Lee and C.-C. Shen, "A transaction-based approach to over-the-air programming in wireless sensor networks," in *Proc. Int. Symp. Commun. Inf. Technol.*, Oct. 2007, pp. 1377–1382.

[136] Z. Cai, A. Cox, and E. T. S. Ng, "Maestro: A system for scalable OpenFlow control," Rice Univ., Houston, TX, USA, Tech. Rep. TR10-08, 2010.

[137] D. Erickson, "The beacon OpenFlow controller," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 13–18.

[138] *Floodlight OpenFlow Controller-Project Floodlight*, accessed on Nov. 24, 2015. [Online]. Available: http://www.projectfloodlight.org/floodlight/

[139] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 7–12, Oct. 2013.

[140] A. A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "ElastiCon: An elastic distributed SDN controller," in *Proc. 10th ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, 2014, pp. 17–28.
[141] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 655–685, 1st Quart., 2015.
[142] P. Lin, J. Bi, Z. Chen, Y. Wang, H. Hu, and A. Xu, "WE-bridge: West-east bridge for SDN inter-domain network peering," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr./May 2014, pp. 111–112.
[143] A. Sehgal, "Using the contiki cooja simulator," Dept. Comput. Sci., Jacobs Univ. Bremen, Bremen, Germany, 2013.
[144] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *ACM SIGOPS Oper. Syst. Rev.*, vol. 35, no. 11, pp. 93–104, Nov. 2000.
[145] P. Levis *et al.*, "TinyOS: An operating system for sensor networks," *Ambient Intell.*, pp. 115–148, 2005.
[146] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw. (Hotnets)*, 2010, pp. 1–6.
[147] S.-Y. Wang, C.-L. Chou, and C.-M. Yang, "EstiNet OpenFlow network simulator and emulator," *IEEE Commun. Mag.*, vol. 51, no. 9, pp. 110–117, Sep. 2013.
[148] S.-Y. Wang, "Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2014, pp. 1–6.
[149] M. Gupta, J. Sommers, and P. Barford, "Fast, accurate simulation for SDN prototyping," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2013, pp. 31–36.
[150] *POX Controller*, accessed on Jun. 9, 2016. [Online]. Available: http://www.noxrepo.org/pox/about-pox/
[151] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, and R. Buyya, "CloudSimSDN: Modeling and simulation of software-defined cloud data centers," in *Proc. 15th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2015, pp. 475–484.
[152] T. R. Henderson, T. R. Henderson, M. Lacage, and G. F. Riley, "Network simulations with the ns-3 simulator," in *Proc. SIGCOMM DEMO*, 2008, p. 527.
[153] *Trema*, accessed on Jun. 9, 2016. [Online]. Available: https://trema.github.io/trema/
[154] M. B. Al-Somaidai and E. B. Yahya, "Survey of software components to emulate OpenFlow protocol as an SDN implementation," *Amer. J. Softw. Eng. Appl.*, vol. 3, no. 6, pp. 74–82, Dec. 2014.
[155] A. Mahmud, R. Rahmani, and T. Kanter, "Deployment of flow-sensors in Internet of Things' virtualization via OpenFlow," in *Proc. 3rd FTRA Int. Conf. Mobile, Ubiquitous, Intell. Comput.*, Jun. 2012, pp. 195–200.
[156] A. Hakiri, P. Berthou, A. Gokhale, and S. Abdellatif, "Publish/subscribe-enabled software defined networking for efficient and scalable IoT communications," *IEEE Commun. Mag.*, vol. 53, no. 9, pp. 48–54, Sep. 2015.
[157] S. Shanmugapriya and M. Shivakumar, "Context based route model for policy based routing in WSN using SDN approach," in *Proc. BGSIT Nat. Conf. Emerg. Trends Electron. Commun.*, 2015, pp. 1–8.
[158] R. Sherwood *et al.*, "FlowVisor: A network virtualization layer," Stanford Univ., Palo Alto, CA USA, Tech. Rep. OPENFLOW-TR-2009-1, 2009, pp. 1–13.
[159] R. Sayyed, S. Kundu, C. Warty, and S. Nema, "Resource optimization using software defined networking for smart grid wireless sensor network," in *Proc. 3rd Int. Conf. Eco-Friendly Comput. Commun. Syst.*, Dec. 2014, pp. 200–205.
[160] A. S. Yuan, H.-T. Fang, and Q. Wu, "OpenFlow based hybrid routing in wireless sensor networks," in *Proc. IEEE 9th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process. (ISSNIP)*, Apr. 2014, pp. 1–5.
[161] C. Perkins, E. Belding-Royer, and S. Das, *Ad Hoc On-Demand Distance Vector (AODV) Routing*, document RFC 3561, Internet RFCs, 2003, pp. 1–37.
[162] *Open vSwitch*, accessed on Nov. 24, 2015. [Online]. Available: http://openvswitch.org/
[163] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, Feb. 2013.
[164] A. Voellmy, H. Kim, and N. Feamster, "Procera: A language for high-level reactive network control," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2012, pp. 43–48.
[165] N. Foster *et al.*, "Frenetic: A network programming language," *ACM SIGPLAN Notices*, vol. 46, no. 9, pp. 279–291, Sep. 2011.
[166] A. Voellmy and P. Hudak, "Nettle: Taking the sting out of programming network routers," in *Proc. 13th Int. Conf. Practical Aspects Declarative Lang.*, 2011, pp. 235–249.
[167] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing software-defined networks," in *Proc. 10th USENIX Conf. Netw. Syst. Design Implement.*, 2013, pp. 1–14.
[168] T. L. Hinrichs, N. S. Gude, M. Casado, J. C. Mitchell, and S. Shenker, "Practical declarative network management," *Proc. 1st ACM Workshop Res. Enterprise Netw.*, 2009, pp. 1–10.
[169] K.-K. Yap *et al.*, "Blueprint for introducing innovation into wireless mobile networks," in *Proc. 2nd ACM SIGCOMM Workshop Virtualized Infrastruct. Syst. Archit. (VISA)*, 2010, pp. 25–32.
[170] H. Huang, J. Zhu, and L. Zhang, "An SDN_based management framework for IoT devices," in *Proc. 25th IET Irish Signals Syst. Conf. China-Ireland Int. Conf. Inf. Commun. Technol. (ISSC/CIICT)*, Jun. 2014, pp. 175–179.
[171] K. Feng, X. Huang, and Z. Su, "A network management architecture for 6LoWPAN network," in *Proc. 4th IEEE Int. Conf. Broadband Netw. Multimedia Technol. (IC-BNMT)*, Oct. 2011, pp. 430–434.
[172] H. Mukhtar, K. Kang-Myo, S. A. Chaudhry, A. H. Akbar, K. Ki-Hyung, and S.-W. Yoo, "LNMP-management architecture for IPv6 based low-power wireless personal area networks (6LoWPAN)," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2008, pp. 417–424.
[173] H. Choi, N. Kim, and H. Cha, "6LoWPAN-SNMP: Simple network management protocol for 6LoWPAN," in *Proc. 11th IEEE Int. Conf. High Perform. Comput. Commun.*, Jun. 2009, pp. 305–313.
[174] R. Wallner and R. Cannistra, "An SDN approach: Quality of service using big switch's floodlight open-source controller," in *Proc. Asia-Pacific Adv. Netw.*, vol. 35. 2013, p. 14.
[175] *Ryu SDN Framework*, accessed on Jan. 10, 2017. [Online]. Available: https://osrg.github.io/ryu/
[176] T. Tsou, P. Aranda, H. Xie, R. Sidi, H. Yin, and D. Lopez, "SDNi: A message exchange protocol for software defined networks (SDNS) across multiple domains," in *Proc. Internet Eng. Task Force*, 2012, pp. 1–14.
[177] J. Stribling *et al.*, "Flexible, wide-area storage for distributed systems with WheelFS," in *Proc. 6th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Apr. 2009, pp. 43–58.
[178] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a model-driven SDN controller architecture," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, Jun. 2014, pp. 1–6.
[179] Z. K. Khattak, M. Awais, and A. Iqbal, "Performance evaluation of OpenDaylight SDN controller," in *Proc. 20th IEEE Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2014, pp. 671–676.
[180] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "Rules placement problem in OpenFlow networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1273–1286, 2nd Quart., 2015.
[181] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of Things: Vision, applications and research challenges," *Ad Hoc Netw.*, vol. 10, no. 7, pp. 1497–1516, Sep. 2012.
[182] J. Saraswat and P. P. Bhattacharya, "Effect of duty cycle on energy," *Int. J. Comput. Netw. Commun.*, vol. 5, no. 1, pp. 125–140, 2013.
[183] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient OpenFlow-based networking," in *Proc. IEEE Netw. Oper. Manage. Symp.*, Apr. 2012, pp. 933–939.
[184] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, p. 473, 2012.
[185] M. Sneps-Sneppe and D. Namiot, "Metadata in SDN API for WSN," in *Proc. 7th Int. Conf. New Technol., Mobility Secur. (NTMS)*, Jul. 2015, pp. 1–5.
[186] X. Feng, J. Shen, and Y. Fan, "REST: An alternative to RPC for Web services architecture," in *Proc. 1st Int. Conf. Future Inf. Netw. (ICFIN)*, Oct. 2009, pp. 7–10.
[187] R. Perrey and M. Lycett, "Service-oriented architecture," in *Proc. Symp. Appl. Internet Workshops*, Jan. 2003, pp. 116–119.

HLABISHI I. KOBO received the B.Sc. and M.Sc. degree in computer science from the University of the Western Cape in 2010 and 2012, respectively. He is currently pursuing the Ph.D. degree with the University of Pretoria. He was with Telkom SA as a Technology Architect. He is currently a Ph.D. Researcher with the Council for Scientific and Industrial Research. His main interests are software defined networking, software defined wireless sensor networks, and software architecture.

ADNAN M. ABU-MAHFOUZ received the M.Eng. and Ph.D. degrees in computer engineering from the University of Pretoria. He is currently a Principal Research Engineer with the Council for Scientific and Industrial Research. He is also an Adjunct Research and Innovation Associate with the Faculty of Engineering and Built Environment, Department of Electrical Engineering/ French South African Institute of Technology, Tshwane University of Technology. He is the Chair of the Tshwane Water Resource Management Network. His research interests are wireless sensor networks, software-defined wireless sensor networks, network management, network security, localization systems, and low-power wide area networks.

GERHARD P. HANCKE is currently an Assistant Professor with the Department of Computer Science, City University of Hong Kong. He received the B. Eng. and M.Eng. degrees in computer engineering from the University of Pretoria, South Africa, in 2002 and 2003, respectively, and the Ph.D. in computer science for the Computer Laboratory, Security Group, University of Cambridge, in 2008. He was with the Smart Card Center and Information Security Group, Royal Holloway, University of London. His main interests are sensing applications and security of embedded systems.