Can SDN technology be transported to software-defined WSN/IoT?

Thi Minh Chau Nguyen, Doan B. Hoang, Zenon Chaczko
University of Technology Sydney
Faculty of Engineering and IT
Sydney, Australia
ThiMinhChau.Nguyen@student.uts.edu.au, Doan.Hoang@uts.edu.au, Zenon.Chaczko@uts.edu.au

***Abstract*— Wireless sensor networks (WSNs) are essential elements of the Internet of Things ecosystem, as such, they encounter numerous IoT challenging architectural, management and application issues. These include inflexible control, manual configuration and management of sensor nodes, difficulty in an orchestration of resources, and virtualizing sensor network resources for on-demand applications and services. Addressing these issues presents a real challenge for WSNs and IoTs. By separating the network control plane from the data forwarding plane, Software-defined networking (SDN) has emerged as network technology that addresses similar problems of current switched-networks. Despite the differences between switched network and wireless sensor network domains, the SDN technology has a real potential to revolutionize WSNs/IoTs and address their challenging issues. However, very little has been attempted to bring the SDN paradigm to WSNs. This paper identifies weaknesses of existing research efforts that aims to bring the benefits of SDN to WSNs by mapping the control plane, the OpenFlow protocol, and the functionality between the two network domains. In particular, the paper investigates the difficulties and challenges in the development of software-defined wireless sensor networking (SDWSN). Finally, the paper proposes VSensor, SDIoT controller, SFlow components with specific and relevant functionality for an architecture of an SDWSN or SDIoT infrastructure.**

Keywords—Software-defined networking, software-defined wireless sensor network, software-defined IoT

I. Introduction

Wireless sensor networks (WSNs) were developed in the early 2000s and are now widely used in monitoring and tracking applications, e.g., seismic and structural monitoring, inventory location monitoring, indoor or outdoor environmental monitoring, power monitoring, and health and wellness monitoring as well as humans, vehicles, objects, or animals tracking [1]. However, as part of the IoTs era where every network-capable devices are connected as part of some Internet applications, WSNs have to overcome their limitations such as limited storage, low processing capacity short transmission range, high energy consumption, and underutilization. They are also application-dependent [1] and hard to manage with manual configuration and rigid policy [2]. Architecture, routing protocols, energy minimization algorithms and management schemes have been proposed, but their complexity prevents them from widespread deployment.

The Internet has grown into a huge and global interconnecting infrastructure, yet this conventional network is still using complicated and cumbersome network management systems that deal individually and manually with numerous network elements; complex and intertwined distribution of network control and transport protocols [3]; and rigid support for applications and services. In particular, a network device cannot be updated, replaced, or reconfigured easily without affecting other network devices because its distributed control plane and data plane are both embedded in the device itself. New protocols or network architectures cannot be introduced for innovative and emerging applications. A new networking technology is needed and SDN is not only that technology but also a new networking paradigm because the ideas and principles behind SDN are applicable to control, management, and other networking domains. SDN attracts not only academia but also the networking industry with its four key benefits [3]: 1) the separation of the control plane and the data plane, allowing them to evolve independently and leaving networking devices simple to forward data efficiently; 2) the centralization of network control at a controller external from the network device (the SDN controller or a Network Operating System (NOS)); 3) the network programmability via software applications running at the control or application planes; and 4) the use of flow-based forwarding rules instead of destination-based decisions. Despite the differences between switched network and wireless sensor network domains, the SDN technology has a real potential to revolutionize WSNs/IoTs and address their challenging issues. However, very little has been attempted to bring the SDN paradigm to WSNs. Several surveys on various efforts in extending SDN paradigm to WSNs appeared recently in [4, 5]. Reference [5] presents a general view of work on SDN to SDWSN migration without analyses of any specific SDWSN aspects. State of the art of SDWSN research is discussed in [4] which reviews recent studies extending SDN paradigm not only to sensor networks but also to other wireless networks including cellular, mesh and home networks. In [4], related works are classified into three groups: solutions to challenges of SDWSN design, resource allocation and management, and hierarchical scalable architectures. However, there is a little in-depth discussion on SDWSN style of networking. This paper reviews recent efforts in transporting SDN to SDWSN with full benefits. In particular, the paper discusses issues and challenges in applying software-defined paradigm to WSNs/IoTs. Finally, the paper proposes an architecture with relevant SDIoT/SDWSN controller, SFlow, and VSensor components for software-defined WSN and software-defined IoT environment.

The remainder of this paper comprises four sections. Section II discusses advantages and challenges in adapting software-define paradigm to WSNs. Section III synthesizes recent works regarding OpenFlow-based SDN architectures, SDWSN controller's placement, SDWSN southbound interface (SBI), the protocol stack of sensor nodes and sink nodes. Section IV discusses and proposes the functionality of the control plane, the SBI, and the data plane with virtual sensors. Section V concludes this paper.

II. SDN PARADIGM: BENEFITS AND CHALLENGES TO WSNS

A. *SDN architecture*

SDN components consist of SDN devices, controllers and applications, falling into three main planes, namely data, control, and application planes respectively [3, 6] (Fig. 1). Communication between the control plane and the others is via two main interfaces, called SBI and northbound interface (NBI).

1) *Data plane*

The data plane comprises both virtual and physical SDN devices, known as SDN switches. These devices consist of functional elements including a packet-processing function, an abstraction layer, and an application programming interface (API), known as a SBI [7]. The packet-processing function is responsible for handling arriving packets based on specifications of flow entries in flow tables provided by the controller. The abstraction layer abstracts the SDN device as a set of flow tables. To allow SDN devices to upward interact with the control plane, a SBI is needed to define communication approaches, message types, and a secure communication channel between the two entities.

2) *Control plane*

The SDN control plane mainly 1) manages the infrastructure layer and implements policies to the data plane via the SBI, and 2) provides a global view of the underlying network for the application plane via the NBI [3, 6]. It includes basic components such as device manager, packet processing unit, topology manager, routing and SBI [8].

3) *Application plane*

The application plane houses network services and applications. Via a highlevel programming language in the control plane, the applications can access the global network view and use the underlying services to execute a function. In particular, requirements for a SDN application are defined and translated into commands to program SDN switches [3]. Network services are used to execute network applications and provide them with APIs to communicate with other planes.

4) *Northbound interface (NBI)*

The NBI is a bridge between the application plane and the control plane. It allows end-host applications to autonomously and dynamically to send requests to the underlying network. The NBI enables application developers to control and program the network [6]. It provides the application plane with the abstraction of low-level instructions to allow the SBI to configure SDN devices. The NBI is defined as a software system, not a hardware one. Requirements for different network applications are different, so NBIs are various. Currently, there is a wide range of NBIs as RESTful APIs, Ad Hoc APIs, file systems, and other specific APIs like SDMN API, NVP NBAPI. However, none of them is considered as a standardized NBI [3].

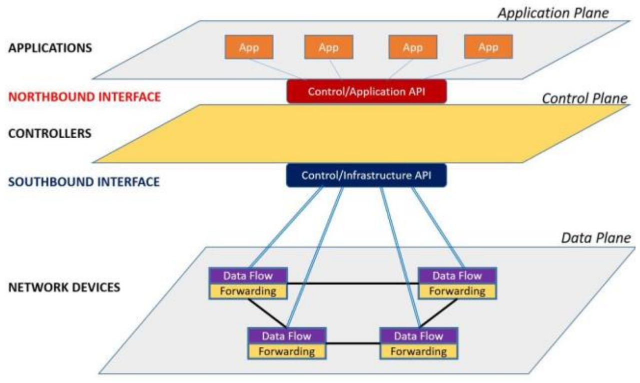

Fig. 1. SDN architecture [8]

5) *Southbound interface (SBI)*

SBI is a bridge between the control plane and the data plane [3]. It offers an interaction method between the control and data elements, as well as a set of instructions to forwarding devices. Some SDN controllers may support a single type of SBI. SDN SBI proposals include SoftRouter, ForCES, and OpenFlow. They are different in terms of architecture, design, protocol interface and forwarding model. By comparing their differences and similarities, OpenFlow-based SDN provides higher flexibility and control in terms of development, administration, and network management [3, 9]. It is not surprised that OpenFlow-based SDN paradigm has been applied to a number of proposed SDWSN architecture [2, 10-15].

The OpenFlow protocol allows SDN switches to interact with the control plane. SDN switches with OpenFlow implementation include three main components, namely flow tables, OpenFlow secure channel, and OpenFlow specification.

B. *SDN paradigm and its implication on WSNs*

Software-defined networking is not just a new networking technology; it is a new paradigm that opens up explorations and solutions in provisioning and management of resources from infrastructure to applications and services. The logically centralized control allows controllers to gain a complete information base of the underlying network for optional and real-time provisioning of network services. Programmability of the control plane allows autonomous configuration and management of network devices. Virtualization of resources allows physical resources to be virtualized and support simultaneously multiple services and users.

WSNs do not really operate the same way as switched networks, but they exhibit many similar characteristics. Networked sensors are network devices and they have to be configured and managed by their controllers. WSNs often organized into clusters and managed by cluster controllers. In a comprehensive application, a WSN may employ a large number of sensor nodes. Clearly, the above SDN paradigm would bring benefits to WSNs if applied appropriately. Efforts have been made to realize these benefits as follows.

Simple sensor node and energy saving [11, 16]: sensor nodes simply forward data based on the decision of the control logic. Energy consumption of a sensor node is thus reduced.

Routing protocol: Routing technique can be highly improved in OpenFlow-based SDWSN structure. Yuan, et al. [17] design a new routing protocol integrating OpenFlow protocol and a wireless sensor link-state routing protocol which can send less than a half control packets per minute and independently operate when the controller fails. With similar concern, Han and Ren [18] has proposed a new routing protocol in a clustered SDN-based WSN structure. Compared to LEACH, LEACHM, and DEEC, the routing protocol has better performance concerning the death node number, the network lifetime, and especially a greater advantage in data transmission.

Network management: network characteristics can be remotely and centrally configured instead of manually and individually reconfigured. A smart network management has been proposed in [11] to address WSN problems such as power consumption, sensor node mobility, localization and topology discovery, and network management. Nonetheless, the proposal has not been evaluated for its efficiency; the authors suggest as a future research, to estimate the method's performance regarding reliability and security. Furthermore, it focuses on the controller architecture without discussion of the SBI.

Network programmability and innovation: network policies can be programmed by the software running on the control plane [3, 8] by defining flow entries. This enables a higher degree of innovation in designing network protocols.

Network function virtualization [8]: a sensor node can be virtualized to perform the desired network function through a hypervisor in the controller. This opens up a new dimension of services and services provisioning.

Efficient network utilization and services development: with the global view of the underlying network, the controller is able to create virtual networks based on its network abstraction and virtualizes sensor nodes to handle specific-purpose applications. This allows deployment of multiple WSN applications over a single physical WSN [2, 19]. Moreover, the controller can flexibly allocate appropriate network resources to an application. This allows the developments of infrastructure as a service [19] and platform as a service.

Energy efficiency: This can be achieved with appropriate SDWSN architectures that enable efficient transmission of packets over WSNs [16].

Cloud integration: A cloud can play an integral part in WSN applications by releasing sensor nodes the burden of data storage and data processing. With the extension, sensing as a service can be developed with SDWSN.

C. Challenges in translating SDN to SDWSN

SDN is originally designed for wide area networks with powerful switching and routing devices, so it is difficult to completely apply SDN principles to WSNs because of the constraints of sensor nodes and the limitations of the wireless medium. Essentially, sensors are not switched network devices and hence they are limited in their capability. Furthermore, they do not always use IP protocol for communication. Developing an OpenFlow-based SDWSN model may encounter many technical challenges.

1) Designing an OpenFlow-based SDWSN SBI

Many difficulties are encountered in emulating an OpenFlow-style SBI because of the differences in the functionality of a switched network device and a sensor.

- Data Plane – Flow creating: typical WSNs employ different addressing as attribute-based naming instead of using IP-like addressing, while packets are processed based on flow entries using IP addresses [2]. This prevents the SDWSN SBI from creating flow entries, so two methods are suggested: 1) modifying matching field of flow tables or 2) using uIP/uIPv6 or Blip [2].
- Secure channel establishment: a TCP/IP connection between the control plane and the data plane is established based on the IP addresses from two participants in the communication [2].
- In-network processing module is needed to wirelessly and remotely update sensor firmware and software according to future demands [2].
- Control traffic overhead: the secure channel can be only hosted with in-band management or out-of-band management in wired networks, whereas only in-band management is supported in wireless networks, leading to an overhead of both data and control traffic [2, 10].
- Traffic generation: traffic is has to be generated to conform with the flow entry definition of OpenFlow specifications, so a traf-gen module is needed for each sensor node [2].
- Power efficiency: it is essential to support duty cycles [15] to periodically turn off the radio, and in-network data aggregation to remove redundant data [2, 15]. This is addressed by an in-net proc module [2] or an additional aggregation player in the protocol architecture of sensor nodes, or new actions in flow tables [15].
- Backward and peer compatibility: SDWSN design is expected to be compatible with traditional networks without SDWSN SBI or OpenFlow support [2].

2) Designing a SDWSN node

Without changing the basic functionality of a sensor, the challenge is to empower it with adequate capability for software-defined control: a capacity for flow table storage, and capability for handling flow requests from low-tier nodes or the controller. Following aspects should be taken into consideration in designing a SDWSN node.

- New hardware architecture for sensor nodes may be needed to allow flow-table implementation from the controller. Additionally, memory capacity can sufficiently store the implemented flow tables.
- Processing speed: SDWSN nodes have to handle a large number of requests from applications and other nodes.

Current switches are unlikely to tackle flow demands from applications because the SDN devices only forward data and frequently request instructions from the controller to handle arriving packets. This leads to poor performance of the controller regarding processing power and switch-controller link congestions [9].

- Standardized protocol stack: to support network virtualization requires accesses to heterogeneous sensor nodes to create multiple virtual WSNs for various WSN applications. Moreover, sensor nodes with different higher protocol layers are difficult to migrate between different networks and communicate with sensor nodes in these networks. Well-defined functional layers are needed for a proper interaction between the nodes and the control software.

III. CURRENT RESEARCH ON SDWSN

Most of the ongoing research efforts propose their models based on the OpenFlow-based SDN principles. These proposals can be classified into four groups in terms of SDWSN architecture, SDNWSN controller position, SBI design, the protocol stack of sensor nodes and sink nodes.

A. *SDWSN architecture*

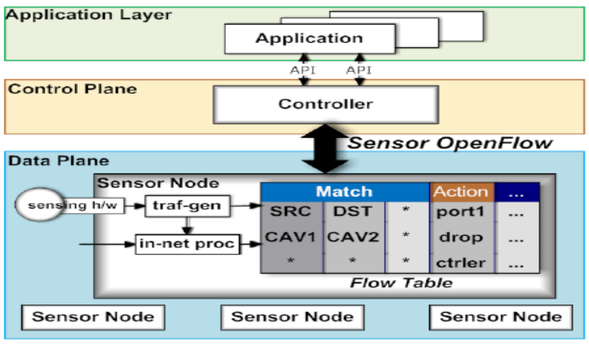

Fig. 2. Software-defined wireless sensor network architecture [2]

Early research efforts in SDN-WSN integration have focused on defining a meaningful and effective architecture. They focus on different aspects of the architecture: SDN architecture for WSNs [20], multi-controller support [10], controller placement [11], and controller core functions [11]. Generally, following the OpenFlow-based SDN structure, SDWSN architecture is structured into three main planes: application, control and data planes [2, 11, 20] (Fig. 2). The data plane composes of sensor nodes forwarding flows of packets. The control plane is responsible for deploying the desired network management policy [15], routing and performing routing and QoS network control [2]. Specifically, the controller is responsible for flow- table definition, mapping function, and mapping information. SDWSN applications can be implemented on top of the control plane [20]. The applications can be associated with the networking of the WSN, e.g., topology control and routing [20]. APIs are needed to allow communications between the planes. However, no attention is paid to the SDWSN NBI and only a few designs of SDWSN SBIs are proposed in [2, 12, 15], but they mainly rely on the OpenFlow.

B. *Controller placement*

TABLE I. SDWSN CONTROLLER POSITIONS

SDWSN structure	Controller Position	SDWSN switch function
Layered SDWSN architecture		
Type 1	Remote server	Sink or gateway
Type 2	Base station	N/A
Clustered SDWSN architecture		
Type 1	Cluster head	Gateway/sink/base station
Type 2	Cluster head	Sensor nodes
Type 3	Remote server	Cluster head

Looking from the architectural angle, the primary concern of SDWSN control plane is if it is distributed or centralized [3]. The position of the controller significantly impacts the SDWSN performance in terms of energy consumption, scalability and reliability. Most of SDWSN architectures are proposed with both centralized [11, 14, 16, 18, 20] and distributed control plane [10, 12, 15, 21, 22]. The two approaches result in differences in position and function of the SDWSN controller and SDWSN nodes (Table I).

The controller can be deployed in a base station [11, 16], a remote server [15], a sink [15], or in a cluster head [16, 18, 22]. In particular, in the centralized model, the central controller is responsible for the whole network and it can reduce synchronization time between distributed control planes. However, a network with only one controller may face with an insufficient undertaking of a huge number of flows, and difficult management of a large scale network [6]. Moreover, it may become a single point of failure and limited in scalability [3]. Alternatively, the distributed structure has a flexible scalability and resilience varying to different network scales [3], but they may encounter a consistency issue. Placing the controller at the cluster head can bring benefits such as power reduction [16], network stability, routing efficiency [18], and fewer messages exchanged among sensor nodes . In a clustered network, a sensor node can act as a gateway, sensor node enabling SDWSN SBI, a local cluster head or a SDWSN switch.

C. *SDWSN SBI*

To be an OpenFlow-based SBI, the SDWSN SBI must preserve essentially the propertied of the OpenFlow protocol. However, because of the different characteristics between the switched network and the wireless sensor network, major modifications to the original design of the OpenFlow protocol (mentioned in section II.C) are required for it to be applicable to WSNs. To meet those requirements, three proposals to SDWSN SBIs are proposed, namely sensor OpenFlow protocol [2], SDWN protocol [15], and SDN-WISE protocol [12], an extension of the SDWN protocol. Properties of the SDWSN SBIs are presented in Table II.

Lou, et al. [2] has initially analyzed challenges of applying SDN to SDWSN and suggested corresponding solutions. However, they mainly focus on ideas without performance evaluation of their proposal or specifications for sensor

OpenFlow in terms of message type, packet format and operation. These limitations have been improved by the proposal of SDWN [15] which consists of sensor OpenFlow features and offers other significant features like duty cycles configuration, flexible definition of flow entries (or called flow rules), in-network data aggregation, and actions to enable cross-layer optimizations. Nonetheless, the proposal fails to provide protocol details and performance evaluation. Only SDN-WISE protocol [12] presents a more completed proposal supporting flexible rules extended from the rule definitions proposed in [15]. Specially, the SDN-WISE is stateful protocol compared to the stateless traditional OpenFlow. Furthermore, SDN-WISE has been simulated with OMNet++ simulator [12]. To optimize the power usage in WSNs, the SDN-WISE protocol is designed to support duty cycle and data aggregation, but the there is no evaluation for the two features.

TABLE II. SPECIFICATIONS OF CURRENT PROPOSALS FOR SDWSN SBI

Features	Sensor OpenFlow [2]	SDWN [15]	SDN-WISE [12]
Flow entry details	Yes	Yes	Yes
Matching field	Yes	Yes	Yes
Action field	No	Yes	Yes
Statistic	No	No	Yes
Packet header details	No	Yes	Yes
Message type details	No	Yes	Yes
Duty cycle	No	Yes	Yes
Data aggregation	No	Yes	Yes
In-band management	Yes	Yes	Yes
Out-of-band management	Yes	No	No
Implementation	No	No	Yes
Stateful protocol	No	No	Yes

D. Protocol stack of sensor nodes and sink node

To support SDWSN SBIs that behave like OpenFlow, supporting protocol stacks of both controller and sensor nodes are firstly proposed in [15] and extended in [12].

1) The sink node

The control logic may be deployed in a sink node [15] or in a remote server [12], called controller node. The controller node consists of two different parts (Fig. 3): the lower part for handling the communication with generic nodes and the upper controller part for handling the controller's functionality and network virtualization [12, 15]. The controller part comprises three layers, called Adaption, Virtualizer [15], [12] and Control layers. They are responsible for formatting messages according to the SDN-WISE protocol, for creating abstractions of the underlying networks used for network virtualization, and for network management policy (e.g., creating flow tables and defining appropriate rules for receiving packets) respectively.

2) The sensor node

Regular sensor nodes enabling SDWSN SBI are called software-defined wireless sensor nodes (SDWS nodes). The protocol stack of SDWS node is defined in [12, 15], as depicted in Fig. 4. In addition to the regular layers, MAC and PHY, other three layers are needed, such as FWD (processing incoming packets in accordance to flow rules specified by the controller), AGGR [15] or INPP [12] (performing tasks related to in-network processing or data aggregation), and NOS or TD (controlling layers as FWD, AGGR or INPP).

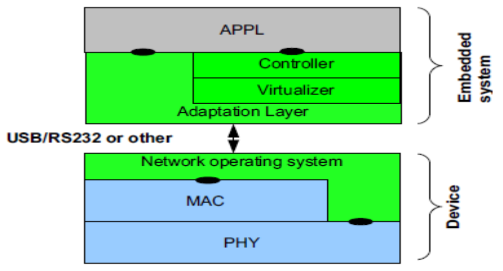

Fig. 3. Protocol stack of SDWS controller node [15]

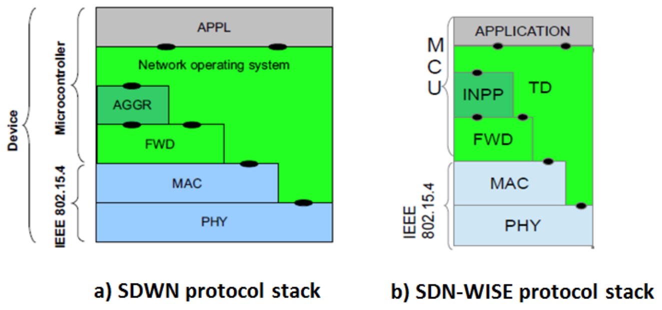

Fig. 4. a) SDWN [15] and b) SDN-WISE [12] generic-node protocol stack

IV. SDWSN/SDIOT PROPOSAL

Although many research efforts have extended the SDN paradigm into WSNs, not much attention has been directed to the design of the controller in the control plane, the functionality of sensors in the data plane, and the API between them. Most try to reuse the OpenFlow with minor modifications and with little consideration of the WSN environment. It is clear from our review and discussion; we observe the following.

Sensors are not network devices in the traditional switched network and hence they should not be concerned with the heavy-duty source-destination routing functionality. The functionality and characteristics of sensors should be preserved: simple data sensing point, minimum energy consumption, and simple operation.

A controller in this SDWSN environment should not be burdened with the heavy-duty of a SDN controller with complex network operating system functionality; it should only be concerned with managing and configuring its sensors and sharing sensor recourses among its applications. Its scope should be confined to the sensor services that can be offered to the applications.

OpenFlow protocol a) is relevant for SDN switched devices but is far too heavy and many features are unnecessary for sensor devices, and b) does not configure devices but requires OF-CONFIG to do so. From these observations, we propose the following:

Creating a VSensor class: VSensor is a software-defined virtual sensor. VSensor can be seen as an entity that represents a single or multiple physical sensors, a single or multiple software sensors (software interrupts, software alerts, software agents, etc.) VSensor can be programed, configured and managed by its controller in only relevant aspects of a sensor, not an SDN network device.

Creating a SDWSN or SDIoT controller: The controller will have few specific functions: managing, configuring, programming VSensors, virtualizing sensor resources under its control and providing sensor services as part of an overall application. An SDIoT controller an also be considered as an end user of a network endpoint that generates sensor data and launches it into the interconnecting network infrastructure (the IoT Internet, SDN networks, switched networks, clouds, etc).

Creating a SFlow southbound protocol: The protocol is created in the same spirit/style as the OpenFlow, but it is not for SDN devices. That means that it is not designed for routing TCP/IP or UDP/IP ultimate source and destination flows. Furthermore, SFlow will incorporate simple protocol for configuring VSensor as an integral part of the SBI.

From the proposed three components, it is clear that the three planes of SDN along with the benefits of the technology are preserved. However, SDWSN/SDIoT can operate separately from SDN or below the data plane of SDN. With this proposal, SDWSN or SDIoT can easily be integrated into a global software defined infrastructure (SDN, NFV, Cloud, IoTs) and still preserves the simplicity and economy of sensors and benefits of software-defined centralized control, virtualization, programmability, and autonomous management and configuration. For example, with the provision of classes of VSensor, any request of WSN/IoT applications can be handled by tailoring or extending the VSensor class. The SDWSN/SDIoT controller can create on-demand VSensors or networks of VSensors via the proposed SFlow southbound protocol. The controller is able to create a graph of virtual sensor networks and to configure VSensors in accordance to the application's demands. The noting point in the design is that the SDWSN/SDIoT network would enable application designers to design their applications without knowledge of the underlying infrastructure, but they can control any connecting underlying sensors through the VSensor. Currently, we are in the process of designing and implementing the proposed architecture and its components.

V. CONCLUSION

This paper discusses advantages and challenges to the SDN-WSN integration. Ongoing research efforts extending SDN paradigm into WSNs are discussed in terms of the proposed SDWSN architectures, SDWSN controllers and their placement, SBI specifications, and protocol stack of SDWSN sensor nodes. Based on these studies and observations, the paper proposes a SDN-style architecture and defines the functionality of components of the control plane, the data plane, the interface between the two planes for SDWSN/SDIoT, and trust aspects of IoTs [23].

REFERENCES

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks,* vol. 52, pp. 2292-2330, 2008.

[2] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor openflow: enabling software-defined wireless sensor networks," *Communications Letters, IEEE,* vol. 16, pp. 1896-1899, 2012.

[3] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE,* vol. 103, pp. 14-76, 2015.

[4] I. T. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: a survey and taxonomy," *IEEE Communications Surveys & Tutorials,* vol. PP, pp. 1-1, 2016.

[5] N. A. Jagadeesan and B. Krishnamachari, "Software-defined networking paradigms in wireless networks: a survey," *ACM Computing Surveys (CSUR),* vol. 47, p. 27, 2014.

[6] J. Xie, D. Guo, Z. Hu, and T. Qu, "Control plane of software defined networks: a survey," 2015.

[7] N. V. R. Gupta and M. Ramakrishna, "A Road Map for SDN-Open Flow Networks," *International Journal of Modern Communication Technologies & Research (IJMCTR),* vol. 3, 2015.

[8] D. Hoang, "Software defined networking–shaping up for the next disruptive step?," *Australian Journal of Telecommunications and the Digital Economy,* vol. 3, 2015.

[9] H. Fei, H. Qi, and B. Ke, "A survey on software-defined network and openflow: from concept to implementation," *IEEE Communications Surveys & Tutorials,* vol. 16, pp. 2181-2206, 2014.

[10] B. Trevizan de Oliveira, M. C. Borges, and G. L. Batista, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," in *Communications (LATINCOM), 2014 IEEE Latin-America Conference on*, 2014, pp. 1-6.

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

[12] 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 *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 513-521.

[13] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "Reprogramming wireless sensor networks by using sdn-wise: a hands-on demo," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS),* 2015, pp. 19-20.

[14] A. Mahmud and R. Rahmani, "Exploitation of OpenFlow in wireless sensor networks," in *2011 International Conference on Computer Science and Network Technology (ICCSNT)*, 2011, pp. 594-600.

[15] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks: unbridling sdns," in *European Workshop on Software Defined Networking (EWSDN)*, 2012, pp. 1-6.

[16] P. Jayashree and F. Infant Princy, "Leveraging sdn to conserve energy in wsn-an analysis," in *3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, 2015, pp. 1-6.

[17] A. S. Yuan, H.-T. Fang, and Q. Wu, "OpenFlow based hybrid routing in Wireless Sensor Networks," in *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2014, pp. 1-5.

[18] Z.-j. Han and W. Ren, "A novel wireless sensor networks structure based on the SDN," *International Journal of Distributed Sensor Networks,* vol. 2014, pp. 1-7, 2014.

[19] A. Mahmud, R. Rahmani, and T. Kanter, "Deployment of flow-sensors in internet of things' virtualization via openflow," in *Third FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing (MUSIC),* 2012, pp. 195-200.

[20] M. Jacobsson and C. Orfanidis, "Using software-defined networking principles for wireless sensor networks," in *11th Swedish National Computer Networking Workshop (SNCNW)*, Karlstad, Sweden, 2015.

[21] R. Sayyed, S. Kundu, C. Warty, and S. Nema, "Resource optimization using software defined networking for smart grid wireless sensor network," in *3rd International Conference on Eco-friendly Computing and Communication Systems (ICECCS),* 2014, pp. 200-205.

[22] F. Olivier, G. Carlos, and N. Florent, "SDN based architecture for clustered WSN," in *9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS),* 2015, pp. 342-347.

[23] T. Nguyen, D. Hoang, and A. Seneviratne, "Challenge-response trust assessment model for personal space IoT," in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, 2016, pp. 1-6.