Advances in Wireless Communications and Networks
2017; 3(2): 10-22
http://www.sciencepublishinggroup.com/j/awcn
doi: 10.11648/j.awcn.20170302.11

SciencePG
Science Publishing Group

Review Article

Software Defined Networking for Wireless Sensor Networks: A Survey

Muhammad Ali Hassan, Quoc-Tuan Vien, Mahdi Aiash

School of Science and Technology, Middlesex University, London, United Kingdom

Email address:

MH1516@live.mdx.ac.uk (M. A. Hassan), Q.Vien@mdx.ac.uk (Q.-T. Vien), M.Aiash@mdx.ac.uk (M. Aiash)

To cite this article:

Muhammad Ali Hassan, Quoc-Tuan Vien, Mahdi Aiash. Software Defined Networking for Wireless Sensor Networks: A Survey. *Advances in Wireless Communications and Networks.* Vol. 3, No. 2, 2017, pp. 10-22. doi: 10.11648/j.awcn.20170302.11

Received: April 24, 2017; **Accepted**: May 11, 2017; **Published**: May 28, 2017

Abstract: One main feature of Software Defined Networking (SDN) is the basic principle of decoupling a device's control plane from its data plane. This simplifies network management and gives network administrators a remarkable control over the network elements. As the control plane for each device within the network is now implemented on a separate controller, this reliefs individual devices from the overhead caused by complex routing. Specifically, this feature has been shown to be extremely beneficial in the case of resource-constrained Wireless Sensor Networks (WSNs). By keeping the control logic away from the low-powered nodes, the WSNs can resolve their major issues of resource underutilisation and counter-productivity. This paper highlights the importance of adopting the SDN in the WSNs as a relatively new networking paradigm. This is introduced through a comprehensive survey on relevant networking paradigms and protocols supported by a critical evaluation of the advantages and disadvantages of these mechanisms. Furthermore, open research issues and challenges are pointed out shedding a light on future innovations in this field.

Keywords: Software Defined Networking (SDN), Wireless Sensor Network (WSN), OpenDaylight, OpenFlow, Virtualised WSN

1. Introduction

The Internet has made an enormous impact on the world of communications. It has interconnected billions of networking devices all over the world [1]. These devices share information with each other in the form of digital data packets. The networking protocols are responsible for the delivery of these packets to their respective destinations. Despite their significance, these protocols have not evolved over the years, which are complex and thus restrict innovation. In the traditional network management, network administrators had to configure each device manually by using proprietary commands. Newer policies or protocols cannot be introduced in the network on the fly, while automatic reconfiguration and response mechanisms are almost non-existent. The still ongoing transition of protocol change from IPv4 to IPv6 shows how difficult it is to introduce new changes [2].

Network management is even more complicated when it comes to large data centres. In the past, storage, computing, and networking resources within a data centre were all kept separated from each other. This was done for the ease of management and also for the sake of security enhancement. With the growing demand of networking and computing resources, it became difficult to provide separated resources for these entities and the organisations were forced to consolidate resources. The reduced cost of micro-electromechanical systems and the advent of operating system (OS) virtualisation have recently facilitated such practical requirement by allowing the deployment of hundreds and even thousands of virtual machines on a few physical servers [3]. This however has brought issues of operational efficiency and power consumption. Furthermore, the virtualised environment also demands a unique IP address for each virtual machine (VM). This in turn presents a hurdle of managing and provisioning IP addresses and networking resources to a large number of VMs simultaneously, which is likely to cause data bottlenecks.

The aforementioned issues were the motivation behind the invention of Software Defined Networking (SDN) (see [4]-[6] and references therein). The micro-controller technology was

booming at a steady pace whereas the networking side merely made any significant progress in feature development or introducing new ideas. The SDN has therefore emerged as the future of modern day networking offering simplicity, scalability, versatility and innovation over the traditional networking models. The basic architecture of SDN separates the device's control plane from its data plane**.** The control plane for all of the devices inside the network is relocated to a remote site where the controller overlooks and manages the entire network. The decision making is done by the controller and then the instructions are sent to the data plane to implement those decisions. For example, in case of data congestion, the controller will make the decision to redirect the flow of traffic and order the devices to update their flow tables accordingly. This feature of traffic management is not possible with the traditional networking models as changes to the routing paths cannot be implemented directly. Some research also have a different view of SDN that they refer to as the software driven networks**.** They present a middle approach whereby some parts of the network are managed by the controller, while others are still managed by the more traditional control plane. Nevertheless, both have the same idea of a greater and more flexible network device programmability.

Additionally, SDN offers an energy efficient solution for power-constrained network elements such as in Wireless Sensor Networks (WSNs) which consists of a group of embedded devices called sensor nodes. WSNs is an example of a system that can benefit from this feature of the SDN. The sensor nodes collect numerous environmental data, such as temperature, light, humidity, pressure, sound, etc., and send it to base station [7], [8]. These nodes are deployed in areas where physical access to these devices might not always be possible and most of the times these nodes run on small limited batteries and may not have any renewable energy resource. Hence, these devices cannot either run complex protocols or perform heavy computational activities, which limits the functionality and efficiency of the entire network. The SDN provides an alternative networking model for these devices enabling them to not only run complex protocols but also customise the functionality of the network according to the needs. It simplifies the management of networking models and utilises networking resources more efficiently. These features are indeed highly suitable for the low-powered nodes in the WSNs. However, there has not been a significant approach towards introducing this concept to the WSN domain and the architectures for deploying the SDN in WSNs have not been practically tested and validated considering various scenarios in reality.

In this paper we aim at accomplishing the following:

i) To analyse of SDN and its effectiveness in the present networking environment.
ii) To investigate the current SDN solutions and their practicality for WSNs.
iii) To compare the SDN solutions available for wired and wireless networks with those available for the WSNs.
iv) To summarise the findings and to propose future work in this area.

Our contribution in this paper is to provide a comprehensive study on highlighting the importance of using SDN in WSN. We plan on achieving this by investigating the research work that has been done so far in this area and pointing out the key factors in SDN that can be of benefit to the WSN. We then present our findings along with the proposed future work.

The rest of this paper is organised as follows: Section 2 starts with the background of SDN and WSNs. We then describe various architectures of SDN for WSNs with relevant research works in Section 3. Section 4 summarises the findings and proposes future work. Finally, Section 5 draws the main conclusions from this paper.

2. Literature Review on WSN and SDN

2.1. Wireless Sensor Network (WSN)

A WSN is made up of a large number of small, low-cost, low powered sensor nodes. These nodes monitor environmental conditions, such as temperature, sound, pressure, humidity, etc., and then send that information wirelessly over the network to a host system where it is processed, analysed and presented in a readable format [7], [8]. These networks, as illustrated in Figure 1, have a wide range of applications. They can be used to monitor weather conditions on farm fields or to detect enemy's movements in warzones. They can also be used to monitor the traffic to keep it away from jams and accidents or to predict natural disasters such as volcanoes and earthquakes.

Figure 1. *WSN Environment [8].*

The history of WSNs dates back to several decades. According to a report published by the Silicon Labs on the evolution of WSNs [9], the Sound Surveillance System (SOSUS) was the first wireless system that shows any resemblance to the modern day WSN. It was invented by the US military to keep track on Soviet submarines. The system consisted of a large number of submerged acoustic sensors called hydrophones that were dispersed all over the Atlantic and Pacific Ocean. This detecting technology is still being used in some areas to monitor natural disasters.

The weakness of WSN lies in their limited processing power, storage memory, and communication infrastructure. In order to improve the overall performance, ensure reliability,

and make the system more secure, the system engineer and designers have to make trade-offs among the choice of underlying hardware, power resources, and networking protocols.

In spite of all the drawbacks, today's WSN differs greatly from the ones that were developed just a few years ago. In past, the aforementioned factors were a major hindrance in the development of these device but the importance of WSN in various modern applications and advancements in semiconductor and networking technologies have led to their large-scale production. These networks are now easier to manage, the devices have longer lifetimes and they are more resilient.

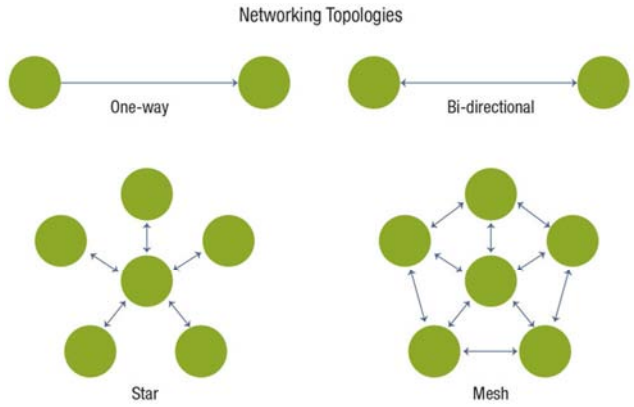

Figure 2. *WSN Topologies [10].*

WSNs are generally classified in four groups [10] including one-way networks, bi-directional networks, star networks and mesh networks. As shown in Figure 2, one-way WSN network topology is the simplest one with only a single, one-way communication link. An example of its use is in the pressure measuring systems. The advancement in technology leads to the need of more efficient topologies and cost-effective protocols for these designs. For example a star topology finds its use in easily scaling the number of lights in a room or a router in a house can use the mesh topology to overcome shadowing and ensure uniform signal strength throughout the house. The major area of concern is the security of these devices. The WSNs are generally set up for gathering records from insecure environments. The sensitive nature of the information carried by the nodes poses a great challenge for the developers to implement a secure framework for these devices so that the data cannot be corrupted.

2.2. Software Defined Networking (SDN)

SDN is a new networking model that separates the control and data planes of a device and makes the control plane programmable by using various APIs [4]-[6]. This results in an efficient, low-cost and dynamic networking architecture that provides network administrators with unprecedented control over the networking elements [9]. As a leading organisation that aims at promoting the SDN, the Open Networking Foundation (ONF) is supported by various companies, such as Cisco, Microsoft, Google, Deutsche Telekom, etc. [11].

The SDN is also defined as a network design approach that makes network management easier by closing the gaps between applications, network services, and devices [12]. This can be achieved by deploying a single centralised point of control which is commonly referred to as the SDN controller. The controller orchestrates and facilitates the correspondence between the applications and network devices. It exposes and abstracts the network functions and operations via programmatic interfaces to the network administrators, which gives them more control over the network functionality.

In a traditional networking environment, the control and data planes reside on the same device. The control plane, which can also be thought of as the brain of the networking device makes all the decisions regarding the routing tables. The data plane utilises these routing tables to forward the data packets. A device with a local control plane will have to be manually and separately configured. In a scenario where hundreds of such devices are to be managed, this can prove to be a tedious task. Moreover, no single device has the visibility of the entire network. In other words, each device has to work on its own and share information with its neighbours to form some sort of view of the network. Also, with the traditional networks, new routing protocols cannot be implemented readily. It is also difficult to integrate devices of different brands to run on the same network as they run proprietary software. For example, a network consisting of Cisco switches will only have those switches running in the network with their proprietary operating systems. It will be difficult to introduce a Dell switch within the network and to make it work smoothly alongside the other switches.

With SDN, rather than each device having its own control plane, a common control plane is implemented on a remote controller for all the devices in the network. This introduces a centralised control policy management. The devices are to become simple packet forwarding elements while all the decision makings are carried out at the remote controller. The controller can manipulate the flow of traffic throughout the network. This relieves the individual devices of the overhead to manage the routing protocols and policies on their own and also helps to manage the network traffic which prevents congestion in the network. With SDN, a user can run multiple operating systems on devices that are not application specific. For example, Facebook compute switches do not require a proprietary software so we can run different operating systems of our choice on different devices on the same network. In fact, all tend to agree that SDN does make the network management simpler [13].

An SDN architecture has three layers (see Figure 3) as follows:

a) Application Layer: It supports applications that communicate with the controller and direct it to perform the desired functions on the underlying physical network infrastructure. These applications also use data supplied by the controller to create a logical view of the entire network. This helps the network administrators in decision making regarding the network management.

These applications can also be used to perform data analysis. Business oriented applications are used to run large data centres or to detect suspicious network activity within the data centre for security purposes.

b) Control Layer: It holds the network controller which is the main entity that interlinks the application and infrastructure layer. The controller is responsible for managing the communication between the two layers. It conveys the instruction received from the applications to the underlying physical or virtualised devices and collects the data from these devices and send it back to the applications.

c) Infrastructure Layer: The infrastructure layer consists of physical networking devices that execute the actual data forwarding. This also includes the virtualised elements.

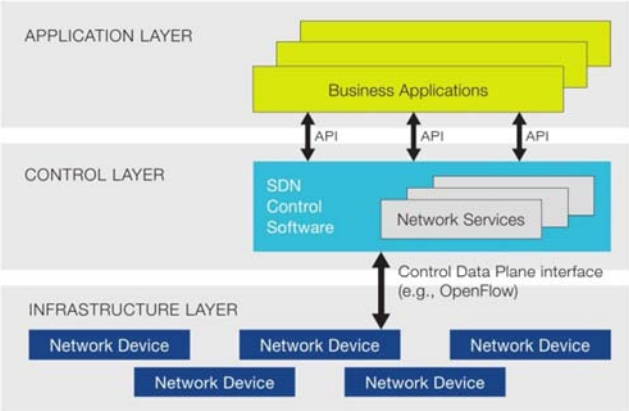

Figure 3. *SDN Architecture [14].*

The SDN architecture is usually described by two interfaces, namely the Northbound interface and the Southbound interface. The connection between the controller and applications is referred to as the Northbound interface, while the connection between the controller and the physical networking hardware is known as the Southbound interface. The SDN is basically based on four main pillars as follows [5]:

a) The control and data planes are to be separated from one another.

b) The forwarding decisions are to be flow based rather than destination based.

c) The control logic is to be moved to an external SDN controller.

d) The network control plane is to be made directly programmable.

Here, three major components of the SDN that can be listed are:

a) Control Plane: The main task of a control plane is to create data forwarding tables for the data plane [12]. The control plane makes these decisions based on the information provided by the Routing Information Base (RIB). RIB is the entity that stores the network topology. It gathers information through observation, manual programming or integrating with other entities of the control plane. Once these decisions are made, they are then stored in the Forwarding Information Base (FIB) which is responsible for forwarding the packets to their proper interfaces. The control plane can be of three following types:

a) Strictly Centralised: This approach to SDN model is referred to as "revolutionary approach" because it proposes a complete separation of the device's control plane from its physical infrastructure. In this model no control plane functions exist at a device and it acts under the total control of the remotely located centralised controller.

b) Semi Centralised: A semi centralised control plane is referred to as the "evolutionary approach". It provides some new capabilities but does not completely remove the control plane from the device. Some control plane functionalities such as learning of MAC addresses is still carried out on the device while the centralised controller is given more authority over other areas of network functionality. This model utilises the best features of both strictly centralised and fully distributed control planes.

c) Fully Distributed: In this model each device runs a complete control plane for each data plane. All the control planes are interlinked to form a cohesive network. This approach offers nothing new and is therefore of little significance.

b) Data Plane: A data plane in SDN is what carries out the actual data packet forwarding. The packets on a device are forwarded based on the flow tables assigned to them by the controller. A flow is a set of packet field values that filter the incoming packets. If a packet matches the criteria defined in a particular flow then corresponding actions are taken on that packet based on the instructions provided by the controller. All packets belonging to a particular flow will receive identical treatment. In case a packet does not belong to the listed flow table entries the device will then ask the controller to provide new instructions on dealing with that packet. The flow tables can be readily updated in case of any policy changes. Several methods have been proposed for cost-effective, fast packet forwarding [4]. Hardware classification can be used to increase processing throughput as using software in switching devices may result in inefficient performance. Another method is to classify the flows into "elephant flows" and "mice flows" categories. Mice flows are generally numerous but each of them have few packets. They also have little impact on the overall network performance. A proposed idea is to send "elephant flows" to the Application Specific Integrated Circuit (ASIC) and allow the Central Processing Unit (CPU) to deal with the "mice flows".

c) Management Plane: The Management plane is responsible for performing tasks that are outside the scope of control and data planes. It manages resource allocations, client-vendor business agreements, setting up of physical networking infrastructure, and configuring bootstraps. Every business organisation has its own administrative entities.

The entity that a control plane utilises to manage the flow controls in a network is called an SDN controller. The SDN controller, for example OpenDaylight (ODL) [15], [16] (see Figure 4 for a typical ODL architecture), should have the following [12]:

a) A database that stores information regarding network state, network configuration and network topology
b) A high-level data model that establishes relationships between the resources and the services provided by the controller
c) An API that offers the controller services to the application layer
d) A TCP control session between the controller and the devices
e) A standards based protocol
f) A topology discovery mechanism for path computation

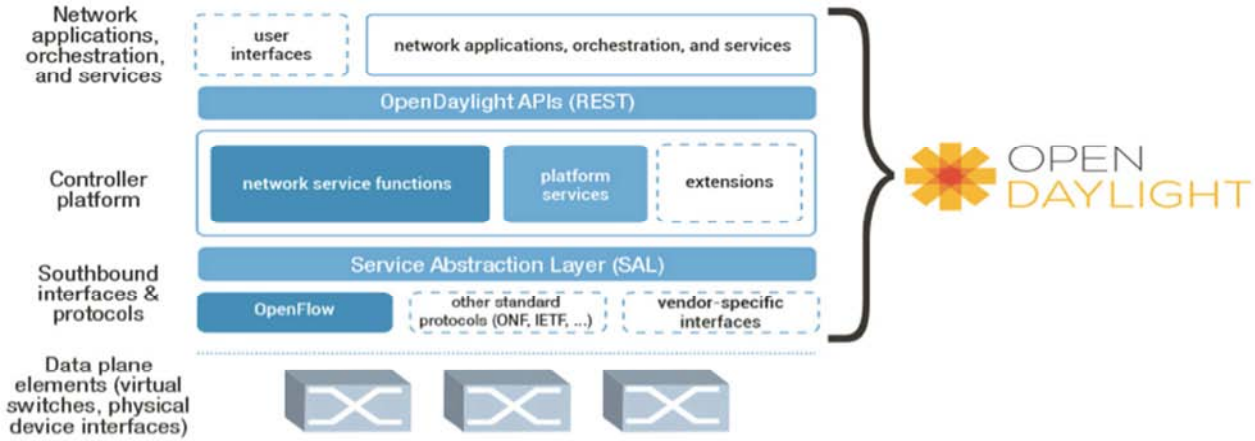

Figure 4. *ODL Architecture [17].*

The ODL project is an initiative by The Linux Foundation to highlight the importance of SDN. The ODL offers the largest open source SDN controller that is being used in various organisations and universities. As shown in Figure 4, the architecture of ODL has a Southbound interface that supports multi-vendor environment and a Northbound interface that offers multiple functionalities to various applications via different APIs. In addition, there is a Service Abstraction Layer (SAL) that not only interlinks service requests to the relevant plugins, but also provides a basic platform for building higher-level services [12]. Open protocol standards such as OpenFlow or standard protocols can be used to communicate with the physical or virtualised hardware.

Some of the key features offered by ODL controller are as follows [16]:

a) On demand services: It provides readily available services on bandwidth scheduling.
b) Cloud computing and virtualisation: It offers quality service on cloud infrastructures. Here, OpenStack is most commonly used.
c) Resource optimisation: It dynamically optimises the network resources based on load balancing.
d) Reliable networking model: It provides highly active and automated networking models for government, university and private sector networks.
e) Network visibility and control: It offers a centralized administration of the entire network using a single or multiple controllers.

2.3. SDN in WSN

The application specific nature restricts the WSN from utilising their full potential [18]. Multiple WSNs are deployed for multiple applications in the same area. Similarly, vendors fail to utilise the common functionalities as they develop WSN in isolation. Furthermore, the remote deployment nature of the WSN requires highly autonomous and self-configurable devices that are not feasible due to the resource limitations of these devices [19]. Some of the common issues in the WSN are energy saving, sensor node mobility, network management, localisation accuracy and virtualised WSN [20].

All of the aforementioned challenges can be effectively tackled by using SDN. The SDN encourages the development of cost-effective protocols that can lead to considerable increase in the productivity of the WSN. The separation of forwarding plane from the control logic allows easier network management and enables network virtualisation. Furthermore, recent boost in the popularity of Internet of Things (IoT) has resulted in the large-scale production and deployment of the WSNs [20]. The next decade could see billions of interconnected sensor-nodes linked through the Internet in which the SDN can provide a solid platform for handling such large number of networked devices and also resolve some of the key issues encountered by the WSN.

In particular, the most significant features that can be achieved by using SDN enabled WSN nodes are node and resource management [21]. A controller can take into account the energy available to different nodes while making the routing decisions to ensure the best network lifetime. Usually WSN nodes are considered as application-specific, disposable devices. But considering their use in Smart Cities where sensor nodes have to collect, process and transmit different types of data for different applications, they need a solid framework in which a much better usage of underlying infrastructure can be achieved through the SDN deployment.

Another key advantage of using SDN is that if a tap in a network indicates to the controller that a device is showing signs of being hijacked, then the controller can steer the traffic away from that device to an Intrusion Detection System (IDS) for further analysis [22]. This approach can prove very helpful for WSN domain.

2.3.1. Energy Saving with SDN

With limited energy resources in WSNs, the nodes are however deployed in situations where they have limited access to any renewable energy resources. This accordingly restricts the development of energy efficient protocols, which in turns affects the overall performance of the network. With SDN, the power consumed by the nodes can be considerably saved [20]. The controller can determine the best routing policies and thus relieves the nodes from making those decisions on their own. In case the node is about to run out of battery, it will send a warning to the controller so that it can make changes to the routing tables in time [23]. Furthermore, since the controller takes over the control plane functionalities, the traffic management, resource allocation, and Quality of Service (QoS) can be efficiently achieved with a lower energy overhead.

2.3.2. Sensor Node Mobility with SDN

In case of mobile sensors, the network topology frequently changes which results in delayed convergence time for the vector based networking protocols. This affects the overall performance of the network given the fact that a WSN has a specific topology to a specific application [24]. When an application changes, the corresponding network topology also changes. In doing so, the sensor nodes lose energy and their lifetimes are shortened. With SDN, a centralised controller can either inject or modify the network policies on the fly. This will result in lower convergence time for protocols. The controller also assigns employ a mobility management protocol that directs the nodes to continuously inform the controller of their location information. By this way, the controller will keeps updating the flow tables with new routing decisions and ensure optimal network performance.

2.3.3. Network Management with SDN

Network management is a complex and challenging process for WSN administrators. Traditional networking requires the management of proprietary software on proprietary hardware devices. In case of sensor nodes, using network components of different vendors makes the management process even more complicated [21]. The cost of managing a WSN is relatively high and any new policy or protocol implementation would require the need of altering the nodes' hardware. Such process requires physical access to all the nodes which might not always be possible. Therefore, SDN can help transform the network administration problem to a network programming one. Complexity of the sensor network is dramatically eased with SDN. New routing protocols can be readily employed on the network and also facilitates the compilation of different versions of the same network applications for different types of sensor nodes.

2.3.4. Localisation Accuracy with SDN

Data provided by a sensor node without correct location information could be considered useless. Due to the energy-constrained nature of the nodes, traditional networking cannot achieve highly accurate location information as it requires running sophisticated localisation algorithms that can prove to be an overhead for these devices. It is shown in [21] that with SDN, a highly accurate location information can be obtained by using a centralised routing algorithm. The gathered location information can be used by a network topology discovery algorithm to further improve the routing decisions made by the controller. This location information data can then be of use to various sensor applications.

2.3.5. Virtualised WSNs with SDN

It is suggested in [20] that applying SDN in WSNs will enable different organisations and applications to share the same underlying physical infrastructure instead of deploying separate networks. This will result in reduced cost to customers, reduced cost of ownership and will allow the network to expand economically. Although the SDN was not designed for resource-constrained WSN, its features can be leveraged to form a virtualised environment for WSN.

3. Architecture for SDN in WSN

The novel idea of exploiting OpenFlow technology to address reliability issues in sensor networks was presented by [25], while the first architectural proposal was presented by [18] in the form of Software Defined Wireless Sensor Network (SD-WSN). Some of the notable, proposed architectures for using SDN in WSN are as follows:

a) **S**oftware Defined Wireless Sensor Network (SD-WSN)
b) TinySDN
c) Service-centric networking for URban-scale Feedback Systems (SURF)
d) Software Defined Networking in WIreless Sensor nEtworks (SDN-WISE)

3.1. Software Defined Wireless Sensor Network (SD-WSN)

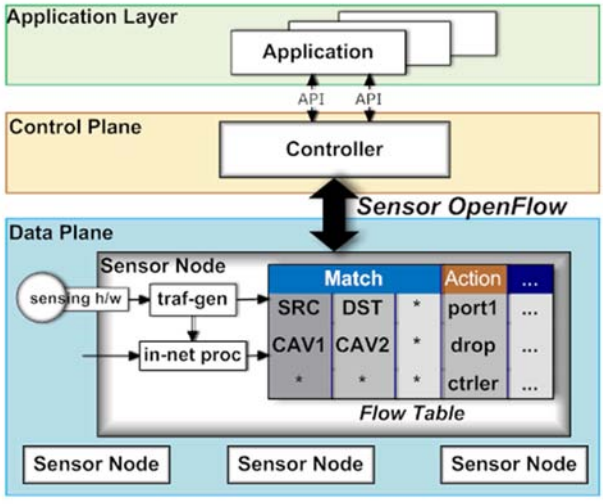

Figure 5. *SD-WSN Architecture [18].*

SD-WSN presents the first effort of combining SDN and WSNs. The aim of the SD-WSN design is to tackle the problems of resource underutilisation, counter-productivity, rigidity in policy changes, and network management in WSNs [18]. The fundamental assumption made by the OpenFlow protocol is that the underlying network is composed of highly sophisticated networking devices. While on the other side, the WSN networks are composed of devices with low power and specifications. As OpenFlow was primarily designed as a wired protocol hence its direct implementation on the WSN domain would not be fruitful. SD-WSN proposes some changes to it and present a new solution that can work with the WSN. The core component of SD-WSN is the Sensor OpenFlow (SOF) protocol. It is used as a standard communication protocol between the data plane and control plane. The aim is to make the underlying network more programmable by deploying user configurable flow tables.

As shown in Figure 5, the architecture of SD-WSN offers the following features [18]:

3.1.1. Data Plane (Creating Flows)

Note that the WSNs are mostly data-centric and the actual data has more importance than where it came from. Therefore, they employ a different addressing scheme which also includes attributes. For example, "nodes with temperature > 30". This will need to be catered during the creation of flow tables. Here, WSN addressing schemes can be classified into Class-1 and Class-2. Class-1 comprises of unique 16 bit addresses, whereas Class-2 consists of Concatenated Attribute Value (CAV) pairs. There are two methods for flow creation, including:

a) Redefining flow tables: SD-WSN handles Class-1 addressing scheme as shown by an example in Table 1. It exploits an OpenFlow eXtensible Match (OXM) like format which is used to define flow Matches. Two new oxm_type fields are introduced by SOF which are OXM_SOF_SRC (source) and OXM_SOF_DST (destination) while the rest of the fields are same as that in OpenFlow. Class-2 addressing scheme is handled by introducing a CAV format which is a quadruple as shown by an example in Table 2 and Table 3. By adding a new oxm_type field of OXM_SOF_CAV any Class-2 flows can be formed.

Table 1. *Class-1 Flow [18].*

oxm_type= OXM_SOF_SRC	oxm_hasmask =1	oxm_length=4	oxm_value= 0x796F	oxm_mask= 0xFF00	oxm_type= OXM_SOF_DST	oxm_mask=0	oxm_length=2	oxm_value=0

Table 2. *Class-2 Flow (30<temperature<60) [18].*

oxm_type= OXM_SOF_CAV	cav_offset =48	cav_cast= int32	cav_op= ">"	cav_value=30	oxm_type= OXM_SOF_CAV	cav_offset=48	cav_cast= int32	cav_op= "<"	cav_value=60

Table 3. *Class-2 Flow (Zone-ID=7 and x-coordinate>150) [18].*

oxm_type= OXM_SOF_CAV	cav_offset =40	cav_cast= int16	cav_op= "="	cav_value=7	oxm_type= OXM_SOF_CAV	cav_offset=42	cav_cast= int16	cav_op= ">"	cav_value= 150

b) Augmenting with IP: The second method is to augment WSN with IP. Two off-the-shelf IP stacks are recommended, including uIP/uIPv6 and Blip.

3.1.2. Control Plane (SOF Channel)

It offers reliable TCP/IP connectivity which also ensures orderly message delivery. The two parties are identified using IP addresses. These addresses are generally unavailable in Wireless Sensor Networks (WSN). This issue is addressed by SD-WSN. Out of the two methods described in the previous section if the network operator selects the first method of non-IP addressing, then Sensor OpenFlow (SOF) channel can be directly implemented on the WSN. If however the network operator decides to augment WSN with IP then SOF channels will be self-sufficient as those IP stacks come with ready-made TCP implementations. The SOF channel needs to be hosted within the same WSN. This can be problematic for the energy-constrained WSN since it has to carry the additional control traffic between the controller and the sensor nodes. Furthermore, the control traffic in the WSN is large and without a proper mechanism, it will overload the entire network.

The control traffic mainly comprises of two types of messages, namely Packet-in and Packet-out. A packet-in is a request sent by the node to the controller to seek instructions on how to deal with a packet that does not match any flow entry. A packet-out is the response from the controller giving instruction on how to deal with the packet. The control traffic in WSN is often bursty in nature and multiple requests are sent to the controller. In case of several different sensors sending several flow setup request to the controller simultaneously, the network will overload. Furthermore this scenario will often occur as each flow has an expiration timer. To tackle with this problem, the SD-WSN instructs the sensor nodes to send only one packet-in request for the first time and withdraw any further requests having same destination address as the first packet until the corresponding packet-out is received. This will prevent the data bottlenecks.

Unlike other networks, the nodes in WSN act like end devices that generate data packets on their own instead of just forwarding them. Therefore, in SD-WSN a new traf-gen module is added on each sensor node for traffic generation. Depending on the implementation, it can run in blocking (synchronously awaiting sensory data to become available), call-back (asynchronously triggered by a "data-available" event) or round-robin (periodically checking if data is

available) manner.

Furthermore, the WSN at times need to perform data aggregation to reduce data redundancy. However, such feature is absent in SDN. To tackle this issue, SD-WSN model offers an in-net proc module. If processing is not needed, then it simply forwards it to the flow table. In case of making any changes to the algorithm, an over the air programming (OTA) technology can be used to direct the changes.

3.2. TinySDN

TinySDN is a TinyOS-based SDN framework [21]. The TinySDN introduces multiple controllers in WSN and has two main components which are SDN sensor node and SDN controller node (see Figure 6). The TinySDN design focuses on the key issues of energy supply, communication latency and smaller link layer frames. Most of these issues were not addressed by the previously proposed architectures for SDN in WSN. It is also the first SDN based design for devices running TinyOS. Typical WSN devices have only one radio module that transmits or receives signals at a given time. Therefore, data and control planes have to share the same communication link and available bandwidth. This in-band control causes delays in the network. Furthermore the control and data flows must also be separated. It is shown in [21] that the IEEE 802.15.4 standard provides a very limited bandwidth which results in an average of 250 Kbps increased latency per hop until reaching the controller. If the controller is placed directly on the sink, then it can reduce the latency considerably by exploiting the positioning of the nodes. The TinySDN proposes a new model in which multiple controllers are used in WSN and one of them is placed closer to the end nodes.

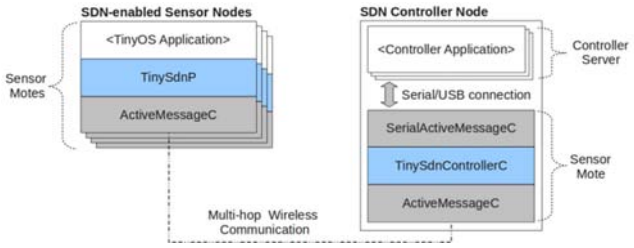

Figure 6. *TinySDN Design [21].*

3.2.1. SDN-Enabled Sensor Node

As the end devices are considered peripheral to SDN, they are out of the scope for OpenFlow but on the other hand sensor nodes in WSN behave like end devices by generating data packets. Keeping that in view, the TinySDN can deploys an SDN-enabled node which plays both roles of an SDN switch and an SDN end device. Each SDN-enabled node must find an SDN controller node to join and then receive flow specifications. An SDN enabled sensor node is split into three parts.

a) TinyOS Application: This portion is the equivalent to the end device. It generates data packets and then places them on the network using the programming interface provided by the TinySDN component. The network designer of programmer writes it according to the WSN application.
b) TinySdnP: It is the main component of TinySDN which checks whether the received packet matches a flow entry in the flow table and then performs the related action. If not then it sends a flow setup request named “packet-in” to an SDN controller. It is also responsible for performing a flow table update when it receives a flow setup response named “packet-out”.
c) ActiveMessageC: This TinyOS component manages and provides a programming interface to interact with the radio module of the sensor node. It performs all tasks related to wireless communication such as data forwarding and topology information discovery.

3.2.2. SDN Controller Node

The SDN controller node(s) that are responsible for creating network flows with two different modules as follows:

a) Sensor Mote Module: It runs on sensor mote and communicates with SDN-enabled sensor node using ActiveMessageC. It acts as an intermediate between the controller server module and the network. It forwards the received messages to the controller server and receives the messages from the network for the controller server module.
b) Controller Server Module: It contains the control plane logic and is responsible for hosting controller applications and managing the network flow and topology information.

3.2.3. Specification of Flows and Actions in Tiny SDN

Two actions are specified in TinySDN including forward and drop. The forward action performs packet forwarding to the next hop while the drop action indicates that a packet should be dropped. In terms of flows, the TinySDN has two types of flows, i.e. control flows and data flows. The control flows are meant for control traffic between SDN-enabled sensor nodes and SDN controller nodes while the data flows are meant for application’s data traffic. Specifically, flows are classified as flow entries in the flow tables where each entry is composed of four field as shown in Tables 4 and 5. In case of data flow table the identification field is Flow ID whereas in case of control flow table the identification field is the Destination node ID.

Table 4. *Data Flow [21].*

Flow ID	Action	Value	Count
1	Drop	N/A	100
2	Forward	5	10
101	Forward	10	27

Table 5. *Control Flow [21].*

Destination Node ID	Action	Value	Count
0	Forward	4	5
1	Forward	4	2
7	Forward	6	2

At network start-up, the first task for an SDN-enabled Sensor Node is to find an SDN controller Node and enlist itself with it. For this discovery process, the TinySDN runs the

Collection Tree Protocol (CTP). This protocol is widely used in multi-hop TinyOS-based applications and has two main advantages of hardware Independence and multiple SDN controllers. The network topology information collection comprises of two steps:

a) Step 1: Each TinySDN-enabled sensor node recognises its neighbour and measures the link quality between them;
b) Step 2: The information regarding the link state is sent to the TinySDN controller node through a CTP path or control flow.

3.3. Service-Centric Networking for Urban-Scale Feedback Systems (SURF)

In WSNs, the sensing applications are considered to be more significant as compared to typical network applications such as firewall. As described in [20], the SURF is an architecture that acknowledges the differences between a typical SDN and WSNs. It recognises that nodes are not only switches as in the traditional OpenFlow SDN networks, but they also have one or more application components. It also addresses the issue of different stakeholders with different requirements sharing a single large infrastructure.

The SURF controller has the following capabilities [20]:

a) It sets up and manages data flows through the network that maintain a required level of QoS.
b) It finds the optimal subset of nodes that can service an external sensing request in terms of quality of sensing and communication.
c) It dynamically adjusts the allocations of data flows and sensing applications by migrating flows or applications in order to respond to external changes or reallocation requests.

The SURF architecture has following layers:

3.3.1. Network Applications

This layer handles the business and network applications that control and monitor a set of resources managed by a single or multiple SDN controllers. In case of multiple parties using the same virtualised WSN infrastructure, the applications at this layer receive events and notifications about the state of virtual networks (VNs), and then can alter them with varying levels of QoS and bandwidth.

3.3.2. Controller

The main responsibility of the controller is to execute the requests of the applications coming through the northbound APIs. These applications have access to the Network Information Base (NIB) and they direct the controller to perform tasks such as resource management and re-optimisation, responding to and generating events as a result of changes in the underlying network; and computing a collection of packet forwarding rules. These rules are then installed into the WSN nodes via the southbound API. One of the key motivations behind their design of SURF is to enable network virtualisation. To support this feature, the SURF SDN control logic introduces four entities in the controller, including

a) Resource Allocator: It is responsible for determining whether a VN or network function virtualisation (NFV) request can be accommodated by the network. If the service request can be supported then the resource allocator interacts with the virtualiser to allocate the physical resources that will form part of the VN.
b) Virtualiser: This entity is responsible for creating a VN agent that represent the resources through a subset view of the NIB and actions available to the application.
c) Orchestrator: It ensures that the service function chains which are responsible for running network services are flexibly compose network functions and are working as independent functions by following the service-oriented principles. The orchestrator is also responsible for resolving conflicts between different applications and to ensure optimal performance in terms of resource utilisation, overhead, sleep schedules and routing.
d) Management: The management plane in the controller consists of a service manager, a tenant manager, a physical network model that keeps track of the physical infrastructure and an Operation Support Service (OSS). The network model maintains a database of network dynamics, the tenant manager has a database of tenant functions, and the service manager maintains a database of VN application and functions.

3.3.3. Physical and Virtual WSN

This layer consist of the physical or virtualised network elements which implement the decisions made in the controller layer issued via the southbound interface. An extension of Constrained Application Protocol (CoAP) can be a suitable choice as it is well established within the WSN field. The southbound plane of the controller is expected to support multiple protocols that are designed by keeping in consideration the limitations of the underlying infrastructure. This plane of the controller also has a topology manager which updates the NIB for the SDN control logic. It also considers the modifications necessary on sensor node protocol stack required to support communication with the controller via the southbound API.

3.4. Software Defined Networking in WIreless Sensor Networks (SDN-WISE)

Although the previously mentioned architectures have been shown to provide a number of advantages over the traditional WSNs without SDN, there exist a few shortcomings as follows [26]:

a) Protocol details are not provided which are fundamental for the correct operation of the network.
b) The architectures are not practically implemented and hence no performance evaluations of the proposed solutions have been carried out.

SDN-WISE designed in [26] is the first practical implementation of an OpenFlow like SDN solution designed specifically for WSNs. Unlike other architectures, the SDN-WISE aims to limit the exchange of information

between the nodes and the controller, and also to make sensor nodes directly programmable.

Furthermore, the SDN-WISE offer the ease of implementing the SDN controller logic. This represents a major advantage as compared to the previously proposed solutions as it increases the flexibility and simplicity in network programming. The SDN-WISE also offer the opportunity to run its controller in a simulated environment. Simulation software such as OMNET++ and COOJA can be used to test its functionality.

The SDN-WISE endeavour to be proficient in the utilisation of sensor resources regardless of the fact that such productivity may result in a lower data rate. To be energy efficient, it encourages the use of duty cycle to periodically turn the radio module on and off, which would help in conserving the energy. Moreover, since the WSNs are inherently information driven, the SDN-WISE makes the system more aware of the packet content. The nodes can deal with packets based on the information available in their header and payload. More complex relational operators are also introduced in the flow tables. In OpenFlow, the system resources are separated by the FlowVisor into small portions. Every portion is associated with only one controller at a time. A further notice in WSN is that the same bit of information can be of significance to another application running on another controller. The SDN-WISE therefore also permit various controllers to specify different rules for the same packet according to their needs.

3.4.1. SDN-WISE Sensor Nodes

The behaviour of SDN-WISE sensor nodes is completely encoded in three data structures including WISE States Array, Accepted IDs Array and WISE Flow Table [26]. These structures are then filled with the instructions originating from the controllers running at the distant servers. The controllers define the systems administration arrangements which are then implemented by the sensor nodes. At any time, each node is described by one current state for every active controller. Specifically, the WISE state array is the data structure that contains those values. The broadcast nature of wireless medium will enable the nodes to receive all data packets of which some maybe not meant for them. The Accepted IDs Array permits each node to choose just those packets that are meant for it. In case the ID is enlisted in the Accepted IDs array, the packet will be further processed by filtering it through the matching rules specified in the WISE flow table. In case the packet does not match any rule, a request is sent to the controller to specify the new rules for that packet.

In order to contact the controller, the node needs to specify the next best hop towards one of the sinks. For this purpose, the Topology Discovery (TD) layer runs a protocol which is based on the exchange of TD packets between the nodes. These packets contain information regarding battery level of the node and distance from the sink in terms of the number of hops. Every time a node receives such packet, it compares it with its own next best hop information and chooses the best amongst them. This information is also used to populate a WISE Neighbours List which is periodically sent to the Topology Management (TM) layer which that generates a graphical view of the network.

3.4.2. SDN-WISE Protocol Architecture

The protocol architecture is shown in Figure 7.

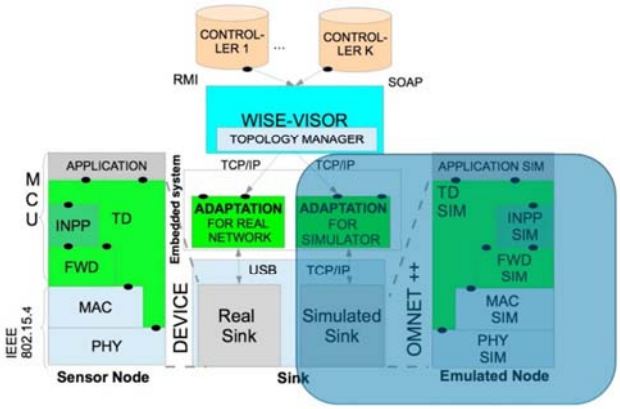

Figure 7. *SDN-WISE Protocol Architecture [26].*

Sensor nodes have an IEEE 802.15.4 transceiver and a Micro Control Unit (MCU). The MCU entertains the forwarding layer and it executes the forwarding decisions according to the WISE flow table. It also continuously updates the WISE flow table according to the configuration commands sent by the controller. The In-Network Packet Processing (INPP) layer is present on top of the Forwarding (FWD) layer. It performs data aggregation and other processing tasks. In SDN-WISE, the INPP layer reduces the network overhead by combining small packets that are to travel on the same routes. Another under development feature of INPP is to perform network coding which will prove very efficient for several WSN scenarios. Topology Discovery layer can access all layer. It controls the behaviour of the nodes at all levels. The network control logic is directed by a WISE-VISOR. It includes a Topology Management (TM) layer which abstracts the networks resources so that different logical networks with different management policies set by different controllers can run over the same set of physical devices. The Adaption layer is responsible for formatting the messages received from the sink in such way that they can be handled by the WISE-VISOR. In case the controller runs in the same node which is hosting the TM layer, the interactions will occur through the JAVA methods offered by the TM layer or else the interactions can occur through the JAVA Remote Method Invocation (RMI) or Simple Object Access Protocol (SOAP).

3.4.3. Topology Discovery

The topology manager module in the WISE-Visor builds a consistent view of the entire network by collecting local topology information through TD packets generated by the sensor nodes. For instance, when a random sensor node A receives a TD packet from a node B, it will perform the following operations:

a) Node A will enlist node B's ID in its current neighbours along with node B's current RSSI and battery level.

b) Node A will perform a check on whether the recently received TD packet from node B has a lower value of the current distance from the sink then the already enlisted packets. If it is true, then node A will update its value to current value plus one and it will set its next hop towards the controller equal to that of node B.
c) Node A will set its battery level in the corresponding field of the TD packet.
d) Node A will transmit the updated TD packet over the broadcast wireless channel.
e) Each sensor node generates a packet containing its current list of neighbours and sends it periodically to the WISE-Visor. The list of neighbours is also periodically cleared. If a node receives a packet directed towards the WISE-Visor then it redirects it to the node set as their next hop towards the controller.

The fields in SDN-WISE packet header and SDN-WISE Flow table are shown in Figures 8 and 9, respectively.

Figure 8. *SDN-WISE Packet Header [26].*

Matching Rule					Matching Rule					Matching Rule					Action					Statistics	
Op.	Size	S	Addr.	Value	Op.	Size	S	Addr.	Value	Op.	Size	S	Addr.	Value	Type	M	S	Addr.	Value	TTL	Counter
=	2	0	2	B	>	2	0	10	x_{thr}	=	1	1	0	0	Modify	1	1	0	1	122	23
=	2	0	2	B	≤	2	0	10	x_{thr}	=	1	1	0	1	Modify	1	1	0	0	122	120
=	2	0	2	B	-	0	-	-	-	-	0	-	-	-	Forward	0	0	0	D	122	143
=	2	0	2	A	=	1	1	0	0	-	0	-	-	-	Drop	0	0	-	-	100	42
=	2	0	2	A	=	1	1	0	1	-	0	-	-	-	Forward	0	0	0	D	100	32

Figure 9. *SDN-WISE Flow Table [26].*

As shown in Figure 9, the SDN-WISE Flow table has three sections including Matching Rules, Actions and Statistics. The Matching Rules specify three conditions to be fulfilled. Actions are executed in case the matching takes place and then the corresponding statistics section is updated accordingly. The S field in each matching rules specifies whether the current packet (i.e. s=0) or the current state (i.e. s=1) is under consideration. Offset and Size specify the first byte and the size of the string of bytes in the packet. The Operator field gives the relational operator to be checked against the Value given in the rule. Actions field specify the corresponding actions to be taken in case a packet satisfies the matching rules. Here, an Action is specified by five fields. The type specifies the type of action for which the possible values can be "forward to", "drop", "modify", "turn on/off radio", "send to INPP". M states whether the entry is exclusive (i.e. M=0) or not (i.e. M=1). In case of M=0, if the conditions are satisfied then the sensor node executes the action and then stops browsing the WISE flow table. In case of M=1, after executing the action the sensor node will continue to browse the WISE flow table and execute other actions if the corresponding conditions specified in the Matching Rules section are satisfied. The offset and value field depend on the type of action. If for example the action is to Forward the packet then they must specify the next hop ID. If it the action is to Drop then they will give the drop probability as well as the next hop OD in case the packet is not dropped. If the action is to Modify then they specify the offset and the new value to be written. If it is to send to INPP then they must specify the type of processing that must be executed and in case of Turn off radio they must state the time after which the radio should turn on again. In case the action is to Modify then the flag S shows whether the action is to be taken on the packet or the state. Statistics in SDN-WISE are used in a similar way as in OpenFlow. In case of a match the relevant actions will be executed and the TTL field in will reduce by one on each hop and the counter will increment by one.

3.4.4. Exemplary Topology of SDN-WISE

Figure 10 illustrates the functionality of SDN-WISE by considering a scenario in a network where the data measured by node A is significant only if the data measured by node B is higher than a given threshold. An energy efficient policy is needed to guide node C to drop packets if the packets received by node B contains a measured data lower than the threshold. In the traditional OpenFlow, such strategy cannot be implemented because of the following limitations:

a) Complex relational operators are not supported.
b) Packet handling cannot be done based on the result of comparison between other packets.

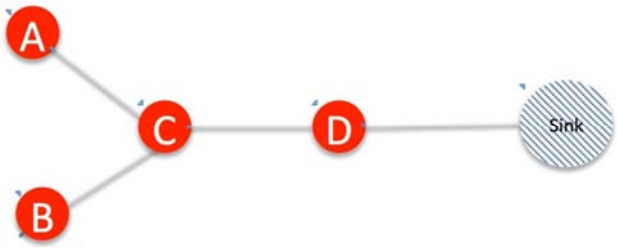

Figure 10. *Exemplary Topology [26].*

This policy can be easily implement in SDN-WISE through a finite state machine as shown in Figure 11 which is implemented through the five WISE flow table entries (see Figure 9).

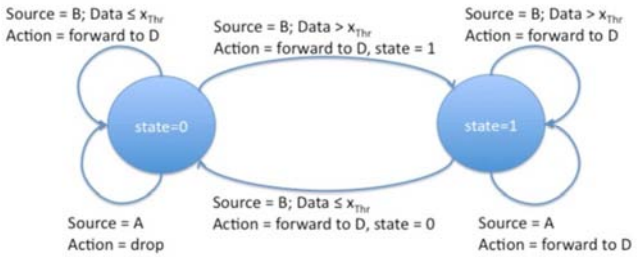

Figure 11. *Finite State Machine [26].*

The first two lines specify the transitions between states 0 and 1. In the first entry of the WISE flow table the first Matching Rule selects packets coming from node B. The second matching rule selects those that have in the 10^{th} and 11^{th} bytes a value higher than the threshold. Finally the third matching rule selects the cases in which the current state of the node is 0. If all these rules are satisfied then the state is set to 1 in the Actions Section.

The second entry selects the cases in which the incoming packet has been generated by B. It contains data whose value is lower than or equal to xthr and the current state is 1. In this case set the state to 0.

The third entry specifies that the packets coming from B must be forwarded to D in any scenario. The fourth and fifth entry state that the packets coming from A be dropped if the current state is 0 or forwarded to D if the current state is 1.

4. Discussion and Future Works

After exploring different features offered by an SDN controller, it is clear beyond doubt that the SDN offers a great alternative to the traditional networking model and it can cater for some of the most common issues in the current networking environment such as data congestion. The SDN makes the management of the network easier and thus highly suitable for scalable data centres and mobile network services. Although the traditional SDN models offer a versatile controller, they did not take into account the limitations of a WSN. Therefore, they cannot be directly applied to a resource-constrained WSN. On the other hand, the first practical implementation of SDN in WSN offered by SDN-WISE does take into account the limitations of WSN's nodes and offer more suitable alternatives for WSN. However, at the moment the SDN-WISE differs greatly from the other SDN controllers available for the Wired and Wireless Networks. The SDN-WISE also does not offer a proper GUI to manage flow tables and data traffic. Specifically, it lacks a proper modelling interface such as Yang UI offered by the ODL, and thus requires further development and enhancement. In addition, the ODL foundation is supported by the leading IT companies around the world, such as Cisco, ERICSSON, Intel, etc., whereas the SDN-WISE is a prospect for implementing the SDN in WSN.

The first and foremost area of concern for SDN in WSN is the security of the controller with the development of technologies and the tremendous increase of number of devices. New ideas are needed for the development of a solid framework for these novel designs. An analysis on the security of SDN can be referred to in [27]. Considering the deployment of SDN for WSN, various features and modifications are required to be investigated, especially when taking into account the practical security issues. Specifically, a major challenge is how to develop an efficient algorithm to deal with security threats over a number of sensors employing various applications.

The concept of deploying distributed controller has already been proposed but whether it will produce the desired results or what effects it will have on the energy consumption of the sensor nodes is yet to be practically investigated. The functionality of SDN with mobile sensors also needs to be verified. Furthermore, with the growing popularity of the Internet of Things (IoTs) where SDN is promising to be an enabling technology (e.g. [28] and references therein), the integration of the WSN with these networks would raise diverse open research problems. In particular, optimisation and automation of network functions via SDN and machine learning techniques for various applications and services are giving good grounds for expecting an intelligent WSN in the near future.

5. Conclusion

In this paper, we have briefly introduced the background of SDN as well as its feasibility in WSNs. After carefully examining various features offered by the SDN, the SDN has been shown to not only offer a more simplified network management with more control over the network devices, but also to provide richer programmatic interfaces. The ability to shape and control data traffic ensures guaranteed content delivery which can be very useful for VoIP and multimedia transmissions. It has also been shown that the SDN and large-scale deployment of the WSNs are the future of networking. The WSNs are key parts of the Internet of Things (IoTs) which will lead to billions of wireless sensor nodes connected to the Internet over the next decade. To cope with this issue, only the SDN can provide an efficient management mechanism for the smooth functionality of their design, and thus promote more research works towards this new approach. The centralised nature of the SDN controller also presents a great challenge to the security of the whole network as it makes the network vulnerable to a single point of failure. The concept of SDN itself is relatively new to the networking world. However, the growing demand of WSN deployments will certainly lead to more innovations in this field.

References

[1] J. D. McCabe, *Network Analysis, Architecture, and Design*, 3rd edition, Elsevier, 2007, ISBN: 9780123704801.

[2] M. Dye, R. McDonald, and A. Rufi, Network Fundamentals, CCNA Exploration Companion Guide, 2007, ISBN: 9781587132087.

[3] Q. Zhang and L. Liu, "Workload Adaptive Shared Memory Management for High Performance Network I/O in Virtualized Cloud," in *IEEE Transactions on Computers*, vol. 65, no. 11, pp. 3480-3494, Nov. 2016.

[4] W. Xia, Y. Wen, C. H. Foh, D. Niyato and H. Xie, "A Survey on Software-Defined Networking," in *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27-51, Firstquarter 2015.

[5] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, Jan. 2015.

[6] T. Mizrahi and Y. Moses, "Time4: Time for SDN," in *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 433-446, Sept. 2016.

[7] I. F. Akyildiz, Weilian Su, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor networks," in *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102-114, Aug. 2002.

[8] Johanna, "Wireless sensor network design," 2014. Available at: https://wirelessmeshsensornetworks.wordpress.com/tag/wireless-sensor-network-technology-and-its-application-using-vlsi/.

[9] Silicon Labs (White Paper), "The Evolution of Wireless Sensor Networks," 2013. Available at: http://www.silabs.com/documents/public/white-papers/evolution-of-wireless-sensor-networks.pdf.

[10] D. Cooley, "Wireless Sensor Networks Evolve to Meet Mainstream Needs," 2012. Available at: http://rtcmagazine.com/articles/view/102871.

[11] J. Tourrilhes, P. Sharma, S. Banerjee and J. Pettit, "SDN and OpenFlow Evolution: A Standards Perspective," in *Computer*, vol. 47, no. 11, pp. 22-29, Nov. 2014.

[12] T. D. Nadeau and K. Gray, *SDN: Software Defined Networks*, O'Reilly Media, 2013, ISBN: 9781449342302.

[13] S. Costanzo, L. Galluccio, G. Morabito and S. Palazzo, "Software Defined Wireless Networks: Unbridling SDNs," 2012 European Workshop on Software Defined Networking, Darmstadt, 2012, pp. 1-6.

[14] SDX Central, "Understanding the SDN architecture," 2015. Available at: https://www.sdxcentral.com/sdn/definitions/inside-sdn-architecture/.

[15] D. Suh, S. Jang, S. Han, S. Pack, M.-S. Kim, T. Kim and C.-G. Lim, "Toward Highly Available and Scalable Software Defined Networks for Service Providers," in *IEEE Communications Magazine*, vol. 55, no. 4, pp. 100-107, April 2017.

[16] OpenDaylight, "OpenDaylight: Open Source SDN Platform," 2013. Available at: https://www.opendaylight.org/.

[17] D. Ward, "OpenDaylight: Building an Open Source Community around SDN," 2013. Available at: http://blogs.cisco.com/news/opendaylight.

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

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

[20] D. O'Shea, V. Cionca and D. Pesch, "The Presidium of Wireless Sensor Networks - A Software Defined Wireless Sensor Network Architecture," in: R. Agüero, T. Zinner, M. García-Lozano, BL. Wenning, A. Timm-Giel (eds), *Mobile Networks and Management*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 158. Springer, 2015.

[21] B. Trevizan de Oliveira, L. Batista Gabriel and C. Borges Margi, "TinySDN: Enabling Multiple Controllers for Software-Defined Wireless Sensor Networks," in *IEEE Latin America Transactions*, vol. 13, no. 11, pp. 3690-3696, Nov. 2015.

[22] K. Slavov, D. Migault and M. Pourzandi, "Identifying and addressing the vulnerabilities and security issues of SDN," in *Ericsson Technology Review*, Vol. 92, No. 7, Aug. 2015. Available at: https://www.ericsson.com/assets/local/publications/ericsson-technology-review/docs/2015/etr-sdn-security.pdf.

[23] Y. Choi, Y. Choi and Y.-G. Hong, "Study on coupling of software-defined networking and wireless sensor networks," *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, Vienna, 2016, pp. 900-902.

[24] A. Boonsongsrikul, S. Kocijancic and S. Suppharangsan, "Effective energy consumption on wireless sensor networks: Survey and challenges," *2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, 2013, pp. 469-473.

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

[26] L. Galluccio, S. Milardo, G. Morabito and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," *2015 IEEE Conference on Computer Communications (INFOCOM)*, Kowloon, 2015, pp. 513-521.

[27] I. Ahmad, S. Namal, M. Ylianttila and A. Gurtov, "Security in Software Defined Networks: A Survey," in *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317-2346, Fourthquarter 2015.

[28] N. Bizanis and F. A. Kuipers, "SDN and Virtualization Solutions for the Internet of Things: A Survey," in *IEEE Access*, vol. 4, pp. 5591-5606, 2016.