Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A Survey on Software-Defined Wireless Sensor Networks: Current status, machine learning approaches and major challenges

F. FERNANDO JURADO-LASSO[1,2], (Member, IEEE), LETIZIA MARCHEGIANI[3], (Member, IEEE), J. F. JURADO[4], ADNAN M. ABU-MAHFOUZ[5,6], (Senior Member, IEEE), and XENOFON FAFOUTIS[1], (Senior Member, IEEE),

[1]Embedded Systems Engineering section, DTU Compute, Technical University of Denmark, 2800 Lyngby, Denmark
[2]Department of Electrical and Electronic Engineering, The University of Melbourne, Melbourne, VIC 3010, Australia
[3]Department of Electronic Systems, Aalborg University, 9220 Aalborg Ø, Denmark
[4]Faculty of Engineering and Administration of the Department of Basic Science, Universidad Nacional de Colombia sede Palmira, Palmira, Colombia
[5]Council for Scientific and Industrial Research (CSIR), Pretoria 0184, South Africa
[6]Department of Electrical and Electronic Engineering Science, University of Johannesburg, Johannesburg 0001, South

Corresponding author: F. Fernando Jurado-Lasso (e-mail: ffjla@dtu.dk).

ABSTRACT Wireless Sensor Networks (WSNs), which are enablers of the Internet of Things (IoT) technology, are typically used en-masse in widely physically distributed applications to monitor the dynamic conditions of the environment. They collect raw sensor data, which are further processed in a centralised manner. With the current traditional techniques of state-of-art WSNs programmed for specific tasks, it is hard to react to any dynamic change in the conditions of the environment beyond the scope of the intended task. To solve this problem, a synergised research effort between Software-Defined Networking (SDN) and WSNs has been proposed. This paper is aimed to present a comprehensive survey of relevant research over the period 2012-2021 of Software-Defined Wireless Sensor Network (SDWSN) proposals and Machine Learning (ML) techniques to perform network management, policy enforcement, and network configuration functions. This survey provides helpful information and insights to the scientific and industrial communities, and professional organisations interested in SDWSNs, mainly the current state-of-art, machine learning techniques, and open issues.

INDEX TERMS Wireless Sensor Networks (WSNs), Internet of Things (IoT), Machine Learning (ML), Software-Defined Wireless Sensor Networks (SDWSNs), Machine Learning Software-Defined Wireless Sensor Networks (ML-SDWSNs).

I. INTRODUCTION

THE Internet of Things (IoT) is an emerging technology that has caught tremendous attention from the scientific and industry communities and professional organisations due to its diverse benefits: including financial, efficiency, management, etc. It is a key enabling technology of the so-called industry 4.0. IoT stakeholders (e.g., governments, industry), which have recently acknowledged that IoT is a real business opportunity. Forecasts estimate that the IoT business can grow into a market worth $7.1 trillion USD by 2025 [1] and that the number of connected "things" will exceed the 75 billion devices [2]. The exponential growth of connected devices implies a huge variety of IoT vendors and protocols. Despite this variety of vendors and protocols, the IoT must, somehow, deliver seamless services to users. The emerging IoT applications including smart agriculture, transportation systems, health systems, etc., expand the scope of the internet to include sensing technologies such as Wireless Sensor Networks (WSNs).

WSNs are built upon the interconnection of large number of Networked Embedded Systems (NESs). A NES, which is also called wireless sensor node, is a tiny energy-constrained device that comprises of a processing unit, a memory unit, a communication transceiver, and some sort of power supply.

TABLE 1. Related survey articles on SDN

Survey article	Topic					
	SDN	SDN-IoT	SDWSNs	SDN-SG	SDN-UWSNs	ML-SDWSNs
[3]	✓					
[4]	✓					
[5]	✓					
[6]		✓	✓			
[7]		✓	✓			
[8]			✓			
[9]			✓			
[10]		✓				
[11]						✓
[12]					✓	
[13]			✓			
[14]			✓			
[15]				✓		
[16]						✓

They are usually deployed to measure physical variables such as humidity, temperature, pressure, air quality, etc., and they work cooperatively to achieve a common goal. The main characteristics of NES are the low cost, size, and limited resources [17], [18]. WSNs are used in range of applications that enable integration of the physical world into the computer-based world, resulting in benefits and improvements in remotely managing the physical world, keeping an electronic record of physical variables, early detection of potential threats, predictions, and economical benefits. Their low cost and ease of deployment make WSNs attractive in the practical implementation of the IoT. However, their small size and low cost lead to limitations on resources such as energy supply, memory size, computational speed and communication bandwidth. Therefore, limited resources in WSNs need to be managed effectively so that they can perform optimally for the longest period of time possible.

The Software-Defined Networking (SDN) paradigm has been proposed to alleviate the management complexity currently found in wired networks. SDN breaks the vertical integration of the network by separating it in application-, control- and data-planes. The application plane hosts user applications and programs that explicitly, directly, and programmatically convey information regarding the network requirements and desire network behaviour to the SDN controller. The control plane consists of a logically centralised entity that process requirements from the application plane and deploy them in the data plane, and provides to the application plane with a global view of the network. The data plane is the network infrastructure that consist of networking devices that become forwarding devices with no intelligence. The introduction of SDN abstractions into the WSN forms what we call Software-Defined Wireless Sensor Networks (SDWSNs).

The SDWSN paradigm emerges to solve the management complexity in current WSNs deployments. This new paradigm allows adding new functionalities into the network, no different from adding another application to the control plane [9]. In large WSNs, with thousands of sensor nodes, it is critical to consider and implement management solutions [19]. The SDWSNs centralise the network intelligence in an SDWSN controller, leaving sensor nodes acting as simple forwarding devices. Sensor nodes forward packets to the destination based upon the reprogrammable forwarding table managed by the controller. SDWSN controller leverages the global information of the network (e.g., network statistics, energy levels, interference, etc.) to come up with new powerful and intelligent protocols to achieve the desired network performance.

A. CONTRIBUTION

Despite the diverse benefits brought by SDN to WSNs, without proper countermeasures to minimise the management overhead introduced, it can greatly negatively impact the network performance of the WSN and lead to high energy cost. This paper conducts an extensive literature review by exploring relevant research articles on SDWSNs and Machine Learning Software-Defined Wireless Sensor Networks (ML-SDWSNs) approaches.

Research works that have reviewed papers on SDN are listed in Table 1. Topics on these surveys include SDN basics, SDN for IoT, SDWSNs, SDN for Smart Grids (SG), SDN for underwater WSNs (UWSNs), and ML-SDWSNs. As can be seen from the table, existing surveys have paid little attention to the use of Machine Learning (ML) techniques in SDWSNs. In particular, the article in [11], which was published in 2017, only briefly discusses the use of ML algorithms in SDN, while SDWSN papers were not surveyed. Their article surveys papers mostly based on the use of ML algorithms in SDN in general. Papers that take advantage of the global view of the controller in SDWSNs to improve the network performance were not discussed. The survey in [16], published in 2019, briefly reviews papers that use Artificial Intelligence (AI) for intrusion detection in SDWSNs. It mainly discusses how the security vulnerabilities of SDWSNs can be counteracted by combining cryptography schemes and AI techniques. In contrast, the contributions of this survey article are as follows.

TABLE 2. List of Acronyms

Acronym	Description
μIP	micro Internet Protocol
μIPv6	micro Internet Protocol version 6
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
AE	Autoencoder
AI	Artificial Intelligence
API	Application Program Interface
BLIP	Berkeley Low-power IP
CNN	Convolutional Neural Networks
CTP	Collection Tree Protocol
DL	Deep Learning
DRL	Deep Reinforcement Learning
DT	Decision Tree
EOS	Embedded Operating System
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
IA	Intelligent Agent
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
k-NN	K-Nearest Neighbour
KPI	Key Performance Indicator
LoRaWAN	Long Range Wide Area Network
MCU	Microcontroller Unit
MDP	Markov Decision Process
ML	Machine Learning
ML-SDWSN	Machine Learning Software-Defined Wireless Sensor Network
NES	Networked Embedded System
NN	Neural Network
OS	Operating System
PCA	Principal Component Analysis
PDR	Packet Delivery Ratio
PLR	Packet Loss Rate
PSO	Particle Swarm Optimisation
QoS	Quality of Service
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RPL	Routing Protocol for Low-Power and Lossy Networks
RSSI	Received Signal Strength Indicator
RTT	Round-Trip Time
SDN	Software-Defined Networking
SDWSN	Software-Defined Wireless Sensor Network
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TSCH	Time Slotted Channel Hopping
UDP	User Datagram Protocol
WPAN	Wireless Personal Area Network
WSAN	Wireless Sensor and Actuator Network
WSN	Wireless Sensor Network

1) We firstly provide a comprehensive background on WSNs including evolution of MCU-sensor nodes, networking and standards, and challenges of WSNs.
2) We provide a systematic review of SDWSN proposals found in the current state-of-art and categorised them into *general frameworks*, proposals that seek to improve KPIs (*QoS-related works*), research works that reprogram both hardware and software of sensor nodes (*fully programmable mechanisms & EOS*), scientific articles that leverage the global view of the controller to devise new routing and management protocols (*network topology and management proposals*), and research papers that seek to solve the controller placement problem (*Controller placement works*).
3) The nature of the SDWSN centralised architecture opens up new research opportunities to experiment with AI/ML algorithms embedded in the SDWSN controller to improve the overall WSN performance. Therefore, we provide a comprehensive background on the most widely used ML techniques, and we perform a systematic review of research papers that have combined research efforts of ML and SDWSNs, to improve network performance.
4) We discuss open issues and research directions in SDWSNs.

This review will serve to produce a better understanding and clarify the current status and the potential research directions regarding the open issues of SDWSNs. To the best of our knowledge, there does not exist a survey that covers in-depth the state of the art of SDWSN research works and ML techniques used in SDWSNs.

The remaining of this paper is organised as follows. Section II provides a detailed background on WSNs. Section III provides a background on SDN, SDWSNs and presents the early adopters of SDWSNs. Section IV presents research works that have expanded the state-of-art of SDWSNs. Section V presents a detailed overview of ML techniques. Section VI presents research efforts that have adopted ML techniques in SDWSN. Section VII summarises SDWSN research works. Section VIII discusses SDWSN open research issues. Finally, in Section IX conclusions are drawn. Acronyms used throughout this paper are summarised in Table 2.

II. BACKGROUND

The introduction of WSNs has opened new opportunities for monitoring applications. These can be summarised as follows.

- Home monitoring: This is an example of a Wireless Sensor and Actuator Network (WSAN). This kind of network can collect sensed data such as temperature, humidity and states of other sensors such as magnetic sensor or switches, and is also capable of changing the environment and physical world through actuators such as servos, motors or switches.
- Environmental monitoring: The goal of this WSN is to maintain the sink informed of any environmental changes at the deployed location and surroundings. This term has evolved to cover many monitoring applications of the environment such as sea, volcanoes and forest monitoring, etc.
- Event detection: Thousands of sensor nodes can be deployed in a specific field to detect early hazards to the ecosystem. For example, sensor nodes embedded with temperature, humidity and gas sensors can be used to detect the presence of fire. Early detection of hazards can prevent the loss of lives and valuable resources.

TABLE 3. Historical wireless sensor nodes development platforms

Year	Name	μC				OS	Radio
		Type	RAM [KB]	Flash [KB]	E^2PROM [KB]		
2000	Rene 2 [20]	Atmega 163	1	16	32	TinyOS	TR1000
2000	μAMPS [21]	Strong ARM	-	1M	4M	μOS	National LMX3162
2001	BTnode [22]	Atmega 128L	4	up to 128	4	Nut/OS	CC1000
2003	Mica2 [23]	Atmega 128	4	128	512	TinyOS	CC1000
2004	TelosB [24]	MSP 430	10	48	1M	TinyOS, Contiki[a], RIOT[b]	CC2420
2006	MicaZ [24]	Atmega 128	4	128	512	TinyOS, Contiki[c]	CC2420
2011	WiSMote [25]	MSP 430	16 sram	up to 256	up to 8M	Contiki	CC2520
2013	WiSense [26]	MSP 430	4	56	128	-	CC2520
2013	CC2541DK [27]	CC2541	8	up to 256	-	-	2.4 GHz
2015	CC2538DK [28]	CC2538	32	512	4	Contiki[a], RIOT[b]	2.4 GHz
2015	OpenMote [29]	CC2538	32	512	4	Contiki[a], OpenWSN, FreeRTOS, RIOT	2.4 GHz
2015	CC2650STK [27]	CC2650	20	128	-	Contiki[a]	2.4 GHz
2015	Re mote [30]	CC2538	32	512	4	Contiki[a,d], RIOT[c], MansOS	2.4 GHz, Sub-1 GHz
2017	CC1350STK [31]	CC1350	20	128	-	Contiki-NG[d]	2.4 GHz, Sub-1 GHz
2019	LPSTK-CC1352R [32]	CC1352R	80	352	256	Contiki-NG[d]	2.4 GHz, Sub-1 GHz

[a] It is also supported by Contiki-NG
[b] Basic support
[c] Partially supported
[d] Dynamically switching, at run-time, between the two bands is not supported

- Physical variable monitoring: WSNs can be also used in a simple task such as data logging information of a physical variable of interest. For examples, keeping track of simple things like the temperature of a refrigerator, all the way up to monitoring the water level and flow of a nuclear power plant [33].

As mentioned above, the use of WSNs covers a range of applications that enables integration of the physical world into the computer-based world, resulting in benefits and improvements in our quality of life. Also, a wide variety of wireless sensor devices have been developed to enable wireless connectivity and sensing capabilities in tiny objects, a historical and most popular WSN platforms available in the market are shown in Table 3.

A. NETWORKING AND STANDARDS FOR WSNS

Networking technology sets the form of communications between sensor nodes. Here, the most current Wireless IoT Network Protocols are presented, and it only covers radio waves which are the most common communication technology found in IoT applications. Other forms of wireless communications methods are surveyed in [34].

The most commonly used communication transceiver for WSNs is the low-power radio and the most popular frequency band is the 2.4 GHz as shown in Table 3. 2.4 GHz radios are popular, low-cost, well-supported and the frequency band is standardised in the IEEE 802.15.4 [35]. Among communication protocols, used in this frequency band, are ZigBee [36], Bluetooth [37], [38], and 6LoWPAN [39].

A brief comparison of different communication technologies used in WSNs is shown in Table 4. There is no such thing as the best communication technology for WSNs as the optimum communication protocol largely depends on the application. For home monitoring or smart home, Zigbee and 6LoWPAN can be the appropriate technology as they provide good data rates and support multiple network topologies. For industrial monitoring, 6LoWPAN or LoRaWAN technologies are good solutions, however, 6LoWPAN works better when frequent measurements are needed, and LoRaWAN fits better for large fields, multiple sources of interference, or for infrequent interaction with the gateway.

B. CHALLENGES IN WSNS

The challenges associated with WSNs and IoT can be divided into three different categories: sensor node hardware, heterogeneity and inflexibility.

1) Sensor node hardware

As mentioned before, the main challenges presented in sensor nodes are due to their constrained resources.

- *Energy source:* due to the communication nature of sensor nodes is wireless, most of the applications require sensor nodes to operate in harsh environments or areas with limited access [40], [41]. Thus, it is envisaged that sensor nodes operate without any battery renewal or human intervention for a long time. The power source and individual energy consumption are vital for the Network Lifetime (NL) of WSNs.
- *Memory size:* the memory of sensor nodes stores information regarding the protocol stack and applications running in the node. The integration of the protocol stack, routing protocols and applications into the node imposes a challenge when adding new features in the already constrained memory. The memory has to be

TABLE 4. Examples of popular communication technologies for WSNs

Wireless Technology	Frequency [MHz]	Data Rates [kbps]	Range (LoS)	Network Topology	Power consumption
IEEE 802.15.4	2400, 915, 868	250, 40, 20	100 m+	star	low
Zigbee	2400, 915, 868	250, 40, 20	100 m+	star, tree, mesh	low
Bluetooth/BLE	2400	125-2000	10-100 m	P2P, star, mesh	high/low
6LoWPAN	2400	250	100 m+	star, tree, mesh	low
LoRaWAN	Sub-GHz	0.250-11	10 km+	star	low

managed effectively to assure all applications and program code run efficiently and that the node can host new features as required.

- *Computational speed:* the nature of WSNs is to use low-power microcontrollers which work well for non-resource-intensive tasks such as sensing and radio communications. The use of more powerful processing units directly affects the sensor node size, power consumption and price. However, the use of low-power microcontrollers limits the sensor node when executing tasks of significantly different intensities as occurs with most Internet Protocols (IPs) which require a scheduler and run on top of the firmware. Basically, the IoT requires more processing power to handle the communication overhead involved. On top of this, sensor nodes, considered to be autonomous systems, use complex routing algorithms that add a processing cost to the already constrained device.
- *Communication bandwidth:* when a number of sensor nodes need to transmit in real-time, bandwidth limitations impose restrictions on how many sensor nodes can transmit and the rate at which they can post their data in real-time [42]. Furthermore, wireless communication can take up to 75% of the total energy in some applications [43]. The communications between sensor nodes have to be managed in a way that sensor nodes reliably transmit their data and that the energy consumption does not compromise the NL.

2) Heterogeneity

The IoT enables the interconnection of a large number of heterogeneous devices that creates new user applications to improve the quality of our lives. However, engineers working on the development of new applications face challenges when setting up a network of heterogeneous devices and systems. These heterogeneous devices include a variety of networking devices, manufacturers and software. The wide variety of networking connectivity technologies, protocols and communication methods can present difficulties to engineers and developers when implementing new network designs or protocols. Thus, the IoT must bring seamlessly together all heterogeneous devices to provide services to users.

3) Inflexibility

Since IoT enables the interconnection of objects to the internet, the number of connected devices increases dramatically. WSNs extended the scope of IoT by introducing networked sensing technologies. Currently, state-of-art WSNs are deployed with an inflexible firmware. Where, once the WSN is deployed, any modification to the firmware (e.g., tasks, behaviour in sensor nodes) requires an on-site visit or Over-The-Air (OTA) programming technology to reprogram sensor nodes' firmware. On site-visit, such as the example given in [5], of a WSN that comprises 100 sensor nodes that measures pollution in a lake, that demands for task reprogramming would require taking sensor nodes out of the lake and reprogram their firmware to modify such task, which is not practical and increases the management costs. Whereas OTA permits firmware updates without taking sensor nodes out of the environment and without interrupting the normal operation of sensor nodes, the time required to update an entire WSN is an issue in time-sensitive applications. A smart building application, which has 69 end devices, needs on average seven hours to complete transferring a 125 KB image file to all sensor nodes [44].

Overall, WSNs enable a range of applications from home monitoring to hazard detection in remote areas with difficult access and strict operational requirements such as NL. Wireless sensor nodes are designed to be small, cheap and wireless, so they can be easily embedded, even into the smallest things and used en-masse in widely physically-distributed applications. Such design requirements impose several constraints in the *power supply, memory size, processing power and communication bandwidth*, making smart management of these resources a high priority in the design of practical and cost-efficient WSN applications. The WSN has to work seamlessly with other network devices independently of the vendor who produced it. Furthermore, it must also manage limited resources and provide easy updates of real-time applications. Hence, there is a genuine, real-world need for innovative research efforts into the smart management of resources in wireless sensor networks. Solutions should be independent of the practical application, and the behaviour of sensor nodes and the software running on them easily modified. Therefore, there is a need to tackle the above-mentioned challenges inherent to WSNs and the IoT. SDN has been proposed as a prospective solution to overcoming these challenges.

III. SOFTWARE-DEFINED WIRELESS SENSOR NETWORK (SDWSN)

The SDWSN paradigm is inspired by the SDN technology, which a network management approach that enables to dynamically and programmatically reconfigure the network,

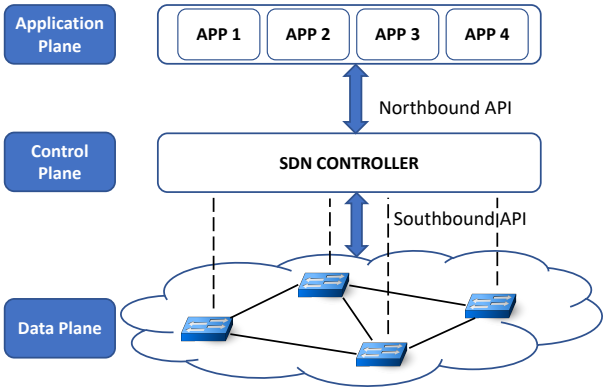


FIGURE 1. A simple representation of an SDN architecture.

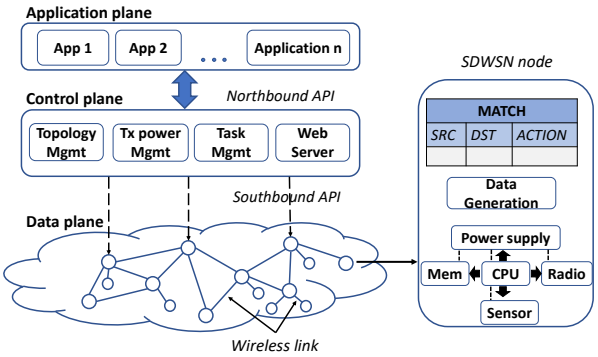


FIGURE 2. Simple representation of an SDWSN architecture.

that is introduced below.

A. SOFTWARE-DEFINED NETWORKING

SDN is a network paradigm solution to the current wired network limitations. It first breaks the vertical integration of the network by separating the control plane or the "control logic" from the underlying networking devices such as routers and switches. Then, the networking devices become a forwarding device with little or no intelligence. The intelligence is instead logically centralised in a controller, facilitating policy enforcement and network reconfiguration [3]. A simple representation of an SDN architecture is shown in Fig 1.

SDN is an approach to network management that enables dynamic network configuration that improves network performance and oversees the network status. SDN is currently widely used in wired networks where architectures are decentralised and complex, and emerging network applications require more flexibility and easy troubleshooting. Although SDN centralises the network intelligence in the control plane, it does not necessarily mean that the data plane depends on a single controller. In fact, the control plane can be built upon multiple controllers which can be physically distributed but logically centralised.

Apart from the three SDN layers, *data plane or infrastructure*, *control plane* and *application plane*, multiple Application Program Interfaces (APIs) also exist: *northbound*, *southbound*, *eastbound*, and *westbound*. The *Northbound* API enables the communication between the application and control plane. Using this API, the control plane provides a global view of the network to the application plane. The *southbound* API is the communication channel between the data- and control-plane. This API is used by the controller to deploy different policies and network management configurations in devices of the data plane. Network devices of the data plane report network status to controllers using the southbound API. The *eastbound* and *westbound* APIs are responsible for orchestrating the communication channel between multiple controllers, so they can make coordinated decisions [10]. The most well-known protocol used in the southbound API is OpenFlow [45]. Researchers have recently applied SDN concepts into WSNs to perform network management, policy enforcement and network reconfiguration functions. The synergy between WSNs and SDN forms the so-called SDWSN paradigm.

B. SOFTWARE-DEFINED WIRELESS SENSOR NETWORK PARADIGM

The SDWSN paradigm emerges to solve the management complexity currently found in state-of-art WSNs. This new paradigm allows adding new functionalities into the network, no different from adding another application to the control plane [9]. In large WSNs, with thousands of sensor nodes, it is critical to consider and implement management solutions [19].

A simple representation of an SDWSN architecture is shown in Fig. 2. The SDWSN architecture differs from the SDN architecture mainly in the data plane. The data plane is based upon wireless sensor nodes that are NESs with constrained resources. SDWSNs centralise the network intelligence in an SDWSN controller, leaving sensor nodes acting as simple forwarding devices. Sensor nodes forward packets to the destination based upon the reprogrammable forwarding table managed by the controller.

1) Challenges of SDWSNs

The main *challenge* of SDWSN architectures are the shared *communication medium* and *constrained resources*. SDN was initially conceived for wired networks, where control packets typically flow through a dedicate communication channel, whereas in WSNs the control packets flow through the same medium. Control packets share the bandwidth with data packets, therefore the bandwidth has to be managed smartly to prevent congestion in the SDWSN. The flexibility of changing the behaviour of sensor nodes implies the introduction of control overhead in the network that may incur increased overhead and energy consumption, and a decrease in the Packet Delivery Ratio (PDR) which is a Key Performance Indicator (KPI) that discloses the amount of data delivered

successfully. The most common principal requirement of WSN applications is to prolong the NL, thus the *constrained resources* of sensor nodes have to be managed in a way that the NL is not drastically reduced. Control packets flowing in the network will increase network energy consumption; therefore, novel control overhead reduction techniques are required to minimise the amount of control overhead and interaction between sensor nodes and the controller, as the work presented in [46].

C. PIONEERS OF SDWSNS

As the SDWSN paradigm is still at its infancy stage, few researchers have started exploring potential architectures for SDWSNs. The introduction of SDN abstractions into WSNs was first introduced by two early adopters: *SOF* [47] and *SDWN* [48].

1) Sensor OpenFlow (SOF)

Luo et al. [47] introduced SOF as a Southbound API to facilitate the communication between the control and data planes. The main objective is to make the WSN infrastructure reprogrammable by customising the flow tables. SOF is motivated by the standard SDN protocol for wired networks, namely OpenFlow [45].

Since WSNs are usually thought to be attribute-based and data-centric networks in comparison to conventional address-centric networks, they offer two approaches for flow creation: (i) *compact network-unique addresses* (ZigBee addressing), and *concatenated attribute-value pairs* that routes packets based on the data attributes, and (ii) the use of the Internet Protocol (IP) in WSNs, and they suggest two IP stacks: μIP or μIPv6 [49], and Berkeley Low-power IP (BLIP) [50]. In comparison to OpenFlow, SOF provides in-networking processing functionalities, but there is no evidence of any type of improvement in network performance with their proposed protocol. Their paper mainly presents SOF as the first research effort that synergizes SDN and WSN; therefore, it lacks specification and details.

2) SDWN

Costanzo et al. [48] introduce *SDWN*. Their approach differs from SOF in many ways: (i) it proposes a Southbound API, namely a flow table, (ii) it states the requirements for the SDWN, such as support for duty cycling and in-network data aggregation, to minimise the overall energy expenditure of the network, (iii) it presents the protocol architectures for the generic and sink nodes, and (iv) it describes the packet format for all packets flowing in the network. Generic nodes are sensor devices in the data plane that forward packets as instructed by the centralised controller. The sink node is the SDN controller which defines the rules for forwarding packets. Their paper tries to analyse the benefits of SDN in WSNs with emphasis on Wireless Personal Area Networks (WPANs).

A brief comparison of the two early adopters is shown in Table 5. SOF and SDWN are considered as the first step towards reprogrammable WSNs, since then multiple research papers have used them as their foundation for new research works.

TABLE 5. Comparison of early adopters of SDWSNs based on the type of networking technology, Network Operating System (NOS) and energy-aware functionalities

FW	Networking	NOS	EAF
SOF	IEEE 802.15.4, IP	-	in-net-proc[a]
SDWSN	IEEE 802.15.4	✓	in-net-proc, duty-cycle

[a] in-net-proc: in-network processing; data aggregation, etc.

IV. EXISTING SDWSNS PROPOSALS

To tackle shortcomings in SOF and SDWN, and the lack of performance evaluation, several authors have proposed SDWSN approaches that aim to improve the overall SDWSN architecture design and performance. This section provides a systematic review of research works found in the current state-of-art of SDWSNs. We group them into five different categories.

- *General frameworks:* This category contains SDWSN research papers that have been proposed to advance in the state-of-art of SDWSNs, but they lack any form of evaluation.
- *Quality of Service (QoS)-related works:* Here, we group research works that guarantee a certain level of service. These works aim to improve KPIs; including energy consumption, control overhead, delay, traffic congestion, packet loss, throughput, etc.
- *Fully reprogrammable mechanisms and EOS:* SDN provides flexibility to reprogram individual sensor nodes functionalities or behaviour; however, there exist research works that extend this to a fully programmable sensor node including both hardware and software.
- *Network topology and management proposals:* This category presents research works that leverage the global view of the controller to devise new topology and management protocols.
- *Controller placement works:* Research works that seek to solve the controller placement in SDWSNs are grouped in this category.

A. GENERAL FRAMEWORKS

It is worth mentioning that the below works are general frameworks that are the first step to synergy research efforts of SDN and WSNs, but they lack evaluation performance. However, some authors have extended these frameworks into a mature and tested framework which we will discuss later in this review.

Previously discussed research works: SOF [47] and SDWN [48], fit in this category. *Alves et al.* [51] proposed IT-SDN as an open-source SDWSN tool to provide a clear separation of protocols: southbound communication, neighbour and controller discovery. IT-SDN is implemented in Contiki and the controller runs on Linux with Qt. This framework

TABLE 6. Comparison of general SDWSN frameworks based on the type of operating system used, controllers, demonstration, and availability to the public

Ref.	Advantage/Disadvantage	EOS	Controller	Availability
TinySDN	First research work that uses TinyOS. It permits the use of multiple controllers, eliminating the dependency on a single controller. Although a demonstration was performed in Cooja to provide an overview of the TinySDN main features, there are no shreds of evidence of improvement with traditional WSNs.	TinyOS	Multiple	✓
IT-SDN	It is an open SDWSN tool inspired by TinySDN. The architecture is independent of the OS and provides detail packet types and formats, and workflow. Although a demonstration was performed in Cooja; it lacks piece of evidence of improvements related to state-of-art WSNs.	Contiki		✓
SDCSN	Cluster head (CH) and multi-controller approach. It provides good architectural design details and security concerns on CHs. The main drawbacks are the lack of details of the implementation and not evidence of performance evaluation provided.	-	Multiple	-
CORAL-SDN	It provides detailed information of the architecture proposed. Tasks handled by the controller and its implementation are explained; but, it lacks details. The demonstration, which is performed in w-iLab.2 and SWN, is well explained; however, it lacks evidence of the evaluation. No charts provided.	Contiki	Single	✓

is intended to be used by researchers to experiment with SDWSNs. Even though there is no evidence of evaluation performance, the authors organised a demonstration of IT-SDN running in Cooja, *i.e.*, the simulator of the Contiki.

De Oliveira et al. [52] present TinySDN which is a TinyOS-based [50] SDWSN implementation. TinySDN offers two types of network devices: SDN-enabled node which comprises of a switch and end-user node, and a SDN controller where the all the intelligence resides. This work was presented as an initial step towards the use of SDN concepts into WSNs using TinyOS. It also details the components and functionalities considered. TinySDN was presented in the form of demonstration, but the paper lacks performance evaluation.

Olivier et al. [53] proposed a Software-Defined Clustered Sensor Network (SDCSN), a cluster-based architecture which has multiple base stations. Base stations host both control plane and cluster heads functionalities. They provide information on design, implementation and security details of SDWSN architectures. They use multiple controllers in their architecture by interconnecting SDN domains. This is a first step towards discovering opportunities and challenges of using cluster techniques in WSNs.

Theodorou et al. proposed CORAL-SDN [54] which is presented as a SDN-based solution for IoT. The architecture design comprises a CORAL-SDN node, a CORAL-SDN controller, and a CORAL Dashboard. The network stack is entirely programmed in C (Compatible with Contiki 3.0). It has a proprietary forwarding protocol that maintains the forwarding table. The CORAL-SDN controller, which is programmed in Java, is designed in a modular scalable approach to easy introduce new algorithms. The CORAL Dashboard is a flexible visualisation tool based on the NODE-RED platform (i.e., https://nodered.org). It has been demonstrated in two testbeds: IMEC w-iLab.2 [55] and SWN (https://www.emulab.swn.uom.gr/). Even though there is not evidence of any evaluation performance, the authors state that experimental results show that CORAL-SDN can improve network control in IoT networks.

A brief comparison of general frameworks is shown in Table 6. The table compares general frameworks stating their advantages and disadvantages, Embedded Operating System (EOS) used, type of controller architecture, and their availability to the research and professional community. We can see that they are also the first research works towards SDN-based WSNs as they seek to provide a practical, fully functional SDWSN architecture and implementation but with little or no evidence of evaluation. These research works have evolved and been used by the research community to further investigate SDWSNs.

B. QUALITY OF SERVICE (QOS) RELATED WORKS

1) Energy consumption

This a well-studied metric in WSNs. Sensor nodes are usually deployed in harsh environments where physical access to sensor nodes is difficult; therefore, WSNs require to smartly manage their energy resources in a way that they could achieve the longest lifetime possible.

Table 7 presents and compares research works currently found in the literature whose main objective is to achieve a reduced energy consumption in WSNs by means of SDN. We can see that new research works consider SDN as a viable solution to improve energy consumption in traditional wireless sensor-based networks; however, a common drawback is a lack of demonstrating improvement against traditional WSNs and the viability in real-world deployments i.e. the study of control overhead, WSN architecture setup, to include all protocol stack layers and computational complexities. Also, they lack evaluation with other SDWSN protocols, which can be tightly related to the limited amount of publicly available SDWSN approaches. Moreover, the development of energy consumption algorithms involve a large number of mathematical models, and their evaluation is frequently made using mathematical tools rather than network simulators. Network simulators allow capturing of all physical events happening in a real network i.e. collision, packet loss, etc., and at the hardware level.

2) Security

This is a concern in IoT networks. It is also in centralised architectures such as SDN. This is especially true in SDWSN architectures with a single controller, whereby an attacker

TABLE 7. Relevant SDWSN research works that aim to achieve a reduced energy consumption

Ref.	Specific aim	Approach	Main drawback	Type of evaluation	Comparison with other protocols
[41]	To design an energy-efficient routing algorithm, where the WSN is divided into clusters	To use a Particle Swarm Optimisation (PSO) algorithm to solve the selection of control nodes	An NP-hard problem no suitable for real-time applications	MATLAB	LEACH
[46]	To design an energy-aware routing algorithm that balances the energy across the network and reduce the control overhead	To select paths with the highest remaining energy level, aggregate packets to common destination and compute checksum over known routes at the controller	The controller can exhaust its resources quickly as it is embedded in one of the sensor nodes	Cooja	SP
[56]	To design an energy-efficient SDWSN using wireless power transfer. To minimise the energy consumption of energy transmitters while keeping sensor nodes sufficiently charged.	Network with energy transmitters. To find the minimum number of energy transmitters by solving an optimisation problem satisfying the constraint of minimum energy charge per node.	The practicality of deployments in real large-scale SDWSN needs to be studied in terms of scalability-cost ratio	Numerical no specified	Traditional WSN
[57]	To design an energy-efficient sensor scheduling and management strategy with certain quality-of-sensing	Formulation of an optimisation problem with three objectives; sensor activation, task mapping and sensing scheduling.	Network connectivity and communication energy consumption are not considered	Numerical no specified	None
[58]	To mathematically express the energy expenditure of SDWSNs	To break down the functions involved, namely; neighbor discovery, neighbor advertisement, network configuration and data collection	No performance improvement demonstrated	MATLAB	None
[59]	To design an energy-efficient routing algorithm	To use a multidimensional energy space. The network uses CHs and Layer Heads (LHs), which have direct communication with the controller	No performance evaluation against architectures without LHs	MATLAB	None
[60]	To design an multidimensional energy space algorithm	Sensor nodes are classified into different energy-space dimensions based on their remaining energy. Nodes with scarce energy are moved to a lower energy-space dimension	MAC layer and TDMA mechanism are considered perfect	MATLAB	SP, energy aware
[61]	To design an energy-efficient routing algorithm that minimises the overhead transmitting data	A sorted distance queue model that allows data to be transmitted to the closest neighbour	The model assumes that all sensor nodes are one hop away from the controller	MATLAB	LEACH-PSO
[62]	To minimise the data generated at the data plane (network traffic)	The controller manages sensor nodes transmissions and implements a learning function of the behaviour for each sensor to replace data transmitted by sensors	No performance improvement, against other approaches, was demonstrated	Computer-based	None
[63]	To design an energy-aware routing protocol and a sleep management mechanism	A clustered network managed by the controller. It finds the best energy-efficient path between any sensor pair and manages sensors' sleep time	No performance evaluation against other SDWSN approaches. Control overhead not considered	Mininet	LEACH, SPIN, [56], [41]
[64]	To design an energy-efficient multicast protocol	Leverages overhearing to deliver a multicast message. To control the transmission range of sensor nodes	No performance improvement against other SDWSN protocols	Mininet	Multicast protocols
[65]	To design a dynamic routing protocol for SDWSNs	An optimisation problem to find the best relay node	High computational complexity. Evaluation is based on different parameters of the algorithm and SP	MATLAB, NS-3	SP
[66]	To reduced the energy expenditure in localisation algorithms	To formulate a binary programming problem on the premise of energy satisfaction	No performance improvement demonstrated	Numerical no specified	None
[67]	To address overhearing in asynchronous SDWSNs	Mathematically express the effects of multi-channel operations and control the transmitting range	No performance evaluation against other SDWSN approaches	Numerical no specified	None
[68]	To develop an energy-optimised congestion control algorithm for Wireless Body Area Networks (WBANs)	A routing algorithm that considers the thermal dissipation of nodes, and selects relay nodes considering the temperature and energy	No SDWSN KPIs effects were evaluated including control overhead	MATLAB	Other WBANs protocols

may compromised the entire network by targeting it. Also, securing a large WSN is a high energy intensive task which can lead to sensor nodes deplete their energy faster. However, SDWSN permits the controller to build a global view of the network which help in identifying malicious devices and activities. Table 8 details research works that aim to identify and improve security issues SDWSNs. Cybersecurity in IoT is surveyed in [78].

Security is a critical aspect to consider when design IoT solutions. As seen from Table 8, security in SDWSNs has not been received proper attention as much of the research efforts focus on discussing security through survey papers

TABLE 8. Relevant SDWSN research works that aim to identify and improve security issues

Ref.	Type	Description
[69]	Survey paper	It provides information on the security challenges present in WSNs and SDN, which are transferable to SDWSNs. Threats and countermeasure techniques are also presented.
[14]	Survey paper	This is a survey paper on SDWSNs. However, this paper provides a security section that surveys security challenges of WSNs, discusses security challenges brought by the introduction of SDN into WSNs, and provides information on the security threats present in SDWSNs and their consequences.
[70]	Research paper	This paper performs an analysis of security issues in SDWSNs. It discusses the security issues that need to be addressed and the already proposed solutions. They provide a summary of challenges, countermeasures actions, tools and research directions
[71]	Research paper	It presents a group key distribution scheme based on physical unclonable functions (PUFs) for SDWSNs. They minimised the communication overhead and latency for securely distributing secret keys. They run their experiments using SDN-WISE [72].
[73]	Research paper	They proposed ETMRM, which is an energy-efficient trust management and routing method for SDWSNs. The design goals are to address security and energy aspects simultaneously. ETMRM handles malicious forwarding attacks including new-flow and selective forwarding attack. Simulation results, based on the SDN-WISE project, show that ETMRM detects and responds to forwarding attacks, and improve KPIs including control overhead, NL, and PDR.

TABLE 9. Relevant SDWSN research works that minimise the delay

Ref.	Specific aim	Approach	Disadvantage
[74]	Eliminate dependency on a single controller and minimise the delay	A multi-controller architecture	CTP outperformed TinySDN when packets are sent to the sink after setting up the flow
[75]	To study the viability of fragmented controller architecture as a method for distributed controllers	A fragmentation method that uses a two-level control structure	Higher network traffic than distributed and centralised architectures
[76]	To compare different protocols for the IoT including SDWSN, ZigBee and 6LoWPAN	Experimentally evaluate the protocols based on Packet Loss Rate (PLR), RTT, overhead and throughput	SDWN outperforms ZigBee and 6LoWPAN in terms of RTT and PLR; however, SDWSN showed poor performance in dynamic environments
[77]	The main goal is to reduce redundant data and minimise the latency in the IoT network	A predictive data selection module that makes use of historical data and Mutual Information (MI) as feature selector, an event identification module, and data sensing module with time constraints	Edge servers can increase deployment costs

rather than designing and implementing security schemes in SDWSNs. Also, most research works discuss security from the SDN and WSN perspectives, where some of these concepts can be easily adapted, whereas others might be unfeasible to apply. In WSNs, security solutions are mainly implemented at sensor level where resources are scarce; therefore, such protocols, which tend to be energy-hungry, are not practical. Security aspects in SDWSNs can be addressed individually at each API. At the northbound API, a misconfiguration can open up new channels of attacks or execute a command that leads to abnormal behaviour of the target application or exposed the information flowing between the controller and the application [79]. At the southbound API, most WSN applications share raw environmental data that can be easily secured centralised at the controller. However, if sensitive data need to be secured at the data plane level, then secure communication schemes should be considered such as SSL/TLS, at the expense of an increase in energy consumption. At west- and east-bound APIs, we can find networked devices with ample resources, e.g. controllers; therefore, secure communication channels can be easily created using traditional security schemes. However, this needs to be studied in detail. Readers interested in an extended discussion on SDN and WSN security from the SDWSN perspective can refer to [9], whereas SDN security is discussed in [80].

3) Delay

This metric is of great importance in sensitive applications such as health monitoring, target tracking, control systems and fire hazard monitoring applications that require prompt reactions to prevent loss of lives and valuable resources. Table 9 compares research works that strive to reduce the delay in SDWSNs. We can see that few papers addressed the delay in SDWSNs directly, it is addressed indirectly in other works. Overall, it has been demonstrated that SDN-based WSNs has the potential of reducing the network delay in comparison with traditional WSNs, as most of the processing has been removed from the sensor nodes. However, it has been also demonstrated that SDWSN works better for static or quasi-static WSN deployments than in dynamic environments as the increased overhead. There is a call for research efforts to make the most of SDWSNs and take advantage of the global view of the network to create new approaches that minimise the delay even in dynamic environments while maintaining a low control overhead.

4) Reliability

This metric assures that the collected data is delivered correctly to the receiver. Table 10 compares research works that aim to improve the reliability of SDWSNs. Similar to the network delay, the network reliability has also been addresses indirectly in other research works. SDWSN architectures grant centralised network monitoring to anticipate potential

TABLE 10. Relevant SDWSN research works that improves the reliability of SDWSNs

Ref.	Specific aim	Approach	Disadvantage
[81]	To address mobility management in industrial WSNs	They use the Time Slotted Channel Hopping (TSCH) protocol, which has fixed length timeslots, where multiple pair of nodes communicate without collisions by using different channels	Network growth implies higher delays and scheduling complexities. Offline scheduling
[82]	To minimise the traffic load	Optimisation problem that selects optimal relay sensor nodes and minimise the transmission of redundant packets.	No improvements demonstrated against other SDWSN and WSN approaches
[83]	To maximise the network reliability by adopting adapting flow schemes	They formulate an integer linear programming problem to obtain the optimal number of APs and the flow manager implements the flow rules at the APs	Redundant flows and increased overhead due to wrong location predictions

TABLE 11. Relevant SDWSN research works that address control overhead

Ref.	Aim	Approach	Disadvantage
[46]	To reduce the control overhead	Aggregate packets to common destination and a checksum function that prevents the controller from sending configuration packets with routes that are already known by the destination	A single communication dead link can trigger a generation of a new control packet
[84]	To reduce both control and data packets resulting in an improved network lifetime	A threshold function whose value is automatically calculated using the data collected from the network. Nodes forward flow setup request packets whose data value is equal to or greater than the calculated threshold value	The threshold function can compromise the network performance, e.g. delay of the controller's response to a change in the network
[85]	To reduce the control overhead traffic in topology discovery and packet forwarding	CHs and neighbouring nodes discover the controller using its nearest CH. Sensor nodes send data packets through CH nodes	CHs can exhaust their resources faster
[72]	To reduce the control overhead	Sensor nodes are programmed as Finite State Machines (FSMs) so they can still make decisions without contacting the controller	The controller may not react to changes promptly
[86]	To minimise the control overhead and to balance the sender waiting time and duplicate packets when sensor nodes are in duty-cycle	MINI-FLOW southbound protocol. Control overhead is reduced using a heuristic algorithm that manages up-, down- and intra-links flows	Periodically flow update increases network consumption
[87]	To reduce the control overhead in SDWSN	A hybrid approach where each sensor node runs an in-cluster routing mechanism and the controller manages routing among clusters	The controller does not have full control of individual sensor nodes

issues that may impact negatively the network reliability. We can see that an increase in network reliability compromises the performance of other key network metrics. There exist a trade-off between network reliability and other KPIs (this also applies to traditional WSNs) such as energy consumption, control overhead, delay, etc. This has to be studied in detail to evaluate and quantify the impact on network performance when increasing network reliability. However, it is expected that centralised architectures such as SDWSN bring more advantages over traditional WSNs to come up with new innovative algorithms to predict network performance indicators to make better network decisions.

Other research has studied distributed control plane architectures for SDWSNs [88], [89]. These works investigate the viability of distributed hierarchical architectures for SDWSNs to reduce the control traffic and dependency on a single controller.

5) Control overhead

Since control packets in SDWSNs share the same communication medium with data packets, it is of great importance to maintain a low level of control packets to avoid negatively impact KPIs such as residual energy of sensor nodes and the PDR. Many research works [53], [88], [89] have indirectly addressed this metric.

Control overhead is a key performance metric to consider when designing SDN-based WSNs. From Table 11, we can see that there exist multiple approaches to minimise the control overhead. They can range from architectural designs such as cluster based architectures, intra-cluster routing and SDN control routing, and techniques to avoid the extra control overhead such as routes checksum, FSMs, threshold functions, etc. The best technique for control overhead reduction is closely related to the application requirements as there exist evident performance trade-offs between them. The overall benefit that SDN brings to WSNs can be overshadowed by the unmanageable control overhead that can be generated if not proper design measures are put in place.

C. FULLY REPROGRAMMABLE MECHANISMS AND EMBEDDED OPERATING SYSTEMS (EOSS)

There are other research works that considered alternative architectures where the WSN can be fully reprogrammable, which includes both software and hardware.

Portilla et al. [90] proposed a modular architecture for wireless sensor nodes using a microcontroller and a Field-

Programmable Gate Array (FPGA) for the processing layer, and Bluetooth radio for communications. The microcontroller manages the radio communications and the analog and digital sensors, whereas the FPGA processes complex operations. *Natheswaran et al.* [91] proposed a remote reconfigurable wireless sensor node with a soft processor which is a microprocessor core that can be implemented using logic synthesis. *Miyazaki et al.* [92] proposed an SDWSN that uses a role generation and delivery system in a reconfigurable WSN. They used a combination of FPGA and MCU to avoid overloading the MCU. The MCU handles the network behaviour while the FPGA performs energy-intensive functions. Although these works bring flexibility to reconfigure sensor nodes, the utilisation of reprogrammable hardware enlarges the complexity of the design and cost. Besides, energy consumption in FPGAs is an issue as discussed in [93]. However, the greatest advances in FPGAs with ultra-low power consumption characteristics have extended their use to WSNs [94]–[96].

In order to achieve the full promise of SDWSNs, the wireless sensor nodes should allow top-layer applications to reconfigure their functionalities by executing different programs. In this way, sensor nodes can be seen as small-scale computers with multiple sensing capabilities. However, due to the limited resources available, sensor nodes require a lightweight OS [8], [97]. The two EOSs that have achieved most attention by the research SDWSN community so far are: (i) Contiki, which is an open-source OS for IoT, designed for resource-constrained sensor nodes [49]. In its core uses C language and has three network stacks; RIME, Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6). Contiki-NG [98] has been presented as a new version of the Contiki project. Contiki-NG started as a fork of the Contiki project and preserves part of its original characteristics. Contiki-NG provides an overall clean-up, updated support for IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH), streamlined RPL implementation, and other features for resource-constrained IoT devices. (ii) TinyOS is also designed for resource-constrained sensor nodes but in its core uses the nesC programming language [50] and supports IPv6 in its protocol stack, namely, BLIP.

There exist a number of EOSs that have not been yet used in SDWSNs: FreeRTOS [99] is an open-source real-time OS kernel for NESs, designed to be small and simple. The footprint can be as low as 9KB and supports over 40 MCU architectures. Key features include a small memory footprint, low overhead, and very fast execution. Zephyr [100] is an stable and open-source real-time OS for resource-constrained embedded systems. It supports multitasking, multiple network stacks, and multiple architectures. One of the network function provided by Zephyr is the dual-stack that enables simultaneously use of IPv4 and IPv6. OpenWSN is not an operating system, but an open-source implementation of a fully standards-based protocol stack for short-range networks, such as the IEEE802.15.4e Time-slotted Channel Hopping standard [101]. IEE802.15.4e, along with IoT protocols, such as 6LoWPAN, Routing Protocol for Low-Power and Lossy Networks (RPL) and CoAP, allows ultra-low power and highly reliable mesh networks that are fully merged into the Internet. RIOT presented as an open-source real-time multi-threading OS that supports a wide range of IoT devices such as low-power sensor boards and microcontrollers including 8-, 16- and 32-bit architectures, that are normally used in the IoT [102]. The RIOT design principle is to be energy-efficient and reliable that supports real-time and small memory applications. It also provides an API access, which is independent of the hardware. Multiple open standard protocols have been ported to RIOT such as the IPv6 network protocol stack that includes the IETF for connecting constrained systems to the Internet (6LoWPAN, IPv6, RPL, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP)).

A brief comparison of the above mentioned operating systems is presented in Table 12. The table shows a comparison of the MCUs supported, the memory footprint, support for RPL, UDP and TCP. Although the memory footprint is platform-dependent, the memory values given in the table can be used as references to perceive how low the memory footprint can be for the specified operating system to run. It shows that Contiki, Contiki-NG, OpenWSN, RIOT and Zephyr are the only operating systems that provide full support for TCP over 6LoWPAN and that FreeRTOS and Contiki support the largest range of MCUs. TinyOS currently supports 8- and 16-bit CPU architectures and the support for TCP still in the experimental phase, which limits the sensor nodes in supporting higher application protocols such as HTTP.

D. NETWORK TOPOLOGY AND MANAGEMENT PROPOSALS

Network management is complex and challenging in networks. Some functionalities include network provisioning, configuration, and maintenance [103]. The implementation of management tasks can lead to a steep increase in the use of sensor resources. But, SDN was introduced to facilitate network management.

One of the main goals of SDN is to facilitate network management. It is envisaged that SDN architectures can help to make smarter decisions and improve the management of vital WSN resources. From Table 13, we can see that implementing network management solutions implies an increment in control overhead. For example, add-on systems on top of 6LoWPAN grant a global view of network resources but large and complex processing functions still are in the protocol stack. Besides, 6LoWPAN are IP-enabled WSNs; therefore, they present a large overhead in their packet formats. Also, some works lack control overhead analysis and the implication in network performance when making the WSN manageable using SDN concepts.

TABLE 12. Comparison of operating systems for WSNs

OS	Contiki	Contiki-NG	TinyOS	FreeRTOS	OpenWSN	RIOT	Zephyr
MCU	MSP430	MSP430	MSP430	MSP430	MSP430	MSP430	ARM
	AVR	Cortex-M	AVR	AVR	Cortex-M	ARM 7	x86
	Cortex-M	JN516x		Cortex-M		Cortex-M	Xtensa
	ARM 7			Cortex-A		x86	RISC-V
	8051			ARM7		AVR	ARC
	RL78			Cyclone V SOC		ESP8266	Nios II
	6502			ARM9		RISC-V	POSIX/NATIVE
	x86			PIC32			SPARC
				NIOS II			
				8051			
				x86			
				Microblaze			
				APS3			
				78K0R			
				TMS570			
RAM [KB]	10	10	10	4-8	-	1.5	8
Flash [KB]	30	∼100	48	32-64	-	5	-
RPL	✓	✓	✓		✓	✓	✓
UDP	✓	✓	✓	✓	✓	✓	✓
TCP	✓	✓	Experimental		✓	✓	✓

TABLE 13. Relevant SDWSN research works that address network topology management

Ref.	Aim	Approach	Disadvantage
[104]	A management system for IoT	Device management to control sensor nodes individually and topology management to control routing paths	Different communication technologies IEEE 802.15.4 [35] and IEEE 802.11 [105] can lead to an increased network design complexities and energy consumption
[103]	An SDN-based management solution for WSNs	Controller placements at base stations	Preliminary proposal with no evidence of improving network management
[106]	To facilitate network service adaptability and network management in mission critical applications	A practical implementation in NS-3 based on the OpenFlow protocol	No evidence of network performance improvement achieved related to traditional WSNs
[107]	To enable network management in 6LoWPANs	Network management over 6LoWPAN layer	High energy-intensive functions still reside in the 6LoWPAN layer. Large control overhead
[108]	Network management for 6LoWPANs	To avoid altering the working principles of nodes, SD-6LN installs SDN features in the existing network infrastructure as an add-on system. SD-6LN merges common features of the SDN and 6LoWPAN protocol stack to manage nodes and process packets more efficiently	High energy-intensive functions still reside in the 6LoWPAN layer. Large control overhead
[109]	A generic SDN-based modular management system for WSNs	They introduced the concept of management modularity using a Management Service Interface (MSI) that eases the insertion of management units as modules	Control overhead still an issue: fusion and flow-rule aggregation techniques needs to be studied in-depth
[110]	A SDN-based measurement architecture for WSNs	Practical implementation, on TinyOS, of the management of multiple measurement tasks	No evidence of network performance improvement achieved related to traditional WSNs
[111]	An SDN-base management solution designed for edge computing multidomain WSNs	Dynamically provision devices, detects operational failures and control devices over the IoT network. It is deployed at the edge computing nodes and uses the cloud	No control overhead analysis and improvement achieved related to traditional WSNs
[112]	A QoS-based technique to actively manage network resources in SDWSNs	It dynamically performs path computation to control network traffic. It provides flexibility to perform resource alignment on different network tasks	No control overhead analysis and and improvement achieved related to traditional WSNs

E. CONTROLLER PLACEMENT WORKS

The placement of the controller directly influences the WSN performance. Among the most important performance metrics to optimise are energy consumption and NL. The SDN controller can be placed in such a way that minimises the energy consumption of sensor nodes; however, this not always the optimal solution to prolong the NL of the network because the solution to this optimisation problem can be found in a low density area, resulting in an inefficient resource management in the neighbourhood of the controller [117]. Therefore, sensor nodes that lie in the proximity of the controller drain their energy first, resulting in a shorter NL. Table 14 presents research works that aim to solve the controller placement to improve network performance in SDWSNs.

TABLE 14. Relevant SDWSNs research works that address the controller placement problem

FW	Aim				Approach
	latency	reliability	fault tolerance	syn. overhead	
[113]	✓		✓		Finding the exact number of controllers required for a specific network topology using GA and GRASP algorithms.
[114]	✓	✓		✓	Placement optimisation problem using thee Cuckoo optimisation algorithm.
[115]	✓	✓	✓	✓	Optimal controller placement using the Cuckoo optimisation algorithm.
[116]	✓	✓			Two approaches for optimal placement were discussed: k-means for local controllers and k-centre for global controller.

As we can see, the controller placement in SDWSNs has not been widely studied in the current state of the art; this can be largely influenced as SDWSN is still at the proof-of-concept stage where most of the research efforts lie in the conceptualisation of it. Besides, the controller placement has been extensively studied in SDN; however, it should be studied in detail for SDWSNs as they impose different resource requirements. A survey on controller placement in SDN can be found in [118], [119], a study on performance evaluation in [120].

V. MACHINE LEARNING OVERVIEW

ML is part of AI that studies computer algorithms to mimic human learning and gradually improving its accuracy. ML is a hot topic and a growing field that has caught tremendous attention among IoT stakeholders. ML algorithms are trained to perform prediction and classification tasks, uncovering vital characteristics within the data. Typical tasks involved in the solution of a ML problem are:

(i) *Data collection:* it usually requires a considerable amount of time to complete this task. It can consist of data acquisition tasks, data labelling and adding new data to already existing datasets.
(ii) *Data preparation:* it is a key step to process raw data and turn it into meaningful and clean data before any training is performed (training is explained in (iv)). Feature engineering is often used to make the collected data better suited to the problem at hand. Tasks include data normalisation, dealing with missing values, data transformation, etc.
(iii) *Choosing a model:* this step consists of selecting the right model for the problem. There exist multiple ML models for different purposes. Some are introduced in this section.
(iv) *Training:* training the model is the bulk task in ML. This is an iterative task that aims to use the training set to improve the prediction of the model at each cycle. Supervised learning uses labelled sample data, whereas unsupervised learning makes inferences from unlabelled data.
(v) *Testing:* it evaluates the accuracy of the learned function using test dataset. The test dataset is a slice of the dataset and is used to evaluate the accuracy of the model.
(vi) *Parameter tuning:* testing multiple algorithm parameters (e.g., learning rate) and selecting the one that improves the model precision.
(vii) *Deployment:* deploy the model and test the prediction outcomes of unforeseen data.

The above are generic steps to follow to solve ML problems; however, some ML techniques such as AutoML and Deep Learning (DL) automates much of these tasks.

This section introduces the reader to the most widely used ML techniques currently found in the state-of-art of ML. Readers interested in thorough discussions on ML theory please refer to [121]. ML techniques can be grouped into four different groups: *supervised*, *unsupervised*, *semi-supervised* and *Reinforcement Learning (RL)*. Given their current widespread usage, in a separate subsection, we introduce *Deep Learning (DL)*, which can be employed in supervised, unsupervised and semi-supervised paradigms.

A. SUPERVISED LEARNING

Supervised learning uses a set of input data X and a set of labels Y. For every sample x, a label y has been assigned, where $x \in X$ and $y \in Y$, and these can be represented in pairs $(x_1, y_1)...(x_n, y_n)$. The goal of supervised learning is to learn a mapping function that matches a given input (x_{n+1}) to a label y_i. Since the labels in the training set are known, this set of algorithms are called supervised learning. Supervised learning requires a huge burden when it comes to data labelling, but there are efforts out there to reduce this burden by relying, for instance, on weak supervision. This set of algorithms can be further classified into *regression* and *classification* depending on the type of output label. Regression algorithms are used to predict continuous values such as salary, cost, etc., whereas classification algorithms are used to assign a class label to a given input. Below we introduce the most popular supervised learning algorithms.

1) K-Nearest Neighbour

The K-Nearest Neighbour (k-NN) algorithm is used to solve both classification and regression problems. The k-NN algorithm calculates the distance between the query point (new data sample) and feature vectors of the examples in the data. The algorithm then retrieves k number of examples in the data that are nearest to the query, where k is a given

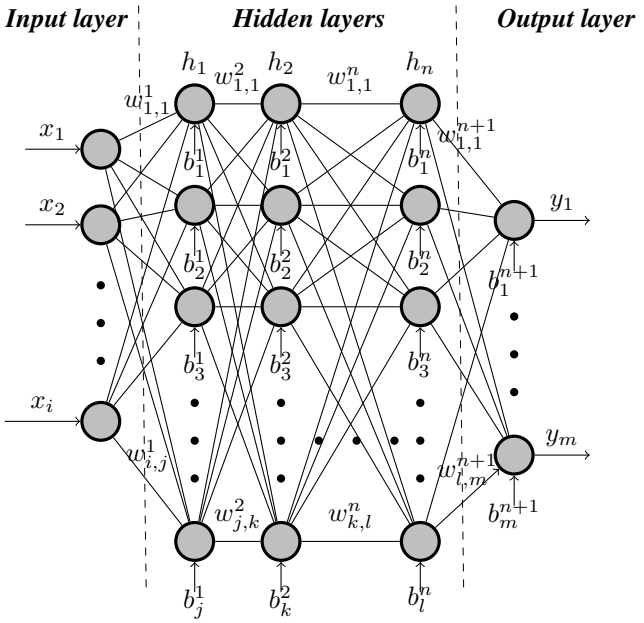


FIGURE 3. Architecture of a deep neural network. The weights between the output of the ith neuron in the $n-1$ layer and the input of j neuron in the nth layer is represented as $w^n_{i,j}$. The bias of the ith neuron in the nth layer is represented as b^n_i.

parameter. It then assigns the class label according to a majority vote among the k examples, in case of classification; or calculates the average of the k number of examples in the data in the case of regression. One of the most common ways to determine the nearest nodes is by calculating the Euclidean distance between the query point and all feature vectors of the examples in the data. The size of the training data directly affects the performance of the k-NN algorithm.

2) Decision Tree (DT)

This is one of the most popular and powerful algorithms for classification and prediction in supervised learning. It creates a tree-like structure that comprises of (i) internal nodes (no-leaf) that test input data X on specific attributes, (ii) branches that represent an outcome of the test, and (iii) leaf nodes that are labelled with a class, for classification trees, or with a meaningful continuous output, for regression trees. A DT is built upon splitting the data, establishing the root node, into subsets. The splitting is based on classification features. This process is then repeated on each individual subset until the splitting no longer adds value to the prediction. This process of building optimal learning trees is NP-complete. In this type of algorithm falls the Random Forest Tree [122] which builds multiple DTs. The output is the mean of each individual tree for regression tasks or the most voted label for classification tasks. In general, random forest trees have greater prediction accuracy than single DTs.

3) Neural Networks

Neural Networks (NNs) are a type of computing system inspired by biological learning systems, and they are the base of deep learning algorithms.. NNs are formed by a collection of nodes called artificial neurons that models neurons in a biological brain. NNs learn to perform tasks by processing examples of the form $(x_1, y_1), ..., (x_n, y_n)$. The general structure of an NN comprises three layers; input, hidden and output layer. The input layers receives the external data. The output layer produces the final result, and the neurons in between are the hidden layers. Each artificial neuron can relay a signal to other neurons. Each neuron processes the received signals and transmits them to other connected neurons. The signal received at each neuron is a real number, and the output is computed using non-linear functions of the sum of the given inputs. A neural network with a single hidden layer is conventionally called "shallow", whereas a neural network with multiple hidden layers, shown in Fig. 3, is called a deep neural network. The training of NNs aims to retrieve the values of the weights $w_{1,1}$, $w_{j,k}$, ..., $w_{l,m}$ of the connections between neurons to best approximate a target function $f(X)$, where X is the input of the network. The values of the weights are obtained by minimising the error between the target output $Y = f(X)$ and the prediction of the network $\hat{f}(X)$. The number of training samples directly affects the prediction accuracy. *Overfitting* may occur when a model performs too well to the given training dataset, learning the detail and noise in the input data; therefore, the model may fail to fit unforeseen data, significantly dropping the accuracy of predictions. *Underfitting* is the opposite of overfitting, where the model fails to fit both the training dataset and the test dataset. As an underfitted model does not perform well in the training set, it will also have poor performance in the test set. The training stage is finalised based on certain termination criteria. One of the algorithms based on NNs is Deep Learning (DL) [123] which is becoming very popular because of its capability to learn complex relationships. DL uses modularity to build complex network from simpler functional units.

B. UNSUPERVISED LEARNING

In comparison with supervised learning, unsupervised learning algorithms just relies on the input data X. The input data is presented to the algorithm without any tags or labels (unlabelled examples). The goal of unsupervised learning is to create a model that automatically learns from the sample data and identify patterns (features) in order to classify them into groups. Data points within groups share similar characteristics (e.g., highest energy level, malicious nodes, etc.). Unsupervised learning uses a probability distribution $P(x)$ given x, whereas supervised learning uses conditional probability distribution $P(x|y)$ given the target vector y. Unsupervised learning is often applied to solve three main applications: (i) clustering groups data points that share similar characteristics, (ii) outlier detection (anomaly detection) that predicts how far a given feature vector is from the unlabelled examples. (iii) reduced dimensionality that aims to reduce the number of features in the input vector. Below we introduce two of the most widely used unsupervised

learning algorithms.

1) K-means clustering

K-means algorithm classifies new data entries into different clusters, where each data point within a cluster share similar characteristics. K-means creates clusters by deploying k data points, called centroids, within the feature space (X). Each feature vector ($x \in X$) is assigned to the closest centroid. K is a parameter given by the data scientist. K-means essentially creates a Voronoi tessallation that partitions the feature space into regions close to each $x \in X$. The algorithm is simple and involves the following steps. (i) The algorithm randomly initialises k clusters, (ii) it then assigns each $x \in X$ to the nearest cluster, (iii) centroids are moved to minimise the distance to its x, (iv) iterate over the last two steps until it reaches a convergence condition or until it reaches the maximum number of iterations. The initial random position of centroids can lead to poor clustering, therefore, the algorithm runs multiple times and returns the clustering with the least total distance between x and its near cluster.

2) Principal Component Analysis (PCA)

This method is one of the most popular for data compression and dimensionality reduction. PCA extracts the most relevant features by performing a rotation of the feature space, this rotation is presented as new orthogonal variables called principal components of the data. The first component is the one that has the highest variance in the data, and the consecutive component will take the orthogonal direction to the previous component. Components with the least data variance can be discarded as they provide the least significant information. Therefore, PCA is key for preprocessing data for ML applications to turn a highly dimensional data into intelligent, actionable information.

Overall, supervised learning uses labelled data to train the model. Labelling the data may be a complex and time-consuming task as it requires human intervention, special instrumentation, experiments, etc. It also requires more computing resources for training, especially for large datasets. Whereas unsupervised learning learns the data, classifies and make inferences of it without any labels (unlabelled data is easy to collect). It is less complex than supervised learning as it is not required to fully understand the data. It is very useful in finding patterns. But, it has less accuracy than supervised learning.

C. SEMI-SUPERVISED LEARNING

Semi-supervised learning is a ML technique that is built-upon a synergy between supervised and unsupervised learning. In its feature space, semi-supervised learning uses a small set of labelled data ($x_1, ..., x_n \in X$) along with a large set of unlabelled data ($x_{n+1}, ..., x_{n+u} \in X$). The used of labelled and unlabelled data can significantly improve learning accuracy. It is often found that the collection of labelled data is a costly task as it requires skilled human intervention. It can lead to large and fully training sets infeasible. In contrast, the collection of unlabelled data is relatively inexpensive. In such applications, the use of semi-supervised learning is a good choice. Semi-supervised learning strategies focus on extending either supervised or unsupervised learning by using information known by the other learning paradigm. It can be used in two main settings:

1) *Semi-supervised classification:* this can be seen as an extension of the supervised classification problem that assumes there are much less labelled data than unlabelled data. The main goal is to train a model from both data types (labelled and unlabelled) such that the resulting accuracy is much better than the supervised model trained on the labelled data only.
2) *Constrained clustering:* this can be seen as an extension of unsupervised clustering. It uses some supervised information about the clusters as well as unlabelled data. The main goal is to form better clusters than the clustering obtained using unlabelled data only.

There exist other semi-supervised learning settings such as regression, dimensionality reduction, etc. [124]. Overall, semi-supervised learning may achieve the same or better performance than supervised learning but using less amount of labelled data leading to a reduction in costs, and better clustering than other clustering algorithms that rely on unlabelled data only. But, semi-supervised learning may increase computational resources as it process more data and it requires more memory. In addition, the outcome accuracy may deteriorate with the use of unlabelled data as the use of more data does not necessary means that the algorithm will perform better. More detailed information on semi-supervised learning can be found in [124].

D. REINFORCEMENT LEARNING (RL)

In contrast with supervised and unsupervised learning, RL uses Intelligent Agents (IAs) to take actions in the environment so it can maximise the notion of the accumulative reward. Also, it does not need labelled examples as in supervised learning. RL uses the trial and error approach, where decisions are made sequential (one after the other). RL is typically modeled as a Markov Decision Process (MDP), where the set of environment and agent states is defined as S, the set of actions taken by the IA is defined as A, the probability of transition from state s to state s' under action a is defined as $P_a(s, s')$, and the immediate reward after the previous transition is defined as $R_a(s, s')$. The main goal of RL is to learn an optimised policy that maximises the reward function [125].

E. DEEP LEARNING (DL)

DL can be seen as an extension of NNs. In general, a NN with an input layer, multiple hidden layers with non-linear activation functions and an output layer is considered a DL network. Here, the use of non-linear activation functions is key as it allows the network to solve complex non-liner problems. As in NNs, each layer in DL contains units (neurons).

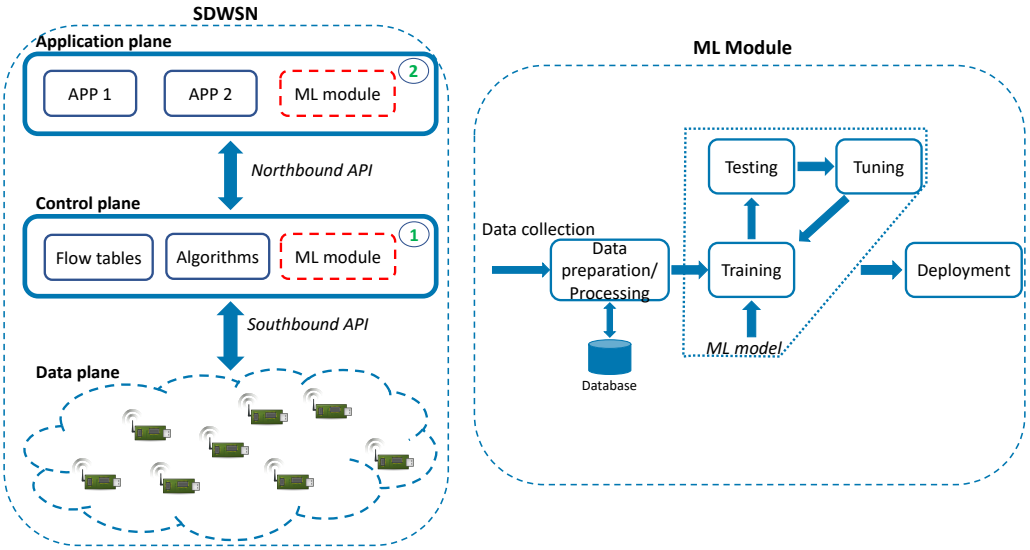


FIGURE 4. A simple representation of an ML-SDWSN architecture with (1) the ML module embedded in the control plane and (2) the ML module embedded in the application plane.

They can have multiple inputs and make weight associations that are updated based on the error and learning rules. DL architectures that have been applied to WSN applications include Convolutional Neural Networks (CNN) [126], Recurrent Neural Networks (RNNs) [127], and Autoencoder (AE) [128].

VI. MACHINE LEARNING SOFTWARE-DEFINED WIRELESS SENSOR NETWORK (ML-SDWSN)

A typical ML-SDWSN architecture comprises of the three SDN planes and a machine learning module. The ML module works as an add-on system that can be easily installed within the SDWSN architecture as shown in Fig. 4. It can be found in two distinct locations: at the control plane (1) or the application plane (2). The location of the ML module within the SDWSN architecture is upon the network designer, user- and application-specific requirements, and available network resources. Installing the ML module at the control plane, which can be built upon multiple controllers, will require the layer to supply all the resources needed for the correct functioning of the network such as enough CPU power to cope with the ML processing needs and memory requirements. The module relies entirely on a single plane; therefore, minimising system failure and network latency as it removes eventual communication outages at upper layers and reducing communication bottlenecks. Whereas, installing the ML module at the application plane frees computing resources at the control plane. It also permits to compute of high processing-intensive functions in a remote location with higher processing resources, therefore, reducing the processing delay. However, the network outage at the upper layers can limit the ML-SDWSN system to act immediately to changes in the data plane; therefore, impacting negatively the network performance.

This section provides relevant research efforts in theoretical works and strategies of adopting ML techniques in the context of SDWSNs. The nature of the SDWSN centralised architecture opens up new research opportunities to experiment with ML techniques embedded in the SDWSN architecture to improve the overall WSN performance. Here, we first group research works based on the specific network problem they address. At the end of this section, we discuss and compare the surveyed ML-SDWSN approaches. Readers interested in ML techniques applied to SDN please refer to [129].

A. MOBILITY

Technological advances and the introduction of the IoT have enabled new emerging mobile IoT applications such as monitoring and tracking systems for a variety of everyday human activities including sports, health care and entertainment [130]. Current routing protocols of choice for IoT have not been designed for such applications. Researchers have lately used ML techniques to tackle mobility in WSNs through SDN.

Theodorou et al. [131] proposed SD-MIoT, which is an SDN-based solution for mobile IoT applications. SD-MIoT aims to reduce the control overhead by detecting the mobility behaviour of sensor nodes. The mobility detector uses network adjacency matrices built upon collected sensor data at the controller. Given a simple mobility scenario as shown in Fig 5, the mobility detector build a connected graph $G = (N, E)$ where N is the set of sensor nodes and E the set of communication links between sensor nodes. It then builds the adjacent matrix A_t, at time t, of G. Where each element

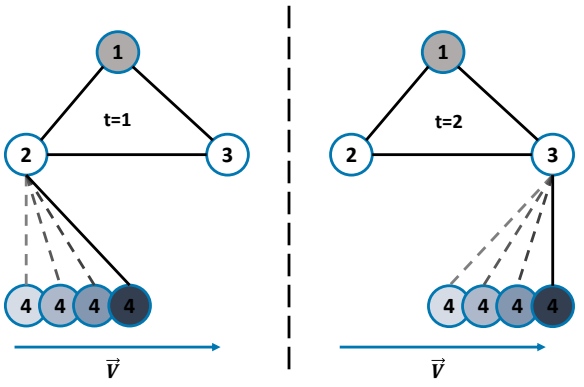


FIGURE 5. Mobility detector scenario [131].

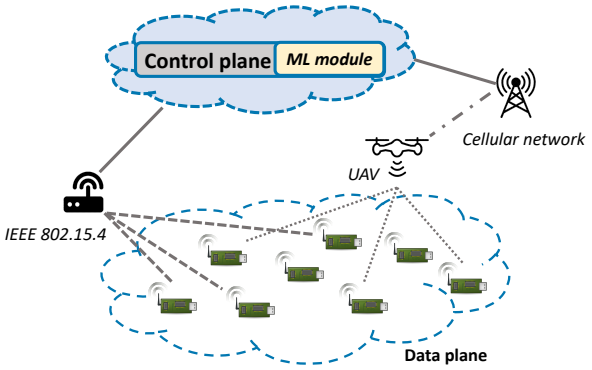


FIGURE 6. An ML-SDWSN architecture with an Unmanned aerial vehicle (UAV).

of $A_{t(i,j)}$ is defined as:

$$A_{t(i,j)} = \begin{cases} 1 & \forall i,j \text{ if node } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

To detect connectivity changes, a square transition matrix is calculated at two subsequent adjacent matrices as follows:

$$\begin{aligned} T_t &= ||A_t - A_{t-1}|| \\ &\vdots \\ T_{t-(k-2)} &= ||A_{t-(k-2)} - A_{t-(k-1)}|| \end{aligned} \quad (2)$$

The transition matrix will contain rows, which represents sensor nodes, with connectivity changes. If all elements of a particular row have a zero value indicates that there are no changes for that row (node); therefore, it is assumed that the sensor node is a fixed node. When multiple connectivity changes are detected in a row (sensor node), it is assumed to be a mobile node. When a single connectivity change is detected, the mobility status of the sensor cannot be defined; however, a simple moving average is tuned to find the best window to allow early connectivity detection while minimising the number of false positives. Then, the mobility detector applies the *k-means* cluster algorithm to separate static nodes from mobile nodes. The routing protocol proactively and constantly deploys forwarding rules to mobile nodes, therefore, reducing the control overhead. The decision module based on ML is placed in the application plane of the SDWSN architecture.

SDN-(UAV)ISE is introduced in [132] for WSNs with data mules. The network architecture, shown in Fig. 6, comprises a data plane based on low power sensor nodes, a cellular network base station to enable communication with the UAV and the control plane that host the ML module. The drone, which acts as a mobile node, serves as a relay node to the SDN controller. The 'set cover problem' is used to find the optimal position to reduce the number of destination to visit, thus, minimising energy consumption and time. A *DT* algorithm is used to predict the medium-long term mobility of the drone. The training dataset is constantly update using the collected data of sensor nodes. The forecasted movements of the drone permit to forecast of the topology changes, so the flow table is created beforehand to reach the drone, thus, reducing the number of control packets generated. SDN-(UAV)ISE reduces the control overhead specially when the topology changes.

Roy et al. [133] proposed a *Reinforcement Learning (RL)* based adaptive topology control approach. This approach is used in a WSN with mobile nodes to improve network latency, PDR and energy efficiency. It is then demonstrated that RL presents poor overall QoS when mobility is erratic. They then discuss the use of supervised learning algorithms (e.g. *Recurrent Neural Network (RNN)*) to identify nodes with low periodicity to mitigate their impacts on QoS.

B. SECURITY

The broadcast nature of WSNs imposes unique challenges. Traditional security solutions cannot be applied directly. Sensor nodes are resource-constrained devices, while most of the traditional techniques require processing-intensive functions. Sensor nodes are also deployed in harsh environments, making them susceptible to physical attacks, and finally, sensor nodes often interact closely with the physical environment and people, creating new security issues [134]. A simple representation of an ML-SDWSN architecture with watermark enabled is depicted in Fig. 7. SDN-based approaches open up new opportunities to solve the above-mentioned challenges in WSNs.

Miranda et al. [135] proposed a collaborative security framework for SDWSNs. It includes an Intrusion Detection System (IDS) in the data plane and an anomaly detection solution near the data plane. A smart monitoring system along with an *Support Vector Machine (SVM)* algorithm is used to improve anomaly detection and mitigation by isolating malicious nodes. At the data plane, CHs generate and embed watermark to data and the sink node runs a watermark detection algorithm to ensure the accuracy of

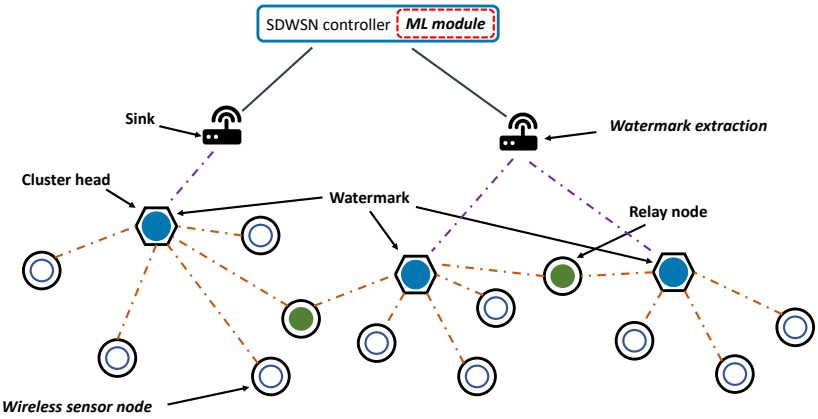


FIGURE 7. A simple representation of an ML-SDWSN architecture with watermark enabled.

recurrent authentications while implementing data integrity inspections. *Kgogo et al.* [136] also proposed an IDS using ML to identify which ML algorithm performs better in the detection of threats and attacks. The algorithms tested were *DT*, *SVM*, and *logistic regression*. Results demonstrated that the SVM model is the most effective in detecting both normal and anomaly instances, followed by DT. However, DT is the most efficient and effective in detecting network intrusion in real-time, so the SDWSN can react to any intrusion instantaneously. *Chen et al.* [137] presented a ML-based DDoS attack detection system. They deployed various wireless sensor nodes in eight poles to collect the data. They extracted the features based on the execution of multiple DDoS attacks including ICMP flood, SYN flood, and UDP flood, with different periods and duration times. Results show that DT achieved over 97% in accuracy. *Zhao et al.* [138] proposed a trusted link-separation method for SDWSNs in adversarial environments. They consider both routing efficiency and security. They use a *Bayesian-based* model to evaluate sensor nodes' trustworthiness based on their communication interactions. They formulate a multi-objective optimisation problem for the trusted link-separation multipath. The optimisation problem is solved using greedy algorithm.

C. ENERGY EFFICIENCY

This metric has been previously introduced in Section IV-B1. Here, we group research works that use ML techniques to improve energy efficiency in SDWSNs.

Huang et al. [139] proposed an SDWSN prototype to improve energy efficiency in environmental monitoring applications. They use *RL* to perform value-redundancy filtering and load-balancing routing that can adapt to environmental variations and network status, improving energy efficiency and adaptability of WSNs for environmental monitoring applications.

Banerjee et al. [140] proposed an *RL* approach to control the transmission range of SDWSNs with moving nodes. Sensor nodes have multiple transmission power levels, and to decide the optimum power level an Epsilon(ϵ)-greedy algorithm is used. This RL approach gains knowledge from velocities of successors and link quality metrics such as Received Signal Strength Indicator (RSSI), packet reception rate, and attenuation.

To prolong the NL of the SDWSN, an *RL* approach that trains the SDN controller to optimise the routing paths is proposed in [141]. The controller gets the rewards in terms of estimated path lifetime loss. The RL uses four reward functions aimed to extend the NL and reduce energy consumption. Results show a NL improvement of 23%-30% as compared to RL-based WSN.

Abdolmaleki et al. [142] proposed a *Fuzzy* topology discovery protocol for SDWSNs. They implemented a fuzzy logic based SDN controller to improve network performance. The fuzzy logic controller considers the neighbours, traffic, workload level, and remaining energy of each sensor node to choose the best forwarding node. Results show that the proposed approach extended the NL by 45% and the PLR by 50%.

A reduced energy consumption and control overhead can be achieved by using a model that predict the energy consumption of each sensor node. *Rahimifar et al.* [143] proposed a *Markov-based* model to predict future energy consumption of sensor nodes. The controller predicts individual energy consumption of sensor nodes; thus, sensor nodes avoid reporting energy levels to the controller.

D. RELIABILITY

In order to minimise power outage, which are due to persistent fault and over utilisation of distribution transformers (DTs), of electrical distribution systems, a remote IoT monitoring and fault prediction system is proposed in [144]. Their approach is a low-cost implementation of a distributed controller architecture with wireless sensor nodes attached to transformers. The LoRa sensor nodes are equipped with a temperature, oil level, humming noise, and overloading

sensor. They act as a health tracker of the transformers. The prediction system uses an *NN* algorithm, which runs on the management plane for prediction on real-time sensor traffic, to improve the smart-grid reliability, transformers health check, and maintenance practises. This is a practical implementation of SDN-based WSNs, and the use of ML to improve the overall system performance.

Leveraging the global view of the controller, monitoring the network infrastructure allows employing suitable traffic engineering techniques to improve network performance. An SDN-based IoT architecture is presented in [145] to perform time granular analysis of network traffic for efficient network management. They used different supervised learning algorithms including DT, SVM, and k-NN to examine the network traffic. Results showed an overall accuracy rate of over 90%, but k-NN achieved 98% accuracy. Other research work that addresses network traffic by means of non-supervised DL but from the wireless medium perspective, in general, can be found in [151].

With the advent of Internet technologies, new applications have emerged. Each application impose different bandwidth requirements. It is of great importance to have network resources balanced to comply with strict QoS requirements. The research work presented in [146] aims to minimise the number of unsatisfied user equipment while maximising the throughput of the network by means of load balancing. They used an *NN*, which was improved using the fruit fly optimisation (FOA) algorithm, to solve this problem.

Since the network infrastructure should dynamically adapt to the user requirements, there should be a decision-making stage that chooses the routing protocol that meets the user-specific requirements. *Misra et al.* [147] proposed a situation-aware protocol switching for SDWSNs. They designed an adaptive controller that deploys the appropriate routing protocol based on the network conditions and application-specific requirements. The decision-making stage is based on a *supervised learning* algorithm, which trains the SDN controller, therefore, it can dynamically switch among routing protocols, as per user-specific requirements.

As the location of SDWSN controllers is key to enhance the network performance, it is of paramount importance to find the best location that satisfies the user requirements. ML has been recently being used to solve the multi-controller placement problem in SDWSNs. In [148] an energy-aware multi-controller placement solution using a *PSO* for minimising energy consumption is presented. Moreover, a *Deep Reinforcement Learning (DRL)* algorithm resource allocation strategy is conceived to reduce the waiting time of tasks.

Researchers have realised that cognitive radio technology can be effectively used along with SDN abstractions to enhance the utilisation of spectrum resources. In [149] a sustainable SDWSN architecture with cognitive radio technology for efficient power management, channel handoffs and spectrum utilisation is proposed. The proposed work has an *RL* algorithm for efficient spectrum utilisation. The network performance is improved by introducing new capabilities such as dynamically adaptation to spectrum and interference conditions. *Orfanidis et al.* [150] also intended to refine the robustness of the network by identifying multiple sources of interference altering the network. They planned to use a *supervised statistical ML* approach. A multivariate linear regression algorithm was planned to use which runs in the SDN controller. A testbed with multiple sources of interference, such as Bluetooth [35] and WiFi [105] networks, was proposed. The feature vector for the statistical model proposed includes PDR, energy consumption, interference, RSSI, end-to-end delay, and noise.

E. DISCUSSION

ML-SDWSN is a new paradigm that has emerged due to (i) the increasing popularity and demonstrated capability of SDWSNs to enhance network performance, (ii) the ML potential to further improve network performance of SDWSNs, and (iii) the ML potential to overcome the concerns raised when introducing SDN concepts in WSNs. From Table 15, we can observe that ML-SDWSNs are still in an early development stage. However, a notable exploration has been already achieved. ML techniques has been applied to a range of network issues. To highlight, ML has been shown great ability to reduce the amount of control overhead (packets) flowing in the network. Given the global view, granted through the SDWSN architecture, ML provides accurate predictions, so the controller can act promptly to changes in the network, reconfiguring proactively network resources, thus, reducing the control overhead and energy consumption. Security, the collected data e.g. network statistics, raw data, etc., give the ML the necessary data to make precise predictions and decisions in identifying network pitfalls, intrusions, etc., and mitigate their impact promptly. Energy is one of the most popular metric to consider in low power sensor networks, ML has been used to balance the overall energy consumption to prolong the NL. ML shows good performance in predicting sensor nodes remaining energy, relay nodes, transmission range; therefore, minimising the control overhead and energy consumption. The logically centralised architecture of SDWSN and the power of ML in predicting network traffic, selecting the best routing protocol, and spectrum utilisation make ML-SDWSN a good candidate for enhancing WSN reliability.

Overall, ML-SDWSN is built upon a multidisciplinary area that puts together the best of communication networks, software-defined networking and machine learning concepts to go beyond the current state-of-art knowledge in SDWSNs to facilitate WSN programmability without putting at risk the network performance. However, there still is room to explore ML techniques in SDWSNs, but, most importantly to evaluate the benefits that ML brings to SDWSN, especially, against traditional WSN.

VII. SUMMARY OF SDWSN PROPOSALS

In this section, we provide simple statistics of previously discussed SDWSN proposals. This will allow us to uncover

TABLE 15. Comparison of relevant ML-SDWSN approaches

Ref.	Aim	ML technique	Improvement	Main drawback
[131]	Mobility	K-means to categorise static and mobile nodes	Reduced control overhead and improved PDR by proactively deploying flow rules to sensor nodes	Mobility false positives
[132]	Mobility & reliability	DT to predict mobility of the drone	Reduced control overhead and energy consumption and improved PDR	Additional complexities involved in the operation of the drone
[133]	Periodic mobility	RL for periodic mobility & nearest centroid	Lower network latency	Considerable learning time for movements of high periodicity
[135]	Security	SVM for anomaly detection and mitigation	Improves anomaly detection rate, lower computational complexity, and reduces false alarms	An increase in packet sizes and computational resources at sensor nodes
[136]	Security (IDS)	DT, SVM, & logistic regression	Evaluation of ML algorithms that perform better in detecting threats and attacks. Real-time detection	Detection rate relatively low
[137]	Security (DDoS)	DT to identify different types of DDoS attacks	A ML-SDWSN system that detects and mitigates three types of attacks with high accuracy	A relative high packet overhead due to the use of IP-enabled network
[138]	Security (trust)	Bayesian approach to compute nodes' reputation	Improved routing security and efficiency of transmission paths	Problem complexity
[139]	Energy	RL for value-redundancy filtering and balancing the routing path	Improved energy	Scalability issues
[140]	Energy (Tx range)	RL for transmission range control	Improved energy consumption, delay and throughput	Network reliability due to the adaptive transmission range
[141]	Energy (NL)	RL for extended NL	Extended NL	A relative high packet overhead due to the use of IP-enabled network
[142]	Energy	Fuzzy logic to choose the best relay node	Extended NL and reduced PLR	Relatively high control overhead due to sensor reporting
[143]	Energy & control overhead	Markov model to predict the energy consumption of sensor nodes	Reduced control overhead	Higher processing energy and memory use in sensor nodes
[144]	Reliability	NN that predicts network traffic	Improved reliability for DTs by handling future interruption and faults	Locating the SDN controller in sensor nodes can lead to exhaust its resources faster
[145]	Network traffic	DT, SVM and K-NN to inspect network traffic	Timely decisions based on the ML predictions	Tasks were not reprogrammed at the sensor level
[146]	Throughput	NN to minimise unsatisfied user equipment and maximise the throughput	Resources balanced to improve the QoS	It lacks experimental validation
[147]	Dynamic/real-time routing	Multiple supervised learning algorithms	Improves network performance by selecting the best candidate for routing protocol given the current network status	A drop in PDR performance due to retransmitted packets in the switching phase
[148]	Multi-controller placement	RL (DRL), and PSO	Reduced waiting tasks and energy consumption for controllers	It lacks experimental validation
[149]	Spectrum	RL for spectrum utilisation	Eliminates channel handoffs, which are energy-intensive tasks, by predicting users traffic	Static CHs can exhaust their resources faster
[150]	Interference	Multivariate linear regression	An improvement in network reliability by taking prompt actions when identifying sources of interference	It lacks practical details regarding physical implementation and performance metrics.

research open issues and future trends in SDWSNs.

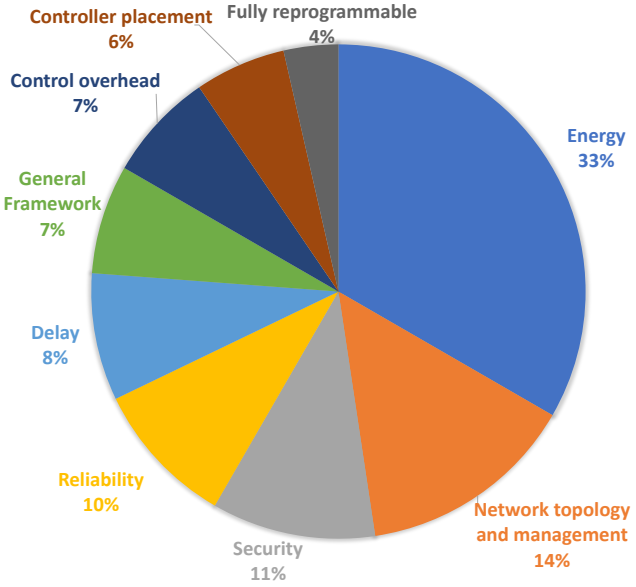
A. SUMMARY

Fig. 8a shows the percentage of research works for each category. This let us discover where most of the research efforts in SDWSN has focused. Most of the proposed research works leverage SDN concepts to reduce energy consumption and management complexities currently found in WSNs. In contrast, the least number of research works focused on making the sensors fully reprogrammable.

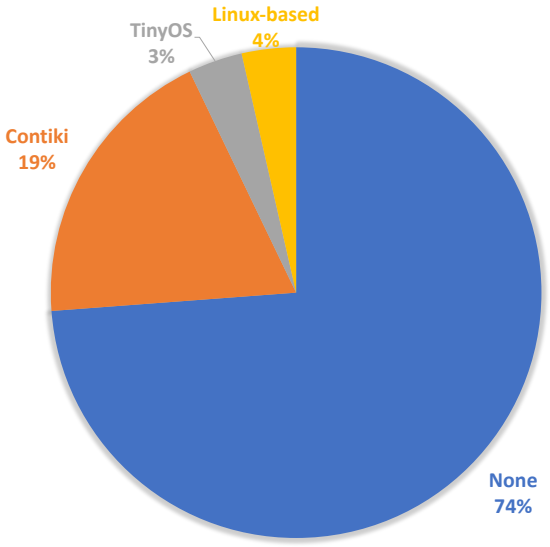
The most popular embedded operating system used in SDWSN is Contiki as shown in Fig. 8b. Research works that have not used any type of operating system are largely influenced by research works that aim to reduce energy consumption in SDWSNs in which most of them used a numerical tool such as MATLAB.

It is of great importance to identify the most used performance metrics as they also help to pinpoint where most of the research effort resides. Similar to WSNs, the most popular performance metric to improve is energy consumption as shown in Fig. 8c. The control overhead, which is among the most important metrics, is considered in the 11% of the surveyed works. Packet delivery metrics such as PDR and PLR are considered in the 8% of the proposals.

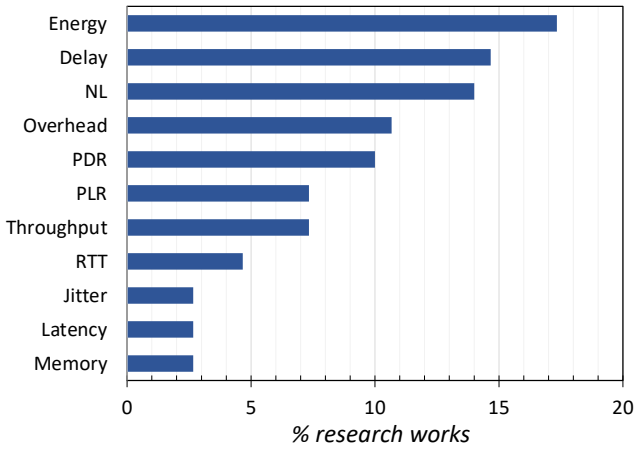
Fig. 8d shows the percentage of number of research works that have used any type of evaluation. Even though most of the research efforts aim to reduce energy consumption, which largely influences numerical evaluation methods, in SDWSNs, the most popular network simulator is Cooja, which is the Contiki network simulator. Mininet and NS-3 that offer add-on modules (e.g., WiFi, OpenFlow, etc.) to reduce the time to design a simulation environment was used in 6% and 4% of the surveyed works, respectively. 10% of the research works did not have any form of simulation



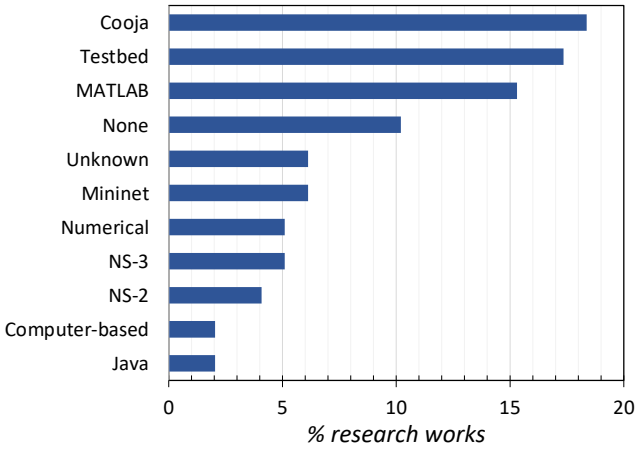
(a) Percentage of research works per category.



(b) Most popular embedded operating systems.



(c) Most popular performance metrics.



(d) Most popular type of evaluation platforms.

nor experimental evaluation. Overall, 41% of the surveyed works were evaluated using simulations tools, 22% through testbeds, 21% employing numerical approaches. The remaining 16% of the works did not use any evaluation method or it is unknown.

B. POPULARITY OF SDWSN AND VENUES OF PUBLICATION

The first research works that start exploring the use of SDN concepts in the WSN architecture appeared around 2012. Then, a number of research works start appearing to extend the use of SDWSNs to a vast variety of IoT applications. However, exponential growth is perceived from 2017. This agrees with the number of research works on ML techniques in SDWSNs that started to emerge. In 2019 and 2020+, the growth continued exponentially. This is influenced by the number of research works that have used previous works, which have their code freely available, to devise new solutions to improve network performance. This exponential growth shows that the research community sees SDWSNs as a potential pathway to overcome the management complexity currently found in the current state-of-art WSNs.

The publication venues of scientific publications reporting on SDWSNs is shown in Fig. 8. As can be seen from the figure, the most popular dissemination method, by far, is journals, followed by conference proceedings. Workshops and forums are the least popular dissemination methods. The journal publications are widespread across different venues. However, looking at specific journals venues, not shown here due to space constraints, the most popular journals are IEEE Internet of Things Journal with 9 publications, followed by IEEE Systems Journal with 6 publications and Sensors (MDPI), IEEE Sensors Journal and Journal of Ambient Intelligence and Humanized Computing (JAIHC) with

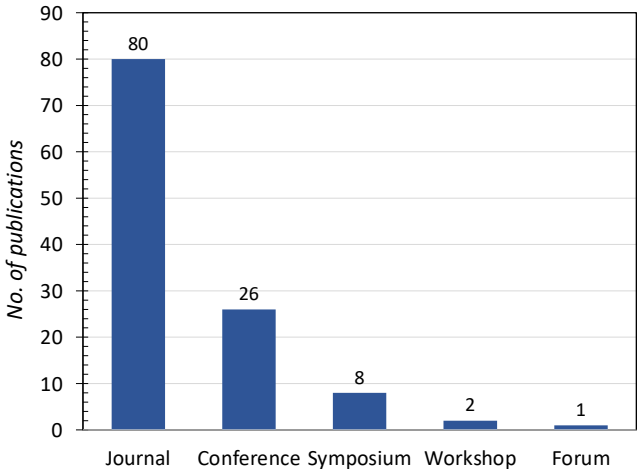


FIGURE 8. Publication venues of scientific articles reporting on SDWSNs.

5 publications.

VIII. MAJOR CHALLENGES

SDWSNs is a relative new and continuous evolving research area. Previous sections provided a comprehensive review and discussions of SDWSN and ML-SDWSN research works. The objective of this section is to group and discuss open issues currently found in state-of-art SDWSNs.

A. STANDARDISATION

SDWSNs have to deal with the exponential growth of wireless sensor devices, a vast variety of manufacturers, and protocols. The creation of standards for such rapidly evolving technology, with various group of stakeholders, is not an easy task [152]. Some SDWSN papers share similar architectural designs and protocols, while others have their own new architectures and protocols. The standardisation of SDWSN should be seen as a holistic architecture that covers all layers involved in the model. The exponential growth of scientific articles calls for an urgent standardisation. Otherwise, this will result in incompatible architectures, and protocols that will go against the SDN principles [9]. Therefore, affecting the rate at which new SDWSN proposals are emerging.

B. CONTROL OVERHEAD

One major concern of adopting SDN principles in WSNs is the control overhead. SDN was originally designed for wired networks where control packets flow through a dedicated control channel. In contrast, SDWSNs share the same communication medium for both control packets and data packets. Even though control overhead has been indirectly addressed in many research works (see Fig. 8c), papers that specifically focus on reducing the control overhead is still low as shown in Fig. 8a. Minimising the number of control packets is of a great deal to avoid impacting the network performance negatively.

Research works have applied multiple techniques to reduce the control overhead as shown in Table 11. Research works that synergy all those techniques simultaneously with ML techniques can lead to a significant improvement in control overhead. For example, the use of ML techniques to tackle mobility in WSNs can greatly reduce the control overhead by proactively and constantly setting the path for packets generated by mobile nodes. This reduces the amount of *packet-in* messages, which are flow setup request sent by sensor nodes to the controller to seek instructions on how to handle an incoming packet that is not present in its forwarding table.

1) Neighbour advertisement and network configuration

SDWSNs have two main functions that generate control packets [58]. (i) *Neighbour advertisement (NA)* which is a key function in the initial phase of the SDWSNs setup. Sensor nodes use NA messages to advertise their current and neighbour status. The SDN controller builds a global view of the network using NA messages. Sensor nodes also use NA message to keep the controller updated on any change in the network. The frequency of NA messages directly affects the network performance. Frequent NA messages immediately warn the controller about any change in the network (e.g., dead node, interference, battery depletion, etc.) but at the cost of increased control overhead and energy consumption, while infrequent NA messages reduce the impact on network performance, the controller would not be able to react immediately to changes in the network. (ii) *Network Configuration (NC)* is used by the controller to manage and control the overall behaviour of the network. Literature review reveals that NC packets are mainly used to dynamically program forwarding tables of sensor nodes. Overall, there still are research gaps to reduce control overhead in SDWSNs. What should be the optimal frequency of NA messages without affecting the network performance, also how to deliver NC messages effectively and at the right timing while minimising the impacts on network performance.

C. SECURITY

Along with the control overhead, security is one of the main concerns in SDWSNs. Security in WSNs, in general, is one of the research areas that have caught most of the researchers' attention. WSNs impose unique challenges due to the dynamic behaviour of communication links. Moreover, sensors nodes have limited resources that restrain the use of traditional security solutions. However, the centralised architecture of SDN brings advantages when devising new countermeasure solutions for security threats. The global view of the network at the controller facilitates constantly and proactively detecting changes in the network. Also, the centralised network information calls for the use of ML-based solutions. Security in SDWSN is still in its initial stage as shown in this survey. But, it makes sense to use ML algorithms in SDWSNs due to the centralised architecture. The centralised architecture offloads the power-intensive compu-

tational tasks from the network infrastructure, then security applications can be easily implemented at the controller. The advantages and disadvantages of centralised or distributed security solutions based on ML need to be studied in detail. Centralised architectures have an overall view of the network facilitating the detection of abnormal behaviours but at the expenses of more network information. In contrast, in distributed architectures sensor nodes can also perform some amount of processing to run lightweight ML solutions, which minimises the control overhead, but it may increase the energy consumption due to the processing.

D. CONTROLLER PLACEMENT

The location of the controller in the network directly affects the network performance. Controller placement has been widely studied in SDN [118], whereas controller placement in SDWSNs still in its infancy stage. Although SDWSN is inspired by SDN, the communication medium differs. Therefore, the optimal placement of the SDWSN controller can be based on previous research works on SDN, however, the placement has to be subject to specific characteristics of the transmission medium, in this case, wireless. The controller placement is also tightly related to scalability problems in SDWSNs.

E. EMBEDDED OPERATING SYSTEM (EOS)

Fig. 8d reveals that most of the research works, in this survey, did not adopt any type of EOS. In fact, there still are a number of EOSs that have not been yet used in SDWSNs. For instance, there is not evidence of any SDWSN solution that have used a Real Time Operating System (RTOS). An RTOS works on strict processing time requirements. This can serve for SDWSN applications that require some level of reliability. In general, the use of EOSs aligns with SDN principles. It brings flexibility when adding new applications to sensors' programs. The use of EOSs makes sensor nodes to be seen as small-scale computers with multiple sensing capabilities, and they are also supported in a variety of sensor platforms, shrinking the interoperability breach.

F. SCALABILITY

This is another big concern in centralised architectures such as SDN. It is known that the management overhead increases as the network increases. Several techniques have been proposed to address scalability issues in SDWSNs. Among the most widely used techniques is the use of multiple controllers. The control plane may include physically distributed controllers. The location of the controllers directly affects the network performance, as discussed in Section IV-E. The network management load can be balanced across multiple controllers. Each controller oversees a specific zone of the network topology. However, one concern that rises up is to find the optimal number of controllers required before affecting network performance. Also, how to cope with the dynamic nature of WSNs. The use of static controllers can directly affect the NL.

G. MACHINE LEARNING (ML)

The 25% of the research works surveyed here adopted ML techniques in their proposals. The first ML-SDWSN articles started appearing in 2015; however, ML-based works took off in 2018. The year with the most numbers of publications in ML-SDWSN was 2020+ with 9 publications. This increasing popularity shows that ML has been seen as an attractive solution to improve network performance on SDWSNs. The adoption of ML in SDWSNs has shown good performance in reducing control overhead, prolonging NL, and intrusion detection. However, there still areas to explore and ML techniques to use. For example, the dynamic nature of WSNs unfolds new opportunities to envision ML techniques that automatically continuous learning including AutoML and transfer learning. The use of an online AutoML structure will allow the system to continuously adapt to new situations while reducing the need for a long training phase on a big dataset that might not even be available. Transfer learning will permit learning from simulation or controlled environments and deploy them in real-world applications, which might improve the learning rate, accuracy or the need for less training data. DL could be useful in unveiling which kind of features or parameters are actually more relevant to the specific user application. Besides, the use of multiple architectures such as centralised or distributed ML techniques should be studied in depth. The time complexity of algorithms should be also considered, especially for real-time applications with strict time constraints and resource-constrained IoT devices.

H. TESTBEDS FOR SDWSNS

SDWSNs have different network topologies. Some topologies have the controller embedded in one of the sensor nodes. This imposes strict hardware requirements such as sensor nodes with enough resources to run centralised protocols, store network information and with access to main powers. Other topologies require multiple embedded controllers; therefore, the network infrastructure must provide multiple sensor nodes with large resources. In contrast, SDWSN topologies with the controller connected directly to the sink node (e.g. via serial interface, USB) requires fewer resources from sensor nodes but requires a higher computing machine connected to the sensor node such as a PC, Raspberry Pi, etc. Therefore, a testbed for SDWSNs needs to account for different network topologies, provide an accurate and high dynamic range for power measurements, CPU resources, multiple sensor platforms and EOSs, and debug tools including packet sniffer.

IX. CONCLUSION

The SDWSN paradigm is built upon the synergies research efforts between SDN and WSNs. SDWSN has been envisioned to solve the management complexities currently found in the current state-of-art WSNs. Overall, SDWSN will help industrial and research organisations accelerate the designing, building, and testing of emerging IoT applica-

tions, by simplifying the introduction of new abstractions, removing the management complexities, and costs. This paper presented a comprehensive review of SDWSN research works and ML techniques to perform network management and reconfiguration, and policy enforcement. Additionally, we also provided helpful information and insights to stakeholders interested in state-of-art SDWSNs, ML techniques, testbeds and open issues. This survey has unveiled that although the introduction of SDN abstractions into WSNs is a relatively new topic, notable exploration has already been achieved. The surveyed scientific articles have demonstrated that SDWSN is an effective solution for improving network performance and management, which would not have been possible with traditional WSN architectures. Despite these major achievements, there are several open issues such as standardisation, control overhead, scalability and security that need to be addressed adequately to reach the real promise of a fully reprogrammable network for IoT applications. This survey also reveals that the use of ML algorithms over the SDWSN is becoming popular and shows good performance in tackling the major issues in SDWSN. According to the surveyed articles and statistics performed, we believe that the synergy between ML and SDWSNs can shape networking decisions smarter and robust, and that ML will play a major role in the creation of new applications and protocols for SDWSNs.

REFERENCES

[1] F. Wortmann and K. Flüchter, "Internet of Things," *Business & Information Systems Engineering*, vol. 57, no. 3, pp. 221–224, 2015.

[2] F. Computing, "The Internet of Things: Extend the cloud to where the things are," Cisco Syst., San Jose, CA, USA, Report, 2016.

[3] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[4] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE communications surveys and tutorials*, vol. 16, no. 4, pp. 1955–1980, 2014.

[5] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

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

[7] K. Sood, S. Yu, and Y. Xiang, "Software-defined wireless networking opportunities and challenges for Internet-of-Things: A review," *IEEE Internet of Things Journal*, vol. 3, no. 4, pp. 453–463, 2015.

[8] M. Ndiaye, G. P. Hancke, and A. M. Abu-Mahfouz, "Software defined networking for improved wireless sensor network management: A survey," *Sensors*, vol. 17, no. 5:1031, pp. 1–32, 2017.

[9] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "A survey on software-defined wireless sensor networks: Challenges and design requirements," *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

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

[11] O. G. Matlou and A. M. Abu-Mahfouz, "Utilising artificial intelligence in software defined wireless sensor network," in *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2017, Conference Proceedings, pp. 6131–6136.

[12] H. Luo, K. Wu, R. Ruby, Y. Liang, Z. Guo, and L. M. Ni, "Software-defined architectures and technologies for underwater wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2855–2888, 2018.

[13] K. M. Modieginyane, B. B. Letswamotse, R. Malekian, and A. M. Abu-Mahfouz, "Software defined wireless sensor networks application opportunities for efficient network management: A survey," *Computers & Electrical Engineering*, vol. 66, pp. 274–287, 2018.

[14] H. Mostafaei and M. Menth, "Software-defined wireless sensor networks: A survey," *Journal of Network and Computer Applications*, vol. 119, pp. 42–56, 2018.

[15] M. Abujubbeh, F. Al-Turjman, and M. Fahrioglu, "Software-defined wireless sensor networks in smart grids: An overview," *Sustainable Cities and Society*, vol. 51, p. 101754, 2019.

[16] S. M. W. Umba, A. M. Abu-Mahfouz, T. Ramotsoela, and G. P. Hancke, "A review of artificial intelligence based intrusion detection for software-defined wireless sensor networks," in *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*. IEEE, 2019, Conference Proceedings, pp. 1277–1282.

[17] W. Dargie and C. Poellabauer, *Fundamentals of wireless sensor networks: theory and practice*. John Wiley & Sons, 2010.

[18] K. Sohraby, D. Minoli, and T. Znati, *Wireless sensor networks: technology, protocols, and applications*. John wiley & sons, 2007.

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

[20] F. Karray, M. W. Jmal, A. Garcia-Ortiz, M. Abid, and A. M. Obeid, "A comprehensive survey on wireless sensor node hardware platforms," *Computer Networks*, vol. 144, pp. 89–110, 2018.

[21] N. Ickes, F. Lee, and P. Phanaphat, "Hardware architecture for a power-aware microsensor node."

[22] J. Beutel, O. Kasten, and M. Ringwald, "BTnodes–a distributed platform for sensor nodes," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, Conference Proceedings, pp. 292–293.

[23] M. Healy, T. Newe, and E. Lewis, "Wireless sensor node hardware: A review," in *SENSORS, 2008 IEEE*. IEEE, 2008, Conference Proceedings, pp. 621–624.

[24] T. Datasheet, "Crossbow inc," 2013.

[25] A. Systems, "Wismote." [Online]. Available: http://www.aragosystems.com/produits/wisnet/wismote/

[26] WiSense, "System overview," 2020. [Online]. Available: https://wisense.in/

[27] T. Instruments, "CC2541DK-SENSOR Development kit," 2013. [Online]. Available: https://www.ti.com/tool/CC2541DK-SENSOR

[28] ——, "CC2538 powerful wireless microcontroller system-on-chip for 2.4-GHz IEEE 802.15. 4, 6LoWPAN, and Zigbee applications," *CC2538 datasheet (April 2015)*, 2015.

[29] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister, "OpenMote: Open-source prototyping platform for the industrial IoT," in *International Conference on Ad Hoc Networks*. Springer, 2015, Conference Proceedings, pp. 211–222.

[30] Zolertia, "Re-mote." [Online]. Available: https://zolertia.io/product/re-mote/

[31] T. Instruments, "CC1350STK Development kit," 2017. [Online]. Available: https://www.ti.com/tool/CC1350STK

[32] ——, "LPSTK-CC1352R Evaluation board," 2019. [Online]. Available: https://www.ti.com/tool/LPSTK-CC1352R

[33] M. A. Alwadi, "Energy efficient wireless sensor networks based on machine learning," Thesis, 2015.

[34] G. M. Bragg, "Standards-based Internet of Things sub-GHz environmental sensor networks," Thesis, 2017.

[35] IEEE, "Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPAN)," 2004.

[36] Z. Specification, "ZigBee alliance IEEE standard 802.15.4k2013," 2014. [Online]. Available: https://www.zigbee.org/zigbee-for-developers/network-specifications/

[37] V. C. Gungor and G. P. Hancke, "Industrial wireless sensor networks: Challenges, design principles, and technical approaches," *IEEE Transactions on industrial electronics*, vol. 56, no. 10, pp. 4258–4265, 2009.

[38] M. Ehrlich, L. Wisniewski, and J. Jasperneite, *State of the art and future applications of industrial wireless sensor networks*. Springer, 2018, pp. 28–39.

[39] I. L. W. Group, "IPv6 over low power WPAN (6LoWPAN)." [Online]. Available: https://datatracker.ietf.org/wg/6lowpan/charter/

[40] L. Militano, M. Erdelj, A. Molinaro, N. Mitton, and A. Iera, "Recharging versus replacing sensor nodes using mobile robots for network maintenance," *Telecommunication Systems*, vol. 63, no. 4, pp. 625–642, 2016.

[41] W. Xiang, N. Wang, and Y. Zhou, "An energy-efficient routing algorithm for software-defined wireless sensor networks," *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7393–7400, 2016.
[42] J. Beuchert, F. Solowjow, S. Trimpe, and T. Seel, "Overcoming bandwidth limitations in wireless sensor networks by exploitation of cyclic signal patterns: An event-triggered learning approach," *Sensors*, vol. 20, no. 1, p. 260, 2020.
[43] H.-L. Shi, K. M. Hou, H.-Y. Zhou, and X. Liu, "Energy efficient and fault tolerant multicore wireless sensor network: E²MWSN," in *2011 7th International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE, 2011, Conference Proceedings, pp. 1–4.
[44] W. Xu, J. Zhang, J. Y. Kim, W. Huang, S. S. Kanhere, S. K. Jha, and W. Hu, "The design, implementation, and deployment of a smart lighting system for smart buildings," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 7266–7281, 2019.
[45] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.
[46] F. F. Jurado-Lasso, K. Clarke, A. N. Cadavid, and A. Nirmalathas, "Energy-aware routing for software-defined multihop wireless sensor networks," *IEEE Sensors Journal*, vol. 21, no. 8, pp. 10 174–10 182, 2021.
[47] T. Luo, H.-P. Tan, and T. Q. Quek, "Sensor OpenFlow: Enabling software-defined wireless sensor networks," *IEEE Communications letters*, vol. 16, no. 11, pp. 1896–1899, 2012.
[48] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks: Unbridling SDNs," in *2012 European Workshop on Software Defined Networking*. IEEE, 2012, Conference Proceedings, pp. 1–6.
[49] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 2004, Conference Proceedings, pp. 455–462.
[50] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, and E. Brewer, "TinyOS: An operating system for sensor networks," *Ambient intelligence*, vol. 35, pp. 115–148, 2005.
[51] R. Alves, D. Oliveira, G. A. Núñez, and C. B. Margi, "IT-SDN: Improved architecture for SDWSN," in *XXXV Brazilian Symposium on Computer Networks and Distributed Systems*, 2017, Conference Proceedings.
[52] B. T. de Oliveira and C. B. Margi, "TinySDN: enabling TinyOS to software-defined wireless sensor networks," in *XXXIV Simpósio Brasileiro de Redes de Computadores. Bahia*, 2016, Conference Proceedings, pp. 1229–1237.
[53] F. Olivier, G. Carlos, and N. Florent, "SDN based architecture for clustered WSN," in *2015 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, 2015, Conference Proceedings, pp. 342–347.
[54] T. Theodorou and L. Mamatas, "CORAL-SDN: A software-defined networking solution for the Internet of Things," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2017, Conference Proceedings, pp. 1–2.
[55] i. iLab.t, "Wireless testlab and officelab," 2021. [Online]. Available: https://doc.ilabt.imec.be/ilabt/wilab/index.html
[56] W. Ejaz, M. Naeem, M. Basharat, A. Anpalagan, and S. Kandeepan, "Efficient wireless power transfer in software-defined wireless sensor networks," *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7409–7420, 2016.
[57] D. Zeng, P. Li, S. Guo, T. Miyazaki, J. Hu, and Y. Xiang, "Energy minimization in multi-task software-defined sensor networks," *IEEE transactions on computers*, vol. 64, no. 11, pp. 3128–3139, 2015.
[58] F. F. Jurado-Lasso, K. Clarke, and A. Nirmalathas, "Performance analysis of software-defined multihop wireless sensor networks," *IEEE Systems Journal*, vol. 14, no. 4, pp. 4653–4662, 2019.
[59] R. Tumuluri, A. Kovi, and B. K. R. Alluri, "An energy-efficient algorithm using layer heads for software-defined wireless sensor networks," in *2018 International Conference on Recent Trends in Advance Computing (ICRTAC)*. IEEE, 2018, Conference Proceedings, pp. 103–108.
[60] L. Wenxing, W. Muqing, and W. Yuewei, "Energy-efficient algorithm based on multi-dimensional energy space for software-defined wireless sensor networks," in *2016 International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2016, Conference Proceedings, pp. 309–314.
[61] F. Junli, W. Yawen, and S. Haibin, "An improved energy-efficient routing algorithm in software defined wireless sensor network," in *2017 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*. IEEE, 2017, Conference Proceedings, pp. 1–5.
[62] H. Bo, W. Muqing, Z. Min, and L. Wenxing, "An energy aware routing algorithm for software defined wireless sensor networks," in *2017 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2017, Conference Proceedings, pp. 1–6.
[63] A. Banerjee and D. Hussain, "SD-EAR: Energy aware routing in software defined wireless sensor networks," *Applied Sciences*, vol. 8, no. 7, p. 1013, 2018.
[64] A. Banerjee and A. Sufian, "Smart-Green-Mult (SGM): overhear from topological kingpins in software defined wireless sensor networks," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–18, 2020.
[65] Z. Ding, L. Shen, H. Chen, F. Yan, and N. Ansari, "Energy-efficient relay-selection-based dynamic routing algorithm for IoT-oriented Software-Defined WSNs," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 9050–9065, 2020.
[66] Y. Zhu, Y. Zhang, W. Xia, and L. Shen, "A software-defined network based node selection algorithm in WSN localization," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*. IEEE, 2016, Conference Proceedings, pp. 1–5.
[67] A. Pal and A. Jolfaei, "On the lifetime of asynchronous software-defined wireless sensor networks," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6069–6077, 2020.
[68] O. Ahmed, F. Ren, A. Hawbani, and Y. Al-Sharabi, "Energy optimized congestion control-based temperature aware routing algorithm for software defined wireless body area networks," *IEEE Access*, vol. 8, pp. 41 085–41 099, 2020.
[69] T. Kgogo, B. Isong, and A. M. Abu-Mahfouz, "Software defined wireless sensor networks security challenges," in *AFRICON, 2017 IEEE*. IEEE, 2017, Conference Proceedings, pp. 1508–1513.
[70] M. Manuel, B. Isong, M. Esiefarienrhe, and A. M. Abu-Mahfouz, "Analysis of notable security issues in SDWSN," in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, Conference Proceedings, pp. 4706–4711.
[71] M. Huang, B. Yu, and S. Li, "PUF-assisted group key distribution scheme for software-defined wireless sensor networks," *IEEE Communications Letters*, vol. 22, no. 2, pp. 404–407, 2017.
[72] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, Conference Proceedings, pp. 513–521.
[73] R. Wang, Z. Zhang, Z. Zhang, and Z. Jia, "ETMRM: an energy-efficient trust management and routing mechanism for SDWSNs," *Computer Networks*, vol. 139, pp. 119–135, 2018.
[74] B. T. De Oliveira, L. B. Gabriel, and C. B. Margi, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," *IEEE Latin America Transactions*, vol. 13, no. 11, pp. 3690–3696, 2015.
[75] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "Fragmentation-based distributed control system for software-defined wireless sensor networks," *IEEE transactions on industrial informatics*, vol. 15, no. 2, pp. 901–910, 2018.
[76] C. Buratti, A. Stajkic, G. Gardasevic, S. Milardo, M. D. Abrignani, S. Mijovic, G. Morabito, and R. Verdone, "Testing protocols for the Internet of Things on the EuWIn platform," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 124–133, 2016.
[77] X. Li, Z. Ma, J. Zheng, Y. Liu, L. Zhu, and N. Zhou, "An effective edge-assisted data collection approach for critical events in the SDWSN-based agricultural Internet of Things," *Electronics*, vol. 9, no. 6, p. 907, 2020.
[78] Y. Lu and L. Da Xu, "Internet of Things (IoT) cybersecurity research: A review of current research topics," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2103–2115, 2018.
[79] J. C. C. Chica, J. C. Imbachi, and J. F. B. Vega, "Security in SDN: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 159, p. 102595, 2020.
[80] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful SDN data planes," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1701–1725, 2017.
[81] L. L. Bello, A. Lombardo, S. Milardo, G. Patti, and M. Reno, "Experimental assessments and analysis of an SDN framework to integrate mobility management in industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5586–5595, 2020.

[82] G. Li, S. Guo, Y. Yang, and Y. Yang, "Traffic load minimization in software defined wireless sensor networks," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1370–1378, 2018.

[83] S. Bera, S. Misra, and M. S. Obaidat, "Mobi-flow: Mobility-aware adaptive flow-rule placement in software-defined access network," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1831–1842, 2018.

[84] S. S. G. Shiny and K. Murugan, "TSDN-WISE: Automatic threshold based low control-flow communication protocol for SDWSN," *IEEE Sensors Journal*, 2021.

[85] S. A. Asakipaam, J. J. Kponyo, J. O. Agyemang, and F. Appiah-Twum, "Design of a minimal overhead control traffic topology discovery and data forwarding protocol for software-defined wireless sensor networks," *International Journal of Communication Networks and Information Security*, vol. 12, no. 3, pp. 450–458, 2020.

[86] A. Hawbani, X. Wang, L. Zhao, A. Al-Dubai, G. Min, and O. Busaileh, "Novel architecture and heuristic algorithms for software-defined wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 6, pp. 2809–2822, 2020.

[87] Q. Liu, L. Cheng, R. Alves, T. Ozcelebi, F. Kuipers, G. Xu, J. Lukkien, and S. Chen, "Cluster-based flow control in hybrid software-defined wireless sensor networks," *Computer Networks*, vol. 187, p. 107788, 2021.

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

[89] H. Yao, C. Qiu, C. Zhao, and L. Shi, "A multicontroller load balancing approach in software-defined wireless networks," *International Journal of Distributed Sensor Networks*, vol. 11, no. 10, p. 454159, 2015.

[90] J. Portilla, A. De Castro, E. De La Torre, and T. Riesgo, "A modular architecture for nodes in wireless sensor networks," *J. UCS*, vol. 12, no. 3, pp. 328–339, 2006.

[91] S. Natheswaran and G. Athisha, "Remote reconfigurable wireless sensor node design for wireless sensor network," in *Communications and Signal Processing (ICCSP), 2014 International Conference on*. IEEE, 2014, Conference Proceedings, pp. 649–652.

[92] T. Miyazaki, S. Yamaguchi, K. Kobayashi, J. Kitamichi, S. Guo, T. Tsukahara, and T. Hayashi, "A software defined wireless sensor network," in *2014 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2014, Conference Proceedings, pp. 847–852.

[93] K. Goh, S. Ong, Y. Joe, P. Kusolpalin, W. Moh, and K. V. Ling, "FPGA based wireless sensor node for distributed process monitoring," in *Industrial Electronics and Applications (ICIEA), 2012 7th IEEE Conference on*. IEEE, 2012, Conference Proceedings, pp. 1934–1939.

[94] H. Qi, O. Ayorinde, and B. H. Calhoun, "An ultra-low-power FPGA for IoT applications," in *2017 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*. IEEE, 2017, Conference Proceedings, pp. 1–3.

[95] A. Razzaq, S. R. Sani, and A. G. Ye, "Designing efficient FPGA tiles for power-constrained ultra-low-power applications," *Integration*, vol. 78, pp. 124–134, 2021.

[96] R. Chéour, S. Khriji, D. El Houssaini, M. Baklouti, M. Abid, and O. Kanoun, "Recent trends of FPGA used for low-power wireless sensor network," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, no. 10, pp. 28–38, 2019.

[97] D. Zeng, T. Miyazaki, S. Guo, T. Tsukahara, J. Kitamichi, and T. Hayashi, "Evolution of software-defined sensor networks," in *Mobile Ad-hoc and Sensor Networks (MSN), 2013 IEEE Ninth International Conference on*. IEEE, 2013, Conference Proceedings, pp. 410–413.

[98] S. Duquennoy, "Contiki-NG: The OS for next generation IoT devices," 2019. [Online]. Available: https://github.com/contiki-ng/contiki-ng

[99] R. Barry, "Freertos," *Internet, Oct*, 2008. [Online]. Available: https://www.freertos.org/RTOS.html

[100] T. L. F. Projects, "Zephyr project," 2021. [Online]. Available: https://www.zephyrproject.org/

[101] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, "OpenWSN: a standards-based low-power wireless development environment," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.

[102] E. Baccelli, C. Gündoğan, O. Hahm, P. Kietzmann, M. S. Lenders, H. Petersen, K. Schleiser, T. C. Schmidt, and M. Wählisch, "RIOT: an open source operating system for low-end embedded devices in the IoT," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4428–4440, 2018.

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

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

[105] IEEE, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," 2012.

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

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

[108] R. K. Das, A. K. Maji, and G. Saha, "SD-6LN: Improved existing IoT framework by incorporating SDN approach," in *International Conference on Innovative Computing and Communications*. Springer, 2021, Conference Proceedings, pp. 599–606.

[109] M. Ndiaye, A. M. Abu-Mahfouz, and G. P. Hancke, "SDNMM a generic SDN-based modular management system for wireless sensor networks," *IEEE Systems Journal*, vol. 14, no. 2, pp. 2347–2357, 2019.

[110] C. Cao, L. Luo, Y. Gao, W. Dong, and C. Chen, "TinySDM: software defined measurement in wireless sensor networks," in *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*. IEEE Press, 2016, Conference Proceedings, p. 18.

[111] A. Mavromatis, C. Colman-Meixner, A. P. Silva, X. Vasilakos, R. Nejabati, and D. Simeonidou, "A software-defined IoT device management framework for edge and cloud computing," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1718–1735, 2019.

[112] B. B. Letswamotse, R. Malekian, and K. M. Modieginyane, "Adaptable QoS provisioning for efficient traffic-to-resource control in software defined wireless sensor networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 6, pp. 2397–2405, 2020.

[113] N. Samarji and M. Salamah, "A fault tolerance metaheuristic-based scheme for controller placement problem in wireless software-defined networks," *International Journal of Communication Systems*, vol. 34, no. 4, p. e4624, 2021.

[114] S. Tahmasebi, M. Safi, S. Zolfi, M. R. Maghsoudi, H. R. Faragardi, and H. Fotouhi, "Cuckoo-pc: An evolutionary synchronization-aware placement of SDN controllers for optimizing the network performance in WSNs," *Sensors*, vol. 20, no. 11, p. 3231, 2020.

[115] S. Tahmasebi, N. Rasouli, A. H. Kashefi, E. Rezabeyk, and H. R. Faragardi, "Syncop: An evolutionary multi-objective placement of SDN controllers for optimizing cost and network performance in WSNs," *Computer Networks*, vol. 185, p. 107727, 2021.

[116] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "Efficient controller placement and reelection mechanism in distributed control system for software defined wireless sensor networks," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 6, p. e3588, 2019.

[117] F. Chen and R. Li, "Single sink node placement strategy in wireless sensor networks," in *2011 International Conference on Electric Information and Control Engineering*. IEEE, 2011, Conference Proceedings, pp. 1700–1703.

[118] T. Das, V. Sridharan, and M. Gurusamy, "A survey on controller placement in SDN," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 472–503, 2019.

[119] G. Ramya and R. Manoharan, "Enhanced optimal placements of multi-controllers in SDN," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–18, 2020.

[120] L. Zhu, M. M. Karim, K. Sharif, C. Xu, F. Li, X. Du, and M. Guizani, "SDN controllers: A comprehensive analysis and performance evaluation study," *ACM Computing Surveys (CSUR)*, vol. 53, no. 6, pp. 1–40, 2020.

[121] S. R. Kulkarni, G. Lugosi, and S. S. Venkatesh, "Learning pattern classification-a survey," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2178–2206, 1998.

[122] M. Pal, "Random forest classifier for remote sensing classification," *International journal of remote sensing*, vol. 26, no. 1, pp. 217–222, 2005.

[123] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[124] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.
[125] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[126] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 146–152, 2018.
[127] A. I. Moustapha and R. R. Selmic, "Wireless sensor network modeling using modified recurrent neural networks: Application to fault detection," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 5, pp. 981–988, 2008.
[128] Z. E. Khatab, A. Hajihoseini, and S. A. Ghorashi, "A fingerprint method for indoor localization using autoencoder based deep extreme learning machine," *IEEE sensors letters*, vol. 2, no. 1, pp. 1–4, 2017.
[129] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393–430, 2018.
[130] E. Borgia, "The Internet of Things vision: Key features, applications and open issues," *Computer Communications*, vol. 54, pp. 1–31, 2014.
[131] T. Theodorou and L. Mamatas, "SD-MIoT: A software-defined networking solution for mobile Internet of Things," *IEEE Internet of Things Journal*, 2020.
[132] J. Mertens, G. Milotta, P. Nagaradjane, and G. Morabito, "SDN-(UAV) ISE: Applying software defined networking to wireless sensor networks with data mules," in *2020 IEEE 21st International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2020, Conference Proceedings, pp. 323–328.
[133] S. Roy, R. Dutta, N. Ghosh, and P. Ghosh, "Leveraging periodicity to improve quality of service in mobile software defined wireless sensor networks," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2021, Conference Proceedings, pp. 1–2.
[134] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 53–57, 2004.
[135] C. Miranda, G. Kaddoum, E. Bou-Harb, S. Garg, and K. Kaur, "A collaborative security framework for software-defined wireless sensor networks," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2602–2615, 2020.
[136] A. T. Kgogo, "Intrusion detection system in software defined wireless sensor networks," Thesis, 2019.
[137] Y.-W. Chen, J.-P. Sheu, Y.-C. Kuo, and N. Van Cuong, "Design and implementation of IoT DDoS attacks detection system based on machine learning," in *2020 European Conference on Networks and Communications (EuCNC)*. IEEE, 2020, Conference Proceedings, pp. 122–127.
[138] P. Zhao, W. Zhao, Q. Liu, and A. Wang, "Trusted link-separation multipath selection for software-defined wireless sensor networks in adversarial environments," in *International Conference on Security and Privacy in Digital Economy*. Springer, 2020, Conference Proceedings, pp. 19–32.
[139] R. Huang, X. Chu, J. Zhang, and Y. H. Hu, "Energy-efficient monitoring in software defined wireless sensor networks using reinforcement learning: A prototype," *International Journal of Distributed Sensor Networks*, vol. 11, no. 10, p. 360428, 2015.
[140] A. Banerjee and A. Sufian, "Reinforcement learning based transmission range control (RL-TRC) in SD-WSN with moving sensors," *arXiv preprint arXiv:2005.08215*, 2020.
[141] M. U. Younus, M. K. Khan, M. R. Anjum, S. Afridi, Z. A. Arain, and A. A. Jamali, "Optimizing the lifetime of software defined wireless sensor network via reinforcement learning," *IEEE Access*, 2020.
[142] N. Abdolmaleki, M. Ahmadi, H. T. Malazi, and S. Milardo, "Fuzzy topology discovery protocol for SDN-based wireless sensor networks," *Simulation Modelling Practice and Theory*, vol. 79, pp. 54–68, 2017.
[143] A. Rahimifar, Y. S. Kavian, H. Kaabi, and M. Soroosh, "Predicting the energy consumption in software defined wireless sensor networks: a probabilistic Markov model approach," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–14, 2020.
[144] A. K. Al Mhdawi and H. S. Al-Raweshidy, "A smart optimization of fault diagnosis in electrical grid using distributed software-defined IoT system," *IEEE Systems Journal*, vol. 14, no. 2, pp. 2780–2790, 2019.
[145] R. Kumar, U. Venkanna, and V. Tiwari, "A time granular analysis of software defined wireless mesh based IoT (SDWM-IoT) network traffic using supervised learning," *Wireless Personal Communications*, vol. 116, no. 3, pp. 2083–2109, 2021.
[146] X. Zeng, Q. Luo, J. Zheng, and G. Chen, "An efficient neural network optimized by fruit fly optimization algorithm for user equipment association in software-defined wireless sensor network," *International Journal of Network Management*, vol. 30, no. 6, p. e2135, 2020.
[147] S. Misra, S. Bera, M. Achuthananda, S. K. Pal, and M. S. Obaidat, "Situation-aware protocol switching in software-defined wireless sensor network systems," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2353–2360, 2018.
[148] F. Li, X. Xu, H. Yao, J. Wang, C. Jiang, and S. Guo, "Multi-controller resource management for software-defined wireless networks," *IEEE Communications Letters*, vol. 23, no. 3, pp. 506–509, 2019.
[149] I. Kakalou and K. E. Psannis, "Sustainable and efficient data collection in cognitive radio sensor networks," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 29–38, 2018.
[150] C. Orfanidis, "Ph. D. forum abstract: Increasing robustness in WSN using software defined network architecture," in *Information Processing in Sensor Networks (IPSN), 2016 15th ACM/IEEE International Conference on*. IEEE, 2016, Conference Proceedings, pp. 1–2.
[151] B. Mao, F. Tang, Z. M. Fadlullah, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "A novel non-supervised deep-learning-based network traffic control method for software defined wireless networks," *IEEE Wireless Communications*, vol. 25, no. 4, pp. 74–81, 2018.
[152] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1191–1221, 2020.

F. FERNANDO JURADO-LASSO (GS'18-M'21) received the Ph.D. degree in Engineering and the M.Eng. degree in Telecommunications Engineering both from The University of Melbourne, Melbourne, VIC, Australia, in 2020 and 2015, respectively; a B.Eng. degree in Electronics Engineering in 2012 from the Universidad del Valle, Cali, Colombia. He is currently a postdoc at the Embedded Systems Engineering (ESE) section of the Department of Applied Mathematics and Computer Science of the Technical University of Denmark (DTU Compute).

His research interests include networked embedded systems, software-defined wireless sensor networks, machine learning, protocols and applications for the Internet of Things.

LETIZIA MARCHEGIANI (M'15) received a PhD degree in Computer Engineering from Sapienza - University of Rome (Italy) in 2012; she also holds an MSc degree in Computer Engineering and a BSc in Computer Engineering from the same university. From 2014 to 2018, she was a researcher at the University of Oxford (UK), where she was a member of the Oxford Robotics Institute (ORI). Previously, she was associated with the INSPIRE (Investigating Speech Processing In Realistic Environments) ITN as a Marie Curie Postdoctoral Research Fellow (2012-2013). Since 2019 he is an Assistant Professor in Robotics with the Automation and Control section of the Department of Electronic Systems of the Aalborg University (Denmark). Her research interests primarily lie in the areas of signal processing, machine learning, and their application to robotics, autonomous systems, cognitive modelling, intelligent transportation, and digital healthcare.

J. F. JURADO received the Doctorate and MSc degree in Physics both from Universidad del Valle, Cali, Colombia, in 2000 and 1986, respectively; he also holds a BSc degree in Physics from the Universidad de Nariño, Pasto, Colombia in 1984.

He is currently a Professor with the Faculty of Engineering and Administration of the Department of Basic Science of The Universidad Nacional de Colombia Sede Palmira, Colombia. His research interests include nanomaterials, magnetic and ionic materials, nanoelectronics, embedded systems and the Internet of Things. He is a senior member of Minciencias in Colombia.

ADNAN M. ABU-MAHFOUZ (M'12-SM'17) received his MEng and PhD degrees in computer engineering from the University of Pretoria. He is currently the Centre Manager of the Emerging Digital Technologies for 4IR (EDT4IR) research centre at the Council for Scientific and Industrial Research (CSIR), Extraordinary Professor at University of Pretoria, Professor Extraordinaire at Tshwane University of Technology and Visiting Professor at University of Johannesburg. His research interests are wireless sensor and actuator network, low power wide area networks, software defined wireless sensor network, cognitive radio, network security, network management, sensor/actuator node development. He is an associate editor at IEEE Access, IEEE Internet of Things and IEEE Transaction on Industrial Informatics, Senior Member of the IEEE and Member of many IEEE Technical Communities. He participated in the formulation of many large and multidisciplinary RD successful proposals (as Principal Investigator or main author/contributor). Abu-Mahfouz is the founder of the Smart Networks collaboration initiative that aims to develop efficient and secure networks for the future smart systems, such as smart cities, smart grid and smart water grid.

XENOFON FAFOUTIS (S'09-M'14-SM'20) received a PhD degree in Embedded Systems Engineering from the Technical University of Denmark in 2014; an MSc degree in Computer Science from the University of Crete (Greece) in 2010; and a BSc in Informatics and Telecommunications from the University of Athens (Greece) in 2007. From 2014 to 2018, he held various researcher positions at the University of Bristol (UK), and he was a core member of SPHERE: UK's flagship Interdisciplinary Research Collaboration on Healthcare Technology. He is currently an Associate Professor with the Embedded Systems Engineering (ESE) section of the Department of Applied Mathematics and Computer Science of the Technical University of Denmark (DTU Compute). His research interests primarily lie in Wireless Embedded Systems as an enabling technology for Digital Health, Smart Cities, and the (Industrial) Internet of Things (IoT).