A Survey on Machine Learning Software-Defined Wireless Sensor Networks (ML-SDWSNs): Current Status and Major Challenges

F. FERNANDO JURADO-LASSO\textsuperscript{1,2}, (Member, IEEE), LETIZIA MARCHEGIANI\textsuperscript{3}, (Member, IEEE), J. F. JURADO\textsuperscript{4}, ADNAN M. ABU-MAHFOUZ\textsuperscript{5,6}, (Senior Member, IEEE), AND XENOFON FAFOUTIS\textsuperscript{1}, (Senior Member, IEEE)

\textsuperscript{1}Embedded Systems Engineering Section, DTU Compute, Technical University of Denmark, 2800 Lyngby, Denmark
\textsuperscript{2}Department of Electrical and Electronic Engineering, The University of Melbourne, Melbourne, VIC 3010, Australia
\textsuperscript{3}Department of Electronic Systems, Aalborg University, 9220 Aalborg Øst, Denmark
\textsuperscript{4}Department of Basic Science, Faculty of Engineering and Administration, Universidad Nacional de Colombia Sede Palmira, Palmira 763531, Colombia
\textsuperscript{5}Council for Scientific and Industrial Research (CSIR), Pretoria 0184, South Africa
\textsuperscript{6}Department of Electrical and Electronic Engineering Science, University of Johannesburg, Johannesburg 0001, South Africa

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

This work was supported in part by the Electronics Components and Systems for European Leadership (ECSEL) Joint Undertaking (JU) through Distributed Artificial Intelligent Systems (DAIS). DAIS (https://dais-project.eu/) under Grant 101007273; in part by the European Union’s Horizon 2020 Research and Innovation Program and Sweden, Spain, Portugal, Belgium, Germany, Slovenia, Czech Republic, The Netherlands, Denmark, Norway, and Turkey, for JU; and in part by the Innovation Fund Denmark under Grant 0228-00004A.

\textbf{ABSTRACT} Wireless Sensor Network (WSN), 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 that is processed centralised. With the current traditional techniques of state-of-art WSN 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 synergy between Software-Defined Networking (SDN) and WSN has been proposed. This paper aims to present the current status of Software-Defined Wireless Sensor Network (SDWSN) proposals and introduce the readers to the emerging research topic that combines Machine Learning (ML) and SDWSN concepts, also called ML-SDWSNs. ML-SDWSN grants an intelligent, centralised and resource-aware architecture to achieve improved network performance and solve the challenges currently found in the practical implementation of SDWSNs. This survey provides helpful information and insights to the scientific and industrial communities, and professional organisations interested in SDWSN, mainly the current state-of-art, ML techniques, and open issues.

\textbf{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).

\section*{I. INTRODUCTION}

The IoT (as a general IoT ecosystem including middlewares, servers, cloud, edge) 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.

The associate editor coordinating the review of this manuscript and approving it for publication was Yafei Hou\textsuperscript{0}.

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 USD 7.1 trillion by 2025 [1] and that the number of connected “things” can exceed the 75 billion devices barrier [2]. The exponential growth of connected devices fosters the creation of a large variety of IoT vendors and protocols. Despite the variety of vendors and protocols,
the IoT ecosystem must, somehow, deliver seamless services to users. Emerging IoT applications such as smart agriculture, transportation systems, health systems, etc., expand the scope of the internet to include sensing technologies such as WSNs.

WSNs, enablers of IoT technology, are built upon the interconnection of a large number of Networked Embedded Systems (NESs). An NES often called a wireless sensor node, is a tiny energy-constrained device comprised of a processing unit, a memory unit, a communication transceiver, and some sort of power supply. 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 [22], [23]. WSNs are used in a 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, the limited resources in WSNs need to be managed effectively; so they can run for the longest time possible.

The SDN paradigm has been proposed to alleviate the management complexity currently found in wired networks. A simple representation of an SDN architecture is shown in Fig. 1. SDN breaks the vertical integration of the network by separating it into 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 the application plane with a global view of the network. The data plane is the network infrastructure that consists of networking devices that become forwarding devices with no intelligence. The introduction of SDN abstractions into the WSN forms what we call 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 [10]. In large WSNs, with thousands of sensor nodes, it is critical to consider and implement management solutions [25]. The SDWSNs centralise the network intelligence in an SDWSN controller, leaving sensor nodes acting as simple forwarding devices (see Fig. 2). 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. Although SDWSN has been demonstrated to improve network performance against other traditional WSNs, there is a need for novel architectures that make the most of the global view of the network assets and balance the expenditure of network resources when making the WSN programmable.

ML-SDWSN has been devised as a potential network architecture solution to exploit the centralised WSN assets information to enhance the overall network performance. The ML component has, at hand, real-time data including network statistics (Received Signal Strength Indicator (RSSI), Packet Delivery Ratio (PDR), etc.), network resources (sensor nodes remaining energy, applications load, etc.), network topology, etc. This makes the ideal environment to deploy ML algorithms tailored to user and application requirements. ML-SDWSN is also seen as a prominent solution to alleviate the communication overhead introduced; thus, making the most of SDWSNs. ML-SDWSN is discussed in detail in Section VI.

A. CONTRIBUTION

Despite the diverse benefits brought by SDN to WSNs, without proper countermeasures to minimise the management overhead introduced, it can negatively impact the network performance of the WSN and lead to high energy costs.
TABLE 1. Summary of related works on SDN-based networks.

| Article | Year | Survey? | SDN | SDWSNs | ML | ML-SDWSNs |
|---------|------|---------|-----|--------|----|-----------|
| [3]     | 2014 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [4]     | 2014 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [5]     | 2014 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [6]     | 2015 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [7]     | 2015 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [8]     | 2016 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [9]     | 2017 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [10]    | 2017 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [11]    | 2017 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [12]    | 2017 | No      | ☑  | ☑      | ☑  | ☑         |
| [13]    | 2018 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [14]    | 2018 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [15]    | 2018 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [16]    | 2018 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [17]    | 2019 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [18]    | 2019 | No      | ☑  | ☑      | ☑  | ☑         |
| [19]    | 2019 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [20]    | 2019 | Yes     | ☑  | ☑      | ☑  | ☑         |
| [21]    | 2020 | Yes     | ☑  | ☑      | ☑  | ☑         |

This paper conducts an extensive literature review by exploring relevant research articles on SDWSNs and 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 ML techniques in SDWSNs. In particular, the article in [12] published in 2017, 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 [18], 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. A survey paper published in 2020 on ML-WSNs is presented in [21], it mainly focuses on the use of Deep Learning (DL) in WSNs, and they also discuss the energy expenditure in the ML training phase. The survey papers in [5], [19] discuss the design challenges of WSNs due to their inherent dynamic behaviour, and the power of ML techniques to improve the ability of WSNs to adapt to such changing behaviour of their surrounding environment. Due to the distributed nature of traditional WSNs, ML techniques are laborious to apply to operate and control traditional WSNs. However, the design concepts of SDN (e.g., centralised architecture) form the perfect habit to easier apply ML techniques. The survey paper in [16], published in 2018, principally focuses on how ML techniques are applied to SDN architectures mainly to traffic classification, routing, Quality of Service (QoS) prediction, security and resource management. The paper also briefly discusses SDWSNs and directions on the use of ML in WSNs. The survey in [20], published in 2019, presents network applications...
that combined SDN and ML concepts. The survey provides thorough discussions on ML methods and SDN-concept networks, their applications and gives future directions on future ML in future SDN. In contrast, the contributions of this survey article are as follows.

1) We firstly provide a comprehensive background on WSNs including the evolution of Microcontroller Unit (MCU)-sensor nodes, networking and standards, and challenges of WSNs.

2) We provide a systematic review of SDWSN proposals that have not been previously covered by other survey papers. We categorised them into general frameworks, proposals that seek to improve Key Performance Indicators (KPIs) (QoS-related works), research works that reprogram both hardware and software of sensor nodes (fully programmable mechanisms), 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 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 ML techniques used in SDWSNs.

### B. STRUCTURE OF THIS PAPER

Fig. 3 provides a visual representation of the organisation of this paper. Section II provides detailed background on WSNs including the networking standards, embedded operating systems and challenges. Section III provides background on SDN, SDWSNs and presents the early adopters of SDWSNs. Section IV presents the current status of research works that have expanded the state-of-art of SDWSNs. Section V presents an overview of the most commonly used ML algorithms in supervised, unsupervised, semi-supervised, reinforcement and deep learning. Section VI presents a survey of research efforts that have applied ML techniques in SDWSN. Section VII summarises both SDWSN and ML-SDWSN research works. Section VIII discusses major challenges and future directions for both SDWSNs and ML-SDWSNs. 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

### 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 |
| GPRS    | General Packet Radio Service |
| IA      | Intelligent Agent |
| IDS     | Intrusion Detection System |
| IETF    | Internet Engineering Task Force |
| IoT     | Industrial Internet of Things |
| 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 |
| LAN     | Local Area Network |
| LoRa    | Long Range |
| LoRaWAN | Long Range Area Network |
| MAC     | Media Access Control |
| 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 |
| SPM     | Support Vector Machine |
| TCP     | Transmission Control Protocol |
| TSCH    | Time Synchronized Channel Hopping |
| UDP     | User Datagram Protocol |
| WPAN    | Wireless Personal Area Network |
| WSN     | Wireless Sensor and Actuator Network |

This table lists the acronyms used throughout the paper.
# Structure of this paper.

## FIGURE 3. Structure of this paper.

## TABLE 3. Historical wireless sensor nodes development platforms. The dash (-) symbol indicates that no information was found on that specific platform.

| Year | Name               | Type      | RAM [KB] | Flash [KB] | \( E^{2}\)PROM [KB] | OS           | Radio              |
|------|--------------------|-----------|----------|------------|------------------------|--------------|--------------------|
| 2000 | Rene 2 [26]        | Atmega 163| 1        | 16         | 32                     | TinyOS       | TR1000             |
| 2000 | \(\mu\)AMPS [27]   | Strong ARM|          | 1M         | 4M                     | \(\mu\)OS     | National LMX3162   |
| 2001 |BTNode [28]         | Atmega 128L| 4        | up to 128  | 4                      | Nut/OS       | CC2500             |
| 2003 | Mica2 [29]         | Atmega 128| 4        | 128        | 512                    | TinyOS       | CC2500             |
| 2004 | TelosB [30]        | MSP 430   | 10       | 48         | 1M                     | TinyOS, Contiki\(^b\), RIOT\(^b\) | CC2420 |
| 2006 | MicaZ [30]         | Atmega 128| 4        | 128        | 512                    | TinyOS, Contiki\(^d\) | CC2420 |
| 2011 | WiSMote [31]       | MSP 430   | 16 sram  | up to 256  | up to 8M               | Contiki\(^d\) | CC2520             |
| 2013 | WiSense [32]       | MSP 430   | 4        | 56         | 128                    | -            | CC2520             |
| 2013 | CC2541DK [33]      | CC2541    | 8        | up to 256  | -                      | Contiki\(^d\), FreeRTOS, RIOT | 2.4 GHz |
| 2015 | CC2538DK [34]      | CC2538    | 32       | 512        | 4                      | Contiki\(^d\) | 2.4 GHz             |
| 2015 | OpenMote [35]      | CC2538    | 32       | 512        | 4                      | Contiki\(^d\) | 2.4 GHz             |
| 2015 | CC2650STK [33]     | CC2650    | 20       | 128        | -                      | Contiki\(^d\) | 2.4 GHz             |
| 2015 | Re mote [36]       | CC2538    | 32       | 512        | 4                      | Contiki\(^d\), RIOT\(^d\), MantisOS | 2.4 GHz, Sub-1 GHz |
| 2017 | CC1350STK [37]     | CC1350    | 20       | 128        | -                      | Contiki-NG\(^d\) | 2.4 GHz, Sub-1 GHz |
| 2019 | LPSSTK-CC1352R [38]| 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
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.

- 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 [39].

As mentioned above, the use of WSNs covers a range of applications that enable 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 widely used communication protocols in WSNs are presented. Other forms of wireless communications methods are surveyed in [40].

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 [41]. Among communication protocols, used in this frequency band, are ZigBee [42], Bluetooth [43], [44], and IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) [45].

1) ZIGBEE

Zigbee was originally designed by the ZigBee Alliance under the specifications of the IEEE 802.15.4 standard [42]. Among its features are low power consumption and support for different network topologies such as mesh, star and tree, which makes ZigBee a good candidate for Industrial Internet of Things (IIoT). However, ZigBee does not meet with all requirements of industrial applications as it cannot serve a large number of sensor nodes and suffers from interference [43], [44].

2) BLUETOOTH

Bluetooth was originally designed to achieve medium data rates for short distances (typically up to 10 m). Due to the power consumption concerns, the Bluetooth-Low-Energy (BLE) specification was proposed. BLE was conceived for embedded systems with low-power requirements and limited processing power. This extension provides up to 1 Mbps over 5-10 m range [43], [44].

3) 6LoWPAN

6LoWPAN was established by the Internet Engineering Task Force (IETF) [45]. 6LoWPAN was conceived under the premise that the Internet Protocol should be applied even to the smallest devices, and that resource-constrained embedded systems should be able to participate in the IoT. Therefore, 6LoWPAN is a lightweight protocol that uses an adaptation layer, that has a set of functions, to enable transmissions of Internet Protocol version 6 (IPv6) packets over IEEE 802.15.4 radios. The great advantage of 6LoWPAN is that enables direct communication with other Internet Protocol (IP) devices locally or via IP network.

There also exist other communications methods that are only used for a set of sensor nodes in WSNs. (i) WiFi is a wireless networking technology based on IEEE 802.11 family of standards [46]. It is commonly used for Local Area Networks (LANs) and to provide wireless high-speed Internet access. It is common to find WiFi modules in gateways or border routers to enable internet connectivity to WSNs. Sensor nodes rarely use WiFi modules as it imposes high power requirements and shortens the network lifetime. (ii) General Packet Radio Service (GPRS) was introduced as a wireless communication packet service that promises data rates from 54 to 114 kbps [47]. GPRS offers a best-effort service that is often used in gateways to communicate with an online monitoring centre. Similar to WiFi, GPRS was not designed for WSNs applications as it also imposes higher power requirements than IEEE 802.15.4. (iii) Long Range Wide Area Network (LoRaWAN) is a technology that enables long-range transmissions (more than 10 km) with low power consumption. LoRaWAN is a cloud-based Media Access Control (MAC) protocol that uses Long Range (LoRa) in its physical layer. Features of LoRaWANs include low bandwidth (250 bps up to 11 kbps), long-range, low cost and low power consumption [48]. Thus, LoRaWAN deployments make sense in applications that use small payloads and transmit data few times a day over long distances, than having hundreds of IEEE 802.15.4 radios interconnected to cover the same area size, resulting in increased energy consumption, and management complexity of sensor nodes.

Overall, 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. EMBEDDED OPERATING SYSTEM (EOS)**

Due to the limited resources available, sensor nodes require a lightweight Operating System (OS) [9], [49]. The two Embedded Operating Systems (EOSs) that have achieved the most attention by the research SDWSN community so far are: (i) Contiki, which is an open-source OS for low-power IoT networks, is designed for resource-constrained sensor nodes [50]. In its core uses C language and has three network stacks; RIME, Internet Protocol version 4 (IPv4) and IPv6. Contiki-NG [51] 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 [52] and supports IPv6 in its protocol stack, namely, Berkeley Low-power IP (BLIP).

There exist some EOSs that have not been yet used in SDWSNs: FreeRTOS [53] 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 [54] is a 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 functions 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 [55]. IEEE802.15.4e, along with low-power 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 IoT networks [56]. The RIOT design principle is to be energy-efficient and reliable that supports real-time and small memory applications. It also provides 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 4. 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.

**C. 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 being wireless, most of the applications require sensor nodes to operate in harsh environments or areas with limited access [57], [58]. 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 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 IPs which require a scheduler and run on top of the firmware. 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 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 [59]. Furthermore, wireless communication can take up to 75% of the total energy in some applications [60]. 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 ecosystem 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. The WSN technology provides the IoT with new sensing capabilities, integrating the physical world into the digital world. State-of-art WSNs are deployed with inflexible firmware. Where, after deployment, 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 [4], of a WSN that comprises 100 sensor nodes that measure 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 [61].

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 them. 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 is a network management approach that enables to dynamically and programatically reconfigure the network, 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 elements become forwarding devices with little or no intelligence. The intelligence is instead logically centralised in a controller, facilitating policy enforcement and network reconfiguration [7]. 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. 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 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 [11]. The most well-known protocol used in the southbound API is OpenFlow [62]. 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 [10]. In large WSNs, with thousands of sensor nodes, it is critical to consider and implement management solutions [25].

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 SDWSNs architectures is the shared communication medium and constrained resources. SDN was initially conceived for wired networks, where control packets typically flow through a dedicated 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 PDR which is a 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 [63].

Readers interested in a detailed background on the SDWSN paradigm, including a comprehensive analysis of challenges, architectures, benefits and design requirements can refer to [9], [10], [15].

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 [64] and SDWN [65].

1) SENSOR OpenFlow (SOF)

Luo et al. [64] 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 [62].

F. F. Jurado-Lasso et al.: Survey on ML-SDWSNs: Current Status and Major Challenges

IEEE Access
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 route packets based on the data attributes, and (ii) the use of the IP in WSNs, and they suggest two IP stacks: micro Internet Protocol (µIP) or micro Internet Protocol version 6 (µIPv6) [50], and BLIP [52]. 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. [65] 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.

| Framework | Networking | NOS | EAF |
|-----------|------------|-----|-----|
| SOF       | IEEE 802.15.4, IP | ✗ | in-net-proc$^*$ |
| SDWN      | IEEE 802.15.4 | ✓ | in-net-proc, duty-cycle |

$^*$ in-net-proc: in-network processing; data aggregation, etc.

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 [64] and SDWN [65], fit in this category.

A brief comparison of general frameworks is shown in Table 6. The table compares general frameworks stating their advantages and disadvantages, EOS used, type of controller architecture, their availability to the research and professional community, and surveys where they have been previously discussed. We can see that they are also the first research works towards SDN-based WSNs as they seek to provide a practical, fully functional SDWN 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. QoS-RELATED WORKS
1) ENERGY CONSUMPTION
This is 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 employing SDN. Works that fit in this category, but, has been previously discussed in other SDWSN survey papers are [58] (discussed in [14], [15]), [78] (discussed in [9], [15]), [83] (discussed in [8], [9], [11]), [84] (discussed in [15]), and [85] (discussed in [9], [14]).
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 may compromise the entire network by targeting it. Also, securing a large WSN is a high energy-intensive task that can lead to sensor nodes depleting 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 [95].

Security is a critical aspect to consider when designing low-power 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 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 the 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 [96]. 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
### 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 |
|------|------------------------------------------------------------------------------|--------------------------------------------------------------------------|--------------------------------------------------------------------------------|--------------------|---------------------------------|
| [71] | To design a QoS-based routing protocol that comprises a clustering and routing algorithm and local network maintenance. | A double cluster head mechanism for delay-sensitive and reliable data transmission applications. Multiple forwarding paths, for sensor nodes, for different data classifications. | Creating multiple forwarding tables on sensor nodes can lead to increased memory size and control overhead, therefore increasing the power consumption in sensor nodes | Cooja | SDN-WISE & clustering protocols |
| [63] | 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 |
| [73] | 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 |
| [74] | 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 |
| [75] | 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 |
| [76] | 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 |
| [77] | 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, [78], [58] |
| [79] | 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 |
| [80] | 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 |
| [81] | 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 |
| [82] | 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 |

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

| Ref. | Type   | Description                                                                                                                                                                                                 |
|------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [86] | 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. |
| [15] | 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. |
| [87] | 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. |
| [88] | 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]. |
| [89] | 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. |
| [90] | Research paper | This paper presents a Denial of Service (DoS) attack detection mechanism for SDWSNs. The proposed approach is lightweight, and it can even run on resource-constrained devices. The algorithm was evaluated using both clustered and non-clustered network architectures. The cluster-based approach used less memory while running the algorithm on every node reduced the packet traffic. |
studied in detail. Readers interested in an extended discussion on SDN and WSN security from the SDWSN perspective can refer to [10], whereas SDN security is discussed in [97].

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 addressed indirectly in other research works. SDWSN architectures grant centralised network monitoring to anticipate potential 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.

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 impacting KPIs such as residual energy of sensor nodes and the PDR. Many research works [66], [102], [103] 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

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

Porilla et al. [109] 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 and Athisha [110] 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. [111] 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 [112]. However, the greatest advances in FPGAs with ultra-low power consumption characteristics have extended their use to WSNs [113]–[115].

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.

### D. NETWORK TOPOLOGY AND MANAGEMENT PROPOSALS

Network management is complex and challenging in networks. Some functionalities include network provisioning, configuration, and maintenance [116]. The implementation of management tasks can lead to a steep increase in the use of sensor resources.

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 12, 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.

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

| Ref. | Specific aim | Approach | Disadvantage |
|------|-------------|----------|--------------|
| [98] | 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 |
| [99] | 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 |
| [100] | 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 |
| [101] | To enhance network reliability and minimise the latency in low-power IoT networks | To use synchronous flooding (SF) architecture to dynamically reconfigure SF protocols to account for SDN control requirements. It divides the network operation into timeslots where control packets are sent through a dedicated control timeslot | Scalability issues due to concurrent transmissions for large dense networks |

### TABLE 11. Relevant SDWSN research works that address control overhead.

| Ref. | Aim | Approach | Disadvantage |
|------|-----|----------|--------------|
| [63] | 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 |
| [104] | 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 |
| [105] | 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 |
| [106] | 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 |
| [107] | 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 |
| [108] | To remove the control overhead from the data communication channel | The control traffic is separated from the data communication channel. They use a long-range SubGHz interface to send control packets in a one-hop fashion. The data communication channel uses a short-range interface with a data rate of 250kbps | Increased energy consumption due to the dual-stack network radio interfaces |
stack. Also, some works lack control overhead analysis and the implication in network performance when making the WSN manageable using SDN concepts.

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 [130]. Therefore, sensor nodes that lie in the proximity of the controller drain their energy first, resulting in a shorter NL. Table 13 presents research works that aim to solve the controller placement to improve network performance in SDWSNs. 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 [131], [132], a study on performance evaluation in [133].

V. MACHINE LEARNING OVERVIEW

ML is part of AI that studies computer algorithms to mimic human learning and gradually improve 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 V). 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,
TABLE 13. Relevant SDWSNs research works that address the controller placement problem. The checkmark (✓) and cross (✗) symbols depict whether the specific aim is addressed in the referred article or not.

| FW       | energy | latency | reliability | fault tolerance | syn. overhead | Approach                                                                 |
|----------|--------|---------|-------------|-----------------|---------------|--------------------------------------------------------------------------|
| [125]    | ✓      | ✓       | ✓           | ✓               | ✓             | Finding the exact number of controllers required for a specific network topology using GA and GRASP algorithms. |
| [126]    | ✓      | ✓       | ✓           | ✓               | ✓             | Placement optimisation problem using the Cuckoo optimisation algorithm.   |
| [127]    | ✓      | ✓       | ✓           | ✓               | ✓             | Optimal controller placement using the Cuckoo optimisation algorithm.     |
| [128]    | ✓      | ✓       | ✓           | ✓               | ✓             | Two approaches for optimal placement were discussed: k-means for local controllers and k-centre for global controller. |
| [129]    | ✓      | ✓       | ✓           | ✓               | ✓             | A multi-objective optimisation solution to select the SDN controllers that route, uplink, control and data packets considering multi-objective parameters. |

whereas unsupervised learning makes inferences from unlabelled data.

(v) Testing: it evaluates the accuracy of the learned function using the 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 DL automates much of these tasks.

This section briefly 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 [134]. 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 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 ∈ X and y ∈ Y, and these can be represented in pairs (x₁, y₁) … (xₙ, yₙ). The goal of supervised learning is to learn a mapping function that matches a given input (xₙ₊₁) to a label yₙ₊₁. 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. Between the most popular supervised learning algorithms, we can find K-Nearest Neighbour (k-NN), Naive Bayes, Decision Tree (DT), Neural Networks (NNs), and Support Vector Machines (SVMs), which are discussed in [19], [21].

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. The most widely used unsupervised learning algorithms are K-means clustering and Principal Component Analysis (PCA). A thorough discussion on unsupervised learning techniques and applications can be found in [19], [135].

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 \((x_1, \ldots , x_n \in X)\) along with a large set of unlabelled data \((x_{n+1}, \ldots , x_{n+m} \in X)\). 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. [136]. 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 processes more data and 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 mean that the algorithm will perform better. More detailed information on semi-supervised learning can be found in [136].

### 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 modelled 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 [137]. More detailed information on RL can be found in [138].

### 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-linear problems. As in NNs, each layer in DL contains units (neurons). 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) [139], Recurrent Neural Networks (RNNs) [140], and Autoencoder (AE) [141]. Readers interested in thorough discussions on DL algorithms, techniques and applications shall refer to [142].

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

A typical ML-SDWSN architecture comprises 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 [16].
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 [143]. 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 and Mamatas [144] proposed SD-MIoT, which is an SDN-based solution for mobile low-power 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 $G$.

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

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

$$T_t = ||A_t - A_{t-1}||$$

$$T_{t-(k-2)} = ||A_{t-(k-2)} - A_{t-(k-1)}||$$ \hfill (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 [145] 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 destinations 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 updated 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. [146] proposed a 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. RNN) to identify nodes with low periodicity to mitigate their impacts on QoS. 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. A comparative study of three AI approaches for IDSs using SDWSNs is presented in [152]. The SDWSN controller comprises three main functions: (i) The flow collector which collects the network information, (ii) the anomaly detector which detects any abnormal behaviour in the networks, and (iii) the anomaly mitigator which serves to counteract the detected anomaly. The three AI-based approaches used are DT, Naive Bayes, and DL. The three AI-based approaches used are DT, Naive Bayes, and DL. Results show that the Naive Bayes approach is best suited for SDWSN applications where the controller has restricted memory capabilities, e.g., the controller is embedded in one of the sensor nodes, and it also shows fewer energy consumption requirements. For SDWSN applications where the controller memory size is not a concern, e.g., external or cloud-based controllers, the DL or DT anomaly detector can be used. However, the DT approach presents the best overall performance for detecting anomalies, especially, for delay-sensitive applications. Chen et al. [149] presented an 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

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 [147]. 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. [151] 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 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 recurrent authentications while implementing data integrity inspections. Kgogo et al. [148] 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. A comparative study of three AI approaches for IDSs using SDWSNs is presented in [152]. The SDWSN controller comprises three main functions: (i) The flow collector which collects the network information, (ii) the anomaly detector which detects any abnormal behaviour in the networks, and (iii) the anomaly mitigator which serves to counteract the detected anomaly. The three AI-based approaches used are DT, Naive Bayes, and DL. Results show that the Naive Bayes approach is best suited for SDWSN applications where the controller has restricted memory capabilities, e.g., the controller is embedded in one of the sensor nodes, and it also shows fewer energy consumption requirements. For SDWSN applications where the controller memory size is not a concern, e.g., external or cloud-based controllers, the DL or DT anomaly detector can be used. However, the DT approach presents the best overall performance for detecting anomalies, especially, for delay-sensitive applications. Chen et al. [149] presented an 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

### TABLE 14. Research works that applied ML in the SDWSN architecture to tackle mobility issues present in traditional WSNs.

| Ref. | Year | Aim | ML technique | Improvement | Main drawback |
|------|------|-----|--------------|-------------|---------------|
| [144] | 2020 | 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 |
| [145] | 2020 | 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 |
| [146] | 2021 | Periodic mobility | RL for periodic mobility & nearest centroid | Lower network latency | Considerable learning time for movements of high periodicity |
DDoS attacks including ICMP flood, SYN flood, and UDP flood, with different periods and duration times. Results show that DT achieved over 97% accuracy. Zhao et al. [150] 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 a greedy algorithm.

Table 15 presents a qualitative comparison of research works that aim to tackle security vulnerabilities in WSNs using ML-SDWSNs. These works have demonstrated that ML is a good candidate to overcome the security vulnerabilities currently present in traditional WSNs and SDWSNs, without putting at risk valuable and scarce network resources.

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. [153] 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 and Sufian [154] 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 the velocities of successors and link quality metrics such as RSSI, packet reception rate, and attenuation. Younus et al. [155] combined RL and SDN concepts to devise a new routing algorithm for SDN-based WSNs that enhance the overall network performance. For the RL algorithms, they used the Q-learning [156] approach to choosing the best routing path from the routing list obtained by the Spanning Tree Protocol (STP). Simulation results show a prolonged NL and an improved PDR.

To prolong the NL of the SDWSN, an RL approach that trains the SDN controller to optimise the routing paths is proposed in [159]. 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. Training the SDWSN controller to find alternative energy-efficient routing paths has been studied in [160]. They used a Deep Reinforcement Learning (DRL) approach that configures routing paths avoiding the use of sensor nodes with low energy levels. The reward expected
for forwarding packets to the next hop is estimated using a deep neural network, mainly CNN. Results demonstrated that the proposed approach achieved a prolonged NL compared to existing state-of-art methods. This approach increases the number of hops a packet needs to travel to reach the destination by finding alternative paths, rather than the traditional SP, to avoid exhausting the energy of sensor nodes with low remaining energy.

Abdolmaleki et al. [157] 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 predicts the energy consumption of each sensor node. Rahimifar et al. [158] proposed a Markov-based model to predict the future energy consumption of sensor nodes. The controller predicts the individual energy consumption of sensor nodes; thus, sensor nodes avoid reporting energy levels to the controller. Nunez Segura and Borges Margi [161] proposed a Markov chain prediction mechanism for energy consumption in SDWSNs. They compared the prediction model by running it on every sensor node of the WSN and solely in the controller. Experiments show that running the prediction algorithm on the controller (moving the prediction out of sensor nodes) increases the prediction accuracy and PDR while reducing the delay, energy consumption, control overhead and sensor nodes’ processing overhead.

Table 16 presents a qualitative comparison of research efforts that have used ML-SDWSN concepts to further improve the energy efficiency in traditional WSNs. These works took advantage of the global view of the network granted by the controller and the power of ML to discover energy-efficient paths, optimal transmission range and energy consumption predictions to extend the NL of WSNs.

### D. RELIABILITY

To minimise power outages, which are due to a persistent fault and over utilisation of distribution transformers (DTs), of electrical distribution systems, a remote IoT monitoring and fault prediction system is proposed in [162]. 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 [168] to perform a 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 [170].

---

**Table 16. Qualitative comparison for ML-SDWSN research works that strive to improve the energy efficiency in traditional WSNs.**

| Ref. | Year | Aim | ML technique | Improvement | Main drawback |
|------|------|-----|--------------|-------------|---------------|
| [153] | 2015 | Energy | RL for value-redundancy filtering and balancing the routing path | Improved energy | Scalability issues |
| [157] | 2017 | Energy | Fuzzy logic to choose the best relay node | Extended NL and reduced PLR | Relatively high control overhead due to sensor reporting |
| [158] | 2020 | 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 |
| [154] | 2020 | Energy (Tx range) | RL for transmission range control | Improved energy consumption, delay and throughput | Network reliability due to the adaptive transmission range |
| [159] | 2020 | Energy (NL) | RL for extended NL | Extended NL | A relative high packet overhead due to the use of IP-enabled network |
| [160] | 2021 | Energy (NL) | A DRL approach that finds energy-efficient paths by using a reward function based on CNN | Extended NL | An increase in the number of hops a packet needs to travel |
| [161] | 2021 | Energy | Markov chain prediction mechanism for energy consumption in SDWSNs | Improved prediction accuracy, PDR, delay, energy consumption, control overhead, sensor nodes’ processing overhead | The performance can decay for large WSN deployments |
| [155] | 2021 | Energy | RL based on Q-learning for routing path | Improved NL and PDR | It was not evaluated in large scale deployments and no real-time data used |
TABLE 17. Qualitative comparison of relevant ML-SDWSN works that strive to improve the network reliability of traditional WSNs.

| Ref  | Year | Aim               | ML technique                        | Improvement                                                                 | Main drawback                                                                 |
|------|------|-------------------|-------------------------------------|------------------------------------------------------------------------------|------------------------------------------------------------------------------|
| [163]| 2016 | 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. |
| [164]| 2018 | Spectrum          | RL for spectrum utilisation         | Eliminates channel handoffs, which are energy-intensive tasks, by predicting users traffic | Static CHs can exhaust their resources faster.                                |
| [165]| 2018 | 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. |
| [166]| 2019 | Multi-controller placement | RL (DRL), and PSO                  | Reduced waiting tasks and energy consumption for controllers                  | It lacks experimental validation.                                            |
| [162]| 2019 | 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. |
| [167]| 2020 | Throughput        | NN to minimise unsatisfied user equipment and maximise the throughput | Resources balanced to improve the QoS.                                      | It lacks experimental validation.                                            |
| [168]| 2021 | 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.                             |
| [169]| 2021 | Reliability       | regression, DT, SVM and NN          | Link quality                                                                 | The deployment of ML models within sensor nodes can negatively impact the network lifetime. |

With the advent of Internet technologies, new applications have emerged. Each application imposes different bandwidth requirements. It is of great importance to have network resources balanced to comply with strict QoS requirements. The research work presented in [167] aims to minimise the number of unsatisfied user equipment while maximising the throughput of the network through load balancing. They used an NN, which was improved using the fruit fly optimisation (FOA) algorithm, to solve this problem. To comply with strict network reliability requirements, a link quality prediction model for SDWSNs is presented in [169]. The model focuses on predicting the link quality between neighbouring nodes, therefore, improving the overall stability of the routing paths. They use multiple ML models such as; regression, DT, SVM and NN with physical and logical parameters as inputs. The physical parameter includes the RSSI metric, whereas the logical parameter includes the reception of the historical discovery packets. The trained model is then run at the sensor nodes level. Simulation results show that the SDWSN and ML, at the link-layer level, improve the network reliability by avoiding the use of unstable wireless communication links.

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. [165] 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 enhancing 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 [166] an energy-aware multi-controller placement solution using a Particle Swarm Optimisation (PSO) for minimising energy consumption is presented. Moreover, a 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 [164] 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 [163] 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 be used which runs in the SDN controller. A testbed with multiple sources of interference, such as Bluetooth [41] and WiFi [46] networks, was proposed. The feature vector for the statistical model proposed includes PDR, energy consumption, interference, RSSI, end-to-end delay, and noise.

The use of ML in SDWSNs has already been explored in the agriculture industry. To enhance the grain quality sold to customers, an ML-based approach is proposed, in [171]. The main objective is to classify the quality of the stored grain. In the deployment, key environmental factors including
temperature, moisture and CO₂ concentration levels are considered and used as input for the ML models. The SDWSN controller runs the ML models including the KNN, random forest, and linear regression. Experimental results show that the random forest algorithm performs better than the other classifiers in separating high-quality grains from the infested ones.

Table 17 presents a qualitative comparison of ML-SDWSN research works that aim to improve the WSN reliability using ML-SDWSN concepts. These research papers have demonstrated that by having real-time network information (e.g., statistics) and using the power of ML, the controller can promptly react to any network change (e.g., interference, traffic, etc.) by setting up a new network configuration. This allows the ML-SDWSN architecture to proactively provision optimal resources to deal with potential threats that hamper the network performance.

E. A CASE STUDY: IN-VEHICLE WSNs AND 6G

Due to the increasing number of sensors deployed in modern cars, a growing interest has emerged in reducing the number of wires connecting sensors to cars’ microcontrollers [172]. One way to minimise the wiring in modern cars is to use WSN technology. Wireless sensor nodes, in small environments such as cars, are usually in one-hop distance from the sink. A star topology may be used to connect all sensor nodes. However, the high density of sensor nodes can lead to high network interference and latency in a contention-based MAC protocol [173]. The TSCH protocol provides both time and frequency diversity for transmissions boosting the network reliability [55]. TSCH reduces the communication and power overhead. TSCH relies on the scheduler that sets the communication links for each cell (a specific time and channel) in the slotframe. The transmission schedules highly impact the performance of the WSN. They are usually designed and scheduled to meet a specific requirement (e.g., reliability, latency, energy, etc.). A star topology (case 1 in Fig. 8) in TSCH leads to a large slotframe increasing the network latency as the network density grows, whereas a tree topology (case 2 in Fig. 8) enables parallel transmissions reducing the network latency.

1) THE ROLE OF ML-SDWSN AND 6G

In TSCH networks, the communication schedules are assigned autonomously (e.g., orchestra [174]) or centralised. SDWSN technology enables new ways to assign
communications schedules. Network data is collected centralised such as packet loss, link qualities, energy, etc. The control plane has a global view of the network which makes the perfect environment as it has all the network information and resources to decide on the best communication schedules that satisfy the user or application requirements. Having the network data at hand permits engineers and scientists to deploy bespoke scheduling algorithms. These communication schedules can also be assigned with the aid of ML algorithms. ML offers the potential to anticipate communication links that will suffer from interference when the car passes a specific road or source of interference (a source of interference coming from a motorbike in Fig. 8). ML can set the schedules that reduce the latency of a sensor that is sending more frequent critical data. ML can also dynamically update the schedules based on the remaining energy of sensor nodes.

ML-SDWSN technology can be easily applied to intra-car WSNs either utilising the car technology or the 6G infrastructure. Modern cars have powerful processing units in which the control plane can run complex computational operations with strict time and processing requirements. However, the designing and planning of the upcoming sixth-generation (6G) communication network has already begun. 6G is seen as a disruptive technology that will go beyond the mobile internet and will support ubiquitous AI technology at the edge of the network. 6G is envisioned to offer computational efficient dedicated hardware capable of running AI/ML algorithms locally at the edge (see Fig. 8). The 6G infrastructure creates the perfect computational environment to deploy ML-SDWSN applications that impose stringent computational requirements. Offloading the control plane from cars to the 6G network can significantly improve the processing and communication latency, which is of high priority for delayed sensitive applications.

F. 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 the network performance of SDWSNs, and (iii) the ML potential to overcome the concerns raised when introducing SDN concepts in WSNs. From the research works that adopted ML in the context of SDWSNs, we can observe that ML-SDWSNs are still in an early development stage. However, a notable exploration has been already achieved. ML techniques have 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, improving the network security and energy.

1) CONTROL OVERHEAD

SDWSN has shown great performance in solving challenges currently present in traditional WSNs (see Section IV), and reacting to dynamic changes in the condition of the environment (not being able to be solved with the current traditional techniques of state-of-art WSNs). However, it has also shown that the amount of control overhead needed to implement SDN abstractions into the WSNs required appropriate attention. The ML-SDWSN paradigm is seen as a promising solution to reduce the amount of control overhead packets required to implement SDN abstractions into WSNs.

The global view granted through the SDWSN architecture permits the ML module to make accurate predictions allowing the controller to act promptly to changes in the network, provisioning proactively network resources, thus, reducing the control overhead and energy consumption. An example can be found in [144], where the SDWSN architecture collects network information such as reports of neighbouring nodes (also known as Neighbour Advertisements (NA)), and the ML module classifies static nodes from mobile nodes. The use of both SDWSN and ML technologies permits the controller to configure optimal routes in mobile nodes, at a precise time, to avoid them generating flow requests to the controller to find the path to their destination.

2) SECURITY

WSNs and SDWSNs are susceptible to security threats due to their broadcast nature and centralised architectures. Intruders can tamper with sensors and the overall network, putting at risk valuable network assets and systems. Traditional security solutions applied to wired networks cannot be applied directly in WSNs as most of the solutions require processing-intensive functions. The use of both SDWSN and ML technologies creates a new pathway to solve security issues inherent to WSNs and not being easy to solve in state-of-art WSNs due to their limited resources. The SDWSN realises the network collection (e.g., sensor behaviour, energy levels, raw data) and reconfiguration, while the ML module runs a suite of algorithms that can easily classify problematic nodes, identify network pitfalls, etc. Both technologies enable the execution of appropriate actions to mitigate the impact promptly.

ML-SDWSN grants an intelligent, centralised and resource-aware mechanism to protect the network against cyber-physical attacks. It frees up the processing and communication load of sensor nodes to implement security countermeasures.

3) ENERGY EFFICIENCY

Monitoring applications are often deployed in harsh environments with difficult access to the electrical network. These types of networks aim to run the programmed task for the longest time possible. To achieve the longest NL multiple solutions has been proposed [175]. SDWSN offers innovative mechanisms to bring forward new solutions to such problems (see Section IV-B1). The centralised architecture creates a new setting to run novel algorithms at the logically centralised control plane. ML has been used in SDWSNs to balance the overall energy consumption to prolong the NL. ML learns and identifies patterns from the information collected from the sensor nodes. This data is used by the ML module to
configure e.g., new routing paths, at a precise time, to minimize the main objective (e.g., overall energy consumption, individual energy consumption, etc.). Research results of ML-SDWSN works that aim to minimize the energy consumption show that ML-SDWSN technology is a good candidate to further extend the NL of traditional WSNs and SDWSNs. ML-SDWSN has not only been used for finding the routing path that reduces the energy consumption but it has also been used at the individual sensor node level. ML can tune the ideal transmission range for sensor nodes, thus minimizing the transmission energy. Also, one of the performance metrics to consider when devising a new routing path for SDWSNs is the individual remaining energy of sensor nodes. ML plays a role in predicting sensor nodes remaining energy, minimizing the need for sensor nodes to report their energy level, therefore, minimizing the control traffic and energy in the network.

The reduction of energy consumption is a key performance metric to consider when deploying monitoring applications. ML has shown great potential to achieve this goal. However, care must be taken with the frequency of the network configuration tasks, as this can negatively affect the network performance.

4) NETWORK RELIABILITY
ML plays a big role when it comes to improving the network reliability of traditional WSNs. ML uses the centralized information collected through the SDWSN architecture to identify patterns. These patterns (e.g., traffic congestion times, interference, nodes failures, task loads, etc.) are then used by the control plane to reconfigure the network (provision the network) to avoid a drop in the network reliability. For example, the use of ML in SDWSNs has been used to detect sources of interference and to trigger timely actions to mitigate them. ML has also been used to detect periodical heavy traffic links and to anticipate them by setting up new routing paths.

Network reliability is a key objective when designing WSN applications. There exist multiple solutions to enhance the network reliability in WSNs. They can range from verification at individual layers of the protocol stack up to end-to-end verification. Although they are state-of-art mechanisms to improve network reliability, they struggle to overcome network-level traffic issues.

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 WSNs. Besides, a comparison of the two locations of the ML module is needed (see Fig. 4) to appreciate the significance and the applications for both architectures. ML-SDWSN is a promising technology envisioned to evolve along with the deployment of 5G and 6G networks including SDN, ML, cloud computing, and Network Function Virtualization (NFV).

VII. SUMMARY OF SDWSN PROPOSALS
In this section, we provide simple statistics of previously discussed SDWSN proposals. This will allow us to uncover research open issues and future trends in SDWSNs.

A. SUMMARY
Fig. 9a shows the percentage of research works for each category. This lets 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 EOS used in SDWSN is Contiki as shown in Fig. 9b. 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. 9c. The control overhead, which is among the most important metrics, is considered in 11% of the surveyed works. Packet delivery metrics such as PDR and PLR are considered in the 8% of the proposals.

Fig. 9d shows the percentage of the 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 or 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, several 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. 10. 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 10 publications, followed by IEEE Access with 8, IEEE Systems Journal with 6 publications, and Sensors (MDPI), IEEE Sensors Journal and Journal of Ambient Intelligence and Humanized Computing (JAIHC) with 5 publications.

VIII. MAJOR CHALLENGES AND FUTURE DIRECTIONS
SDWSNs is a relatively 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 groups of stakeholders, is not an easy task [176]. Some SDWSN papers share similar architectural designs and protocols, while others have their own new architectures and protocols. There is currently no established
technical standard for SDWSNs that defines the set of functions and protocols for sensors nodes and controllers [15], [177], [178]. 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 [10]. 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. 9c), papers that specifically focus on reducing the control overhead is still low as shown in Fig. 9a. 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 requests 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 [73]. (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 messages 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 computational 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 expense 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 [131], whereas controller placement
in SDWSNs is 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. The use of distributed and dynamic SDWSN controllers (embedded in the sensor nodes) can potentially balance the expenditure of key network resources, e.g., energy. The use of ML algorithms to predict, and pinpoint the best locations where sensor nodes can run key controller functionalities may lead to an overall network improvement and a reduced control overhead.

### E. EOS

Fig. 9b 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 no 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 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 48% of the research works surveyed here adopted ML techniques in their proposals. The number of ML-SDWSN research works has been exponentially increasing, with a steep increase in 2020. 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 17 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 are 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 applications, 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. DL, for example, will be useful in reducing the complexity of model training, especially for large-scale WSN deployments, due to its ability to uncover patterns in the data to build more efficient decision rules. Some ML–SDWSN applications may have strict latency requirements, for such applications, DL could be useful in reducing the training phase and allowing the controller to react fast enough to changes in the network. Lastly, the advent of 6G, mainly its architecture and resources, and the flexibility gained in SDWSN architectures, set the perfect environment to run state-of-art ML algorithms and support upcoming ML approaches. 6G provides a powerful, flexible and multi-node architecture to run, deploy and manage ML-based distributed control architectures and logically centralised control schemes for large scale SDWSNs.

ACKNOWLEDGMENT

The document reflects only the authors’ view and the Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] F. Wortmann and K. Flüchter, “Internet of Things,” Bas. Inf. Syst. Eng., vol. 57, no. 3, pp. 221–224, 2015.
[2] Fog Computing and the Internet of Things: Extend the Cloud to Where the Things are, Cisco Syst., San Jose, CA, USA, 2016.
[3] Y. Jarraya, T. Madi, and M. Debabhi, “A survey and layered taxonomy of software-defined networking,” IEEE Commun. Surveys Tuts., vol. 16, no. 4, pp. 1955–1980, 4th Quart., 2014.
[4] F. Hu, Q. Hao, and K. Bao, “A survey on software-defined network and OpenFlow: From concept to implementation.” IEEE Commun. Surveys Tuts., vol. 16, no. 4, pp. 2181–2206, 4th Quart., 2014.
[5] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, “Machine learning in wireless sensor networks: Algorithms, strategies, and applications,” IEEE Commun. Surveys Tuts., vol. 16, no. 4, pp. 1996–2018, 4th Quart., 2014.
[6] K. Sood, S. Yu, and X. Xiang, “Software-defined wireless networking opportunities and challenges for Internet-of-Things: A review,” IEEE Internet Things J., vol. 3, no. 4, pp. 453–463, Aug. 2016.
[7] D. Kreutz, F. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” Proc. IEEE, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[8] I. T. Haque and N. Abu-Ghazaleh, “Wireless software defined networking: A survey and taxonomy,” IEEE Commun. Surveys Tuts., vol. 18, no. 4, pp. 2713–2737, 4th Quart., 2016.
[9] 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, pp. 1–32, 2017.
[10] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, “A survey on software-defined wireless sensor networks: Challenges and design requirements,” IEEE Commun. Surveys Tuts., vol. 5, pp. 159–179, 2017.
[11] S. Bera, S. Misra, and A. V. Vasilakos, “Software-defined networking for Internet of Things: A survey,” IEEE Internet Things J., vol. 4, no. 6, pp. 1994–2008, Dec. 2017.
[12] O. G. Mathou and A. M. Abu-Mahfouz, “Utilising artificial intelligence in software defined wireless sensor network,” in Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc. (IECON), Oct. 2017, pp. 6131–6136.
[13] 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 Commun. Surveys Tuts., vol. 20, no. 4, pp. 2855–2888, 4th Quart., 2018.
[14] K. M. Modieginyane, B. B. Letswanamote, R. Malekian, and A. M. Abu-Mahfouz, “Software defined wireless sensor networks application opportunities for efficient network management: A survey,” Comput. Electr. Eng., vol. 66, pp. 274–287, Feb. 2018.
[15] H. Mostafaei and M. Menth, “Software-defined wireless sensor networks: A survey,” J. Netw. Comput. Appl., vol. 119, pp. 42–56, Oct. 2018.
[16] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, “A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges,” IEEE Commun. Surveys Tuts., vol. 21, no. 1, pp. 393–430, 1st Quart., 2019.
[17] M. Abujubbeh, F. Al-Turjman, and M. Fahrioglu, “Software-defined wireless sensor networks in smart grids: An overview,” Sustain. Cities Soc., vol. 51, Nov. 2019, Art. no. 101754.
[18] S. M. W. Umba, A. M. Abu-Mahfouz, T. D. Ramotsela, and G. P. Hancke, “A review of artificial intelligence based intrusion detection for software-defined wireless sensor networks,” in Proc. IEEE 28th Int. Symp. Ind. Electron. (ISIE), Jun. 2019, pp. 1277–1282.
[19] D. P. Kumar, A. Tarachand, and C. S. R. Annavarapu, “Machine learning algorithms for wireless sensor networks: A survey,” Inf. Fusion, vol. 49, pp. 1–25, Sep. 2019.
[20] Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, “A survey of networking applications applying the software defined networking concept based on machine learning,” IEEE Access, vol. 7, pp. 95397–95417, 2019.
[21] T. Kim, L. F. Vecchietti, K. Choi, S. Lee, and D. Har, “Machine learning for advanced wireless sensor networks: A review,” IEEE Sensors J., vol. 21, no. 11, pp. 12379–12397, Jun. 2021.
[22] W. Dargie and C. Poellabauer, Fundamentals of Wireless Sensor Networks: Theory and Practice. Hoboken, NJ, USA: Wiley, 2010.
[23] K. Sohraby, D. Minoli, and T. Znati, Wireless Sensor Networks: Technology, Protocols, and Applications. Hoboken, NJ, USA: Wiley, 2007.
[24] F. F. Jurado-Lasso, E. Aragón-Gració, and M. A. B. A. Varela, “Software-defined networking for improved wireless sensor network management: A survey,” in Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc. (IECON), Oct. 2017, pp. 6131–6136.
[25] Texas Instruments, “CC2538 Powerful Wireless Microcontroller System-on-Chip for 2.4-GHz IEEE 802.15.4, 6LoWPAN, and ZigBee Applications, CC2538 Datasheet,” Texas Instrument, Dallas, TX, USA, Apr. 2015.
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 Wirel. Commun., vol. 25, no. 4, pp. 74–81, Aug. 2018.

R. Kaushik and J. Singhai, “Classification of quality of granary using machine learning based on software-defined wireless sensor network,” in Proc. 8th Int. Conf. Signal Process. Integr. Netw. (SPIN), Aug. 2021, pp. 652–656.

H.-M. Tsai, O. K. Tonguz, C. Saraydar, T. Talty, M. Ames, and A. Macdonald, “ZigBee-based intra-car wireless sensor networks: A case study,” IEEE Wireless Commun., vol. 14, no. 6, pp. 67–77, Dec. 2007.

R. Tavakoli, M. Nabi, T. Basten, and K. Goossens, “Topology management and TSCH scheduling for low-latency convergecast in in-vehicle WSNs,” IEEE Trans. Ind. Informat., vol. 15, no. 2, pp. 1082–1093, Jul. 2019.

S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, “Orchestra: Robust mesh networks through autonomously scheduled TSCH,” in Proc. 13th ACM Conf. Embedded Netw. Sensor Syst., Nov. 2015, pp. 337–350.

F. F. Jurado-Lasso (Member, IEEE) received the B.Eng. and Ph.D. 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, Council for Scientific and Industrial Research (CSIR); an Extraordinary Professor at the University of Pretoria; a Professor Extraordinaire at the Tshwane University of Technology; and a Visiting Professor at the University of Johannesburg. He has participated in the formulation of many large and multidisciplinary research and development successful proposals (as a principal investigator or a main author/contributor). He 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. His research interests include wireless sensor and actuator networks, low power wide area networks, software defined wireless sensor networks, cognitive radio, network security, network management, and sensor/actuator node development. He is a member of many IEEE technical communities. He is an Associate Editor at IEEE Access, IEEE INTERNET OF THINGS JOURNAL, and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.

F. F. Jurado-Lasso (Member, IEEE) received the B.Eng. degree in electronics engineering from the Universidad del Valle, Cali, Colombia, in 2012, and the M.Eng. degree in telecommunications engineering and the Ph.D. degree in engineering from The University of Melbourne, Melbourne, VIC, Australia, in 2015 and 2020, respectively.

He is currently a Postdoctoral Researcher at the Embedded Systems Engineering (ESE) Section, Department of Applied Mathematics and Computer Science, Technical University of Denmark (DTU Compute). His research interests include networked embedded systems, software-defined wireless sensor networks, machine learning, and protocols and applications for the Internet of Things.

J. F. JURADO received the B.Sc. degree in physics from the Universidad de Nariño, Pasto, Colombia, in 1984, and the M.Sc. and Ph.D. degrees in physics from the Universidad del Valle, Cali, Colombia, in 1986 and 2000, respectively.

He is currently a Professor with the Department of Basic Science, Faculty of Engineering and Administration, 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.

LETIZIA MARCHEGIANI (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer engineering from the Sapienza University of Rome, Italy, and the Ph.D. degree in computer engineering from the Sapienza University of Rome, in 2012. From 2014 to 2018, she was a Researcher at the University of Oxford, U.K., where she was a member of the Oxford Robotics Institute (ORI). Previously, she was associated with the Investigating Speech Processing In Realistic Environments (INSPIRE) ITN as a Marie Curie Postdoctoral Research Fellow, from 2012 to 2013. Since 2019, she has been an Assistant Professor in robotics with the Automation and Control Section, Department of Electronic Systems, Aalborg University, Denmark. Her research interests include signal processing, machine learning, and their application to robotics, autonomous systems, cognitive modeling, intelligent transportation, and digital healthcare.

XENOFON FAFOUTIS (Senior Member, IEEE) received the B.Sc. degree in informatics and telecommunications from the University of Athens, Greece, in 2007, the M.Sc. degree in computer science from the University of Crete, Greece, in 2010, and the Ph.D. degree in embedded systems engineering from the Technical University of Denmark, in 2014. From 2014 to 2018, he held various researcher positions at the University of Bristol, U.K. He was a Core Member of SPHERE: U.K.’s Flagship Interdisciplinary Research Collaboration on Healthcare Technology. He is currently an Associate Professor with the Embedded Systems Engineering (ESE) Section, Department of Applied Mathematics and Computer Science, Technical University of Denmark (DTU Compute). His research interests include wireless embedded systems as an enabling technology for digital health, smart cities, and the (Industrial) Internet of Things (IoT).