IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 15, 2020, accepted May 20, 2020, date of publication June 1, 2020, date of current version June 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2999087

A Versatile Out-of-Band Software-Defined Networking Solution for the Internet of Things

TRYFON THEODOROU, (Member, IEEE), AND LEFTERIS MAMATAS, (Member, IEEE)
Department of Applied Informatics, University of Macedonia, 546 36 Thessaloniki, Greece

Corresponding author: Tryfon Theodorou (theodorou@uom.edu.gr)

This work was supported in part by the EU's Horizon 2020 Research and Innovation Program through the EU-BRA Horizon 2020 NECOS Project under Grant agr. no 777067, in part by the European Commission and the Brazilian Ministry of Science, Technology, Innovation, and Communication (MCTIC) through the Brazilian National Research and Educational Network (RNP) and Directorate for Computer & Information Science & Engineering (CTIC), and in part by the Open Call Scheme of the Wireless Software and Hardware platforms for Flexible and Unified radio and network controL project (WiSHFUL) under Grant agr. no 645274.

ABSTRACT The Internet of Things (IoT) is gradually incorporating multiple environmental, people, or industrial monitoring deployments with diverse communication and application requirements. The main routing protocols used in the IoT, such as the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), are focusing on the many-to-one communication of resource-constraint devices over wireless multi-hop topologies, i.e., due to their legacy of the Wireless Sensor Networks (WSN). The Software-Defined Networking (SDN) paradigm appeared as a promising approach to implement alternative routing control strategies, enriching the set of IoT applications that can be delivered, by enabling global protocol strategies and programmability of the network environment. However, SDN can be associated with significant network control overhead. In this paper, we propose *VERO-SDN*, an open-source SDN solution for the IoT, bringing the following novelties in contrast to the related works: (i) programmable routing control with reduced control overhead through inherent protocol support of a long-range control channel; and (ii) a modular SDN controller and an OpenFlow-like protocol improving the quality of communication in a wide range of IoT scenarios through supporting two alternative topology discovery and two flow establishment mechanisms. We carried out simulations with various topologies, network sizes and high-volume transmissions with alternative communication patterns. Our results verified the robust performance and reduced control overhead of *VERO-SDN* for alternative IoT deployments, e.g., achieved up to 47% reduction in network's overall end-to-end delay time compared to RPL and a packet delivery ratio of over 99.5%.

INDEX TERMS Internet of Things, out-of-band control, software-defined networks, wireless sensor networks.

I. INTRODUCTION

Wireless Sensor Networks (WSN) operate at the edge of conventional network infrastructures and provide communication means between the digital and the physical world. WSN connect tens or hundreds of tiny wireless network devices equipped with sensors and actuators, called motes, capable of measuring real-world phenomena and interacting with a variety of hardware devices, e.g., industrial control equipment. Their main features are low cost, scalability, and low energy consumption, whereas their drawbacks are the limited computational resources, limited bandwidth, and low-quality radio communication [1]. Nowadays, the WSN motes are getting smaller, cheaper, and smarter, enabling their usage in a wide range of applications, such as on industrial monitoring, environmental or people's surveillance.

The support of Internet technologies from the WSN motes contributes in the new Internet evolution known as the *Internet of Things* (IoT) [2], [3]. The advent of the IoT arises new trends, such as enhanced network management, intelligent-processing capability, efficient use of resources, adaptive network operation to business conditions, and large-scale deployments, while maintaining the network's reliability, performance, and Quality of Service (QoS) [4]. These challenges impose communication requirements that are difficult to be addressed by conventional network protocols, inherited from the WSN world.

For example, the de facto standard distance vector protocol for WSN, the *Routing over Low Power and Lossy Networks* (RPL) protocol [5], is one of the most important

The associate editor coordinating the review of this manuscript and approving it for publication was Prakasam Periasamy.

routing solutions for the IoT. RPL builds a logical routing topology graph as a *Destination Oriented Directed Acyclic Graph* (DODAG) using an objective function and a set of metrics/constrains. It is characterized by significant benefits, including multi-hop communication, efficient operation over noisy channels, and IPv6 support.

RPL is architecturally specialized for many-to-one WSN scenarios (i.e., data collection from all network nodes to a central node, namely the sink). Moreover, it supports the one-to-many communication pattern in storing mode, i.e., for the transmission of queries to sensors or the transmission of actuation commands, when a control loop is present. These communication patterns were mainly motivated by the need to support monitoring applications and their routing requirements, back in 2009 when IETF was specifying RPL's architectural characteristics.

Although monitoring is still a key IoT application, "smarter" IoT applications that emerge are enhancing the application's decision making and communication capabilities. Such independence to decide and act comes with the requirement for more complex communication patterns than the typical many-to-one, including direct communication with other nodes. For example, in a smart traffic light system, apart from the basic needs for communication between the traffic lights and a central point (i.e., implementing many-to-one and one-to-many communication patterns), small control loops among nearby devices can implement fault tolerance tasks, like a green traffic light that informs with node-to-node messaging other traffic lights in its neighborhood of its state, i.e., to avoid conflicting traffic lights. When it comes to such communication requirements, RPL faces performance and reliability issues, which raises the question on whether RPL can support today's IoT requirements [6].

The *Software-Defined Networks* (SDN) provide a new, elastic network paradigm that transforms traditional network backbones into flexible service-delivery platforms and improves the network's utilization [7]. SDN, initially introduced for infrastructure networks, are being gradually expanding to new environments, such as the WSN. To this end, efforts presented in [8] and [9], attempt to *evolutionary* enable the above capabilities through the adaptation of SDN operations on top of existing WSN protocols, e.g., the RPL protocol. Those solutions are predominantly confronted by the restrictions of the underlying protocol operation (e.g., mainly supporting the many-to-one type of communication) and the intensification of the control message overhead in the wireless medium.

Other research endeavors follow a *clean-slate* approach [10], [11] and integrate SDN Openflow-like architectures with the WSN technologies to provide new perspectives and grounds for the IoT applications. This new approach, called the *S*oftware-*D*efined *W*ireless *S*ensor *N*etworks (*SDWSN*), brings new ways of controlling and operating the WSN through applying logically-centralized network control. For example, it improves routing and topology control techniques by offloading network management intelligence to a central controller, reducing the computational process requirements from the low memory, storage, and processing power motes. Such approaches are inheriting the advantages of the traditional OpenFlow-based solutions over fixed networks, but are not yet fully aligned with the unique requirements of the IoT networks, e.g., the resource-constraints of the nodes and lossy nature of the wireless medium [12].

As a bottom line, this first generation of solutions should evolve towards: (i) supporting alternative application communication patterns (i.e., many-to-one, one-to-many, one-to-one [13]); (ii) mitigating the increased amount of control packets due to the frequent communication between the network nodes with the SDN controller; and (iii) considering in a greater extent the resource-constraints of the involved IoT devices and the lossy nature of the wireless medium.

Here, we argue that an important feature of an SDN solution for IoT should be a separate control channel, for the following reasons: (i) the control channel associates with different communication requirements compared to the data channel (e.g., in level of robustness), so bespoke protocols can be used for each one of them; and (ii) the control messages should not be causing performance or reliability issues to the data communication. Although a number of IoT motes already support double radio interfaces (e.g., the Zolertia RE-Mote devices [14]), such exercise requires systemic adaptations, spanning from the protocol level to the involved SDN controller and its mechanisms.

Although an additional network interface increases the hardware complexity and construction cost of the mote, we argue that this can be balanced out from the benefits it brings to our solution, i.e., overcoming a major drawback of SDWSN solutions: the unreliable and inefficient control message communication. This approach exploits a natural strength of wireless communication, i.e., the flexibility of medium deployment. The construction cost of small hardware amendments that enhance the usability of the device can be absorbed and not be reflected in its market cost. For example, the first mobile phones with one GSM communication interface were not cheaper than today's mobile phones that contain dual-GSM, GPS, WiFi, Bluetooth, and IR.

Moreover, we argue that the additional energy consumption of the secondary channel is balanced by the energy-savings due to: (i) the transfer of computational power from devices to the infrastructure network [12]; and (ii) the most informed and accurate decisions for the network operation. Furthermore, the second channel can enable new approaches for energy conservation, such as the proposals [15] and [16]. However, this aspect deserves an independent study.

Along these lines, we propose *VERO-SDN*[1], a *VERsatile with Out-of-band control* OpenFlow-like SDWSN solution that improves the communication performance of a wide range of IoT applications while reducing the SDN control overhead. *VERO-SDN* natively supports: (i) a separate

[1]The Italian word "vero" translates to "real", in the English language.

wireless control channel that introduces one-hop communication with the SDN controller, through adopting a double protocol stack and appropriate routing control; and (ii) an SDN protocol and controller dynamically maintaining the IoT topology based on two novel topology discovery and two flow control mechanisms. The impact of *VERO-SDN* operation is reflected in our extended evaluation (i.e., discussed in Sections V-A4 and V-B4), highlighting *VERO-SDN's* achievements in terms of packet delivery ratio, network overhead, and end-to-end delivery time.

An early implementation of *VERO-SDN* with the name *CORAL-SDN* is described in the demo paper [17], while the same *CORAL-SDN* version is supported by the MINOS, which is a multi-protocol network control platform for the IoT [18]. In the conference paper [19], we presented and evaluated relevant topology discovery algorithms with these documented here, e.g., we enriched the latter algorithms with the support of the separate control channel. We released the source code of *VERO-SDN* as an open-source, as well as several videos demonstrating its novel features [20].

The remainder of this paper is organized as follows. Section II contrasts *VERO-SDN* with the state of the art solutions addressing the SDWSN and important WSN protocols. Section III elaborates on the *VERO-SDN* architecture, including its main components and functionalities, while Section IV provides the design details of the main *VERO-SDN* protocol mechanisms. Furthermore, an extensive evaluation is covered from Section V, illustrating the performance and reliability advantages of our solution. Finally, Section VI discusses further improvements and research challenges, while the paper concludes with Section VII.

II. RELATED WORKS

In the research front towards the adoption of the SDN paradigm from the IoT, we single out two categories of approaches, as briefly discussed in the Introduction. The first one proposes SDN-inspired network control facilities that operate on top of existing WSN protocols, such as RPL [5], which centrally fine-tune protocol parameters and processes. The second category of solutions covers SDN protocols and their associated network controllers implementing forwarding mechanisms that are harmonized with the traditional SDN architecture, i.e., separating the control from the data plane.

The first category of proposals is *evolving the WSN towards the SDN world* and includes the work [21] discussing the synergy between the SDWSN protocol TinySDN [22], [23] with RPL and how they can benefit from each other. The paper μSDN [8] presents a lightweight, low-overhead SDN-inspired framework that builds on the current trends towards network control centralization for the IEEE 802.15.4 networks. μSDN supports compatibility with the IPv6 networks, interoperability with the RPL protocol, and implements centralized network optimization techniques to improve the legacy protocol operation and provide QoS for critical network flows. A relevant solution is *Whisper* [24], which manipulates RPL and 6TiSCH operation using a controller that transmits compatible control messages with RPL. The recent work [9], [25], configures the protocol's parameters based on particular behavior of the nodes (e.g., whether they are mobile or not) based on SDN-inspired mechanisms that form closed *monitor-decide-configure* control loops. The prime advantage of such solutions is the backward compatibility with the traditional protocols. However, they fail to improve the protocol's performance significantly, because they cannot fully exploit the SDN features, due to the above interoperability and compatibility limitations.

The second category of solutions is following the reverse path through *bringing the SDN paradigm to the WSN environments*, i.e., radically changing the network environment. The paper [26] is proposing the *Sensor OpenFlow*, a conceptual OpenFlow-based protocol [27]. The same paper is identifying key technical challenges of *Sensor OpenFlow*, e.g., its increased control overhead. Gante *et al.* [28] elaborate on benefits that SDN brings to WSN, e.g., enhanced network management, advanced topology discovery, and improved sensor node mobility handling. Costanzo *et al.* [29] propose *SDWN*, an SDN framework addressing a number of technical requirements, in-network data aggregation, and flexible definition of rules, to improve performance aspects, e.g., in terms of the control messages overhead and the energy consumption. SDN-WISE [30] extends the same *SDWN* framework towards adopting stateful routing tables and proactive routing decisions to reduce the number of interactions with the controller and improve the flow-rule establishment decisions.

Towards a better integration of the SDN-enabled IoT with the fixed SDN environments, Anadiotis *et al.* [31] propose an amalgamation of a network operating system, the *Open Network Operating System (ONOS)*, with the aforementioned *SDN-WISE* platform. In the same context the work [32] presents *SD-WISE*, an SDWSN solution which is integrated with an extended *ONOS* implementation. *SD-WISE* provides abstractions of the nodes' resources, enables network function virtualization in WSN, and leverages the flexibility of flow definition and RDC control to achieve energy efficiency. *TinySDN* [22], [23] implements a distributed TinyOS-based control plane architecture based on multiple controllers. In [33], *TinySDN* enabled nodes are enhanced with the *Spotled*, a 2-level hierarchy of physically distributed global and local SDN controllers, striving with the reduction of control traffic. *SoftSDN* [34] supports application-aware service provisioning in IoT, implementing basic SDN features, i.e., topology control, device, and network management to meet run-time and application-specific requirements. The major drawbacks of the above-quoted studies are, on the one hand, the challenges of the additional complexity and overhead that SDN architecture brings to WSN, and on the other hand the reduced efficiency of the SDN operation due to the dubious transmission of the control messages over Low-power and Lossy Network's (LLN) multi-hop medium.

A most recent work from Baddeley *et al.* [35], [36] towards the alienation of control from data messages, suggests *AtomicSDN*, a time-sliced mechanism that separates the SDN control messages from the WSN data plane layer messages using designated flooding periods for the control messages, and improving the network's latency, reliability, and energy consumption metrics.

All prior frameworks advance the idea of the SDN paradigm utilization in the WSN; nevertheless, the in-band physical coupling of control and data planes leads to undesirable consequences, such as the unreliable control plane operation and the encumbered with control messages data communication plane. Our proposal is the only SDN-based solution that employs a separate control channel to overcome the above issues.

Along these lines, a limited amount of research studies investigates the usage of a separate radio channel for the control messages in more traditional wireless multi-hop environments (e.g., WSN or ad-hoc networks). In [15] and [16], a second radio channel, acts as a wake-up medium for activating and deactivating the primary data communication radio interface, aiming mainly to reduce the mote's power consumption. *WASP* [37] framework implements a data plane for mobile ad-hoc networks using the Wi-Fi Direct and manipulates its operation through an LTE-based control plane. WASP considers smartphone devices but not the low-power motes. Gu *et al.* [38] suggest the physical separation between the control and data plane for a network of Raspberry Pi computers utilizing the Zigbee protocol. They implement one-hop out-of-band control using LoRaWAN and improve the packet delivery ratio with ad hoc interventions to the Collection Tree Protocol (CTP) protocol forwarding decisions, i.e., through the LoRaWAN control channel. This approach does not investigate the additional network performance and manageability aspects towards fully adopting the SDN paradigm.

VERO-SDN supports the idea of using two separate radio channels, one for control and one for data communication, and materializes a complete out-of-band SDN framework for WSN. Our approach exploits further the potential that the SDN paradigm brings to WSN, in terms of performance and reliability, as highlighted in our extended evaluation (i.e., Section V).

III. VERO-SDN FRAMEWORK

In this Section, we present the operational framework of the *VERO-SDN* protocol through a high-level overview of its architecture and interfaces, including a detailed description of the associated software and hardware components.

A. ARCHITECTURE OVERVIEW

VERO-SDN, in contrast to the other approaches implementing the SDN paradigm over the IoT, rather than using the same medium to communicate data and control messages (i.e., in-bound control), it splits out the network communication control to a separate dedicated radio channel (i.e., out-bound control). This approach requires a second radio interface on the IoT mote device, the appropriate secondary network protocol stack and bespoke network control mechanisms.

A number of IoT motes are equipped with double radio interfaces, but for installation flexibility mainly, i.e., employing a single protocol stack that uses one of the two devices only, depending on the installation configuration. For example, the Zolertia RE-Mote platform [14] supports two radio interfaces [39]: (i) one long-range *SubGHz* $868/915MHz$ RF–transceiver with distance ranging from $712m$ to $5km$, depending on the data rate and the amplification level; and (ii) one short-range $2.4GHz$ transceiver with $50-100m$ coverage distance. Consequently, the long-range interface can be used for the SDN control channel and the short-range for the data communication, depending on appropriate protocol and control facilities that enable this strategy. In the context of this work, we fully design, implement and evaluate through a series of simulations the advantages of *VERO-SDN* solution (i.e., SDN controller and data plane mechanisms), in terms of communication performance and reliability.

Fig. 1 gives an overview of *VERO-SDN* operational structure based on a dual-radio interface. In detail, it is composed of the following functional entities:

- The *VERO-SDN Controller*, located in the infrastructure network, constitutes the heart of the WSN. Operating on a computer system with high computational power and memory, performs costly computations and equips the WSN with centralized decision-making features.
- The *VERO-SDN Adapter* is responsible for communicating control messages between the *Controller* and the *Border Router (BR)*. Physically it is located close to the latter and enables the former to be off-site.
- The *BR* handles the control messages from and to all other network nodes. At the same time, acts, if needed, as the WSN sink mote, i.e., for the data collection scenarios. It supports three interfaces: i) a long-range *SubGHz* RF transceiver for the control-plane communication; ii) a short-range $2.4GHz$ RF transceiver for the data-plane; and iii) a connection to the SDN adapter (i.e., currently serial, for simplicity). In order to support very large topologies and a high number of nodes (e.g., for a smart city deployment), *VERO-SDN* protocol is designed to support multiple *BR*. This feature will be discussed in subsection III-B as part of the protocol's *southbound Application Programming Interface (API)*. However, this aspect deserves an independent study due to its level of complexity.
- The *Network Nodes* are low-power IoT motes that support dual RF transceivers and a number of sensors and actuators being responsible for the data acquisition and control.
- The *Control Channel* is responsible for the direct communication between all network nodes and the *BR* through the long-range radio interfaces. The *Control Channel* forms a *cone graph*, where the *BR* is the *universal vertex* of the undirected graph that is adjacent to all other network nodes, or graph vertices. As such, the optimal position for the *BR* is in the center of the network. However, the *BR* can be placed anywhere in

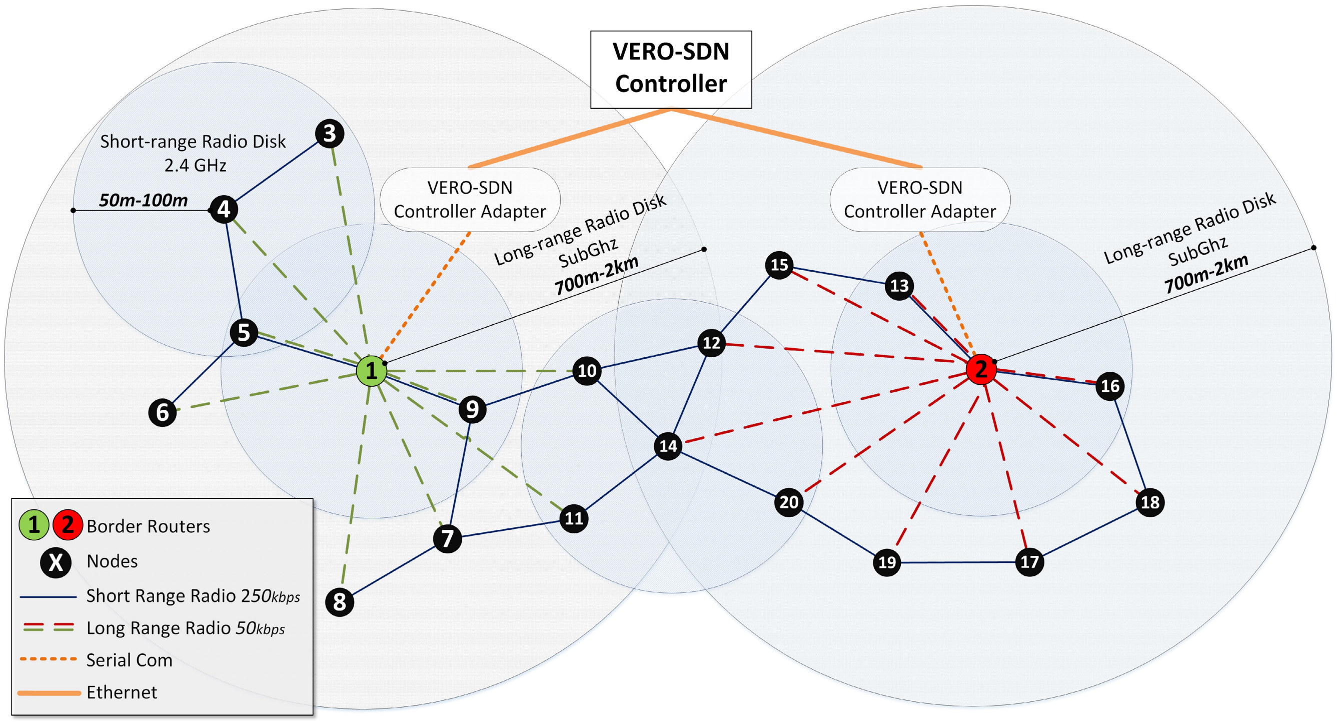

VERO-SDN Controller
VERO-SDN Controller Adapter
VERO-SDN Controller Adapter
Short-range Radio Disk 2.4 GHz
50m-100m
Long-range Radio Disk SubGhz
700m-2km
Long-range Radio Disk SubGhz
700m-2km
1 2 Border Routers
X Nodes
Short Range Radio 250kbps
Long Range Radio 50kbps
Serial Com
Ethernet

FIGURE 1. ***VERO-SDN* operational framework schema where each node espouses a dual radio network interface, for long and short range communication.**

the network terrain, but within the nodes' long-range distance. Control messages were intentionally designed to be small-sized to adapt to the potentially low throughput of the long-range wireless channel, e.g., $50kbps$.

- For the *Data Communication Channel*, *VERO-SDN* designates the short-range radio interfaces at $2.4GHz$, where higher data rates, (e.g., $250kbps$), are important. The *Data Communication Channel* forms a *undirected connected graph*, where messages are transferred to any peer node through multi-hop paths.

Fig. 2 gives a high-level representation of the *VERO-SDN* solution that consists of three planes, aligned to the typical SDN architecture [40]: (i) the *Application plane* providing high-level network management, monitoring, and the IoT applications; (ii) the *Control plane* manipulating an abstracted and logically-centralized anatomy of the infrastructure network through applying sophisticated network control algorithms; and (iii) the *Infrastructure plane* covering the dual network stacks that implement the control and data communication channels, among the neighbor nodes and the *BR*.

We now elaborate on the three *VERO-SDN* planes and their relevant functionalities.

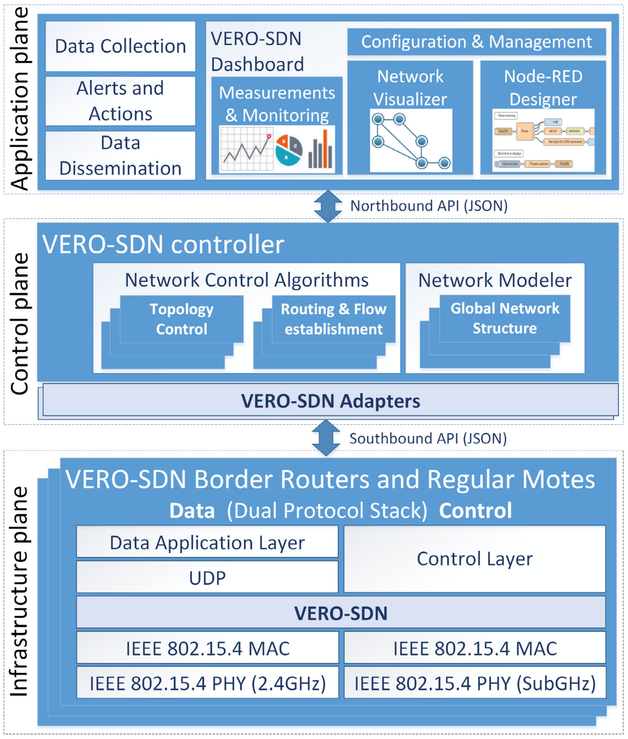
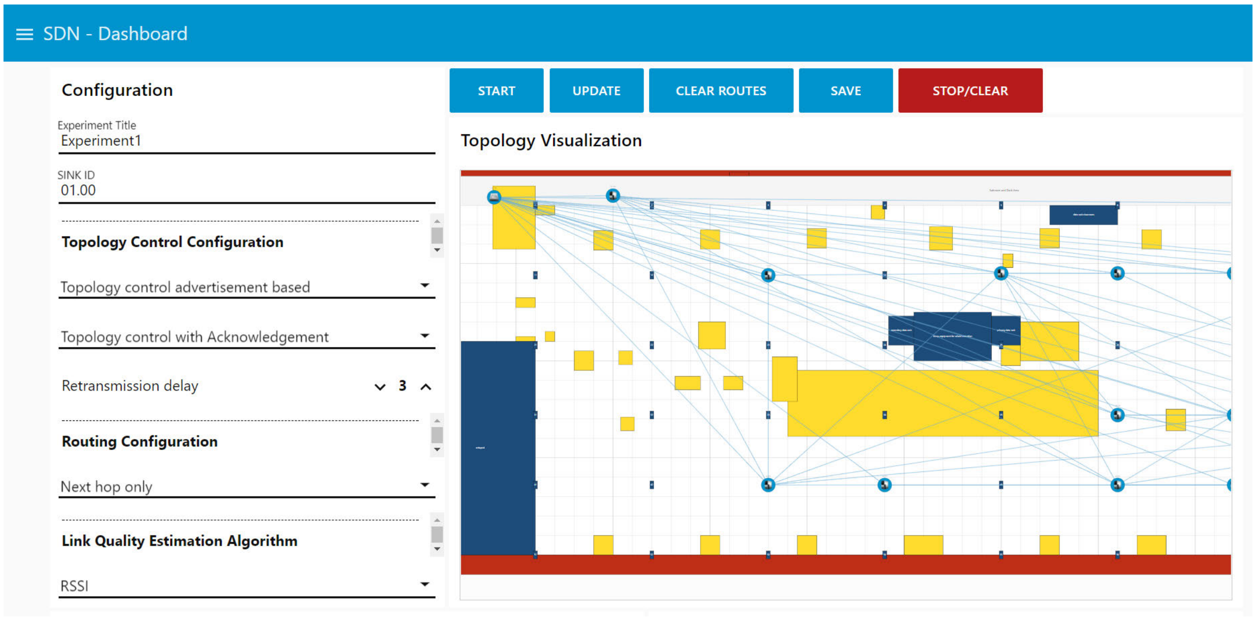


FIGURE 2. ***VERO-SDN* architecture.**

1) THE APPLICATION PLANE

The *Application plane* monitors and manages *VERO-SDN* infrastructure from a high-level viewpoint and enacts as the ground for user-defined IoT applications utilizing the WSN infrastructure, i.e., through the *VERO-SDN northbound API*. According to [41], these applications can be classified as: (i) *Data collection*; (ii) *Alerts and Actions*; and (iii) *Data Dissemination* applications. To demonstrate *VERO-SDN northbound API* functionality and the overall protocol



Configuration

Experiment Title: Experiment1

SINK ID: 01.00

Topology Control Configuration

Topology control advertisement based

Topology control with Acknowledgement

Retransmission delay 3

Routing Configuration

Next hop only

Link Quality Estimation Algorithm

RSSI

FIGURE 3. ***VERO-SDN*** **dashboard GUI; left-hand-side, configuration parameters; right-hand-side, network topology graphical representation.**

operation, we developed our dashboard and visualization facility, the *VERO-SDN Dashboard*.

VERO-SDN Dashboard is a flexible, web-based, and user-friendly GUI for the overall network monitoring and system management, providing advanced system visualization and configuration options, as shown in Fig. 3. The *Dashboard* is implemented with the *Node-RED* framework [42], which is based on the *Node.js* programming environment. Its functionality is divided into three modules:

- The *Network Configuration*, which provides a graphical user interface with a list of network management configuration options and alternative protocol setups, e.g., the type of the topology discovery algorithm, the type of the forwarding rules establishment, or the link quality metric options for the routing decisions.
- The *Network Visualizer* providing graphical visualization of the network topology along with details about the nodes and links. The network administrator can observe various network parameters and performance measurements through the monitoring section of the Visualizer, illustrating as well as evaluation results in charts and tables.
- The *Node-RED Designer* offering a library of pre-implemented Node-RED nodes and flows that can be wired together, implementing and automating different network management processes for *VERO-SDN*. The implementation decision to develop this module in Node-RED provides extensibility since it offers the flexibility to add and modify new *VERO-SDN* features.

2) THE CONTROL PLANE

The *Control plane* acts as a hub between the *Application* and the *Infrastructure* planes. It comprises of the *VERO-SDN Controller*, which constructs and maintains an abstract representation of the infrastructure network, the *Global Network Structure*. This abstract view is an undirected connected graph, which nodes and links correspond to the network devices and the wireless connections among them, respectively. The *Global Network Structure* is kept at the *Network Modeler* module and is updated with information, such as the nodes' network address and energy level, or adjacency information to neighbor nodes, e.g., the Link Quality Indicator (LQI) and the Received Signal Strength Indicator (RSSI) values.

Moreover, *VERO-SDN Controller* is responsible for the centralized management of the network's routing decisions. In particular, it handles the following tasks: (i) maintains an abstract view of the network through the supported topology control algorithms; (ii) takes efficient routing decisions and performs dynamic forwarding rules establishment; and (iii) adjusts the protocol parameters dynamically. *VERO-SDN Controller*, through the *northbound API*, constantly provides the *Application plane* with network monitoring information, for example, the *Dashboard GUI* visualizes the network connectivity graph based on *northbound API* monitoring messages. In addition, it receives high-level configuration options and directives, such as the selection of the link quality estimation metrics exploited from the flow decision algorithm (e.g., RSSI, LQI, or node's energy). It actually provides to the network administrators or to particular intelligent applications with the means to refine the overall network operation. *VERO-SDN Controller* is implemented as open-source software (i.e., [20]) based on the Java programming language. Its modular design easily accommodates new modules or algorithms, facilitating new functionalities.

3) THE INFRASTRUCTURE PLANE

The *Infrastructure plane* is composed of the multi-hop WSN motes. These motes are either a border router or regular IoT motes. All motes contain two radio interfaces, operated by the two *VERO-SDN* network stacks (i.e., the control network and the data network stacks). Both of them are implemented using the C programming language, for the Contiki-OS 3.0 [43], and are embedded into the IoT devices' firmware. In the context of this work, we had to employ the following facilities to implement mote devices with dual-radio interfaces:

1) *Contiki fork with dual-radio features:* Since the Contiki-OS does not support a dual network stack operation in its standard version, we used as a basis a relevant forked version of it [44]. Moreover, we had to ameliorate the Contiki core network modules to enable the two network stacks.
2) *Zolertia RE-Mote devices upgrade:* Although the Zolertia RE-Mote devices contain two radio interfaces, in their standard version, they are not designed to operate at the same time. Applying an upgrade suggested by Zoleria [45], allowed these motes to become capable of using both radio interfaces concurrently.

The *Data Network Stack* consists of the following layers: (i) the IEEE 802.15.4 Physical (PHY) and Media Access Layer (MAC) layers, offering standardized low-power wireless communication and media access control in the band of 2.4*GHz*; and (ii) the *VERO-SDN* forwarding layer, as a core aspect of our data-plane protocol maintaining a forwarding table for the data packets. The *Control Network Stack* consists of: (i) the IEEE 802.15.4 PHY and MAC layers with standardized low-power and wireless communication and media access control in the band of 868*MHz*; and (ii) the *VERO-SDN* control layer that manipulates the control messages and operates the control processes.

B. VERO-SDN API

Aligned with the SDN paradigm, the *VERO-SDN Controller* plane communicates with the *Application* and *Infrastructure* planes using a *northbound* and a *southbound* API, respectively. The communication messages are formed as *JavaScript Object Notation (JSON)* text strings, for simplifying the interaction with third-party IoT applications.

VERO-SDN northbound API offers two categories of communication messages:

- The *Configuration Messages* that are out-bound messages which either set protocol's configuration options (i.e., topology discovery algorithm type, flow establishment type, and link quality estimation metrics), or control the protocol execution parameters (i.e., start, stop, update, and reconfigure). With these commands, administrators or IoT applications can have full control over the protocol's operation.
- The *Monitoring Messages*, which are out-bound messages used for monitoring and the evaluation of the network status. They provide constant updates about the network's connectivity with an abundant number of parameters (e.g., the nodes' energy level, the network's adjacency degree, the connection links' quality). The research community may utilize this information and contribute to applications that enhance further its intelligent network management capabilities. To this end, we already integrated the *Weka* machine learning software tool [46] with the *Feature Extractor* approach for the link quality estimation and prediction proposed in [47]. This way, *VERO-SDN* is able to establish flows based on predicted LQI. The results of this work received the *eWINE Grand Challenge* first runner up award [48]. A further discussion on this matter is out of this paper's scope.

TABLE 1. ***VERO-SDN* Southbound API.**

Protocol Operation	Message Category	Message Type	Payload (bytes)
Topology Control	Border Router	New Border Router Solicitation Request	2
		New Border Router Registration	6
	Node Discovery	New Node Solicitation Request	6
		New Node Response	15
	Neighbor Discovery	Neighbor Request *TC-NA*	9
		Neighbor Request *TC-NR*	13
		Neighbor Response	27
Routing	Missing Route	Missing Forwarding Rule	14
	Add Route	Add Forwarding Rule	23
		Replace Forwarding Rule	23
	Remove Route	Remove Forwarding Rule	14
		Remove All Forwarding Rules	14

VERO-SDN southbound API handles the control messages the *VERO-SDN Controller* exchanges with the network nodes. It is designed with the condition that the *Controller* connects directly with the routing nodes, following a very similar command set to the OpenFlow SDN protocol. Although it is more complicated than the *northbound API*, we intended to keep it as simple as possible, mainly because simple protocol control messages allow easier maintenance and future extensions. In Table 1, we enlist the *southbound API* messages classified into two categories based on their use in the protocol (i.e., Topology Control and Routing). It is worth mentioning that the *Border Router* messages are designed for managing a plethora of *BR* nodes, supporting topologies with a high number of nodes. However, our current simulations (i.e, in Section V) are using one *BR* node and the scaled-up simulations with multiple *BR* nodes are in our future research plans.

The southbound messages are transmitted in two phases: (i) at the first stage, they are JSON messages sent from the *Controller's* Ethernet port to the *Border Router's* serial port through *VERO-SDN Adapter's* conversion; (ii) at the second stage, the *BR* node compacts them by removing the JSON tags and transmits them through the long-range radio interface to the nodes. The size of these messages is directly reflecting the protocol's performance since the long-range radio has a low throughput. As indicated in Table 1, the maximum payload size in the protocol is in the *Neighbor Response* message with 27 bytes.

IV. VERO-SDN MECHANISMS

In this Section, we elaborate on the main network control features of *VERO-SDN* platform and their associated protocol aspects, i.e., residing at the controller and the data plane, respectively.

The network *topology* and *routing control* are two important network control functions of IoT environments. The former detects and maintains the network connectivity, while the latter establishes and retains the communication paths among the network nodes. The efficient design and implementation of these functions have a direct impact on critical network operation performance aspects, e.g., in terms of packet delivery ratio, end-to-end delay, and control overhead. An important issue is their suitability to various network and application contexts, covering both application-specific requirements and dynamic changes in the network environment.

Along these lines, we next elaborate on the main network topology and routing control features of *VERO-SDN*.

A. NETWORK TOPOLOGY CONTROL

Topology control is an important procedure for the efficient operation of an IoT network. Its operation is divided into two parts, i.e., the *topology discovery (or construction)* and the *topology maintenance* processes. For the former, *VERO-SDN* implements two novel algorithms initially introduced in [19] and adapted to utilize the out-of-bound control channel considered here: (i) the *Node's Advertisement Flooding (TC-NA)*; and (ii) the *Node's Neighbors Requests from the Controller (TC-NR)*. For the latter, *VERO-SDN* applies a topology update algorithm that is driven centrally from the *Controller* and adapts dynamically to the context environment. The details of such *VERO-SDN* topology discovery and maintenance processes follow.

1) TOPOLOGY DISCOVERY USING NODE's ADVERTISEMENT FLOODING (TC-NA)

In Fig. 4, we illustrate a sequence diagram that elaborates on the *Node's Advertisement Flooding* topology discovery algorithm. This algorithm acquires the details of the nodes and links through a topology discovery process the *Controller* initiates. In practice, the latter transmits a topology discovery control packet to the *BR*. The *BR*, in turn, is broadcasting a ''Neighbors' Discovery'' short-range beacon message advertising its location to the neighboring nodes in range. The short-range beacon message, as shown in Algorithm 1 (lines 11 – 15), includes information such as the sender's id, the *BR* node id that initiated the topology discovery process, as well as an index number defined by the *Controller* to identify each topology-discovery-run. Each receiving neighbor node creates a response message to inform the *Controller* for the existence of a link between the beacon node and itself. The total amount of these messages in one topology-discovery-run is equal to the amount of unidirectional links in the network.

Since the uncontrolled transmission of these messages could potentially flood the network, especially in dense networks, *TC-NA* utilizes avoidance mechanisms that we detail later in this subsection. The ''New Neighbor'' response message (i.e., lines 18 – 21 of Algorithm 1) contains the identifications of both nodes as well as the received signal strength and the link quality estimate. Moreover, it includes data related to its operation status, e.g., its energy level. The message is transmitted back to the *Controller* through the BR using the long-range radio link, and subsequently, the *Controller* updates the network topology graph it maintains.

Each node receiving a short-range beacon message participates in the algorithm's process by re-transmitting a similar beacon message. As such, *TC-NA* succeeds in collecting the network information in passive mode and reporting to the *Controller* new nodes and links, whenever any node advertises its existence. The repeated operation gradually detects the whole network. To avoid sending recursively short-range messages backward in the network, *TC-NA* utilizes the *Controller's* topology-discovery-run identification number (i.e., lines 8 – 9 of Algorithm 1), which is propagated through the beacon messages—this way each node sends only one beacon message per topology-discovery-run.

The *TC-NA* operation has similarities to the RPL's topology discovery algorithm [5]. The short-range beacon messages act like the RPL DIO control messages, i.e., advertising information to the neighbor nodes, whereas the long-range responses to the BR act similarly to the DAO messages that inform the sink node about the existence of new nodes, but with the difference that the long-range responses are being transmitted in on-hop.

In broadcast-based epidemic algorithms like TC-NA, the multitude of topology control message re-transmissions can cause the well known broadcast-storm problem [49]. Despite the fact that *TC-NA* uses small-sized topology discovery control messages, such phenomena are still critical for the protocol's operation and require appropriate avoidance mechanisms. To alleviate the effect of the broadcast-storm, *TC-NA* adopts two mechanisms that regulate the transmission of the topology discovery control messages:

- Each node randomly selects to wait for an equal or less duration of *maxD* time, before it propagates either a long-range response message to the BR or a next short-range broadcast message that advances the topology discovery procedure. The *maxD* value is an integer parameter representing the maximum time the nodes suspend the transmission of these messages, in hundreds of milliseconds (i.e., 1 is equal to 100 *ms*). The *maxD* is centrally configured from the *Controller* and communicated to the nodes through the short-range broadcast messages.
- To further reduce the probability of collisions, especially for dense networks, *TC-NA* utilizes a maximum transmission suppression threshold value *maxT*. Each time a node receives a neighbor's discovery message, it counts the number of control messages transmitted from other

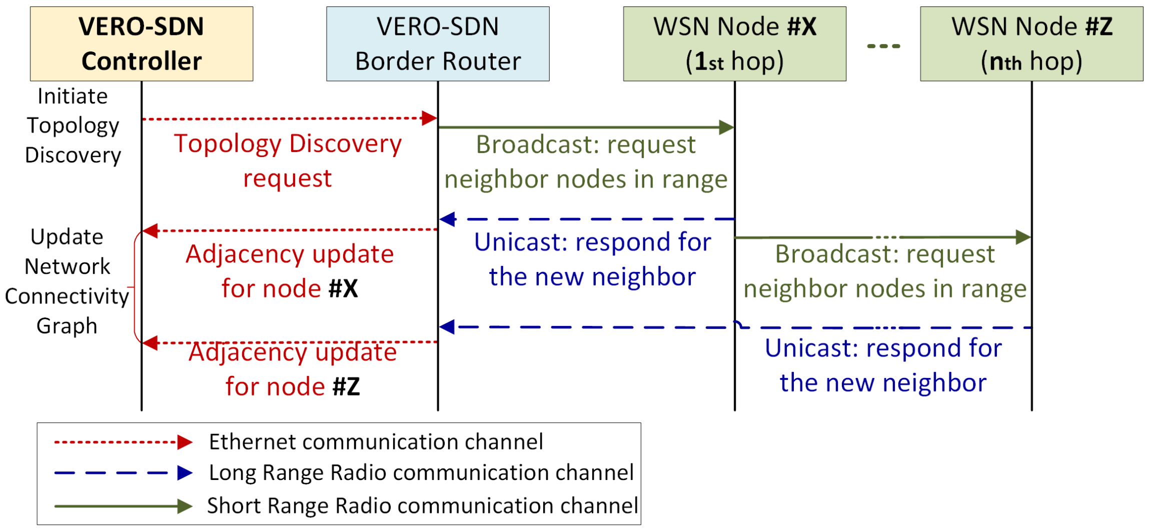

VERO-SDN Controller
VERO-SDN Border Router
WSN Node #X (1st hop)
WSN Node #Z (nth hop)
Initiate Topology Discovery
Topology Discovery request
Broadcast: request neighbor nodes in range
Update Network Connectivity Graph
Adjacency update for node #X
Unicast: respond for the new neighbor
Broadcast: request neighbor nodes in range
Adjacency update for node #Z
Unicast: respond for the new neighbor
Ethernet communication channel
Long Range Radio communication channel
Short Range Radio communication channel

FIGURE 4. Network topology discovery based on the Node's Advertisement Flooding algorithm (TC-NA).

nodes in its neighborhood, until it propagates the message. If the counter value exceeds the value of *maxT*, the algorithm assumes increased traffic and suppresses the transmission of the particular broadcast discovery message, i.e., to reduce the congestion. *maxT* has a default value 10 and is configured dynamically from the *Controller* through a specialized long-range broadcast message. This mechanism resembles RPL's DIO flooding re-transmission suppression threshold value k, in the trickle algorithm.

The *maxD* and *maxT* values can be dynamically configured through the Controller, either from the network administrator (i.e., utilizing the application plane GUI parameters option) as in the current version of *VERO-SDN*, or in a future extension utilizing intelligent algorithms that model the impact of these configuration settings towards a use-case-driven optimization of the protocol's operation. For example, in a linear network topology where the broadcast-storm effect is not intense, the *maxD* value can be substantially lower compared to a dense grid scenario, which results in improved topology discovery time. Moreover, the utilization of two channels reduces significantly, i.e., around in half, the magnitude of the broadcast-storm problem, since the nodes forward the messages through the short-range interface, but reply backwards through the long-range interface. Furthermore, the minimal message size, as well as the reduced number of control messages of *VERO-SDN* provides additional support towards the mitigation of the broadcast-storm problem.

Finally, in order to further reduce such phenomena that may also be associated with large-scale IoT deployments, our solution supports multiple BRs. In addition to the geographic extension of our solution, the BRs achieve the separation of the network in smaller control‘ segments.

In our future goals, we plan to improve the *TC-NA* algorithm towards utilizing combined control messages. For example, a node may apply short delay periods when it waits for neighbor nodes advertisements and then transmits one summarised control message.

To sum up, although *TC-NA* compared to solutions like RPL provides more flexibility (e.g., a dynamic configuration of the broadcast-storm avoidance variable), it is less adaptable compared to mechanisms like the one described next, mainly because it is used for global network discovery only. In Section V, we provide simulations highlighting the improved performance of *TC-NA* compared to RPL, with different network topology scenarios.

2) TOPOLOGY DISCOVERY WITH NODE's NEIGHBORS REQUESTS SOLICITED FROM THE CONTROLLER (TC-NR)

In Fig. 5, we depict the sequence diagram for the topology discovery algorithm based on the *Controller's* direct requests to nodes, i.e., for providing details on their neighbors. The algorithm carries out the detection of nodes and links in two phases, as shown in Algorithm 2:

1) The *Controller* requests from a *BR* node to broadcast a beacon to all nodes in range, through the long-range radio. Each time the nodes receive this solicitation message, they respond with long-range unicast messages to the *BR*, (i.e., a new node registration message, lines 3–10 of Algorithm 2).
2) The *Controller* iterates through the list of newly registered nodes and initiates the neighbor discovery process by sending a long-range control message to each one of the new nodes, i.e., through the *BR*. To avoid the congestion caused by the responses from the neighbor nodes, the *Controller* regulates their rate using a first-come-first-serve policy and applies a delay timer dt, dynamically adjusted to the long-range radio medium traffic. In detail, the *Controller* monitors the number M of responses received in periods of $p = 50\ ms$ and when the traffic increases above a threshold suppression value st, it increases the dt value by an

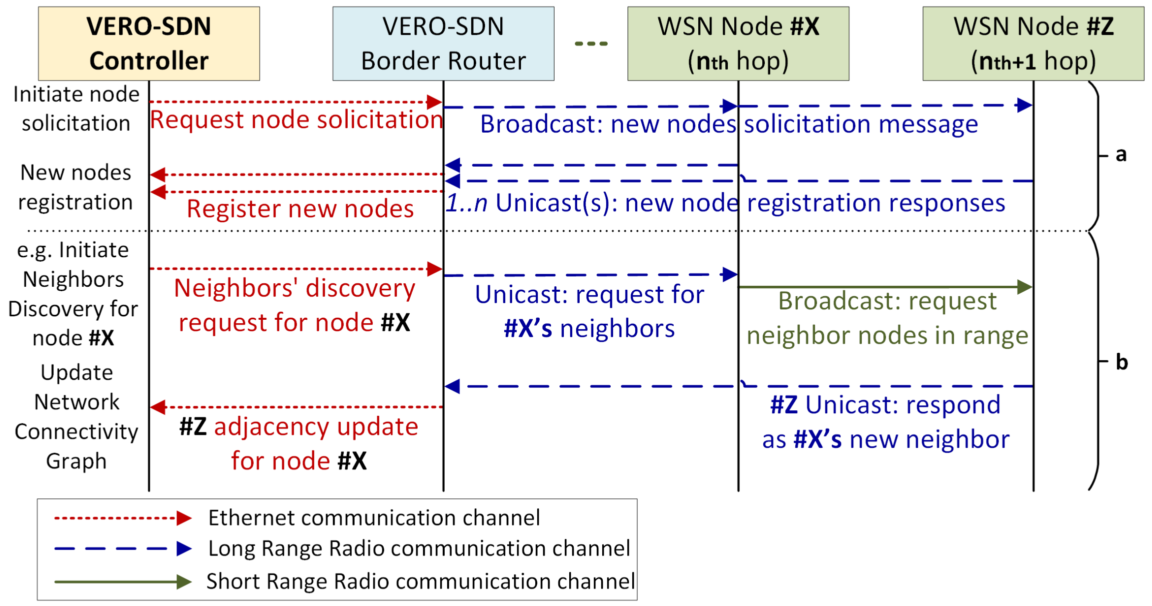

VERO-SDN Controller
VERO-SDN Border Router
WSN Node #X (nth hop)
WSN Node #Z (nth+1 hop)
Initiate node solicitation
Request node solicitation
Broadcast: new nodes solicitation message
New nodes registration
Register new nodes
1..n Unicast(s): new node registration responses
a
e.g. Initiate Neighbors Discovery for node #X
Neighbors' discovery request for node #X
Unicast: request for #X's neighbors
Broadcast: request neighbor nodes in range
Update Network Connectivity Graph
#Z adjacency update for node #X
#Z Unicast: respond as #X's new neighbor
b
Ethernet communication channel
Long Range Radio communication channel
Short Range Radio communication channel

FIGURE 5. Network topology discovery with Node's Neighbors Requests solicited from the Controller (TC-NR).

offset value. To avoid impulsive reactions, it uses an *Exponential Moving Average* (*EMA*) for smoothing and redeeming abrupt changes of the monitored data. The *EMA* is a weighted moving average filter that gives more importance to the most recent data observations. We convey the *EMA* at any given time period p_t in Equation (1), with $n = 10$ denoting the lag parameter.

$$EMA(p_t) = \frac{2}{n+1}M(p_t) + \frac{n-1}{n+1}EMA(p_{t-1}) \quad (1)$$

The *dt* timer is calculated based on Equation (2) with default configuration of *offset* $= 50$ *ms* and threshold suppression value of $st = 3$.

$$dt = \begin{cases} (dt + offset)\ if\ EMA_{(p)} > st \\ dt\ otherwise \end{cases} \quad (2)$$

The above default configuration is successfully tested through simulated scenarios with different topologies in Section V, however it can be further fine-tuned from the *Controller* to improve the topology discovery time with use-case-driven strategies (e.g., align the configuration to the node density).
Each receiving node broadcasts a beacon message to all of its neighbor nodes using the short-range radio interface (i.e., lines 11–19 of Algorithm 2). Each adjacent node that receives the beacon responds to the *Controller* with a long-range unicast packet, containing information about the identification and status of the node (i.e., lines 20–28 of Algorithm 2). The *Controller* updates the network topology graph, accordingly. To avoid a collision in the responses from all neighbors, *TC-NR* utilizes a delay timer mechanism configured from the *Controller* (lines 30–35 of Algorithm 2), likewise as the one used from *TC-NA*.

TC-NR is a centralized topology discovery algorithm collecting the network information in an active mode, i.e., through individual requests to the nodes from the *Controller*. This novel approach fits naturally with the SDN paradigm and exploits the advantages of our out-of-bound network control approach. Although it uses a higher number of control messages compared to *TC-NA*, its notable strength is its flexibility due to the novel advancements the centralized network control brings to the operation of the protocol, as shown in our evaluation results. For example, the algorithm can send targeted topology requests on specific nodes or parts of the network, as many times as needed, without overloading the rest of the network with unnecessary topology control packets (i.e., topology discovery in specific network areas that undergo frequent dynamic topology changes). The investigation of *TC-NR* in mobile environments is an important issue that deserves an independent study, which we consider as future work.

3) TOPOLOGY MAINTENANCE

The *topology maintenance* process retains the network topology representation up to date. Its main task is to have a vivid perception of the network's connectivity structure by balancing the topology discovery time interval in such a way that avoids the excess of node discovery control messages. While the topology maintenance is of paramount importance for the network QoS, its optimal operation is rather challenging, especially for networks with dynamic topologies.

Distributed protocols, like RPL, manage the frequency of topology discovery control messages, with the *trickle timer algorithm* [5]. The trickle timer, in order to reduce the control

Algorithm 1 *TC-NA* – Topology Discovery Using Node's Advertisement Flooding

```
tdID ← 0; // Initialize topology-discovery-run ID
ndi ← 0; // Initialize traffic message counter (global)
while true do
    pktR ← receive();
    if pktR.comm is Broadcast and pktR.radio is ShortR then
        if pktR.type is "ND" then // ND=Neighbors' Discovery
            ndi ← ndi + 1; // Increase traffic message counter
            if tdID not equal pktR.tdID then // New topol. discovery
                tdID ← pktR.tdID;
                // Update topology-discovery-run ID
                ndi ← 1; // Restart traffic message counter
                // Retransmit a Neighbors' Discovery beacon message
                pktB.type ← "ND";
                pktB.sender ← this.nodeAddress;
                pktB.BRaddress ← pktR.BRaddress;
                pktB.tdID ← pktR.tdID;
                pktB.retDelay ← pktR.retDelay;
                broadcast (pktB, ShortR, pktR.maxD) ;
            end
            // Respond to BR New Neighbor message
            pktS.type ← "NB"; // NB=New Neighbor
            pktS.sender ← this.nodeAddress;
            pktS.receiver ← pktR.BRaddress;
            pktS.data ← pktR.rssi&pktR.LQI&this.nodeEnergy;
            unicast (pktS, LongR, pktR.maxD) ;
        end
    end
end
Thread unicast (pktS, radio, maxD)
    sleep(rand(maxD));
    $send_{unicast}$(pktS, radio);
Thread broadcast (pktB, radio, maxD)
    sleep(rand(maxD));
    if ndi <= maxT then // maxT=Maximum Traffic Constant
        $send_{broadcast}$(pktB, radio);
    end
```

Algorithm 2 *TC-NR* – Topology Discovery With Node's Neighbors Requests Solicited From the Controller

```
while true do
    pktR ← receive();
    // Respond to BR New Node message
    if pktR.comm is Broadcast and pktR.radio is LongR then
        if pktR.type is "NN" then // NN=New Node Solicitation
            pktS.type ← "NN"; // NN=NewNode
            pktS.sender ← this.nodeAddress;
            pktS.receiver ← pktR.BRaddress;
            unicast (pktS, LongR, pktR.retDelay) ;
        end
    end
    // Transmit a Neighbors' Discovery beacon message
    if pktR.comm is Unicast and pktR.radio is LongR then
        if pktR.type is "ND" then // ND=Neighbors' Discovery
            pktB.type ← "ND";
            pktB.sender ← this.nodeAddress;
            pktB.BRaddress ← pktR.BRaddress;
            pktB.retDelay ← pktR.retDelay;
            broadcast (pktB, ShortR, 0) ;
        end
    end
    // Respond to BR New Neighbor message
    if pktR.comm is Broadcast and pktR.radio is ShortR then
        if pktR.type is "ND" then // ND=Neighbors' Discovery
            pktS.type ← "NB"; // NB=New Neighbor
            pktS.sender ← this.nodeAddress;
            pktS.receiver ← pktR.BRaddress;
            pktS.data ← pktR.rssi&pktR.LQI&this.nodeEnergy;
            unicast (pktS, LongR, pktR.retDelay) ;
        end
    end
end
Thread unicast (pktS, radio, maxDelay)
    sleep(rand(maxDelay));
    $send_{unicast}$(pktS, radio);
Thread broadcast (pktB, radio, maxDelay)
    sleep(rand(maxDelay));
    $send_{broadcast}$(pktB, radio);
```

traffic overhead, continuously decreases the frequency of sending those messages, unless neighboring nodes are not responding anymore, or when it detects inconsistencies in the protocol version numbers. The algorithm's configuration parameters (I_{min} and $I_{doubling}$) dictate the time intervals between topology discovery processes, starting from the I_{min} value up to $I_{min} \times 2^{I_{doubling}}$. Such a mechanism is oriented to fixed topologies and leads to extensive overhead during frequent network topology changes [25].

Exploiting the centralized control approach and moving towards relevant novel practices, *VERO-SDN* applies efficient topology maintenance, coordinated entirely from the *Controller*. The interval of invoking the global topology discovery processes is determined by the *Controller's* topology refresh time parameter *TRt*. The *TRt* value is an integer number representing in minutes the interval between topology-discovery-runs, and its value is configured from the administrator through the *VERO-SDN* Dashboard GUI. For static network topologies, like the ones we evaluate in Section V, the default configuration value is $TRt = 9$ *min*. We selected this configuration as the rounded mean value between the RPL's I_{min} and I_{max} values, for $I_{min} = 12$. Lower values lead to more frequent topology-discovery-runs that result in a timely representation of the network, but with an excessive number of control messages, while larger values lead to the opposite behavior.

To adapt the topology maintenance process to networks with frequent topology changes, *VERO-SDN* is utilizing interchangeably or simultaneously the topology discovery algorithms mentioned in the previous sections (i.e., imposes frequent *TC-NR* topology requests to dynamic nodes or network areas, instead of continuously flooding the network with frequent global *TC-NA* requests). This targeted approach regulates the number of control messages needed to achieve a vivid picture of the network's topology. For example, in a WSN with mobile nodes, *VERO-SDN* combines *TC-NA* and *TC-NR* algorithms. On the one hand, it triggers global network discoveries using *TC-NA* in *TRt* intervals, as in fixed networks. One the other hand, the *Controller* requests topological information only from the neighbors of the mobile nodes within these intervals, using *TC-NR*. This hybrid strategy avoids the overloading of the network with control data.

In this targeted approach, the *Controller* requires the prior knowledge of the nodes' characteristics (e.g., whether they are mobile or fixed) as well as the targeted topology refresh rate *TTRr*, expressing the number of targeted *TC-NR* requests within a *TRt* interval. In detail, the *TTRr* parameter accepts values from 1 to 10 with a default value of 5. The interval of targeted topology refresh requests, in seconds, is expressed in Equation (3).

$$TTRt = \frac{60 \cdot TRt}{TTRr} \tag{3}$$

The administrator can configure the *TTRr* value through the *Dashboard GUI* to further optimize the topology maintenance process based on various factors, e.g., the number of mobile nodes or their speed.

The evaluation and further development of the targeted topology maintenance approach is part of our future work. For example, since a necessary prerequisite is the prior knowledge of the nodes' behavior, we are currently investigating intelligent mechanisms residing at the *Controller* and being able to identify nodes with special behavior, e.g., a node that changes neighbor nodes often can be classified as mobile.

The effective operation of the *VERO-SDN* topology control mechanisms is intertwined with the robust operation of the network as subsequent operations such as the routing rely entirely on the accuracy of the representation of network connectivity. As we detail in Section V, the use of the centralized SDN approach, in combination with the utilization of the out-of-bound control channel, are the main reasons for the accuracy, flexibility, and reduced control messages of *VERO-SDN* topology control mechanisms. Although it is not within this paper's context, we point out that *VERO-SDN* provides a groundbreaking framework for ongoing research on intelligent solutions that predict or recognize the behavior of a node or a network area, (i.e., mobile nodes, or troublesome network areas), utilizing the centralized panoramic view of the network and the novel topology control algorithms.

B. NETWORK ROUTING

The network *Routing* process determines the end-to-end paths from source to destination nodes, with the requirements of achieving low delays or resource utilization, avoiding loops and deadlocks, as well as providing alternative paths. Network packets advance from one node to the next utilizing the node's flow *forwarding* control mechanism, whereas the *Forwarding or Routing table* constitutes the key instrument of this mechanism. This table contains the flow rules in tuples of *Destination* and *Next Hop* node addresses. *VERO-SDN Infrastructure plane* maintains one table in each node, which is implemented as a dynamically linked list structure. Its maximum size is configured centrally from the *Controller* based on the characteristics of the IoT environment (i.e., size of the topology and physical memory limitations in the motes).

The quality of a routing protocol is strongly related to the flow rule *expiration mechanism*. This mechanism decides which flow rules are going to be removed from the forwarding table to provide space for new rules or to allow the replacement of the former with more suitable ones. Most of the routing protocols employ a ranking parameter in the forwarding table for each flow rule. The value of this parameter is based on metrics that include the usage frequency of the rule, or a fixed *Time To Live (TTL)* value that is being periodically reduced by a time factor. *VERO-SDN flow rule expiration mechanism* is handled entirely by the *Controller*, using the last three *southbound API* commands outlined in Table 1. By moving the intelligence of this mechanism to the *Controller*, we elevate its flexibility and quality of decision making, as its operation is blended with other network processes, like the topology maintenance tuning the control message overhead trade-off.

The *flow establishment process* represents the mechanism that constructs and maintains the *forwarding table*. The routing protocols can be classified into *Reactive*, *Proactive*, and *Predictive*, depending on the adopted flow establishment operation approach:

- The *Reactive* routing establishes forwarding rules using a route discovery process that locates an available path between two nodes (i.e., the Lightweight On-demand Ad-hoc Distance-vector routing protocol LOADng [50]). This process is activated when a node attempts to transmit data packets to an unknown destination. The main drawback of the reactive routing is the potentially increased time to establish a new route.
- The *Proactive* flow establishment proposals usually build a tree of connected nodes considering one node as the root node, i.e., the sink node. For example, RPL is a proactive protocol that builds DODAG using distance vectors. Although DODAG offer efficient routing paths from any node to the sink node, they fail to create efficient node-to-node paths from any to any other node,

especially when these nodes are at the network edge. Moreover, it establishes and maintains routes that may not be used for long periods.

- The *Predictive* routing characterizes emerging flow establishment techniques that attempt to reduce the negative effects of the previous two approaches. They are utilizing intelligent decision-making modules (i.e., neural networks [51]) that predict communication requirements among nodes and set up proactively the routing paths. However, these techniques require substantial processing power and data monitoring from multiple sources. To this end, we argue that the SDN approach is an enabling technology for *Predictive* routing, since it offloads the network intelligence to the infrastructure network, i.e., providing excessive computational power, while maintaining a global view of the network.

Although *VERO-SDN*, due to its inherent flexibility derived from the SDN paradigm, can potentially adopt any variation of the above techniques, we introduce two flow establishment methods: (i) one *reactive* that is aligned to the OpenFlow protocol; and (ii) one hybrid that combines *reactive* and *proactive* characteristics. For example, when a node attempts to send a message to a destination address that does not exist in its forwarding table, it transmits a *miss of forwarding flow rule* request to the *Controller* through the southbound API, as defined in Table 1. The *Controller* selects the best path by making use of the Dijkstra algorithm applied on the *Global Network Structure* connectivity graph, considering as link weights a number of parameters collected during the topology discovery process. These parameters include the signal strength, the link quality, the number of hops, or the node's energy. The *Controller* responds to the requesting node with a flow rule establishment control message. To this regard, *VERO-SDN* protocol provides two methods for flow rule establishment:

1) The *Next-hop only* flow rule establishment method instructs the *Controller* to inform the node with its own forwarding flow rule only, i.e., to reach just the next node. In case the forwarding table of the next node misses the forwarding rule as well, the process is repeated.
2) The *Complete Path* flow rule establishment method operates similarly to the previous one when a node issues a route miss request, but its response differs. The *Controller*, after the selection of the best path, proactively informs all the intermediate nodes participating in the routing path. This way, the *Controller* maintains absolute control over the entire route, with the drawback of sending more control packets. This method is crucial in network installations that require traffic prioritization communication routes. For example, an IoT network in a harsh working environment may require setting up specific priority flows, e.g., for the prevention of an accident.

A combination of the above two is also possible, depending on the context.

VERO-SDN manages to operate the above mechanisms successfully based mainly on the stability through direct communication provided by the out-of-bound control channel between the *Controller* and the network nodes. In addition, the *Controller's* intelligence in establishing efficient routing paths in every direction of the network, achieving the shortest routes and taking into account the specificities of the WSN environment makes *VERO-SDN* capable of operating a variety of applications utilizing any communication model (i.e., many-to-one, one-to-one, or one-to-many).

V. EVALUATION

In this Section, we provide our extensive evaluation and analysis of results that highlights the performance advantages of the *VERO-SDN* platform and its corresponding network control mechanisms. Our main goal is to provide an SDN solution for IoTs that covers many cases of IoT deployments, beyond those adopting the traditional many-to-one communication model of WSN. As a point of reference for our comparisons, we selected RPL, the de-facto WSN routing protocol, for two main reasons: (i) to investigate the advantages of the centralized against the de-centralized approach; and (ii) to retain a common denominator in order to compare our work with any other, now and in the future, since the majority of research papers in SDWSN provide comparisons with RPL.

Our evaluation scenarios probe into the two main network operation processes (i.e., topology control and routing) through two sets of simulations that consider a wide range of network conditions in terms of topology arrangements and sizes, i.e., discussed below in the two respective subsections V-A and V-B. Each subsection includes the corresponding evaluation methodology, setup and results.

A. TOPOLOGY CONTROL SIMULATIONS

The duration time for the discovery of the entire network topology, as the main aspect of topology control, is a critical figure in routing protocols. It is strongly related to the overall network performance because the topology discovery process is repeated regularly from the topology maintenance schema. In our evaluation plans, we include both *VERO-SDN* topology discovery processes, the *TC-NA* and *TC-NR*, and compare them against RPL's discovery mechanisms. We aim to gather results through simulations for each one of them within different network terrains.

1) EVALUATION METHODOLOGY

To assess *VERO-SDN* topology discovery efficiency, we envisage three representative network topology scenarios associated to real-life IoT deployments, namely a *Linear* (Fig. 6a), a *Rectangular Grid* (Fig. 6b), and a *Triangular Grid* (Fig. 6c). To designate theoretically the above scenarios, we consider three respective undirected connected graphs $G_1(V_1, E_1)$, $G_2(V_2, E_2)$, $G_3(V_3, E_3)$, i.e., expressing the three

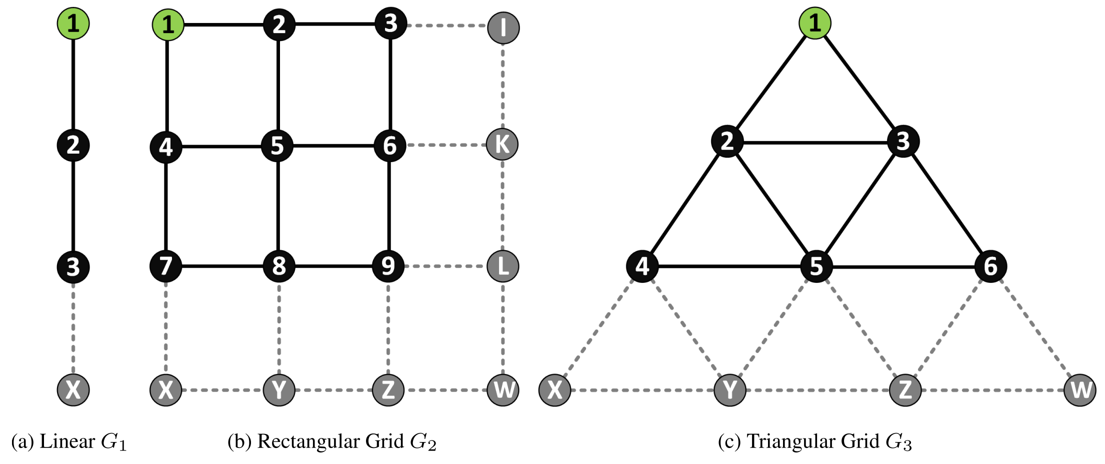


FIGURE 6. Three scenarios of network topology connectivity graphs. The *green* vertex indicates the border router, the *black* vertices represent the regular network nodes, and the *gray* vertices are the potential node expansions for any network size. The graph edges stand for connectivity links.

TABLE 2. Graph properties per topology type.

				$D(G_x)$	
x	$\Delta(G_x)$	$\delta(G_x)$	$\lambda(G_x)$	$V = 30$	$V = 90$
1	2	1	1	29	89
2	4	2	2	9	17
3	6	2	2	7	12

networks connectivity patterns, where V is the number of graph's vertices representing the network nodes, and E is the numbers of graph's edges representing the networks communication links.

In Table 2, we outline the graph properties related to our scenarios, including the *maximum* $\Delta(G)$ *and minimum* $\delta(G)$ *degrees*, defined as the maximum and minimum number of edges incident to its vertices, and the *edge connectivity* $\lambda(G)$, that manifests the size of the smallest edge cut that will disconnect any of the G_1, G_2, G_3 graphs. According to the graph theory, our graphs are *maximally connected* and the network topology complexity is equal to the maximum degree $\Delta(G)$, because $\lambda(G) = \delta(G)$. The *distance* $d(v, u)$ between two vertices v, u of a graph G is the length of the shortest path between those vertices. The *eccentricity* $e(v)$ of vertex v is the maximum distance from v to any other vertex $V(G)$, defined as $e(v) = max\{d(v, u), u \in V(G)\}$. Furthermore, the *diameter* $D(G)$ of graph G is the maximum eccentricity value among the vertices of G, defined as $D(G) = max\{e(v), v \in V(G)\}$. The *diameter* is an important indication in our scenarios, because it represents the longest distance path in the network.

We investigate the following requirements based on our three scenarios:

- The *Linear graph* G_1 represents a low complexity scenario where the maximum connectivity degree is the lowest, i.e., equal to two. However, it is an exemplary scenario because it resembles real-life IoT applications (e.g., smart streets). The main challenge of the scenario is the quality and accuracy of neighbor detection because each node has one chance to detect a neighbor. A failure to detect a neighbor at any point will lead to a disconnected network because $\lambda(G_1) = 1$.
- The *Rectangular Grid graph* G_2 is a moderate scenario in terms of connectivity density compared to the other two, with maximum complexity equal to four ($\Delta(G_2) = 4$). It also represents real-life IoT connectivity scenarios related to monitoring and surveillance applications.
- The *Triangular Grid graph* G_3 offers a dense connectivity environment, where the inner nodes have a maximum complexity of six ($\Delta(G_3) = 6$). In this case, we investigate the behavior of our protocol under an intensive operation due to the multitude of communication links.

To get an insight into the protocols' performance for different network sizes, we nominate one small and one large scale scenario in terms of the number of nodes, with $V = 30$ and $V = 90$, respectively. For all simulations, we are using a radio environment with data loss only related to distance factors, i.e., without external interference. We apply this deterministic methodology because we focus on comparing the effectiveness of our platform and algorithms at the architectural level. Hence, there is no need to confirm the results' statistical accuracy.

The selected network topologies for our evaluation are regularly-shaped graphs in order to enhance our ability to:

- justify our findings utilizing the equivalent theoretical graph characteristics and draw conclusions that can be further used as patterns for the improvement and optimization of our mechanisms per topological structure.
- compare the results on the behavior of the proposed mechanisms in the three scenarios to each other, which process is simpler and clearer with deterministic data.

2) SIMULATION SETUP

In our simulations, we use the *Cooja* [52] simulator with emulated Zolertia Z1 IoT devices. Cooja is a Linux based cross-layer WSN simulator for Contiki OS, which enables the

TABLE 3. The simulation setup of RPL.

Network Layer	Settings	Notes
Transport	UDP	Packet size 128 B
Network	RPL/IPv6	
MAC	CSMA	
Physical	IEEE 802.15.4	
Radio Interface	TX/RX 100%	Transmission Range 50 m
OS	Contiki-OS [43]	ver 3.0

TABLE 4. The simulation setup of *VERO-SDN* data-plane.

Network Layer	Settings	Notes
Transport	UDP	Packet size 128 B
Network	VERO-SDN	Forwarding
MAC	CSMA	
Physical	IEEE 802.15.4	
Radio Interface 1	$SubGHz$, TX/RX 100%	Long Range 700 m
Radio Interface 2	2.4 GHz, TX/RX 100%	Short Range 50 m
OS	Contiki-OS [43]	ver 3.0

creation of virtual WSN scenarios. In Tables 3 and 4, we enlist the setup parameters for both RPL and *VERO-SDN* protocols, respectively.

For both topology discovery mechanisms, we select their default configuration values. We set RPL mode to *storing-mode* and for the trickle timer we keep the default configuration values, as implemented in Contiki OS v3.0, which are: $I_{min} = 12$, $I_{doublings} = 8$ and the redundancy constant $k = 10$. In the case of *VERO-SDN*, the default values of the parameters guiding the intervals of topology discovery for both algorithms are: $maxD = 3$, and $maxT = 10$. These parameters are detailed in subsection IV-A.

For both simulated networks, we set up the radio communication environment quality at the maximum (i.e., TX/RX 100%). Although this configuration is not attainable in real WSN applications, we intentionally consider a radio environment with no signal issues in our evaluation methodology, since we focus our study on processes and algorithms of the network layer. This approach provides a clearer and easier comparison of our simulation results. However, it is important to evaluate the impact of radio interference and other dynamic network conditions (e.g., mobility) on our solution, but such analysis is complex enough to deserve an independent study.

3) PERFORMANCE METRICS

We carry out our simulations and analyze our results using the following two metrics:

- *Topology Discovery Duration (TDD) Time*: The *TDD* time represents the total duration of time in seconds required to collect and construct the connectivity graph for the entire network.
- *Topology Discovery Control (TDC) Overhead*: The *TDC* overhead represents the total number of control messages exchanged among the motes in order to construct the connectivity graph for the complete network.

For both metrics, we target at their lower values, since we desire to form the network connectivity graphs in a short time and with a minimum communication overhead. Since *VERO-SDN* and RPL follow a different approach for the topology construction, i.e., centralized vs distributed, we devised a particular methodology to measure the TDD and TDC metrics. In the case of *VERO-SDN*, we are using the network monitoring data collected from the *Controller* for the network structure construction process. We count as starting time of the topology discovery the first broadcast message from the BR and the new node solicitation broadcast message for *TC-NA* and TC-NR, respectively. As finishing time, we consider the appearance of the last node in the network connectivity graph. For RPL, we are using *Foren6* [53], an external 6LoWPAN network analysis tool that processes the Cooja radio log output and reconstructs a visual and textual representation of the network connectivity graph. As starting time, we consider the first DIO transmission from the sink node and as finishing time the appearance of the last DAO message informing the sink for the last discovered node in the network.

We carry out 15 simulation runs for each scenario and calculate the average values for both metrics. For clarity purposes, we omit the standard deviation values in the figures, since they are insignificant in all scenarios, in contrast to the performance differences we observed.

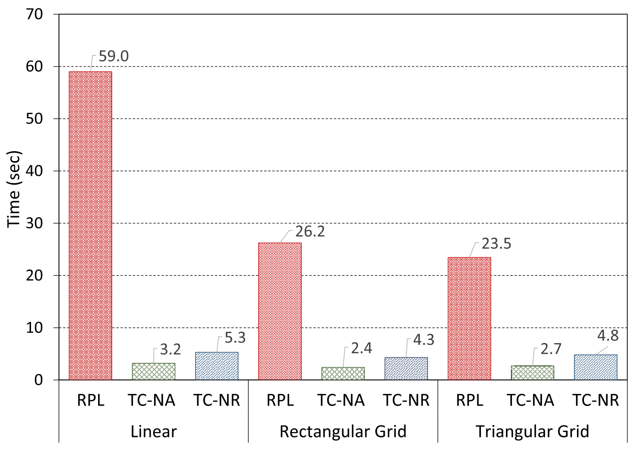
4) EVALUATION RESULTS AND DISCUSSION

In Fig. 7, we illustrate the network topology discovery evaluation results for all the three topology scenarios. We depict our results using bar charts that exhibit each algorithm's performance based on the aforementioned metrics, i.e., *TDD* time and *TDC* overhead, classified per topology scenario for both networks of 30 (i.e., Fig. 7a and Fig. 7b) and 90 nodes (i.e., Fig. 7c and Fig. 7d).

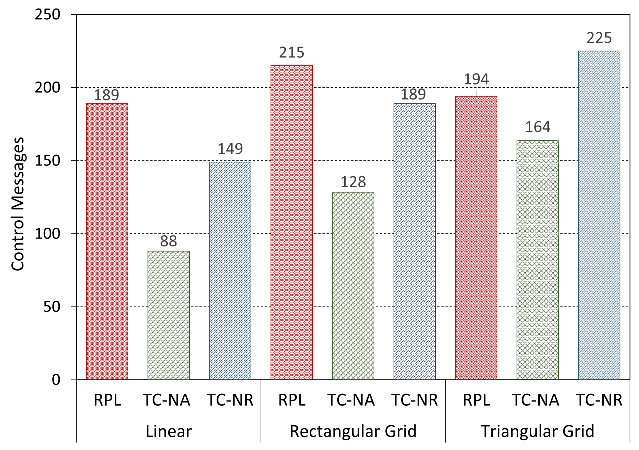
As an initial observation, we underline that *VERO-SDN* significantly outperforms RPL's topology discovery performance both in terms of *TDD* time and the number of *TDC* messages. *TC-NA* algorithm delivers the lowest *TDD* time results in all circumstances while using the lower amount of control messages. *TC-NR* algorithm, although it performs much better than RPL, it is considerably slower compared to *TC-NA*, especially in large topologies.

We analyze our results in detail by comparing in pairs the performance of the algorithms, for each topology scenario:

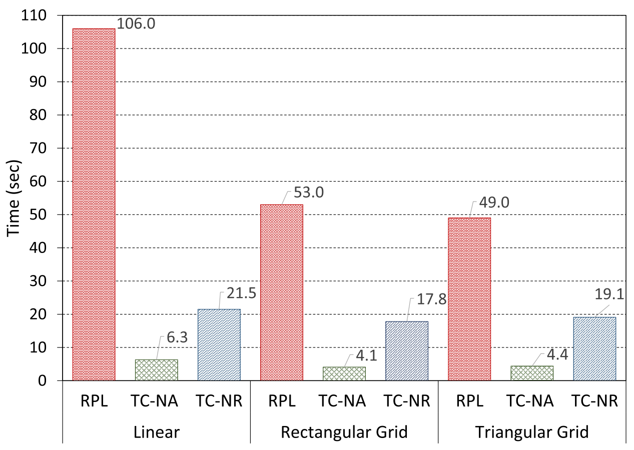
- *RPL vs. TC-NA*: In the Linear topology simulation with 30 nodes, we observe that *TC-NA* is 18 times faster than RPL in respect of *TDD* time, while using lower than the half of control messages (i.e., the 53%). *TC-NA* achieved similar results in the case of the 90 nodes topology, but with even fewer control messages (i.e., the 65%). For the 30 nodes network in the Rectangular and Triangular Grid topologies, the *TC-NA* is 11 and 9 times faster than RPL, respectively, and for the 90 nodes 13 and 11 times faster as well. Although the *TC-NA* maintains similar performance in all topology scenarios, the difference with RPL is reduced compared to the Linear topology because RPL performs better in Grid network topologies due to their reduced DODAG



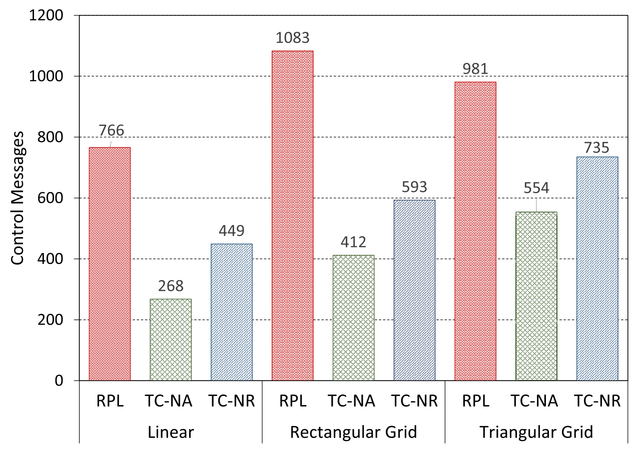
(a) Total discovery duration time for networks of 30 nodes



(b) Control messages overhead for networks of 30 nodes



(c) Total discovery duration time for networks of 90 nodes



(d) Control messages overhead for networks of 90 nodes

FIGURE 7. Network discovery evaluation results for RPL, *VERO-SDN TC-NA* and *TC-NR* with linear, rectangular and triangular grid network topologies.

depth. Although the *TC-NA* retains fewer control messages compared to RPL, we observe that the difference is reduced to 40% and 15% for the Grid topologies (Fig. 7b). Since the *TC-NA* informs the *Controller* for all available links among the nodes, the highest complexity in terms of adjacent nodes leads to increased numbers of control messages. It is interesting to look at the 90 nodes network (Fig. 7d) where we observe that the difference between *TC-NA* and RPL, in terms of control messages, is significantly higher compared to the 30 nodes network, i.e., from 40% to 62% and from 15% to 44% for the two Grid topologies, respectively. That is an expected outcome, since the *TC-NA* control messages increase linearly with respect to the topology size due to the one-hop transmission. In contrast, RPL control messages depend on the size of the network paths.

- *RPL vs. TC-NR*: Comparing the *TC-NR* topology discovery performance against RPL's, we realize directly proportional figures with the ones discussed for *TC-NA*. Although the performance of *TC-NR* is not as good as *TC-NA's*, it still outperforms RPL's results. In Fig. 7a for the 30 nodes *TDD* time we observe improvements in the order of 11, 6 and 5 times faster performance, which for the 90 nodes in Fig. 7c, becomes 5, 3 and 3 for each of the three topology scenarios, respectively. Notable is that although the control messages of *TC-NR* for the 30 nodes scenario are close to RPL's, this is not the case for the 90 nodes simulations for the same reasons with *TC-NA*. The increased amount of control messages in *TC-NR* is the result of its architectural design and operation that becomes more important as the topology complexity increases.
- *TC-NR vs. TC-NA*: Initially, we have to acknowledge that both algorithms achieve the implementation of the topology discovery process. On the one hand, the *TC-NA* algorithm succeeds in collecting the network information in a passive mode, i.e., by reporting to the *Controller* the nodes that advertise their existence. On the other

hand, the *TC-NR* collects the complete network information in an active mode, as it triggers the nodes to request a response from their neighbors. The latter is an important architectural feature since it allows targeted topology discovery requests. However, comparing the *TDD* time results in Fig. 7a between the two topology discovery approaches, we observe that regardless of the topology in the 30 nodes scenario, the *TC-NA* is approximately twice as fast from the *TC-NR*, while in the 90 nodes network, (Fig. 7c), the difference increases to quadruple figures. We conclude that *TC-NR* is more affected by the size of the network than *TC-NA*. The above conclusion is justified by the number of tasks each algorithm executes in relation to the packets sent. *TC-NR* algorithm demonstrates a higher number of executed tasks, and consequently, we argue that the *TC-NR* algorithm produces inferior time performance results compared to *TC-NA*, especially during the topology construction phase.

Generally, we conclude that the main reasons for the enhanced topology discovery performance of the *VERO-SDN* are its architectural characteristics that the SDN paradigm enables in combination with the employment of a separate control channel. For *VERO-SDN*, the *Controller* manages the transmission of control messages in the network based on parameters that maintain linear characteristics, i.e., due to the one-hop transmissions of the former. On the contrary, RPL increases the I_{min} parameter exponentially to avoid the instant flooding of the network with control messages, while the response messages from every node to the sink node (i.e., the DAO messages) use the same multi-hop medium. Consequently, they are overloading the network with control messages and delaying its *TDD* time.

Furthermore, a general conclusion drawn from the simulations is that RPL's discovery performance depends on the network's topology structure and size, while the *VERO-SDN* algorithms do not. That occurs because RPL discovery process is firmly bonded to the depth of the DODAG tree, as we observe an analogous change in the *TDD* time performance with the graph's diameter $d(G)$ property shown in Table 2. On the contrary, for each *VERO-SDN* algorithm, we observe similar *TDD* time results regardless of the topologies used. We argue that this feature can be beneficial in networks that require consistent performance, independent of the topology environment (i.e., networks in industrial or hazardous environments) and also underlines the general applicability of our proposal.

To sum up, in this subsection we demonstrated through simulations that *VERO-SDN* implements successfully and efficiently the topology discovery process based on novel architectural features that combine the support of a separate control channel with the well-fine-tuned network coordination from the *Controller*. Furthermore, we highlighted the applicability of *VERO-SDN* to a wide range of IoT scenarios, since the performance of its topology discovery mechanisms, in terms of discovery time and control overhead, does not depend on the topology structure.

B. ROUTING AND FLOW ESTABLISHMENT SIMULATIONS

The network's flow management and the associated packet delivery times are essential routing protocol factors, which are tightly bonded with the network's QoS and the overall performance. In this subsection, we evaluate the centralized routing operation and performance of *VERO-SDN* and compare it against the distributed approach of RPL. In our evaluation plans, we include the two *VERO-SDN* flow establishment processes: (i) the *Next-hop-only (FE-NH)*, where the *Controller* responds to a miss-table request with one flow rule to the requesting node only; and (ii) the *Complete-path (FE-CP)*, where the *Controller* establishes flow rules to all subsequent nodes that participate in the particular flow's path. We now detail, for this second set of simulations, the evaluation methodology, setup and discuss the corresponding results.

1) EVALUATION METHODOLOGY

In our evaluation, we use a network of 15 stations, arranged in a triangular grid topology with the *BR* placed at the top of the triangle, as shown in Fig. 8. We demonstrate the network connectivity using two types of lines: (i) the *solid black lines* depicting the established connections of the RPL protocol, i.e., illustrating the DODAG; and (ii) the *dashed gray lines* representing other possible wireless connectivity links due to nodes proximity. The triangular graph topology is ideal for providing our simulations with an abundance of different communication paths among the network nodes in order to evaluate the quality of routing path selection and flow establishment. In terms of network operation, we choose a scenario where all nodes send unicast messages to all other nodes. With this approach, we confirm that our proposal achieves very good results in scenarios beyond the traditional many-to-one communication paradigm of WSN. Furthermore, this overall communication exercise with a high multitude of messages confirms the suitability of *VERO-SDN* in many different IoT deployment cases, including communication-demanding environments.

To analyze our simulation theoretically, we consider the network as a triangular graph $G(V, E)$ with $V = 15$ vertices representing the network nodes and $E = 18$ edges representing the radio links. Here, we use the graph theory's concept of distance matrices. A *Distance Matrix M(G)* of a graph $G(V, E)$ is a two-dimensional symmetric matrix $V \times V$ that contains the distances between each pair of vectors. We calculate the *distance* $d(v, u)$ between v and u by counting the number of edges in the shortest path. We define three metrics based on the distance matrix: (i) the *Total distance* representing the summation of all shortest path distances among all nodes, calculated as the addition of all $M(G)$ distances; (ii) the *Average distance* denoting the average shortest path distance, calculated by averaging all $M(G)$ elements; and (iii) the *Max distance* expressing the number of edges of the longest distance path in G.

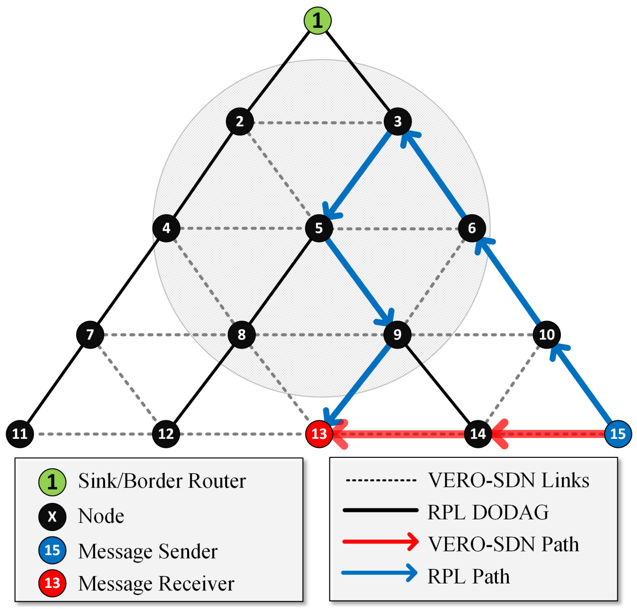


FIGURE 8. Routing evaluation scenario with a triangular grid network topology.

TABLE 5. Graph *distance matrix values* for triangular grid networks of 6, 10 and 15 nodes per protocol.

Graph Vertices	Protocols	Total distance	Average distance	Max distance	% Difference
6	RPL DODAG	55	1.83	4	24%
	VERO-SDN	42	1.40	2	
10	RPL DODAG	267	2.97	6	39%
	VERO-SDN	162	1.80	3	
15	RPL DODAG	807	3.84	8	43%
	VERO-SDN	462	2.20	4	

Table 5 enlists the values of the above three metrics obtained from the Distance Matrices of three triangular graphs. One of 15 nodes, like in Fig. 8 and two sub-graphs of 10 and 6 nodes, respectively. These graphs represent the routing graphs of the two protocols, i.e., RPL and *VERO-SDN*. In the last column, we calculate the *total distance* percentage difference for each pair of protocols, in order to evaluate on a theoretical basis the quality of the paths implemented by each protocol. Since the lower *total distance* numbers indicate shorter paths, we conclude that *VERO-SDN* implements better paths than RPL. We also annotate that the associated performance difference increases with the graph's size. This happens because *VERO-SDN* establishes flows by considering all possible connectivity options, while RPL is using the DODAG's connectivity links only.

2) SIMULATION SETUP

To justify the above theoretical insights, we create the network shown in Fig. 8 and conduct simulations based on the same simulation setup environment described in subsection V-A2. The protocol parameters for RPL and *VERO-SDN* protocols are recorded in Tables 3 and 4, respectively.

Our evaluation plan includes two simulated scenarios:

- *all-to-all*: An intensive data packet traffic scenario that exhibits the one-to-one communication pattern, where each mote transmits in total 350 60-byte unicast messages for 100 minutes to all other motes in the network (i.e., each node transmits 25 messages to each other node), which is equivalent to a rate of 1 data-packet every 17 seconds. The network conveys in total 5, 250 data-packets.
- *many-to-one*: A typical sensor monitoring and data collection scenario where each of the 14 nodes transmits 200 data-packets of 60 bytes to the sink (node-1), within 100 *min*, i.e., with a rate of 1 data-packet every 30 seconds. The network conveys 2, 800 data-packets in total.

3) PERFORMANCE METRICS

We analyze and evaluate our simulations using the following metrics:

1) *Packet Delivery Ratio (PDR)*: This metric measures the protocol's quality in terms of message delivery success ratio. The *PDR* is calculated as the ratio of the received messages (R_x) over the sent messages (T_x) transferred among the motes (i.e., in (4)).

$$PDR = \frac{\sum R_x}{\sum T_x} \tag{4}$$

2) *Total End-to-End Delay (TEED)*: The *End-to-end delay* is the time needed for a packet to be transmitted across a network from the source to the destination node, i.e., the One-way delay. The End-to-end delay [54] includes the: (i) *transmission delay*, the packet transmission time into the transmission medium; (ii) *propagation delay*, the signal traveling time over the distance; and (iii) *packet processing delay*, the time the packet is being processed at a network device. In the SDN paradigm, the End-to-end delay also includes the delays due to the flow establishment, which are the: (i) *table-miss flow delay*, the table-miss request transmission time to the *Controller*; and the (ii) *flow-rule establishment*, the flow rule response time from the *Controller*. These additional details occur mainly at the beginning of each flow communication and should be balanced from the most informed and accurate routing decisions due to the adoption of the SDN paradigm.
Considering the End-to-end delay d as the distance between a pair of network nodes and in analogy with the above mentioned Distance Matrix, we define the *End-to-end Delay Distance Matrix (EED)*. For a network with N nodes the *EED* is an $N \times N$ symmetric matrix with elements the End-to-end delay times $d(i,j), \forall i,j \in N$ among all network nodes, as in (5).

$$\boldsymbol{EED}_{(N,N)} = \begin{bmatrix} 0 & d_{1,2} & d_{1,3} & \cdots \\ d_{2,1} & 0 & d_{2,3} & \cdots \\ d_{3,1} & d_{3,2} & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{5}$$

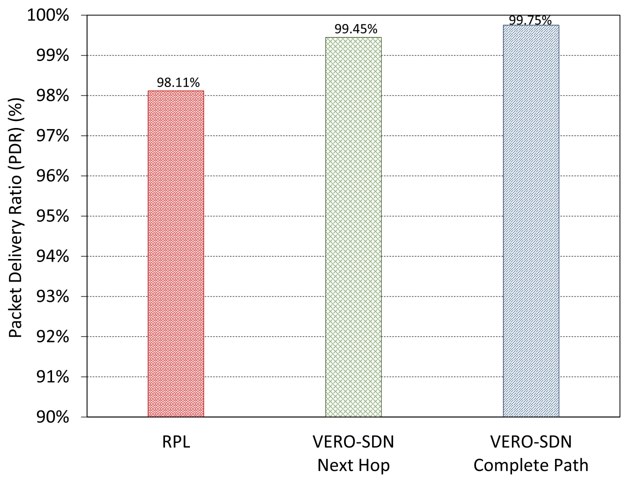


FIGURE 9. The average *PDR* per protocol.

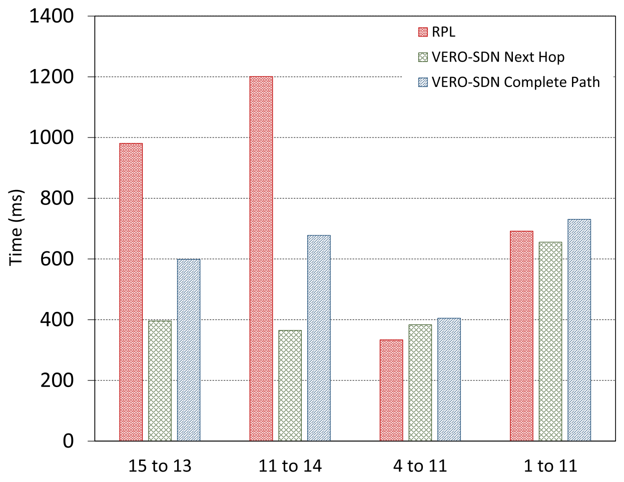


FIGURE 10. Total End-to-End Delay (TEED) time for selected paths from node *X* to node *Y*.

Since it is not reliable to draw conclusions from one *EED* sample, we define an aggregate metric, i.e., the *Total End-to-End Delay (TEED)* matrix, as the summation of *EED* matrices over the course of time for an M number of messages transmitted in the network. We determine the *TEED* metric formula with (6).

$$\boldsymbol{TEED}_{(N,N)} = \sum_{m=1}^{M} \boldsymbol{EED}_m \tag{6}$$

3) *Network Overall End-to-end Delay (NOD)*: To compare the overall network performance of the *VERO-SDN* routing mechanisms with RPL, we define the *NOD* as the total time of all the delay times required to send packets from each node to all other nodes in the network. In (7), we depict the *NOD* metric formula for a network of N nodes. Undelivered packets are excluded from the calculations.

$$NOD = \sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{TEED}_{(i,j)} \tag{7}$$

4) EVALUATION RESULTS OF *all-to-all* SCENARIO

In Fig. 9, we depict *VERO-SDN* performance in terms of *PDR*. Although our simulation environment does not apply external radio interference, the *PDR* does not reach the 100% because the multitude of messages causes radio collisions and packet drops due to the extended routing delays. Nevertheless, we observe that *VERO-SDN* with the *complete-path* flow establishment reaches up to 99.75% packet delivery accuracy, which is by 1.64% higher compared to RPL. *VERO-SDN* next-hop process also achieves 99.45% *PDR*, ascertaining the high reliability and integrity of the *VERO-SDN* protocol. Since in our simulations we consider a reliable radio channel, the improved PDR performance of *VERO-SDN* reflects its efficient communication paths establishment among the network nodes, due to its novel routing and forwarding processes, further justified later in this subsection.

To analyze *VERO-SDN* performance in terms of network communication delays we expedite the evaluation results in three stages: (i) initially, we focus on communication examples of specific pairs of nodes; (ii) then we analyze the communication behavior for each node to all other nodes separately; and (iii) finally, we present the overall picture of network performance to draw general conclusions.

In the first stage, we selectively present the *TEED* evaluation results of four paths between particular pairs of nodes (i.e., as swhon in Fig. 10):

- $TEED_{(15,13)}$ (i.e., from the node 15 to 13, designated as $15 \rightarrow 13$) presents a clear superiority of both *VERO-SDN's* flow establishment processes against RPL with 60% and 40% improvements, respectively. To justify these results, we demonstrate in Fig. 8 the flows that each protocol selects. The dark blue arrows illustrate RPL's flow through the established DODAG, while the thick red arrows represent the *VERO-SDN* path. The proof is evident as RPL needs 6 hops to reach node 13 while *VERO-SDN* selects the shortest path through the neighbor node 14 and delivers the message in 2 hops. We also see that RPL fits naturally to the many-to-one communication model, while *VERO-SDN* can efficiently support IoT applications requiring node-to-node communication.
- $TEED_{(11,14)}$ results in the same pattern with the previous example, which manifests the eminence of our protocol with improved numbers that reach up to 70% and 44% for the next-hop and complete-path flow establishment processes, respectively. These results reinforce the position that *VERO-SDN* selects the shortest communication path.

Both $15 \rightarrow 13$ and $11 \rightarrow 14$ paths are specially selected as challenging communication cases for RPL, as discussed above, that also demonstrate the improved performance and flexibility of *VERO-SDN* over the proactive routing protocols. The next two examples consist of

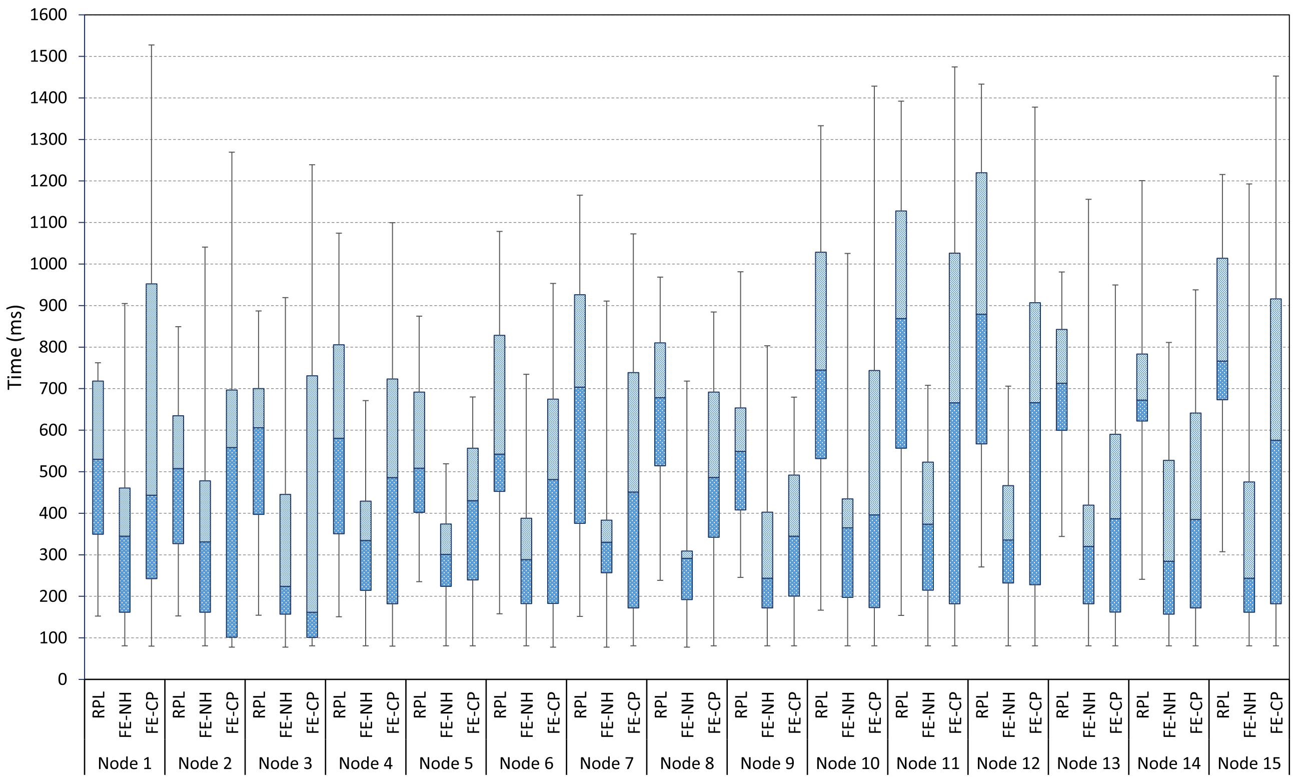


FIGURE 11. The range of ***TEED*** values from each node to all other network nodes.

cases, where the communication paths belong to the established DODAG.

- $TEED_{(4,11)}$ illustrates a case where *VERO-SDN* routing for both flow establishment processes exhibits slightly deteriorating performance, since RPL achieved better results in terms of *TEED*, i.e., 15% and 21%, respectively. This is an expected outcome since both nodes 4 and 11 belong to the same DODAG branch, and as a result, both RPL and *VERO-SDN* select the same 2 hops flow. RPL outruns *VERO-SDN* because the latter includes the flow establishment delay.
- $TEED_{(1,11)}$ represents a similar case with the previous one, where both protocols use the same path, with the difference that $1 \rightarrow 11$ is a four hops path. We observe that these results are very close, with *VERO-SDN next-hop* performing 5% better than RPL and the RPL 6% better than *VERO-SDN complete-path* process. Although *VERO-SDN* maintains the additional delay of the flow establishment, the reason that RPL does not overrun *VERO-SDN*, like in the previous case, is that the forwarding processes in *VERO-SDN* are faster than RPL and as more hops intervene this becomes more obvious in the simulation outcomes.

In the second stage, we further detail our evaluation results through presenting in Fig. 11 a box and whisker plot that demonstrates the *TEED* performance of each node to all other nodes, for all three protocols. This chart illustrates the distribution of the results from minimum to maximum values and compares the concentration of measurements around their mean values. This visual exercise assists us to draw broad conclusions.

Comparing the mean values and the majority of measurements that concentrate around them (i.e., the light and dark blue boxes), we observe that *VERO-SDN* next-hop process produces much better *TEED* for all cases of nodes compared to RPL. It is noteworthy that the lower we go into network topology, the worse the results for RPL (i.e., node 11, 12, 13, 14, and 15). That converges with the earlier mentioned performance results of the $15 \rightarrow 13$ path. However, in many cases (i.e., node 1, 2 and 3), we notice that the highest *TEED* values of *VERO-SDN* are higher than RPL's, due to the initial flow establishment delay time. Evaluating *VERO-SDN's* complete-path performance, we identify in a number of cases (i.e., node 1, 2, 3, 4, 10, 11, 12, and 15) that it produces a considerable spread between very high to low *TEED* values. The main reason is the slower flow rule establishment procedure that causes high *TEED* values.

The network overall end-to-end delay NOD time can be seen in Fig. 12. We observe that *VERO-SDN* routing with next-hop flow establishment mechanism achieves to deliver the full network data workload in a total duration of 73, 430 *ms*, reduced by 47% compared to RPL's performance. Similarly, *VERO-SDN* complete-path performance achieves a reduction of 24% compared to RPL. In Fig. 13, we observe a summary per-protocol of the *TEED* values in quartiles through the box diagram. From this chart, we observe that

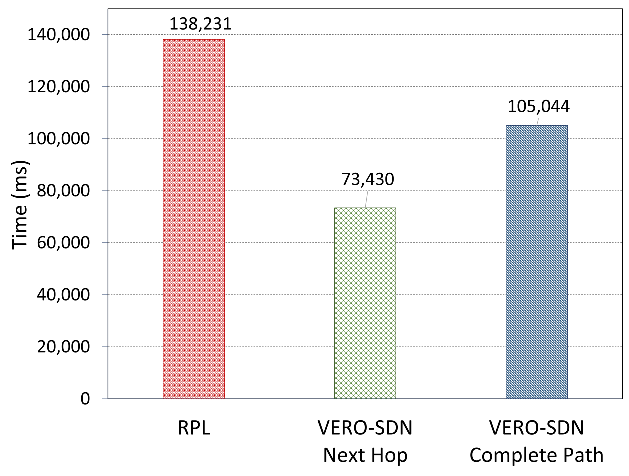


FIGURE 12. **The Network Overall End-to-end Delay (NOD), *all-to-all* scenario.**

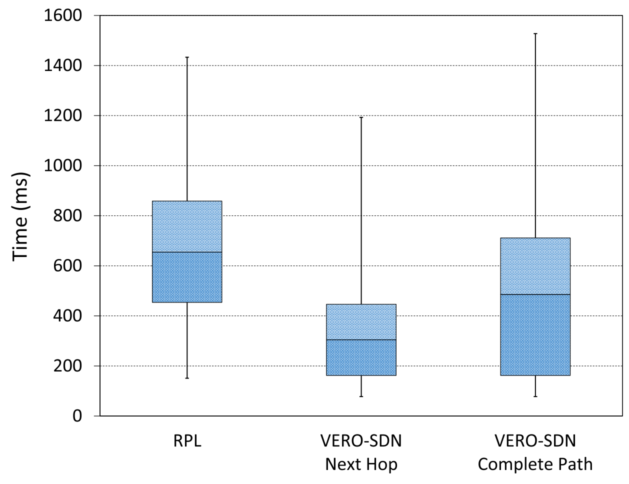


FIGURE 13. **The range of *TEED* values per protocol, *all-to-all* scenario.**

75% of *VERO-SDN* next-hop *TEED* values are below 450 *ms* in total, while the same applies only to the best 25% of RPL's values. Moreover, we find that although *VERO-SDN* complete-path mechanism produces some high numbers in terms of *TEED* delay (i.e., over 1, 400 *ms*), in general, the majority of results and the mean value are better than RPL.

Comparing the *next-hop* with the *complete-path* flow establishment mechanisms of *VERO-SDN*, we observe that *next-hop* obtains 30% reduced NOD time and generally performs better, in most evaluation results. This outcome is not intimidating for the use of the *complete-path* process since our data-intensive simulation scenario is biased in favor of the *next-hop* process due to the multitude of data messages transmitted to all the network nodes. So, the established flow rules in parts of the network are exploited by the *next-hop* for other intermediate transmissions, whereas the *complete-path* has to re-establish the flow rules for all nodes of the path, regardless if some of them already had valid routing information. This excess of control messages that *complete-path* creates and the opportunities that the all-to-all transmission provides to the next-hop can justify the above results.

Our immediate research plans include studying scenarios where the *complete-path* re-establishment of flows can have a valuable effect, e.g., in emergency IoT scenarios, where traffic prioritization, predefined path establishment, and the robustness that the *complete-path* demonstrated in Fig. 9, can play a vital role in the network's operation and performance.

Based on the above, we conclude that *VERO-SDN* is capable of establishing optimal flows for IoT communication requirements in the network, maintaining reliable message delivery and high communication performance. Furthermore, the adopted evaluation methodology highlighted its applicability for a wide range of IoT scenarios, beyond the traditional WSN deployments.

5) EVALUATION RESULTS OF *many-to-one* SCENARIO

At this point, we evaluate the performance of *VERO-SDN* and compare it with RPL in a *many-to-one* communication scenario, commonly used in WSN deployments. These network setups are used in data collection applications, where the network nodes are delivering data collected from sensor devices to a central node, with the intention to be processed in the infrastructure network. In particular, for RPL, the *many-to-one* scenario fits its original architectural specifications, since the DODAG construction process produces optimized routing paths from each node towards the central node, in contrast to its inability to establish efficient node-to-node routing paths, which are characterized by extended delays and packet losses, as discussed in the previous section.

According to the results of *many-to-one* scenario, all protocols achieved 100% performance in terms of PDR, since our analysis focuses on the performance of layer-3 protocols and assumes no radio interference. Moreover, the transmission rate and the number of communication messages are less impacted, compared to the *all-to-all* scenario (i.e., Fig. 14 and 15). These results verify the flawless operation of *VERO-SDN* for both *next-hop* and *complete-path* forwarding mechanisms, as functionalities of the network layer.

In Fig. 14, we depict the network's overall end-to-end delay (NOD) time results. We observe that *VERO-SDN* with *next-hop* flow establishment mechanism achieves the best time to deliver the full network data workload, in a total duration of 46, 412 *ms*, i.e., improved by 19% compared to RPL's performance. However, it is less than half of the 47% difference that is observed in the previous scenario. The main reason is the improved performance of RPL, as, in this scenario, its inability to establish efficient one-to-one communication paths is not present. Since there is a good quality in the routing paths for both protocols in this scenario, the improved performance of *VERO-SDN* can be justified from the better informed decisions for the routing paths, i.e., based on the whole topology.

Regarding the performance of *VERO-SDN's complete-path* flow establishment mechanism, we observe that is slightly outperformed, i.e., by 2%, from the RPL, and by 20% compared to the *next-hop*. As we explained in the previous

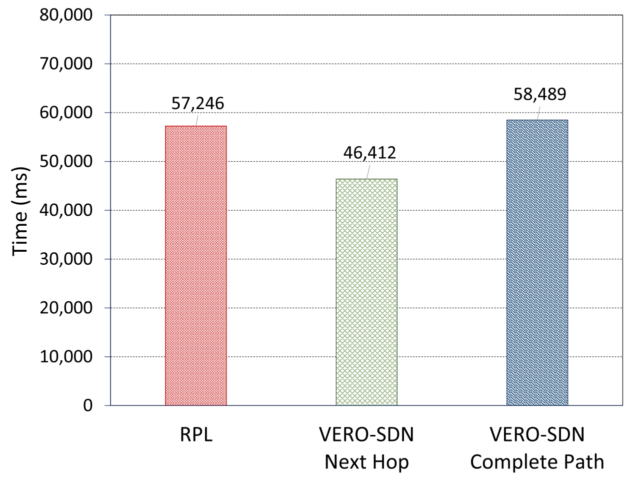


FIGURE 14. **The Network Overall End-to-end Delay (NOD), *many-to-one* scenario.**

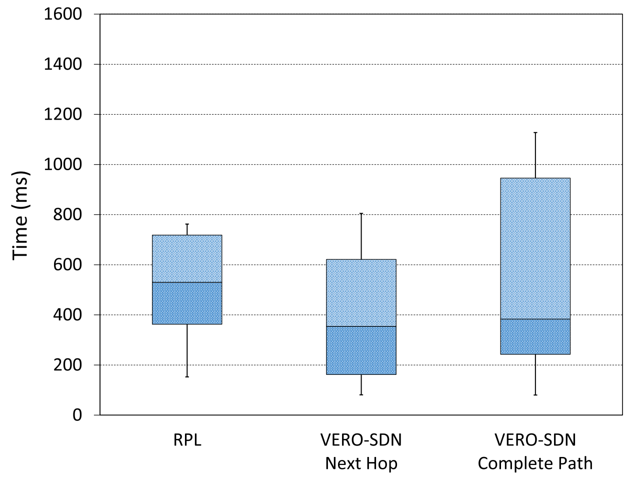


FIGURE 15. **The range of *TEED* values per protocol, *many-to-one* scenario.**

section, this is due to the requirements of the scenario that do not fully comply with the design goals of *complete-path*, preferably targeting ad-hoc communication patterns rather than transmissions from all nodes, which are typical in data collection application. Moreover, the box and whisker diagram (i.e., Fig. 15) depicts a summary of the *TEED* values in quartiles per-protocol. The results verify the above observations and justify, especially for the *complete-path* mechanism, the delays caused by the repeated process of establishing forwarding rules. The latter is the reason for the highest *TEED* values observed in the simulation (i.e., of over $1,100$ *ms*).

As a bottom line, we conclude that *VERO-SDN's* forwarding mechanisms can be successfully applied in traditional data collection communication scenarios, since they are characterized by reliable message delivery and high communication performance.

VI. FURTHER WORK DISCUSSION

Here, we discuss a number of topics that we plan to work on in the near future and can further improve the impact and applicability of the proposed *VERO-SDN* framework, including tackling mobility, energy efficiency, and scalability issues. Our main direction is to introduce a number of relevant intelligent control mechanisms and also exploit the advancements of 5G networks, such as incorporating *Software Defined Radio* (SDR) technologies.

We now further elaborate on the research challenges we plan to investigate:

- *Mobility:* Nowadays, heterogeneous networks consisting of both mobile and fixed nodes are common IoT application environments. Such applications cause challenges to the network's management and operation, e.g., due to the frequent changes in the network topology. In our immediate research interests is to study *VERO-SDN* in mobility environments. We believe that *VERO-SDN*, with its innovative processes of locating and analyzing the network topology, is able to deliver very good results in terms of network performance and QoS.
- *Energy Efficiency:* Energy conservation is an essential feature in the operation of IoT networks. Although at this stage one could argue that *VERO-SDN* may introduce additional energy consumption because of the usage of two radio channels, research works like [15] and [16] showed that a centralized system with a separate control channel, like *VERO-SDN*, can enable new methods in energy management for IoT networks.
- *Scalability:* Scaled up simulations with hundreds of network nodes are also in our near-future plans, justifying the efficient operation of *VERO-SDN* in large scale network scenarios. Although we have tested our facility with networks of 90 nodes in the context of this paper, we anticipate the operation of *VERO-SDN* engaging many BR control nodes (i.e., either connected directly or through a hierarchical structure to the *Controller*) will efficiently address large-scale scenarios, e.g., IoT deployments in Smart-Cities.
- *Industrial use:* The need for robust and adaptable operation in industrial and harsh environments motivate us to study the performance and efficiency of *VERO-SDN* as well as its novel aspects (e.g., its topology control mechanisms and the out-of-band control) in challenging communication environments. As such, we plan to evaluate our solution in test-beds equipped with dual-radio enabled Zolertia Re-Mote devices.

As an aftermath, *VERO-SDN* brings along new novel advancements for the centrally controlled network applications. In the near future, we expect to integrate new intelligent features in the *VERO-SDN Controller*, especially addressing the above research challenges. The modular architecture of *VERO-SDN* and the well-designed API interfaces can also enable the latest developments in networks, such as the network data plane slicing.

Towards the advancements in 5G network technologies and in particular the SDR solutions, where softwarized network interfaces are capable of communicating in a flexible manner over multiple radio bands, we expect that protocols like *VERO-SDN* supporting a dual-radio channel communication will use this infrastructure terrain as the catalyst for expanding further the integration between 5G and IoT networks.

As a bottom line, we consider *VERO-SDN* as an enabling platform for research on SDN-like capabilities for IoT devices. Our goal is to keep building on top of the *VERO-SDN Controller*, keep integrating new intelligence modules, and exploit them to enable new unique protocol features. Since *VERO-SDN* is an open-source project, we plan to keep extending its facilities with the assistance of the research community as well.

VII. CONCLUSIONS

We have presented *VERO-SDN*, our SDN OpenFlow-like framework for IoT networks, along with simulations featuring its novel network control features, validating the suitability of *VERO-SDN* for a wide range of IoT deployment conditions, e.g., topology structures and communication patterns. Our proposal architecturally adopts the usage of a separate long-range wireless channel that connects the network nodes with the *SDN Controller*, within one-hop distance. This innovative approach solves successfully major drawbacks of SDWSN, including the increased control messages overhead and the unreliable communication with the *SDN Controller*. Furthermore, our platform can be easily extended to support new algorithms, network protocol parameters and measurements; its modular architecture makes it also feasible to connect with external entities, such as machine-learning systems, providing intelligent network manipulation decisions based on data inputs from *VERO-SDN*.

Open and flexible architectures like *VERO-SDN* can enable a plethora of new innovative applications that may not even be foreseeable today since the future belongs to frameworks bringing elasticity, intelligence and centralized management into the network operation, emerging from the requirements of today's IoT applications.

Concluding, we envisage that *VERO-SDN* will constitute the basis and the common grounds for the research community to exploit further the benefits that the centralized control brings to WSN and IoT applications. For this reason, we provide *VERO-SDN Controller* and protocol implementation as an open-source platform.

REFERENCES

[1] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: A survey on recent developments and potential synergies," *J. Supercomput.*, vol. 68, no. 1, pp. 1–48, Apr. 2014.

[2] S. Li, L. Xu, and S. Zhao, "The Internet of Things: A survey," *Inf. Syst. Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

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

[4] I. Yaqoob, E. Ahmed, I. A. T. Hashem, A. I. A. Ahmed, A. Gani, M. Imran, and M. Guizani, "Internet of Things architecture: Recent advances, taxonomy, requirements, and open challenges," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 10–16, Jun. 2017.

[5] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, and R. Alexander, "RPL: IPv6 routing protocol for low-power and lossy networks," IETF, Fremont, CA, USA, Tech. Rep. RFC 6550, Mar. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6550

[6] O. Iova, P. Picco, T. Istomin, and C. Kiraly, "RPL: The routing standard for the Internet of Things... or is it?" *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 16–22, Dec. 2016.

[7] D. Kreutz, F. M. V. Ramos, P. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," 2014, *arXiv:1406.0440*. [Online]. Available: https://arxiv.org/abs/1406.0440

[8] M. Baddeley, R. Nejabati, G. Oikonomou, M. Sooriyabandara, and D. Simeonidou, "Evolving SDN for low-power IoT networks," in *Proc. 4th IEEE Conf. Netw. Softwarization Workshops (NetSoft)*, Jun. 2018, pp. 71–79.

[9] G. Violettas, S. Petridou, and L. Mamatas, "Routing under heterogeneity and mobility for the Internet of Things: A centralized control approach," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–7.

[10] K. Sood, S. Yu, and Y. 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.

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

[13] D. Carels, E. De Poorter, I. Moerman, and P. Demeester, "RPL mobility support for point-to-point traffic flows towards mobile nodes," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 6, pp. 1–13, Jun. 2015.

[14] *Zolertia RE-Mote Platform*. Accessed: Dec. 10, 2019. [Online]. Available: https://github.com/Zolertia/Resources/wiki/RE-Mote

[15] M. Del Prete, D. Masotti, A. Costanzo, M. Magno, and L. Benini, "A 2.4 GHz-868 MHz dual-band wake-up radio for wireless sensor network and IoT," in *Proc. IEEE 11th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2015, pp. 322–328.

[16] R. Piyare, T. Istomin, and A. L. Murphy, "WaCo: A wake-up radio COOJA extension for simulating ultra low power radios," in *Proc. Int. Conf. Embedded Wireless Syst. Netw.*, 2017, pp. 48–53.

[17] T. Theodorou and L. Mamatas, "CORAL-SDN: A software-defined networking solution for the Internet of Things," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2017, pp. 1–2.

[18] T. Theodorou, G. Violettas, P. Valsamas, S. Petridou, and L. Mamatas, "A multi-protocol software-defined networking solution for the Internet of Things," *IEEE Commun. Mag.*, vol. 57, no. 10, pp. 42–48, Oct. 2019.

[19] T. Theodorou and L. Mamatas, "Software defined topology control strategies for the Internet of Things," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2017, pp. 236–241.

[20] *VERO-SDN Open-Source Software and Demo Videos*. Accessed: May 1, 2020. [Online]. Available: https://github.com/SWNRG/vero-sdn

[21] B. Trevizan de Oliveira, R. C. A. Alves, and C. Borges Margi, "Software-defined wireless sensor networks and Internet of Things standardization synergism," in *Proc. IEEE Conf. Standards Commun. Netw. (CSCN)*, Oct. 2015, pp. 60–65.

[22] B. T. de Oliveira, L. B. Gabriel, and C. B. Margi, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," *IEEE Latin Amer. Trans.*, vol. 13, no. 11, pp. 3690–3696, Nov. 2015.

[23] B. T. de Oliveira and C. B. Margi, "TinySDN: Enabling TinyOS to software-defined wireless sensor networks," in *Proc. 34th Simpósio Brasileiro de Redes de Computadores*, 2016, pp. 1229–1237.

[24] E. Municio, J. Marquez-Barja, S. Latré, and S. Vissicchio, "Whisper: Programmable and flexible control on industrial IoT networks," *Sensors*, vol. 18, no. 11, pp. 40–48, 2018.

[25] G. Violettas, S. Petridou, and L. Mamatas, "Evolutionary software defined networking-inspired routing control strategies for the Internet of Things," *IEEE Access*, vol. 7, pp. 132173–132192, 2019.

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

[27] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

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

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

[30] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 513–521.

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

[32] A.-C. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SD-WISE: A software-defined WIreless SEnsor network," *Comput. Netw.*, vol. 159, pp. 84–95, Aug. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128618312192

[33] B. T. de Oliveira and C. B. Margi, "Distributed control plane architecture for software-defined wireless sensor networks," in *Proc. IEEE Int. Symp. Consum. Electron. (ISCE)*, Sep. 2016, pp. 85–86.

[34] S. Bera, S. Misra, S. K. Roy, and M. S. Obaidat, "Soft-WSN: Software-defined WSN management system for IoT applications," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2074–2081, Sep. 2018.

[35] M. Baddeley, U. Raza, A. Stanoev, G. Oikonomou, R. Nejabati, M. Sooriyabandara, and D. Simeonidou, "Atomic-SDN: Is synchronous flooding the solution to software-defined networking in IoT?" *IEEE Access*, vol. 7, pp. 96019–96034, 2019.

[36] M. Baddeley, U. Raza, M. Sooriyabandara, G. Oikonomou, R. Nejabati, and D. Simeonidou, "Poster: Atomic-SDN: A synchronous flooding framework for SDN control of low-power wireless," in *Proc. Int. Conf. Embedded Wireless Syst. Netw.*, Feb. 2019, pp. 206–207.

[37] M. Kaplan, C. Zheng, M. Monaco, E. Keller, and D. Sicker, "WASP: A software-defined communication layer for hybrid wireless networks," in *Proc. 10th ACM/IEEE Symp. Archit. for Netw. Commun. Syst. ANCS*, Oct. 2014, pp. 5–15.

[38] C. Gu, R. Tan, X. Lou, and D. Niyato, "One-hop Out-of-Band control planes for low-power multi-hop wireless networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 1187–1195.

[39] R. Wallace, "Achieving optimum radio range," Texas Instruments Incorporated, Dallas, TX, USA, Tech. Rep. SWRA479A, 2017.

[40] Open Networking Foundation, "SDN architecture," Open Netw. Found., Palo Alto, CA, USA, Tech. Rep. ONF TR-521, Feb. 2016. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/TR-521_SDN_Ar% chitecture_issue_1.1.pdf

[41] M. Gigli and S. Koo, "Internet of Things: Services and applications categorization," *Adv. Internet Things*, vol. 1, no. 2, pp. 27–31, 2011.

[42] *Node-RED*. Accessed: Dec. 10, 2019. [Online]. Available: https://nodered.org

[43] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki—A lightweight and flexible operating system for tiny networked sensors," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, Nov. 2004, pp. 455–462.

[44] *Contiki OS Enhanced With Dual-Radio Features Open-Source Software*. Accessed: May 8, 2019. [Online]. Available: https://github.com/clovervnd/Dual-radio-simulation

[45] *RE-Mote 2.4GHz Dual Antenna*. Accessed: Dec. 10, 2019. [Online]. Available: https://github.com/Zolertia/Resources/wiki/RE-Mote-2.4GHz-dual-antenna

[46] *Weka 3: Machine Learning Software in Java*. Accessed: Dec. 10, 2019. [Online]. Available: https://www.cs.waikato.ac.nz/~ml/weka

[47] M. Kulin, C. Fortuna, E. De Poorter, D. Deschrijver, and I. Moerman, "Data-driven design of intelligent wireless networks: An overview and tutorial," *Sensors*, vol. 16, no. 6, p. 790, Jun. 2016.

[48] *eWINE Elastic Wireless Networking Experimentation Grand Challenge Awards*. Accessed: Dec. 10, 2019. [Online]. Available: https://ewine-project.eu/grand-challenge/

[49] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wireless Netw.*, vol. 8, nos. 2–3, pp. 153–167, 2002.

[50] J. V. V. Sobral, J. J. P. C. Rodrigues, R. A. L. Rabêlo, J. Al-Muhtadi, and V. Korotaev, "Routing protocols for low power and lossy networks in Internet of Things applications," *Sensors*, vol. 19, no. 9, p. 2144, May 2019.

[51] S. Milardo, A. Venkatasubramanian, K. Vijayan, P. Nagaradjane, and G. Morabito, "From Reactive to Predictive Flow Instantiation: An artificial Neural Network approach to the SD-IoT," in *Proc. 24th Eur. Wireless Conf.*, May 2018, pp. 1–6.

[52] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proc. 31st IEEE Conf. Local Comput. Netw.*, Nov. 2006, pp. 641–648.

[53] *6LoWPAN Troubleshooting With Foren6*. Accessed: Apr. 10, 2020. [Online]. Available: https://cetic.github.io/foren6

[54] G. Almes, S. Kalidindi, and M. Zekauskas, "A one-way delay metric for IP performance metrics (IPPM)," IETF, Fremont, CA, USA, Tech. Rep. RFC 7679, Jan. 2016, pp. 1–20. [Online]. Available: https://tools.ietf.org/html/rfc7679

TRYFON THEODOROU (Member, IEEE) received the M.Sc. degree in artificial intelligence-knowledge-based systems from the University of Edinburgh, U.K., and the M.Sc. degree in applied informatics from the University of Macedonia, Thessaloniki, Greece, where he is currently pursuing the Ph.D. degree. He has been working in the ICT Sector, since 1993. Over the years, he successfully managed and developed a variety of software applications, either as research or commercial products. He is an active Researcher with several publications. He has participated in various international research projects, such as NECOS (H2020), WiSHFUL OC2 (H2020), and MONROE OC2 (H2020). His academic interests include wireless sensor networks, software-defined networks, communication security, and the Internet of Things.

LEFTERIS MAMATAS (Member, IEEE) is currently an Assistant Professor with the Department of Applied Informatics, University of Macedonia, Greece. He leads the Softwarized and Wireless Networks Research Group (`http://swn.uom.gr`), University of Macedonia. He has worked as a Researcher with the University College London, U.K., the Space Internetworking Center/Democritus University of Thrace, Greece, and DoCoMo Eurolabs, Munich. He has participated in many international research projects, such as NECOS (H2020), FED4FIRE+ OC4 (H2020), WiSHFUL OC2 (H2020), MONROE OC2 (H2020), Dolfin (FP7), UniverSELF (FP7), and Extending Internet into Space (ESA). He has published more than 60 articles in international journals and conferences. His research interests include software-defined networks, the Internet of Things, 5G networks, and multi-access edge computing. He has served as the General Chair for the WWIC 2016 conference and the INFOCOM SWFAN 2016 workshop and the TPC Chair for the INFOCOM SWFAN 2017, E-DTN 2009, and IFIP WWIC 2012 conferences/workshops. He has served as a Guest Editor for *Ad Hoc Networks* (Elsevier) journal.